# Vision tridimensionelle

## TOUR : Introduction à Mathematica

Ce notebook Mathematica a été crée pour introduire les concepts importants de Mathematica.
De plus, on y présente des exercices reliés aux modèles de caméra et à la géométrie projective.

## ■ Operations simples

(Utiliser shift−return pour faire evaluer une cellule)

```
3 + 5
```

```
8
```

```
2^8
```

```
256
```

Les nombres exacts deviennent flottant avec N[]

```
Pi
```

```
π
```

```
N[Pi]
```

```
3.14159
```

Pour avoir de l'aide, utiliser le Help ou faire "?"

```
?Prime
```

```
Prime[n] gives the nth prime number. More...
```

```
Prime[100000]
```

```
1299709
```

## Symboles et accents

Vous pouvez utiliser des symboles et des accents dans les expressions. Ceux−ci sont crées par une séquence
ESC <code du symbole> ESC
Tous les détails sont dans la section **3.10** du **Mathematica Book** (dans le Help Browser)

Pour un é, faire ESC e ' ESC

Pour un symbole $\alpha$, faire ESC alpha ESC

## Vecteurs

Pour ne pas imprimer un résultat, utiliser toujours un ";" à la fin d'une expression

```
u = {1, 2, 3};
```

```
u = {1, 2, 3}
```

```
{1, 2, 3}
```

Suggestion: Utilisez TOUJOURS un ";" sauf si vous voulez vraiment voir le contenu d'une variable

Pour afficher un vecteur ou matrice

```
u // MatrixForm
```

$$\begin{pmatrix} 1 \\ 2 \\ 3 \end{pmatrix}$$

```
MatrixForm[u]
```

$$\begin{pmatrix} 1 \\ 2 \\ 3 \end{pmatrix}$$

```
v = {4, 5, 6}; v // MatrixForm
```

$$\begin{pmatrix} 4 \\ 5 \\ 6 \end{pmatrix}$$

```
u + v
```

```
{5, 7, 9}
```

Produit scalaire

```
u.v
```

```
32
```

En forme symbolique

```
{x1, x2, x3}.{y1, y2, y3}
```

```
x1 y1 + x2 y2 + x3 y3
```

# Matrices

```
m = {{1, 2, 3}, {4, 5, 6}, {7, 8, 9}}; m // MatrixForm
```

$$\begin{pmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{pmatrix}$$

```
n = {{a11, 1}, {2, a22}}; n // MatrixForm
```

$$\begin{pmatrix} a11 & 1 \\ 2 & a22 \end{pmatrix}$$

```
Inverse[n]
```

$$\left\{\left\{\frac{a22}{-2+a11\,a22}, -\frac{1}{-2+a11\,a22}\right\}, \left\{-\frac{2}{-2+a11\,a22}, \frac{a11}{-2+a11\,a22}\right\}\right\}$$

```
Det[n]
```

```
-2 + a11 a22
```

# Calculs symboliques

```
f[x_] := Sqrt[x] Sqrt[1 + x];
```

```
f[y]
```

$$\sqrt{y}\ \sqrt{1+y}$$

```
f[1]
```

$$\sqrt{2}$$

```
N[f[1]]
```

```
1.41421
```

Intégration symbolique

```
Integrate[f[x], x]
```

$$\sqrt{1+x}\left(\frac{\sqrt{x}}{4}+\frac{x^{3/2}}{2}\right)-\frac{\text{ArcSinh}\left[\sqrt{x}\right]}{4}$$

```
g[x_] := x^2 + x;
```

Solution d'équation quadratique

```
s = Solve[g[x] == a, x]; s
```

$$\left\{\left\{x\to\frac{1}{2}\left(-1-\sqrt{1+4\,a}\right)\right\},\ \left\{x\to\frac{1}{2}\left(-1+\sqrt{1+4\,a}\right)\right\}\right\}$$

```
x /. s
```

$$\left\{\frac{1}{2}\left(-1-\sqrt{1+4\,a}\right),\ \frac{1}{2}\left(-1+\sqrt{1+4\,a}\right)\right\}$$

```
(x /. s) /. {a -> 5.0}
```

```
{-2.79129, 1.79129}
```

Pour en savoir plus sur les règles de remplacement (→ ), voir la section 1.4.2 du Mathematica Book.

