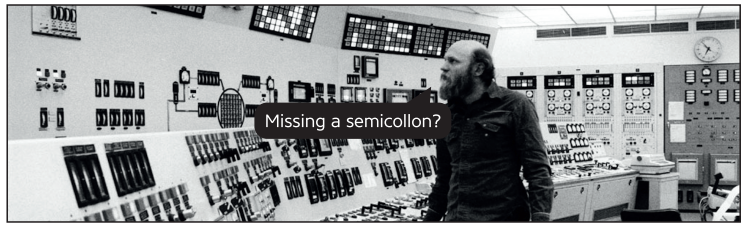


# MÉMO

PROCESSING.ORG  
VERSION 3.5.4



RANDOM(lab) | ESADSE | version 2.0 | Janvier 2020 | <http://randomlab.io>

Mémo (non exhaustif) des usages de Processing à l'usage des Graphistes/Artistes/Designers/Ingénieurs

## COMMENTAIRES

```
// Une ligne de commentaire
/*
Un paragraphe de commentaire
*/
```

## INITIALISATION D'UNE SCÈNE (FONCTION DE BASE)

```
//Une seul fois au début
void setup(){
}
//Tout le temps en boucle
void draw(){
}
```

## FONCTIONS DE CONSTRUCTION

```
//Taille de la fenêtre
size(l,h); //où l et h est un int
size(l,h,P2D); //avec accélération 2D
size(l,h,P3D); //avec accélération 3D

fullscreen(); //Mode plein écran
fullscreen(2D); //Plein écran avec accélération 2D
fullscreen(3D); //Plein écran avec accélération 3D
//nbre d'images/sec
frameRate(c); //où c est un int
```

## FONCTIONS : EVÈNEMENT SOURIS & CLAVIER

```
//Quand une touche est appuyée (type KeyEvent)
void keyPressed(){
}
//Quand une touche est relâchée (type KeyEvent)
void keyReleased(){
}
//Quand la souris est cliquée (type MouseEvent)
void mouseClicked(){
}
//Quand la souris est appuyée (type MouseEvent)
void mousePressed(){
}
//Quand la souris est relâchée (type MouseEvent)
void mouseReleased(){
}
//Quand la souris est en mouvement (type MouseEvent)
void mouseMoved(){
}
//Quand la souris est en mouvement+appuyée
void mouseDragged(){
}
```

## VARIABLES

```
//où bob est le nom que vous souhaitez!
int bob1 = 5;
float bob2 = 5.545;
char bob3 = 'A';
boolean bob4 = true;
color bob5 = color(250,231,32);
String bob6 = "Hello World!"
```

## VARIABLES GLOBALES

```
width // largeur de la fenêtre
height // hauteur de la fenêtre
mouseX // position en X de la souris
mouseY // position en Y de la souris
pmouseX // position en X de la souris à l'image précédente
pmouseY // position en Y de la souris à l'image précédente
frameCount // nombre d'images depuis le début
frameRate // cadence d'images/seconde.
displayWidth // largeur de l'écran
displayHeight // hauteur de l'écran
```

## DANS LES "MOUSEEVENT" :

```
mouseButton //Correspond au bouton de la souris
//Peut être égal à LEFT ou RIGHT
```

## DANS LES "KEYBOARDEVENT" :

```
keyCode //Correspond au int unique d'une touche clavier
(par exemple 32 pour la touche ESPACE)

key //Correspond à la touche clavier char
```

## CONSOLE

```
println("Hello World");
```

## ASTUCE :

```
//println peut recevoir tout type de variable.
//Quelques exemples :
println("frame : "+frameCount);
println("valeur : "+bob);
println("Hello"+"my name is"+" Pat!");
println("Hello"+ random(100));
```

## FONCTION DE DESSIN - GÉNÉRAL

### FORMES :

```
/* Ici,
x est un float - position en x,
y est un float - position en y,
z est un float - position en z,
l est un float - largeur de la forme,
h est un float - hauteur de la forme,
*/
point(x,y); //un point
point(x,y,z); //un point en 3D
rect(x,y,l,h); //un rectangle
rect(x,y,l,h,r); //un rectangle arrondis de r
rect(x,y,l,h,r1,r2,r3,r4);
// r1, r2, r3, r4 désigne les arrondis des 4 coins du
rectangle.
ellipse(x,y,l,h); //une ellipse
line(x1,y1,x2,y2); //une ligne 2D
line(x1,y1,x2,y2,z1,z2) //une ligne 3D
triangle(x1,y1,x2,y2,x3,y3); //un triangle
quad(x1,y1,x2,y2,x3,y3,x4,y4);
//un quadrilatère
```

### COULEURS :

```
/* Ici,
g est un int : valeur de gris,
(255=blanc, 0=noir)
r est un int : quantité de rouge de 0 à 255
g est un int : quantité de vert de 0 à 255
b est un int : quantité de bleu de 0 à 255
a est un int : quantité de transparence de 0 à 255 */
fill(g); //remplissage en gris
fill(g,alpha); //remplissage en gris+alpha
fill(r,v,b); //remplissage en couleur
fill(r,v,b,alpha); //remplissage en couleur+alpha
```

```
stroke(g); //contour gris
stroke(g,alpha); //contour gris+alpha
stroke(r,v,b); //contour couleur
stroke(r,v,b,alpha); //contour couleur+alpha
```

```
noFill(); //pas de remplissage
noStroke(); //pas de contour
```

### ATTRIBUTS :

```
strokeWeight(3); //Épaisseur du contour
```

```
//ici;
//a peut être égal à ROUND
//a peut être égal à SQUARE
//a peut être égal à PROJECT
strokeCaps(a); //Style des fin de ligne
```

```
//ici;
//b peut être égal à MITER
//b peut être égal à ROUND
//b peut être égal à BEVEL
strokeJoin(b); //Style des angles de ligne
```

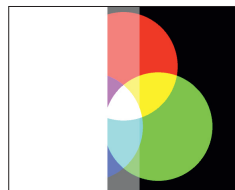
## OPTION DE MÉLANGE : `blendMode()`;

/\* Ajouter un effet de superposition au forme suivante.  
( de type calque de fusion ). \*/

```
blendMode(BLEND);
//Mode normal
```



