CHAPITRE 4

PROGRAMMER SOUS PROCESSING

D’après http://fr.flossmanuals.net/processing/

A voir aussi:

http://www.siteduzero.com/tutoriel-3-268569-processing.html

1. INTRODUCTION

PROCESSING ???

- Conçu par des artistes, pour des artistes
  - **Logiciel libre et gratuit (à noyeau JAVA) !!**
- Environnements de création utilisant le code informatique pour générer des œuvres multimédias sur ordinateur.

ATTRAIT ???

- Simplicité d'utilisation
- Diversité de ses applications
  - Images / sons / animations
  - Applications sur Internet et sur téléphones mobiles,
  - Conception d'objets électroniques interactifs

DESSINER ET CRÉER AVEC DU CODE INFORMATIQUE

- Dessins en 2D ou 3D
- Œuvres sonores et visuelles animées
- Objets communiquant ⟺ (interaction) Environnement

INTÉRÊT DE L'EXPRESSION ARTISTIQUE PAR LE CODE ?

- Rapidité d'exécution
- Automatisation des actions et des taches répétitives
- Interaction etc…

↳ Création d'œuvres originales

2IÈME INTÉRÊT DE PROCESSING

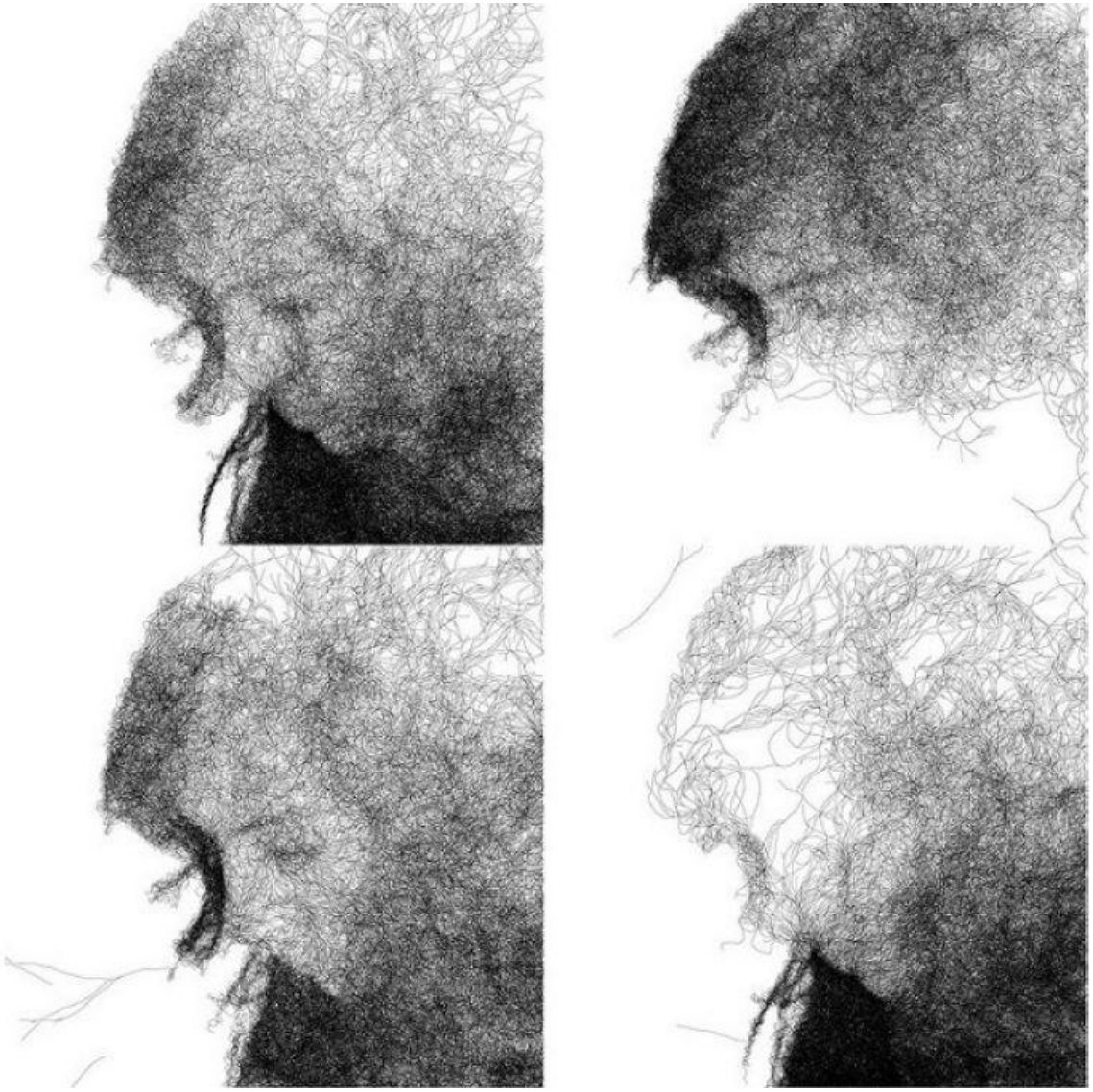
- Programmer des circuits électroniques ⟺ (interaction) Environnement
  - Capteurs sonores
  - Capteurs thermiques
  - Capteurs de mouvement, etc…
- Microcontrôleurs (ARDUINO)
  - Génère des images
  - Actionne des bras articulés
  - Envoie des messages sur internet, etc…