## ■ Calculs numériques

```
NSolve[x^5 - 6 x^3 + 8 x + 1 == 0, x]
```

$$\{\{x\to -2.05411\},\ \{x\to -1.2915\},\ \{x\to -0.126515\},\ \{x\to 1.55053\},\ \{x\to 1.9216\}\}$$

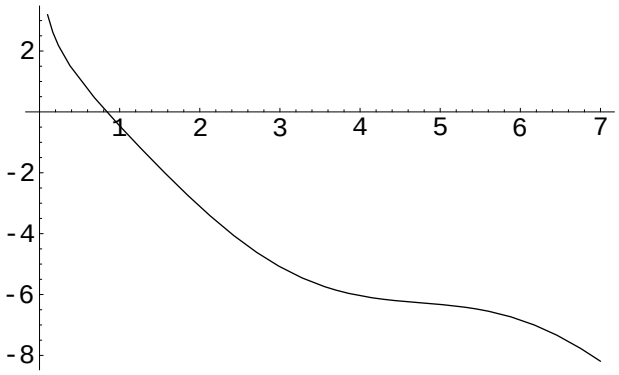
```
FindRoot[Cos[x] == x + Log[x], {x, 1}]
```

$$\{x\to 0.840619\}$$

# ■ Graphiques

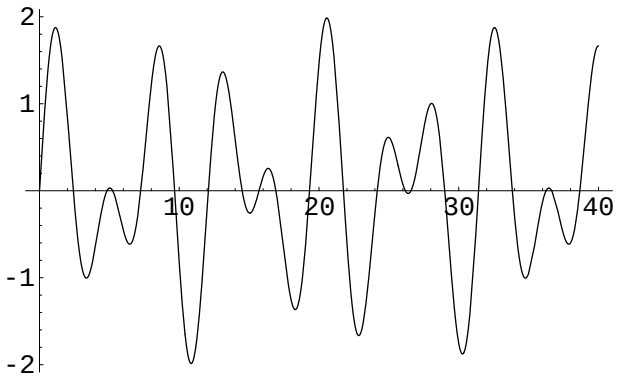
Fonction à une dimension

```
Plot[Cos[x] - x - Log[x], {x, 0.1, 7}];
```



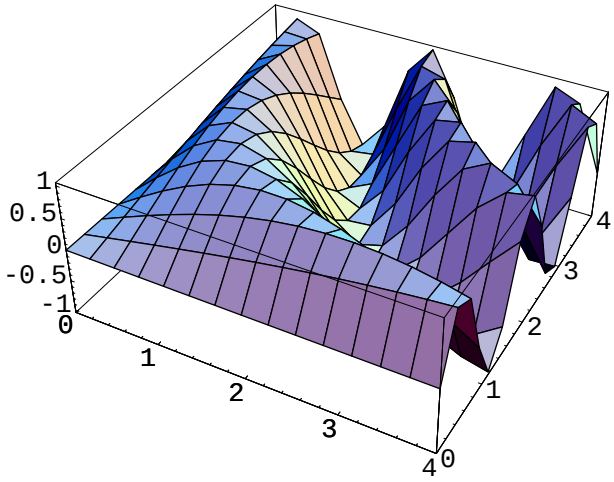

```
f[x_] := Sin[x] + Sin[1.6 x];

Plot[f[x], {x, 0, 40}];
```