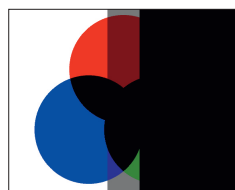
```
blendMode(ADD);
//Mode soustractif :
C = max(B - A*factor, 0)
```



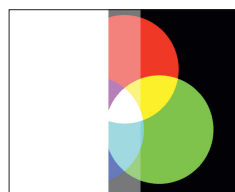
```
blendMode(SUBTRACT);
// Mode soustractif :
C = max(B - A*factor, 0)
```



```
blendMode(DARKEST);
// Mode Obscurcir :
C = min(A*factor, B)
```



```
blendMode(LIGHTEST);
//mode Eclaircir :
C = max(A*factor, B)
```



```
blendMode(DIFFERENCE);
//mode Difference : soustrait
les couleurs de l'image sous-
jacente.
```



**blendMode(EXCLUSION);**  
 //Similaire à différence, mais plus important.



**blendMode(MULTIPLY);**  
 //Multiplie la couleur.



**blendMode(SCREEN);**  
 //Opposé à MULTIPLIE, utilise la valeur inverse des couleurs.



#### Exemple :

```
blendMode(ADD);

fill(255,0,0,150);
ellipse(width/2,height/2-300/2,300,300);

fill(0,255,0,150);
ellipse(width/2-300/3,height/2,300,300);

fill(0,0,255,150);
ellipse(width/2+300/3,height/2,300,300);
```

#### SAUVEGARDES D'IMAGES :

##### SAVE & SAVEFRAME:

```
save("sauvegarde.png");
```

## FONCTION DE DESSIN - AVANCÉ

### LES VERTEX :

```
/* Ici,
x est la position en x en float,
y est la position en y en float,
*/
vertex(x,y); //un point de polygone
beginShape(); //le debut du polygone
endShape(); //la fin du polygone
```

#### Exemple:

```
beginShape();
vertex(20, 20);
vertex(40, 20);
vertex(40, 40);
vertex(60, 40);
vertex(60, 60);
vertex(20, 60);
endShape();
```

### LES TRANSFORMATIONS :

```
/* Ici,
x est le décalage en x en float
y est le décalage en y en float
z est le décalage en z en float */
translate(x,y); //décalage en 2D
translate(x,y,z); //décalage en 3D(OPENGL)
```

```
// Ici,
// r est le décalage d'angle en RADIEN! (PI!) en float
rotate(r); //décalage d'angle en 2D
rotateX(r); //décalage d'angle X en 3D (OPENGL)
rotateY(r); //décalage d'angle Y en 3D (OPENGL)
rotateZ(r); //décalage d'angle Z en 3D (OPENGL)
```

#### Astuce :

//Faire sa rotation en degrés et pas en radians!

```
rotate(radians(45)); //une rotation à 45 degrés!
```

```
/* Ici,
a,x,y et z est un float - indice d'agrandissement
si a,x,y,z est plus petit que 1 la forme devient plus petite
si a,x,y,z est plus grand que 1 la forme devient plus grande
si a, x,y,z est égale à 1 la forme à sa taille d'origine */
scale(a);
//changement de taille sur tout les
//axes X-Y-Z
scale(x,y); //chmgt de taille sur X et Y
scale(x,y,z); //chmgt de taille sur X, Y et Z
pushMatrix(); //Initialiser une transformation
popMatrix(); //Finir une transformation
resetMatrix(); //Redéfinir une transformation
```

## DÉCLARATION :

/\* PGraphics permet de créer un clip graphique dans lequel on peut dessiner, intégrer des images, faire des manipulations de pixels ou même implémenter de la 3D.

Il comprend quasiment toutes les fonctions de PApplet c'est à dire de la scène principale.

On l'affiche grâce à la fonction image();

\*/

PGraphics pg;

```
void setup() {
  size(300, 300);
  pg = createGraphics(40, 40);
  //Crée un objet PGraphic d'une taille de 40 par 40
}

void draw() {
  pg.beginDraw(); //On commence a dessiner

  pg.background(100);
  pg.stroke(255);
  pg.line(20, 20, mouseX, mouseY);

  pg.endDraw(); //On termine de dessiner

  image(pg, 9, 30); //On affiche
  image(pg, 51, 30);
}
```

## GESTION DE L'AFFICHAGE

## DÉCLARATION :

hint(DISABLE\_OPENGL\_2X\_SMOOTH);

//Permet de désactiver le lissage des formes en P3D. voir aussi la fonction smooth();

hint(DISABLE\_DEPTH\_TEST);

//Désactive le «z-buffer», c'est à dire la profondeur 3D, cette option est utilisé pour par exemple dessiner au dessus de scènes 3D, par exemple mettre des boutons de control au-dessus d'une scène en 3D.

hint(ENABLE\_DEPTH\_TEST);

//Permet de réactiver l'option 3D.

hint(ENABLE\_DEPTH\_SORT);

/\*Activer primitive «z-sorting» des triangles et des lignes dans P3D et OpenGL. Cela peut ralentir considérablement les performances, et l'algorithme n'est pas encore parfait. Restaurer la valeur par défaut avec un :\*/

hint(DISABLE\_DEPTH\_SORT);

## FONCTIONS PERSONNELLES

(SANS RETOUR > **void**)

### FONCTION SIMPLE :

```
void draw(){
    maFunction();//utilisation de ma fonction
}
```

```
void maFunction(){
    //mes actions à effectuer.
}
```

### FONCTION AVEC PASSAGE D'ARGUMENTS :

```
void draw(){
    maFunction(55);//utilisation de ma fonction
    maFunction(99);//utilisation de ma fonction
}
```

```
void maFunction(int bob){
    //mes actions à effectuer.
    //bob, un int n'existe que dans cette fonction
    rect(20, 20, bob, bob);
    //utilisation de la variable bob
    //Dans un premier cas bob vaut 55
    //Dans un deuxième cas 99
}
```

### AUTRE EXEMPLE :

```
void draw(){
    maFunction("Salut",55);//utilisation de ma
fonction
    maFunction("Yop!",99);//utilisation de ma
fonction
}
```

```
void maFunction(String pat, int bob){
    /* mes actions à effectuer.
    bob, un int et pat un String n'existe que dans
    cette fonction
    */
    text(pat, bob, bob);
    //utilisation de la variable bob et pat
    //Dans un premier cas bob vaut 55 et pat vaut "Salut"
    //Dans un deuxième cas bob vaut 99 et pat vaut «Yop!»
}
```