HISTORIQUE DE PROCESSING

- Naissance en 2001 au MIT Media Lab
  - Le papa: Ben Fry
  - La maman: Casey Reas
- Prolongement du projet « Design By Numbers »
  - Artiste programmeur John Maeda
  - Simplicité et économie d'action dans la création d'images

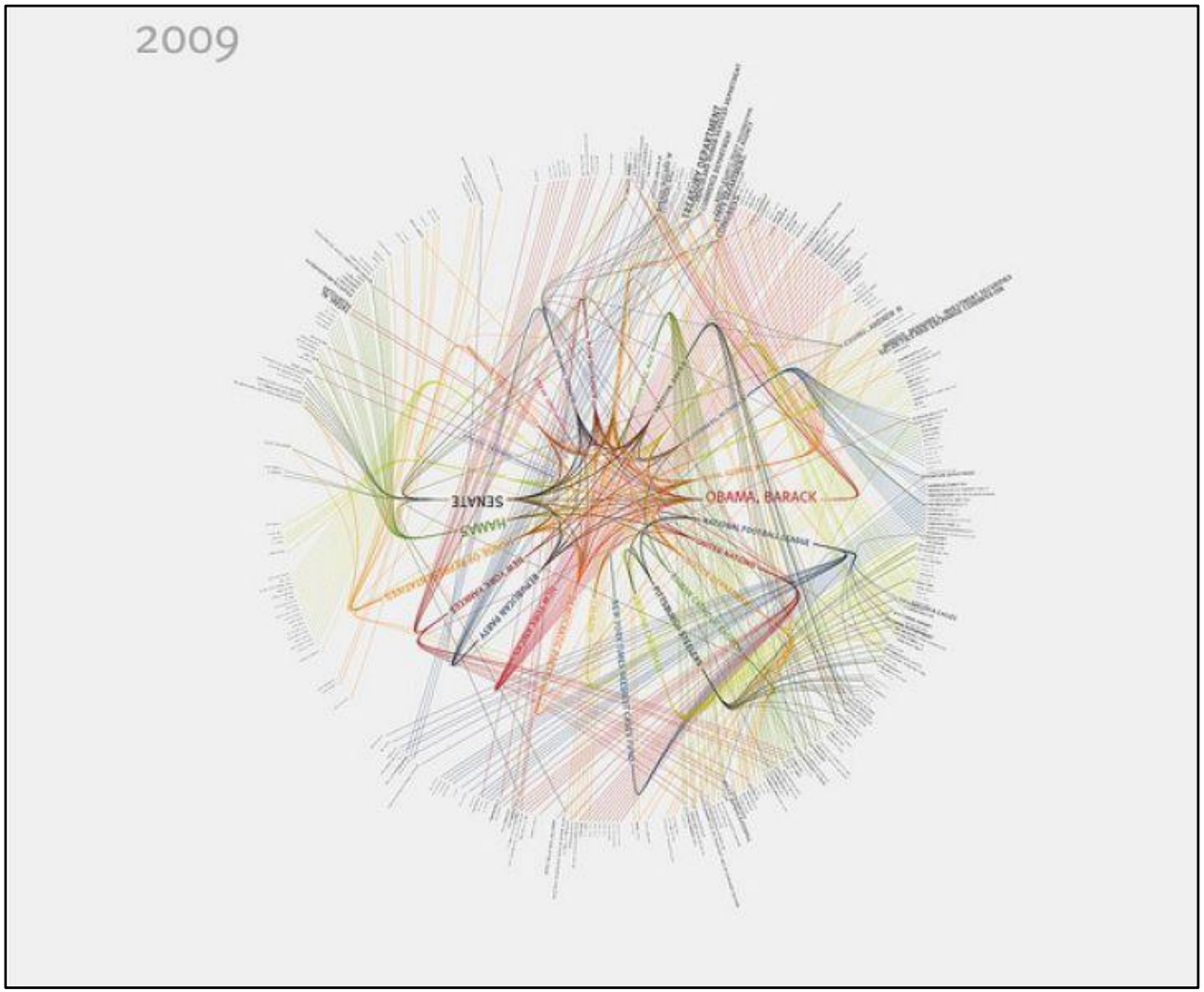
2. EXEMPLES D'UTILISATIONS

Mycelium (Alexander Ryan -2010)