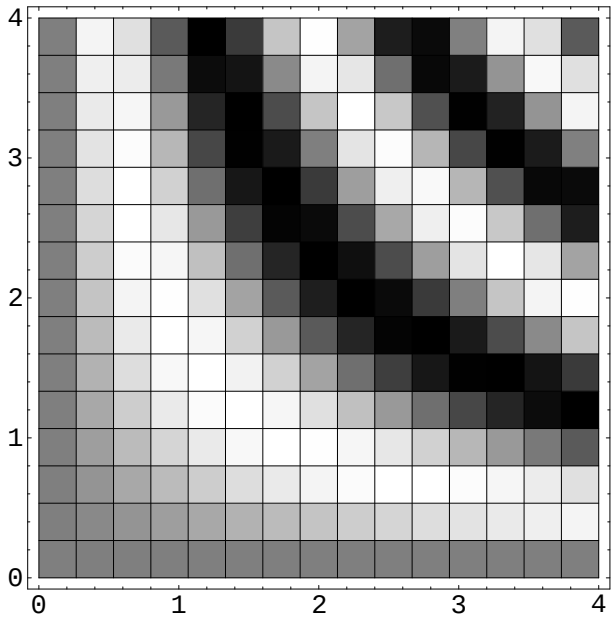



Fonction à deux dimensions

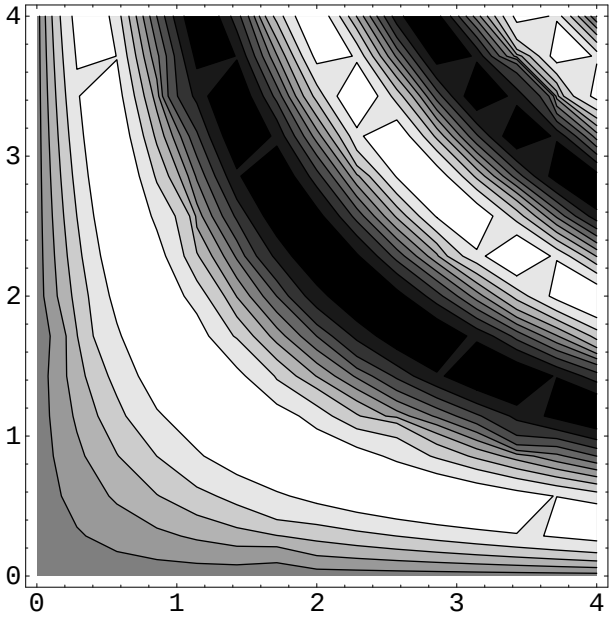
```
Plot3D[Sin[x y], {x, 0, 4}, {y, 0, 4}];
```




```
DensityPlot[Sin[x y], {x, 0, 4}, {y, 0, 4}];
```



```
ContourPlot[Sin[x y], {x, 0, 4}, {y, 0, 4}];
```




Pour connaitre les options possibles d' une commande graphique, utiliser **??**

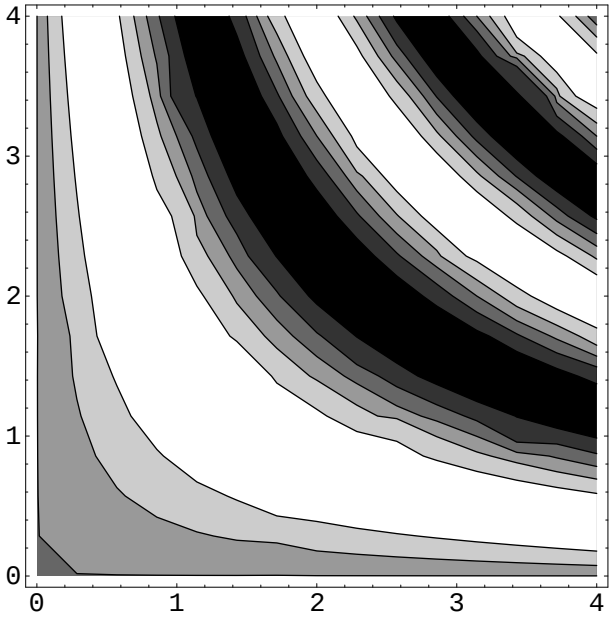
```
?? ContourPlot
```

```
ContourPlot[f, {x, xmin, xmax}, {y, ymin,
   ymax}] generates a contour plot of f as a function of x and y. More...

Attributes[ContourPlot] = {HoldAll, Protected}

Options[ContourPlot] = {AspectRatio → 1, Axes → False, AxesLabel → None, AxesOrigin → Automatic,
  AxesStyle → Automatic, Background → Automatic, ColorFunction → Automatic,
  ColorFunctionScaling → True, ColorOutput → Automatic, Compiled → True,
  ContourLines → True, Contours → 10, ContourShading → True, ContourSmoothing → True,
  ContourStyle → Automatic, DefaultColor → Automatic, Epilog → {}, Frame → True,
  FrameLabel → None, FrameStyle → Automatic, FrameTicks → Automatic, ImageSize → Automatic,
  PlotLabel → None, PlotPoints → 15, PlotRange → Automatic, PlotRegion → Automatic,
  Prolog → {}, RotateLabel → True, Ticks → Automatic, DefaultFont :→ $DefaultFont,
  DisplayFunction :→ $DisplayFunction, FormatType :→ $FormatType, TextStyle :→ $TextStyle}
```

```
ContourPlot[Sin[x y], {x, 0, 4}, {y, 0, 4}, Contours → 5];
```