## FONCTIONS PERSONNELLES

(AVEC RETOUR)

### FONCTION SIMPLE :

```
void draw(){
    int bob = maFunction();
    //Ici bob prend la valeur 5.
}
```

```
int maFunction(){
    //retourne le nombre 5.
    return 5;
}
```

/\* Ici on utilise un retour d'entier, mais tout les types son envisageable comme float, String, boolean, PImage, PGraphics, PShape, Object, .... ou même une classe personnalisée. \*/

### FONCTION AVEC PASSAGE D'ARGUMENTS :

```
//Par exemple une fonction d'addition
void draw(){
    int bob = maFunction(3,2); //bob vaut 5
    int pat = maFunction(100,5); //pat vaut 105
}

int maFunction(int a, int b){
    return a + b;
}
```

### AUTRE EXEMPLE :

```
void draw(){
    println(maFunction("Salut",55));
    /*utilisation de ma fonction avec un retour de chaine
    de caractère, affiche «Salut55»
    */
    println(maFunction("Yop!",99));
    /*utilisation de ma fonction avec un retour de chaine
    de caractère, affiche «Yop99»
    */
}

String maFunction(String pat, int bob){
    String couple = at+str(bob);
    return couple;
}
```

## LES IMAGES (OBJETS BITMAPS) : PImage

//Emplacement mémoire qui contient mon image pour l'instant vide.

PImage bob;

```
void setup() {
    //Chargement dans mon image dans l'emplacement mémoire
    bob = loadImage("bob_l_ponge.png");
}

void draw() {
    //affichage de mon image
    image(bob,40,40);
}
```

### VARIABLES ACCESSIBLES DEPUIS UN OBJET PImage

pixels[] // Tableau de pixels contenant la totalité des pixels d'une image.  
width // Largeur de l'image.  
height // Hauteur de l'image.

### MÉTHODES ACCESSIBLES DEPUIS UN OBJET PImage

**.loadPixels();**

//Charge les pixels de l'image au tableau de pixels accessibles : array[]

**.updatePixels();**

//Charge les pixels du tableau de pixel[] à l'image.

**.resize(w,h);**

//change la taille de l'image en largeur et hauteur  
//où w est un int pour la nouvelle largeur  
//où h est un int pour la nouvelle hauteur

**.get(x,y);**

//Avec 2 paramètres, retourne la couleur d'un pixel.  
//où x est un int et designe la position en x du pixel.  
//ou y est un int et designe la position en y du pixel.

**.get(x,y,w,h);**

//Avec 4 paramètre renvoie un nouvelle objet PImage.  
//où x est un int et designe la position en x du pixel.  
//ou y est un int et designe la position en y du pixel.  
//où w est un int et designe la largeur du carré.  
//ou h est un int et designe la hauteur.

**.set(x,y,c);**

// Attribuer une couleur a un pixel donnée  
//où x est un int et designe la position en x du pixel.  
//ou y est un int et designe la position en y du pixel.  
//où c est un int et designe la nouvelle couleur du pixel.

**.set(x,y,ima);**

// Attribuer un morceau d'image dans l'image.  
//où x est un int et designe la position en x du pixel.  
//ou y est un int et designe la position en y du pixel.  
//où ima est un PImage et designe le morceau d'image a coller.

**.mask(image);**

// Applique un masque alpha à l' image à l'aide d'une autre image.  
//où image est un PImage et designe le morceau d'image à coller.

**.copy();**

// Copies l'objet PImage (Attention une copie n'est pas un "=" )

**.blend();**

// Copie un pixel ou un rectangle de pixel avec l'application d'un filtre.

**.save();**

// Sauvegarde une image au format TIFF, TARGA, JPEG, ou PNG suivant l'extension choisit.

### Exemple :

```
bob.save("MonImage.png");
// Le format jpg peut prendre un paramètre permettant de régler la qualité de l'image.
bob.save("MonImage.jpg", 8);
```

```
.filter(TYPE);
```

//Applique un filtre à l' image  
//où type est le type de filtre à appliquer.

#### LES TYPES POSSIBLES :

// Pour une image bob.

```
Image bob;  
void setup(){  
    bob = loadImage("monImage.png");  
}
```

#### THRESHOLD

// Converti tout les pixels soit en noir soit en blanc. Le paramètre par défaut est 0.5 c'est à dire que tout les pixels ayant une valeur de gris au dessus de  $255 \times 0.5 = 127.5$  passer en blanc et en dessous de 127.5 passer en noir.\*  
// L'ajout d'un paramètre permet d'ajuster cette valeur.

#### Exemple :

```
bob.filter(THRESHOLD);  
bob.filter(THRESHOLD, 0.25);  
bob.filter(THRESHOLD, 0.75);
```

#### GRAY

//Converti tous les pixels d'une image en valeur de gris.  
(Sans paramètre)

#### OPAQUE

// Atribue la chaine alpha complètement opaque. (Sans paramètre)

#### INVERT

// Inverse tous les pixels. (Sans paramètre)

#### POSTERIZE

/\* Limite chaque pixel à une tranche de pixels donnée, le paramètre peut être de 2 à 255.  
Il determine le nombre de la tranche. \*/

#### Exemple :

```
bob.filter(POSTERIZE, 4);
```

#### BLUR

// Exécute un filtre de flou (BLUR) sur l'image, sans paramètre la valeur par défaut est de 1. Mais elle peut être augmentée où diminuée avec un paramètre.\*

#### EXEMPLE :

```
bob.filter(BLUR);  
bob.filter(BLUR, 4);
```

#### ERODE

// Réduit les zones de lumière d'une image. (Sans paramètres)

#### DILATE

// Augmente les zones de lumières d'une image. (Sans paramètres)

## LES SHAPES (OBJETS VECTORIELS) : PShape

//Emplacement mémoire qui contient mon image pour l'instant vide.

PShape bob;

```
void setup() {  
    //Chargement dans mon image vectorielle dans l'emplacement mémoire  
    bob = loadShape("bob_l_ponge.svg");  
}  
void draw() {  
    //affichage de mon image vectorielle ( type *.svg ou *.obj)  
    shape(bob,40,40);  
}
```

## QUELQUES MÉTHODES ITINÉRANTES A PShape

### MÉTHODES LIÉES AUX STYLES DE FORME

**.isVisible()**

//Retourne un boolean pour savoir si la forme est visible ou pas.

**.setVisible()**

//Rend la forme visible ou non.

**.disableStyle()**

//Désactive les styles de la forme pour utiliser ceux de processing.

**.enableStyle()**

//Active les styles de la forme et ignore ceux de processing.

**.translate()**

//Déplace une forme en x,y

**.rotateX()**

//Tourne une forme autour de l'axe X

**.rotateY()**

//Tourne une forme autour de l'axe Y

**.rotateZ()**

//Tourne une forme autour de l'axe Z

**.rotate()**

//Tourne une forme

**.scale()**

//Augmente ou diminue un forme.

**.resetMatrix()**

//Réinitialise les matrices de transformation

### SYSTÈME DE DESSIN

**.beginContour()**

//Commence un nouveau contour.

**.endContour()**

//Termine un contour

**.endShape()**

//Finit la création d'une nouvelle PShape

**.getChildCount()**

//Retourne le nombre d'enfants

**.getChild()**

//Retourne une forme enfant d'un élément PShape.

**.addChild()**

//Ajoute un enfant à un PShape

**.getVertexCount()**

//Retourne le nombre total de vertex. (Points constitutifs d'une forme)

**.getVertex()**

//retourne une vertex suivant sa position d'index.

**.setVertex()**

//installe une vertex dans la position d'index.



//Emplacement mémoire qui contient mon image pour l'instant vide.

PFont bob;

```
void setup() {
    //Chargement dans mon image dans l'emplacement mémoire
    bob = loadFont("arial-48.vlw");
}
void draw() {
    //Sélection de ma typographie
    textFont(bob);
    //Affichage d'un texte
    text("bob",40,20);
    //où 40 désigne la position en x et 20 la position en Y
}
```

textWidth(texte);

Typographie

textAscent();

textDescent();

Origine en X et Y;

## COMMANDES ITINÉRANTES A L'UTILISATION DE PFont :

textSize(a); //où «a» est un float qui détermine la taille de la typographie

textLeading(a); //où «a» est un float qui l'interlignage de la typographie

textAlign(a); //où «a» est une variable qui détermine l'alignement.

// «a» peut valoir CENTER, LEFT ou RIGHT .

Exemple :

textAlign(CENTER); //Alignement au centre

Autre exemple :

textAlign(CENTER,TOP); //Alignement au centre et en haut

## COMMANDES DE CRÉATION DE FONT À PARTIR D'UN .TTF :

//Emplacement mémoire qui contient mon image pour l'instant vide.

PFont bob;

```
void setup() {
    //Chargement dans mon image dans l'emplacement mémoire
    bob = createFont("arial.ttf",48);
}
void draw() {
    //Sélection de ma typographie
    textFont(bob);
    //Affichage d'un texte
    text("bob",40,20);
    //où 40 désigne la position en x et 20 la position en Y
}
```

## CONDITIONS

### CONDITIONS SIMPLE :

```
int bob=1;

if(bob==1){
    //Mes actions si bob est égale à 1
}

if(bob!=1){
    //Mes actions si bob est différente de 1
}

if(bob==1){
    //Mes actions si bob est égale à 1
}else{
    //Mes actions si bob est différente de 1
}

if(bob>1){
    //Mes actions si bob est strictement supérieure à 1
}

if(bob<1){
    //Mes actions si bob est strictement inférieure à 1
}

if(bob<=1){
    //Mes actions si bob est inférieure ou égale à 1
}
```

### CONDITIONS MULTIPLES :

```
if(bob>1 && bob<10){
    //Mes actions si bob est supérieure à 1
    ET inférieure à 10
}

if(bob>1 || bob<10){
    //Mes actions si bob est supérieure à 1
    OU inférieure à 10
}

if(bob>1 && bob<10 && pat>1 && pat>10){
    /* Mes actions si bob est supérieure à 1
    ET inférieure à 10
    ET pat supérieure à 1
    ET pat inférieure à 10. */
}
```

### CONDITION IMBRIQUÉES :

```
if( bob==1 ){
    //Mes actions si bob égale à 1
} else if ( bob== 2 ){
    //Mes actions si bob égale à 5
} else if ( bob == 5 ){
    //Mes actions si bob égale à 5
}
```

## BOUCLE FOR

### BOUCLE «FOR» A 1 DIMENSION :

```
for(int i=0;i<100;i++){
    //Mes actions pour i prenant une valeur de 0 à 99
}

for(int i=0;i<=100;i++){
    //Mes actions pour i prenant une valeur de 0 à 100
}
```

### BOUCLE «FOR» A 2 DIMENSIONS :

```
for(int i=0;i<100;i++){
    for(int j=0;j<100;j++){
        /* Mes actions pour i prenant une valeur de 0
        à 99 et j prenant une valeur de 0 à 99. */
    }
}
```

## OPÉRATEURS

```
float bob=1;
bob = 8 + 9;    //où bob est égale à 8+9 soit 17
bob = 8 - 9;    //où bob est égale à 8-9 soit -1
bob = 8 * 9;    //où bob est égale à 8x9 soit 72
bob = 8 / 9;    //où bob est égale à 8/9 soit 0.88..
bob = bob+1;    //où bob prend +1 toutes les
                images
```

### RACCOURCIS

```
bob = bob + 1;
//peut s'écrire :
bob += 1;
//ou encore peut s'écrire :
bob++;
```

```
bob = bob * 8;
//peut s'écrire :
bob *=8;
```

### OPÉRATIONS MATHÉMATIQUES

```
float bob = 3;
float pat = -4.456;
float gary = 5.11;
```

### VALEURS ABSOLUE : abs()

```
float r = abs(pat);
//Retourne la valeur absolue de -4.456 soit 4.456.
```

### PUISSANCE : sq()

```
float r = sq(bob);
//Retourne la puissance de 2 de bob 32 soit 9.
```

### PUISSANCES DE N : pow()

```
float r = pow(bob,4);
// Retourne la puissance de 3 de bob 3x3x3x3 soit 81.
```

## RACINE CARRÉE : `sqrt()`

```
float r = sqrt(bob);  
// Retourne la racine carrée de bob : √3 soit 1.732...
```

## TRONCAGE HAUT : `ceil()`

```
float r = ceil(gary);  
// Retourne la valeur arrondie haute de 5.11 soit 6.  
float r = ceil(pat);  
// Retourne la valeur arrondie haute de -4.456 soit -4.
```