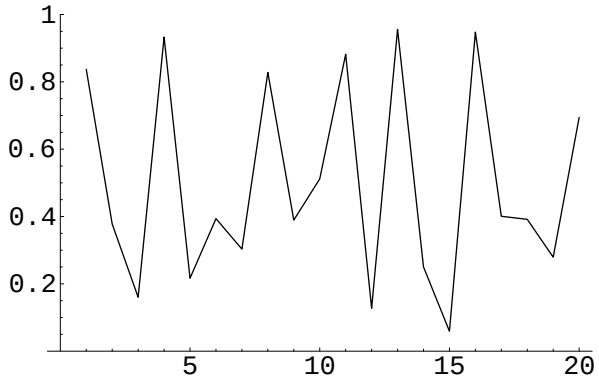

## ■ Tables

```
u = Table[Random[], {i, 1, 20}]
```

```
{0.837154, 0.377615, 0.160213, 0.932984, 0.216752, 0.39375,
 0.303173, 0.827786, 0.389341, 0.511735, 0.881977, 0.127353, 0.955636,
 0.250208, 0.0593088, 0.947581, 0.400639, 0.391573, 0.279231, 0.694211}
```

On peut afficher une graphiquement des listes à une ou plusieurs dimensions

```
ListPlot[u, PlotJoined -> True, PlotRange -> {0, 1}];
```

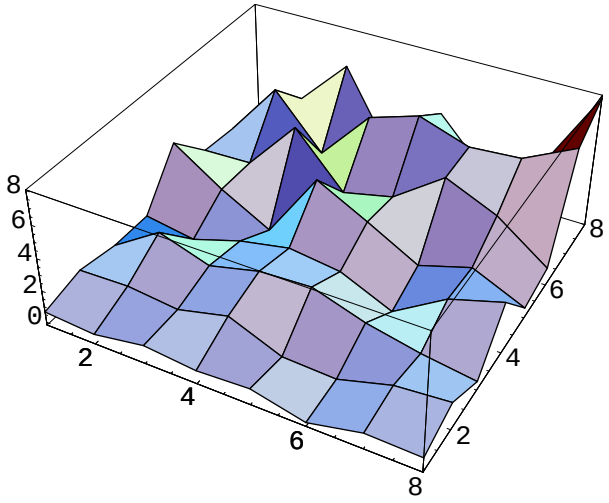



Dans cette table à deux dimensions, i est la rangée, j la colonne

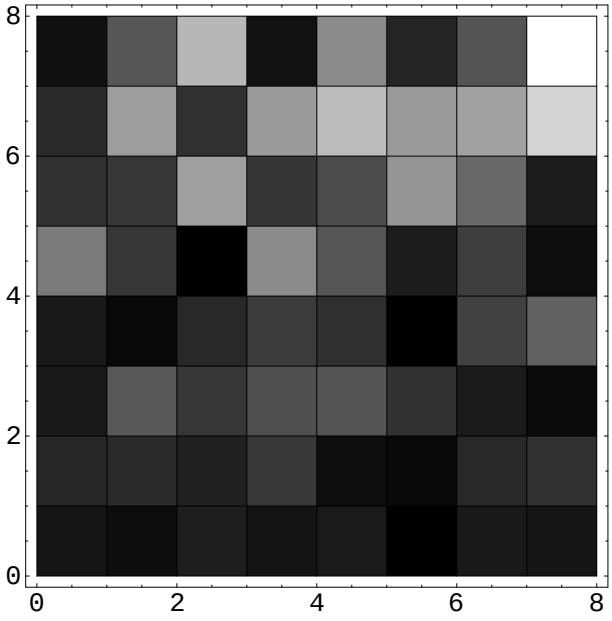
```
m = Table[(i) * Random[], {i, 1, 8}, {j, 1, 8}]; m // MatrixForm
```