## TRONCAGE BAS : `floor()`

```
float r = floor(gary);  
// Retourne la valeur arrondie basse de 5.11 soit 5.  
float r = floor(pat);  
// Retourne la valeur arrondie basse de -4.456 soit -5.
```

## ARRONDIS : `round()`

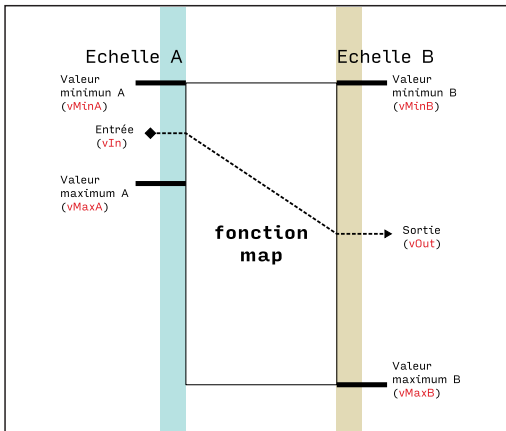
```
float r = round(gary);  
// Retourne l'arrondi basse de 5.11 soit 5.  
float r = round(pat);  
// Retourne l'arrondi basse de -4.456 soit -4.
```

## ALÉATOIRE : `random()`

```
float r = random(59);  
//tir un nombre aléatoire (un float) de 0 à 59  
float r = random(22,59);  
//tir un nombre aléatoire (un float) de 22 à 59
```

## CHANGEMENT D'ÉCHELLE: `map()`

```
//Passe d'une plage de valeur à une autre.  
float vOut  
=map(vIn,vMinA,vMaxA,vMinB,vMaxB);
```



```
float res = map(0.5,0,1,0,100);  
println(res);  
// affiche res = 50;
```

```
float res = map(0.5,0,1,0,100);  
println(res);  
// affiche res = 50;
```

## LES TABLEAUX

### REMPLISSAGE VIA LE SETUP

```
//Déclare un tableau de int à 6 cases vides de type int  
int [] bob = new int[6];  
  
//Remplissage du tableau via le setup;  
void setup(){  
    bob[0]=9;        //la case 0 prend 9  
    bob[1]=81;       //la case 1 prend 81  
    bob[2]=111;      //la case 2 prend 111  
}  
// Et donc les cases 4 et 5 sont égales à 0.
```

### REMPLISSAGE VIA LA DÉCLARATION

```
float [] bob = {65,5.7,67};  
// Ici la case 0 prend 65  
// Ici la case 1 prend 5.7  
// Ici la case 2 prend 67  
// et donc le tableau a une longueur de 3.
```

### AUTRES EXEMPLES, AVEC D'AUTRES TYPES

```
String [] tabPays =  
    {"France","Italie","Espagne","Allemagne"}  
/* Ici la case 0 prend «France»  
   ici la case 1 prend «Italie»  
   ici la case 2 prend «Espagne»  
   ici la case 3 prend «Allemagne»  
   et donc le tableau a une longueur de 4.  
*/
```

```
boolean [] tabBool = {true, false,  
    true,true,false}  
// Initialisation d'un tableau de boolean contenant 5  
valeurs booléennes.
```

```
color [] tabCol=  
    {color(255),color(251,15,21),  
    color(255,255,0) };  
// Initialisation d'un tableau de couleur contenant 3  
couleurs.
```

```
PImage [] tabImage = new PImage[4];  
// Initialisation d'un tableau pouvant contenir 4 images.  
void setup(){  
    tabImage[0]=loadImage("bob.png");  
    tabImage[1]=loadImage("gary.png");  
    tabImage[2]=loadImage("pat.png");  
}  
// et tabImage[3] est égal à null puisque c'est des objets.
```

## LES TABLEAUX (OPÉRATIONS)

### append()

```
//Ajoute un éléments à la fin d'un tableau
String[] pays1={ "France", "Suisse",
    "Italie"};
String[] pays2=append(pays1, "Allemagne");
println(pays2);
// Sort l'actualisation du tableau dans la console
// [0] "France"
// [1] "Suisse"
// [2] "Italie"
// [3] "Allemagne"
```

### arrayCopy()

```
//Copie l'intégralité d'un tableau dans un autre
String[] europe={"France", "Suisse",
    "Italie"};
String[] asie={"Chine", "Japon", "Laos"};
arrayCopy(europe, asie);
println(asie);
//Contenu du tableau asie
// [0] "France"
// [1] "Suisse"
// [2] "Italie"
```

### concat()

```
//Unifie deux tableaux du même type dans un troisième
String[] europe={"France", "Suisse",
    "Italie"};
String[] asie = {"Chine", "Japon", "Laos"};
String[] monde = concat(europe, asie);
println(monde);
//Contenu du tableau monde
// [0] "France"
// [1] "Suisse"
// [2] "Italie"
// [3] "Chine"
// [4] "Japon"
// [5] "Laos"
```

### expand()

```
//Par default double la longueur d'un tableau
int[] data = {0, 1, 3, 4};
println(data.length); // Prints "4"
data = expand(data);
println(data.length); // Prints "8"
data = expand(data, 512);
println(data.length); // Prints "512"
```

### reverse()

```
//Inverses toutes les valeurs d'un tableau
String sa[] = { "OH", "NY", "MA", "CA"};
sa = reverse(sa);
println(sa);
// Sort l'actualisation du tableau dans la console
// [0] "CA"
// [1] "MA"
// [2] "NY"
// [3] "OH"
```

### shorten()

```
//Supprime le dernier élément d'un tableau
String[] sa1={ "OH", "NY", "CA"};
String[] sa2=shorten(sa1);
println(sa1); // 'sa1' contient toujours OH, NY, CA
println(sa2); // 'sa2' maintenant contient OH, NY
```

### sort()

```
//Pour une String : Trie les éléments par ordre
alphabétique.
//Pour un Int, Float, Char, .. : Trie les éléments du plus
grand au plus petit.
String[] monde={"France", "Italie",
    "Allemagne", "Kenya", "Belgique" };
monde = sort(monde);
println(monde);
//Contenu du tableau monde
// [0] "Allemagne"
// [1] "Belgique"
// [2] "France"
// [3] "Italie"
// [4] "Kenya"
```

### splice()

```
String[] a={ "OH", "NY", "CA" };
a = splice(a, "KY", 1); // Ajoute une valeur
dans un tableau a la position désirée.
//Splice ( tableau, élément, position dans le tableau );
println(a);
// Trace le nouveau tableau
// [0] "OH"
// [1] "KY"
// [2] "NY"
// [3] "CA"
```

### subset()

```
String[] sa1={ "OH", "NY", "CA", "VA", "CO",
    "IL" };
String[] sa2=subset(sa1, 1);
println(sa2);
// Sort l'actualisation du tableau dans la console
// [0] "NY"
// [1] "CA"
// [2] "VA"
// [3] "CO"
// [4] "IL"
String[] sa3=subset(sa1, 2, 3);
println(sa3);
// Sort l'actualisation du tableau dans la console
// [0] "CA"
// [1] "VA"
// [2] "CO"
```

## AUTRES TABLEAUX (ArrayList)

```
/*
  Dans cet exemple on suppose une classe Ball
  contenant des méthodes .move(), .display(), .finished()
*/
```

```
ArrayList balls;
```

```
void setup() {
  size(200, 200);
  balls = new ArrayList();// Crée un ArrayList
  vide
  balls.add(new Ball(width/2, 0, 48));
  // Commence à ajouter un éléments.
}

void draw() {
  background(255);
  // On fait une boucle pour parcourir tout notre tableau.
  for(int i=0;i<balls.size()-1;i++){
    // Un ArrayList ne sait pas quel type d'objet il contient
    // Donc nous devons typé l'objet de sortie.
    Ball ball = (Ball) balls.get(i);
    ball.move(); // Ceci est une fonction de Ball
    ball.display();
    if (ball.finished()) {
      // les objets peuvent être supprimée avec .remove();
      balls.remove(i);
    }
  }
}
```

```
// Quand on clic on ajoute un objet à notre tableau
```

```
void mousePressed() {
  // Un nouvel objet est ajouté dans la list, par default, à
  // la dernière position
  balls.add(new Ball(mouseX, mouseY, 30));
}
```

```
/* ArrayList contient donc les méthodes
```

```
.add();      Ajoute un objet
.get();      Prendre un objet
.remove();   Supprimer un objet.
```

```
*/
```

## NOTE SUR LES TABLEAUX

- N'importe quel objet peut être stocké dans un ArrayList.
- Il existe dans Processing d'autres types de Tableaux comme les StringList, IntList, StringDir, IntDir ou encore les Table.
- Pour plus d'informations, consultez la référence.

## VARIABLES DE TEMPS

### TEMPS PAR RAPPORT A L'ORDINATEUR

```
day(); //Retourne le jour en Int
bob = day();
```

```
hour(); //Retourne l'heure en Int
bob = hour();
```

```
minute(); //Retourne la minute en Int
bob = minute();
```

```
month(); //Retourne le mois en Int
bob = month();
```

```
second(); //Retourne la seconde en Int
bob = second();
```

```
year(); //Retourne l'année en Int
bob = year();
```

### EXEMPLE :

```
//Retourne la date du jour.
```

```
String today =
  "Le" + day() + "/" + month() + "/" + year()
  + " à " + hour() + ":"
  + minute() + ":" + second();
```

```
println(today);
```

### TEMPS PAR RAPPORT AU PROGRAMME

```
millis();
```

```
//Retourne le nombre de millisecondes exécutées depuis
le debut du programme*/.
```

## PVECTOR (LES VECTEURS)

```
PVector v1, v2;
```

```
void setup() {  
    v1 = new PVector(40, 20);  
    v2 = new PVector(25, 50);  
}  
  
void draw() {  
    ellipse(v1.x, v1.y, 12, 12);  
    ellipse(v2.x, v2.y, 12, 12);  
    v2.add(v1);  
    ellipse(v2.x, v2.y, 24, 24);  
}
```

### Variables Accessibles

```
.x //Variable x du vecteur  
.y //Variable y du vecteur  
.z //Variable z du vecteur
```

### Exemple :

```
rect(v1.x,v1.y,10,10);  
//Affiche un carré en position x et y de mon vecteur.
```

### METHODES ACCESSIBLES

```
.set()
```

//Attribut les composants x,y ou x,y et z du Vecteur.

### Exemple :

```
v1.set(9,3); //Attribut la valeur 9 en x et 3 en y
```

```
.get();
```

//Récupère un copy du Vecteur

### Exemple :

```
v2 = v1.get();  
//Le Vecteur v2 à maintenant toutes les valeurs de v1
```

```
.mag();
```

//Récupère la magnitude d'un vecteur (sa longueur)

```
float r = v1.mag();  
println(r);  
// r équivaut à 44.72136
```

```
.add();
```

//Ajoute un vecteur avec un autre vecteur.

```
PVector v1, v2;
```

```
void setup() {  
    v1 = new PVector(40, 20);  
    v2 = new PVector(25, 50);  
    v2.add(v1);  
}
```

## QUELQUES AUTRES MÉTHODES ACCESSIBLES...

// Pour plus d'informations, consultez la référence.

```
.random2d()
```

//Donne deux valeurs : x et y aléatoire au vecteur compris entre -1 et 1.

```
.random3d()
```

//Donne trois valeurs : x, y et z aléatoire au vecteur compris entre -1 et 1.

```
.sub()
```

//Soustrait les valeurs x, y et z du vecteur par les valeurs d'un autre, ou de deux vecteurs indépendants.

```
.mult()
```