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| 0.667583 | 0.440182 | 0.967838 | 0.652811 | 0.830429 | 0.0625678 | 0.807625 | 0.719828 |
| 1.22735 | 1.33764 | 1.0089 | 1.78408 | 0.448671 | 0.314164 | 1.24495 | 1.52938 |
| 0.8061 | 2.72062 | 1.6895 | 2.45132 | 2.60418 | 1.5459 | 0.851807 | 0.36869 |
| 0.801915 | 0.300476 | 1.26439 | 1.88034 | 1.4802 | 0.0502045 | 2.03389 | 3.00103 |
| 3.78187 | 1.71867 | 0.0201028 | 4.29108 | 2.66019 | 0.933257 | 1.90773 | 0.467638 |
| 1.58002 | 1.67866 | 4.91027 | 1.65852 | 2.37166 | 4.58685 | 3.20666 | 0.92114 |
| 1.36358 | 4.8255 | 1.52842 | 4.78407 | 5.77323 | 4.73764 | 4.96911 | 6.53227 |
| 0.546997 | 2.66458 | 5.64682 | 0.599717 | 4.2907 | 1.17137 | 2.59446 | 7.8515 |

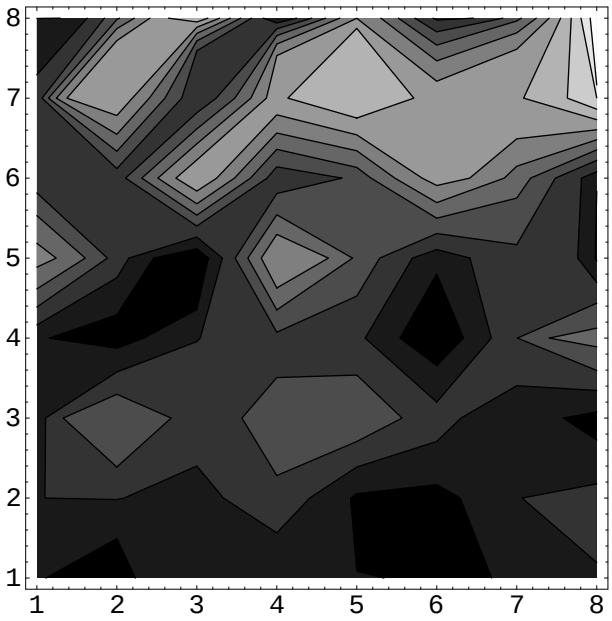
```
ListPlot3D[m, PlotRange -> All];
```




```
ListDensityPlot[m, PlotRange -> All];
```



```
ListContourPlot[m]
```




```
- ContourGraphics -
```

Première rangée d'un tableau

```
m[[1]]
```

```
{0.667583, 0.440182, 0.967838, 0.652811, 0.830429, 0.0625678, 0.807625, 0.719828}
```

Élément de la rangé 1 et colonne 5

```
m[[1, 5]]
```

```
0.830429
```

## ■ Operations sur les tables

```
u = {a, b, c, d, e}
```

```
{a, b, c, d, e}
```

```
Delete[u, 3]
```

```
{a, b, d, e}
```

```
Append[u, 999]
```

```
{a, b, c, d, e, 999}
```

```
Take[u, 3]
```

```
{a, b, c}
```

```
Take[u, -3]
```

```
{c, d, e}
```

```
Partition[{1, 2, 3, 4, 5, 6, 7, 8}, 2] // MatrixForm
```

$$\begin{pmatrix} 1 & 2 \\ 3 & 4 \\ 5 & 6 \\ 7 & 8 \end{pmatrix}$$

```
Partition[{1, 2, 3, 4, 5, 6, 7, 8}, 4] // MatrixForm
```

$$\begin{pmatrix} 1 & 2 & 3 & 4 \\ 5 & 6 & 7 & 8 \end{pmatrix}$$

Applique la fonction toto[] à chaque élement de u (voir aussi Map)

```
toto /@ u
```

```
{toto[a], toto[b], toto[c], toto[d], toto[e]}
```

```
?Map
```

```
Map[f, expr] or f /@ expr applies f to each element on the first level in expr.
   Map[f, expr, levelspec] applies f to parts of expr specified by levelspec. More...
```

Applique la fonction toto à l'ensemble du vecteur u (voir aussi Apply)

```
toto @@ u
```

```
toto[a, b, c, d, e]
```

```
?Apply
```

```
Apply[f, expr] or f @@ expr replaces the head of expr by f. Apply[f,
   expr, levelspec] replaces heads in parts of expr specified by levelspec. More...
```

## ■ Exemples pour les matrices

```
m = {{a, b, c}, {d, e, f}, {g, h, i}}; m // MatrixForm
```

$$\begin{pmatrix} a & b & c \\ d & e & f \\ g & h & i \end{pmatrix}$$

```
toto /@ m // MatrixForm
```

$$\begin{pmatrix} \text{toto}[\{a, b, c\}] \\ \text{toto}[\{d, e, f\}] \\ \text{toto}[\{g, h, i\}] \end{pmatrix}$$

Applique toto à la deuxième dimension seulement...

```
Map[toto, m, {2}] // MatrixForm
```

$$\begin{pmatrix} \text{toto}[a] & \text{toto}[b] & \text{toto}[c] \\ \text{toto}[d] & \text{toto}[e] & \text{toto}[f] \\ \text{toto}[g] & \text{toto}[h] & \text{toto}[i] \end{pmatrix}$$

Applique une fonction à deux (ou plusieurs) tableaux à la fois

```
MapThread[toto, {{a1, b1, c1}, {a2, b2, c2}}]
```

```
{toto[a1, a2], toto[b1, b2], toto[c1, c2]}
```

```
MapThread[toto, {{a1, b1, c1}, {a2, b2, c2}, {a3, b3, c3}}]
```

```
{toto[a1, a2, a3], toto[b1, b2, b3], toto[c1, c2, c3]}
```

un équivalent... moins élégant...

```
toto /@ Transpose[{{a1, b1, c1}, {a2, b2, c2}}]
```

```
{toto[{a1, a2}], toto[{b1, b2}], toto[{c1, c2}]}
```

## ■ Fonctions

```
f[x_] := 2 / x;
```

Les cas spéciaux passe avant les cas généraux

```
f[0] := e;
```

```
f[6] + f[a + b] + f[0]
```

$$\frac{1}{3} + \frac{2}{a + b} + e$$

Efface le symbole f

```
Clear[f];
```

Fonction factorielle

```
f[0] := 1;

f[i_] := i * f[i - 1];

f[4]
```

```
24
```

Les paramètres des fonctions sont des "patterns"

```
Clear[g];

g[x_] := 2 x;

g[{x_, y_}] := x + y;

g[5]
```

```
10
```

```
g[{1, 2}]
```

```
3
```

## ■ Variables locales dans les fonctions

Voici une fonction qui mélange un tableau.
Notez que le résultat de la dernière commande d'un "block" est retournée par la fonction.

```
permute[{}] := {};
```

Les variables pos,t1,t2 sont locales à la fonction.

```
permute[t_] := Module[{pos, t1, t2}, (
            pos = Random[Integer, {1, Length[t]}];
            t1 = permute[Delete[t, pos]];
            t2 = Append[t1, t[[pos]]];
            t2
            )];

permute[{1, 2, 3, 4, 5}]
```

```
{4, 5, 1, 2, 3}
```

Cette fonction inverse une table

```
invtable[{}] := {};

invtable[t_] := Append[invtable[Delete[t, 1]], t[[1]]];

invtable[{1, 2, 3, 4, 5, 6}]
```

```
{6, 5, 4, 3, 2, 1}
```

## ■ Comment définir une fonction pure

```
u = {1, 2, 3, 4};

toto /@ u
```

```
{toto[1], toto[2], toto[3], toto[4]}
```

Les symboles représente un paramètre de la fonction, le & est l'opérateur qui construit la fonction

```
(# + 1) & /@ u
```

```
{2, 3, 4, 5}
```

## ■ Un peu d'Orienté−Objet (!)

Comment peut−on représenter un "objet" et definir des "méthodes"?
On utilise un nom de fonction...