//Multiplie un vecteur par un scalaire, ou par un autre vecteur.

```
.div()
```

//Divise un vecteur par un scalaire, ou par un autre vecteur.

```
.dist()
```

//Calcul la distance entre deux points.

```
.normalize()
```

//Normalise la longueur d'un vecteur à 1.

```
.limit()
```

//Limite la magnitude/longueur d'un vecteur

```
.setMag()
```

//Définit la magnitude/longueur d'un vecteur.

```
.angleBetween()
```

//Calcule et retourne un angle entre deux vecteurs.

```
.array()
```

//Retourne une représentation d'un vecteur par un tableau de float

## CHARGEMENT DE DATAS CSV (SIMPLE)

Table mData;

```
void setup(){
  size(800,600);
  mData = loadTable("data.csv","header");
}

void draw(){
  background(200);
  for(TableRow row : mData.rows()){
    fill(row.getInt("r"),row.getInt("v"),row.getInt("b"));
    noStroke();

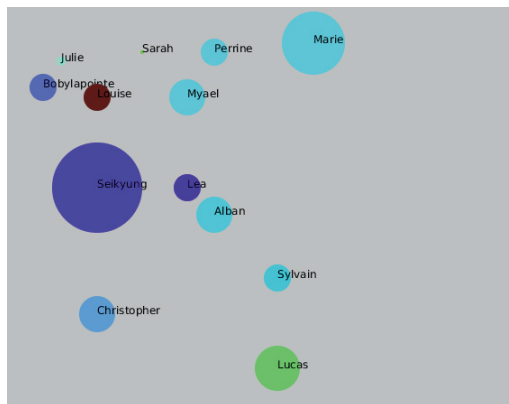
    ellipse(row.getFloat("x"),row.getFloat("y"),
            row.getFloat("diametre"),row.getFloat("diametre"));

    fill(0);
    text(row.getString("Nom"),row.getFloat("x"),row.getFloat("y"));
  }
}
```

### DÉTAIL DU FICHIER CSV

/\*  
La première ligne correspond à l'en-tête de colonne  
\*/

```
Nom,x,y,diametre,r,v,b
Boby Lapointe,40,89,30,100,100,220
Julie,60,60,10,100,255,220
Sarah,150,50,4,100,200,30
Marie,340,40,70,100,200,220
Louise,100,100,30,100,29,20
Myael,200,100,40,100,200,220
Seikyung,100,200,100,100,40,220
Perrine,230,50,30,100,200,220
Lucas,300,400,50,100,200,90
Christopher,100,340,40,100,160,220
Alban,230,230,40,80,200,220
Sylvain,300,300,30,50,200,220
Lea,200,200,30,100,10,220
```



## CONVERSIONS (TRANSTYPAGE)

### binary();

```
//Retourne une représentation binaire d'un char, byte, ou
d'un int sous forme de String
color c = color(255, 204, 0);
println(c); //Prints "-13312"
println(binary(c));
// Prints "111111111111111100110000000000"
println(binary(c, 16));
// Prints «1001100000000000»
```

### boolean();

```
//Converti un int, String ou un array en boolean, pour un
int 0 revient a false et 1 à true.
String s = "true";
println(boolean(s));
//Prints true
```

### BYTE();

```
//Converti un char en byte, de -128 à 127
char c = 'E';
byte b = byte(c);
println(c + " : " + b); // Prints «E : 69»

int i = 130;
byte b = byte(i);
println(i + " : " + b); // Prints «130 : -126»
```

### char();

```
int i = 65;
char c = char(i);
println(i + " : " + c); // Prints "65 : A"

byte b = 65;
c = char(b);
println(b + " : " + c); // Prints "65 : A"
```

### float();

```
// Converti un int, string, ou un tableau en float.
String bob = "659";
float pat = float(bob);
println(pat); // Trace 659.0 dans la console
```

### hex();

```
//Converti un byte, char, int, ou un objet couleur dans la
représentation hexadecimal String (donc la chaine de
caractère correspondante)
color c = color(255, 204, 0);
println(c); // Prints «-13312»
println(hex(c)); // Prints «FFFFCC00»
```

### int();

```
//Converti un datatype (float, boolean, char, byte, color),
string ou tableau dans sa représentation entière.
float f = 65.0;
int i = int(f);
println(f + " : " + i); // Prints «65.0 : 65»

char c = 'E';
int i = int(c);
println(c + " : " + i); // Prints «E : 69»
```

### str(); EN STRING

```
/*Retourne la représentation d'un primitif en chaîne de
caractères */
boolean b = false;
byte y = -28;
char c = 'R';
float f = -32.6;
int i = 1024;

String sb = str(b);
String sy = str(y);
String sc = str(c);
String sf = str(f);
String si = str(i);
sb = sb + sy + sc + sf + si;

println(sb); // Prints 'false-28R-32.61024'
```

### unbinary();

```
//Converti un une représentation binaire String en un
binaire entier.
String s1 = "00010000";
String s2 = "00001000";
String s3 = "00000100";
println(unbinary(s1)); // Prints «16»
println(unbinary(s2)); // Prints «8»
println(unbinary(s3)); // Prints «4»
```

### unhex();

```
//Converti une chaine de caractère en objet hexadécimal.
String hs = "FF006699";
int hi = unhex(hs);
fill(hi);
rect(30, 20, 55, 55);
```



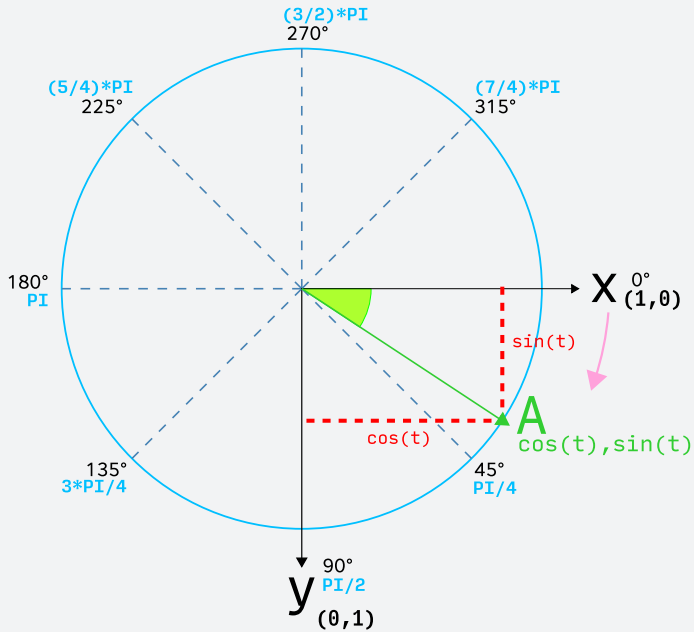
```
// Importation de la librairie vidéo
import processing.video.*;

// Initialisation de l'objet vidéo
Capture cam;

void setup() {
    size(200, 200);
    // Construction de l'objet vidéo
    cam = new Capture(this);
    // Lancement de l'objet vidéo
    cam.start();
}

void draw() {
    // Affichage de l'objet vidéo
    image(cam, 0, 0);
}

// Événement de réception d'image de l'objet vidéo
void captureEvent(Capture c) {
    c.read();
}
```