Constructeurs...

```
Point2d[] := Point2d[0, 0];

Affiche[p_Point2d] := "(" <> ToString[p[[1]]] <> "," <> ToString[p[[2]]] <> ")";

Ajoute[Point2d[x1_, y1_], Point2d[x2_, y2_]] := Point2d[x1 + x2, y1 + y2];

a = Point2d[1, 2];
b = Point2d[3, 4];

Affiche[a]
```

```
(1,2)
```

```
Affiche[a + b]
```

```
Affiche[Point2d[1, 2] + Point2d[3, 4]]
```

```
c = Ajoute[a, b];
```

```
Affiche[c]
```

```
(4,6)
```

## L’ennemi : le FOR

Ne jamais utiliser le FOR pour une boucle. Utiliser Map (/@) ou Apply (@@) à la place.

```
t = Table[Random[], {i, 1, 20000}];
```

```
r1 = Timing[For[sum = 0; i = 1, i <= Length[t], i++, sum += t[[i]]]; sum]
```

```
{0.47 Second, 9979.79}
```

Nettement plus efficace...

```
r2 = Timing[Plus @@ t]
```

```
{0.02 Second, 9979.79}
```

```
ratio = r1[[1, 1]] / r2[[1, 1]];
Print["Le FOR est ", ratio, " fois plus lent que le Apply"];
```

```
Le FOR est 23.5 fois plus lent que le Apply
```

## Graphisme 3D de base

Création de 40 points aléatoires 3D

```
p = Partition[Table[Random[], {i, 1, 40 * 3}], 3];
```

Afficher les 10 premiers points

```
Take[p, 10] // MatrixForm
```

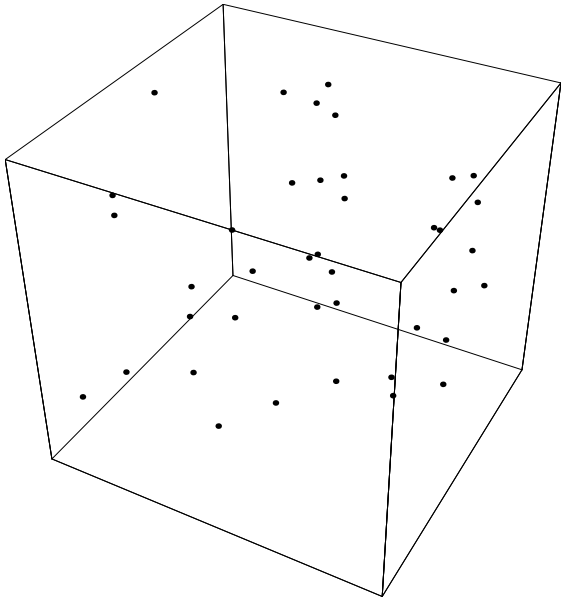
$$\begin{pmatrix} 0.273 & 0.866644 & 0.505928 \\ 0.705017 & 0.877724 & 0.102168 \\ 0.971485 & 0.551494 & 0.682926 \\ 0.412812 & 0.75314 & 0.868055 \\ 0.858179 & 0.736006 & 0.0432672 \\ 0.934875 & 0.789804 & 0.402934 \\ 0.337415 & 0.85991 & 0.253467 \\ 0.256514 & 0.0131079 & 0.878473 \\ 0.980467 & 0.38987 & 0.507179 \\ 0.173456 & 0.102743 & 0.287702 \end{pmatrix}$$

```
s1 = Point /@ p;
```

Ceci est équivalent à la formulation:

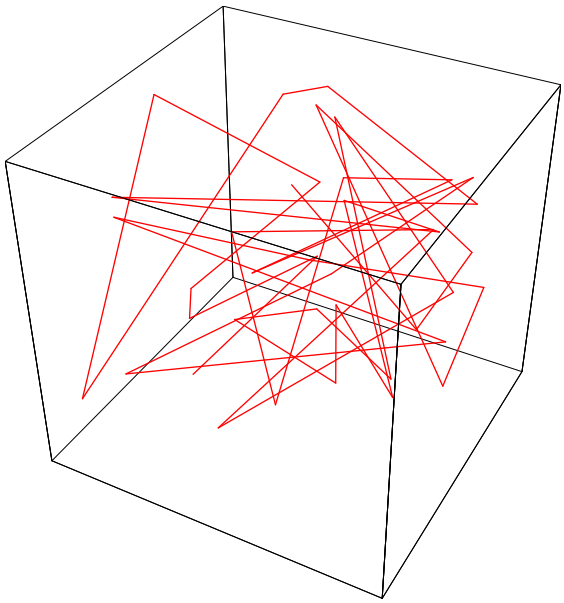
```
s1 = Table[Point[p[[i]]], {i, 1, Length[p]}];
```

```
Show[Graphics3D[s1]];
```



```
s2 = Line[p];
```

```
Show[Graphics3D[{RGBColor[1, 0, 0], s2}]];
```

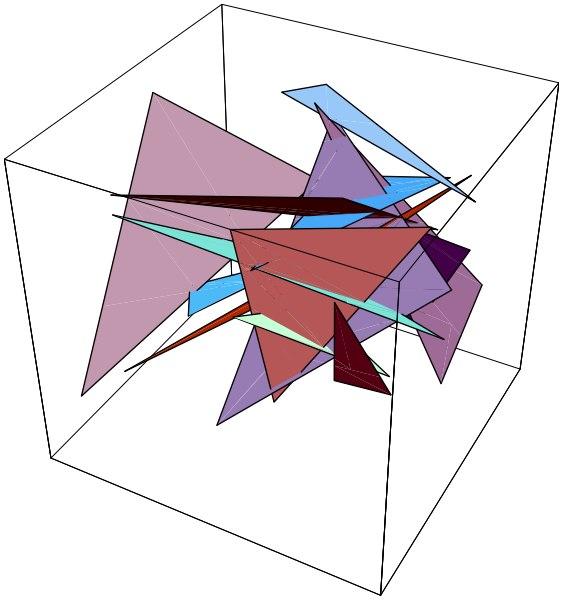


Partitione la liste de point en groupes de 3 points

```
poly = Partition[p, 3];

s3 = Polygon /@ poly;

Show[Graphics3D[s3]];
```

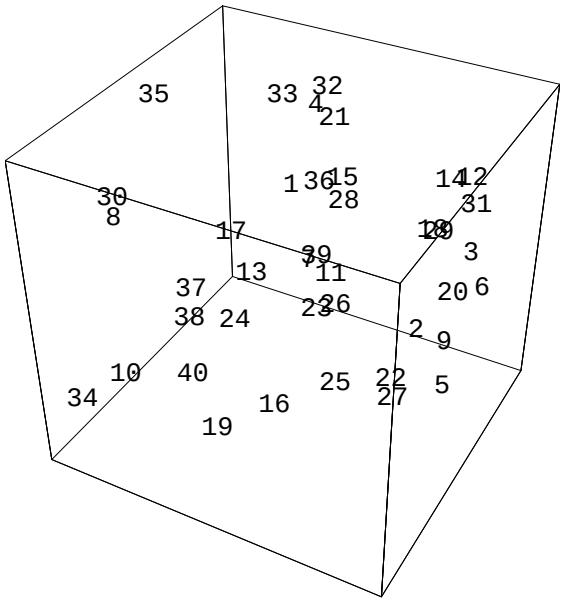


```
? Text
```

```
Text[expr, coords] is a graphics primitive that represents text corresponding
   to the printed form of expr, centered at the point specified by coords. More...
```

```
s4 = Table[Text[i, p[[i]]], {i, 1, Length[p]}];

Show[Graphics3D[s4]];
```




```
Show[Graphics3D[{s2, RGBColor[0, 1, 0], s4}]];
```