## FONCTIONS SINUS ET COSINUS

```
float bob = cos(t);
//la valeur périodique cosinus d'un angle ou d'une
//valeur
float bob = sin(t);
//la valeur périodique sinus d'un angle ou d'une
//valeur
```

## Astuce :

```
//Utilisation régulière de fonction sinus et cosinus
// par exemple pour tracer un cercle de centre px,py et
// de rayon r ;
float t=0;
float px,py=400;
float r = 100;
```

```
void setup() {
  size(800,800);
}
void draw() {
  point(px+cos(t)*r , py+sin(t)*r );
  t+=0.01;
}
```

## PI

```
//Renvoie la valeur de PI
//C'est à dire :
//3.1415927
```

## radians();

```
//Transforme des degrés en radians
float r = PI/2;
println(degrees(r));
//renvoie 90
//donc 90 degrés.
```

## degrees();

```
//Transforme des radians en degrés
float r = 45;
println(radians(r));
//renvoie 0.7853982
//et qui équivaut a PI / 4;
//donc 45 degrés.
```

## dist();

```
//Calcule de la distance entre 2 points
//où x1 est un float
//où x2 est un float
//où y1 est un float
//où y2 est un float
float r=dist(x1,y1,x2,y2);
println(r);
```

## LES STRINGS OU "CHAÎNES DE CARACTÈRE"

```
String bob = "Super bob, super, bob !";
```

### `.equals();`

```
//Comparaison de deux chaines de caractères.  
//Ne pas utiliser == pour comparer deux chaines.  
//Utiliser .equals()  
if(bob.equals("bob")){  
    //si la variable bob est égale à bob.  
}
```

### `.charAt();`

```
//Récupération d'un caractère à un index donné.  
char r = bob.charAt(3);  
println(r);  
// r va retourner "e", car dans  
// "Super bob,super,bob !" le 3ème caractère  
// est le "e".
```

### `.indexOf();`

```
//Recherche d'un caractère ou d'une chaîne donnée et  
// retour de son index  
int r = bob.indexOf("bob");  
println(r);  
// r retourne 6  
//en effet "bob" se trouve au niveau du 6ème caractère
```

### `.length();`

```
// Récupération du nombre de caractères dans la chaîne  
int r = bob.length();  
println(r);  
// r retourne 23
```

### `.substring();`

```
// Avec 1 paramètre ou 11 est un int  
// récupère depuis l'index donné le reste  
// de la chaîne de caractères.  
String r = bob.substring(11);  
println(r);  
// Affiche dans la console :  
// super, bob !  
  
// Avec 2 paramètres où 6 et 3 sont des int  
// récupère depuis l'index et le nombre de caractères  
// à récupérer  
String r = bob.substring(6,3);  
println(r);  
// Affiche dans la console :  
// bob
```

### `.toLowerCase();`

```
//transforme tout les caractères en caractères bas de  
// casse  
String r = bob.toLowerCase();  
println(r);  
// affiche dans la console :  
// super bob, super, bob !
```

### `.toUpperCase();`

```
//transforme tous les caractères en caractères hauts de  
// casse  
String r = bob.toUpperCase();  
println(r);  
// affiche dans la console :  
// SUPER BOB, SUPER, BOB !
```

### `.join();`

```
//Permet de grouper un tableau de String  
//1er paramètre : tableau à joindre  
//2ème paramètre : Chaîne de jointure  
String[] animaux = new String[3];  
animaux[0] = "crabe";  
animaux[1] = "chat";  
animaux[2] = "singe";  
String groupeAnimaux=join(animaux," ");  
println(groupeAnimaux);  
// Ecrit dans la console "crabe: chat: singe"
```

### `.split();`

```
/* Permet de découper une chaîne en "tableau de  
// chaîne"  
1er paramètre : chaîne à découper  
2ème paramètre : chaîne de découpe */  
String animaux = "Chat,Chien,Singe";  
String[] tabAnimaux = split(animaux,",");  
/* tabAnimaux[0] contient "Chat"  
tabAnimaux[1] contient "Chien"  
tabAnimaux[2] contient "Singe"  
*/
```

### `.splitToken();`

```
/* Permet de nettoyer des données mal formatées  
// Le ", " est le délimiteur qui permet d'enlever les  
// caractères superflus. */  
String bob = "a, b c ,,d ";  
String[] tab = splitTokens(bob, ", ");  
println(tab[0]); // retourne "a"  
println(tab[1]); // retourne "b"  
println(tab[2]); // retourne "c"  
println(tab[3]); // retourne "d"
```

## LES CLASSES

```
//-----  
//SCENE PRINCIPALE  
//-----
```

//Initialisation de 2 objets de type Monstre

Monstre bob;

Monstre pat;

```
void setup() {  
    size(500,500);
```

//Création des objets Monstres

bob = new Monstre();

pat = new Monstre(100,20);

```
}
```

```
void draw() {
```

//Utilisation des méthodes internes à Monstre

//Ici void affich(); et run();

bob.run();

bob.affich();

//Utilisation des méthodes internes à Monstre

//Ici void affich();

pat.affich();

```
}
```

## MA CLASS MONSTRE

```
//-----  
//Onglet pour une classe Monstre (de préférence).  
//-----
```

//Déclaration de ma classe Monstre

class Monstre{ //Début de classe

//Variable interne à une classe

int px=0;

int py=0;

//Constructeur sans paramètres

Monstre() {

}

//Constructeur avec 2 paramètres

Monstre(int px,int py){

//Récupération des paramètres

this.px=px;

this.py=py;

}

//Fonction personnelle

void affich() {

//Affichage par exemple

rect(px,py,10,10);

}

//Autre fonction personnelle sans paramètre

void run() {

px++;

}

} //Fin de classe

## RACCOURCIS CLAVIER :

Ctrl+R > compiler et lancer le code.

Ctrl+T > Reformater le code.

Ctrl+Espace > Ouvrir l'auto-complétation.