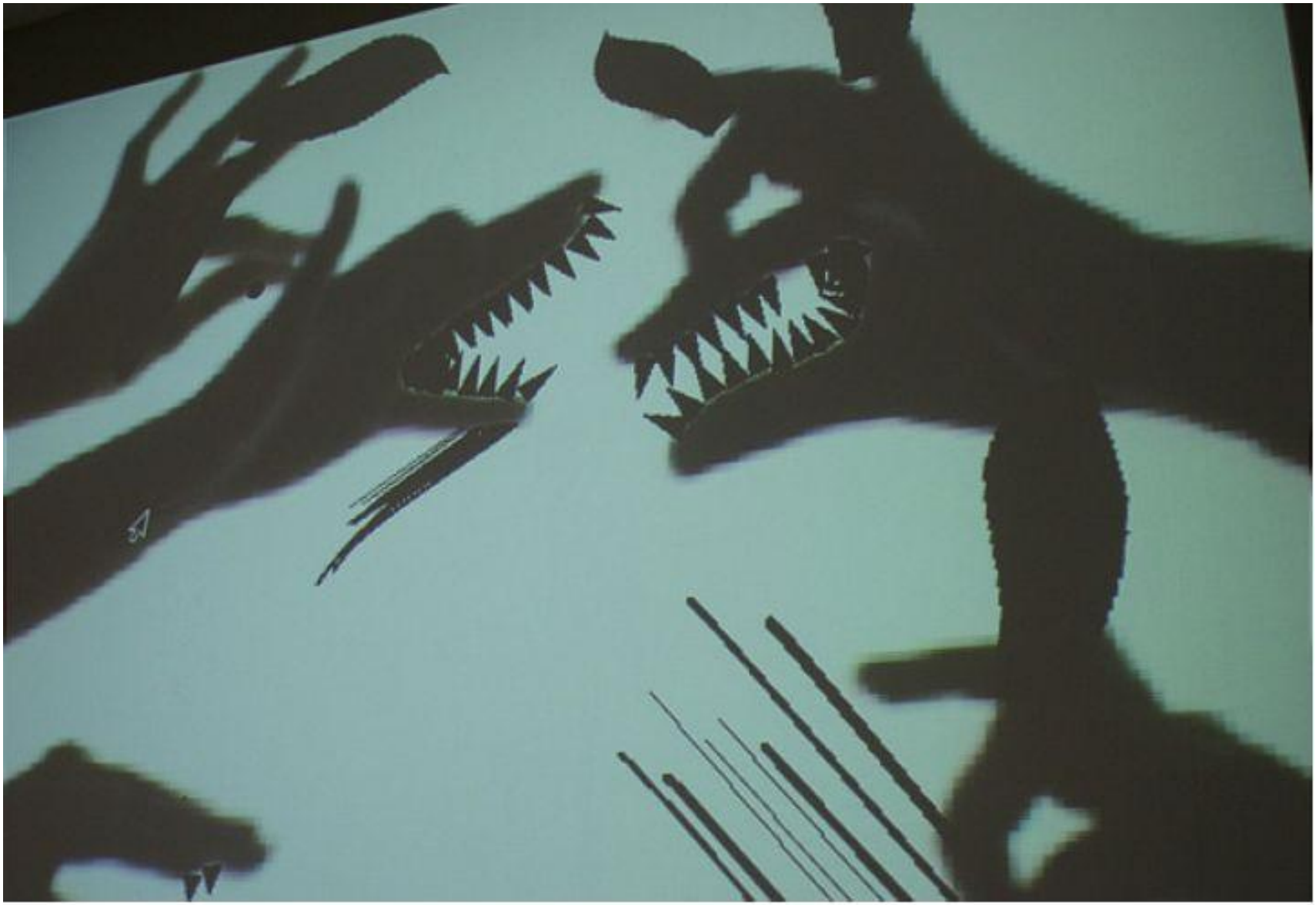
http://onecm.com/projects/mycelium/

NYTIMES (Jer Thrope-2009)



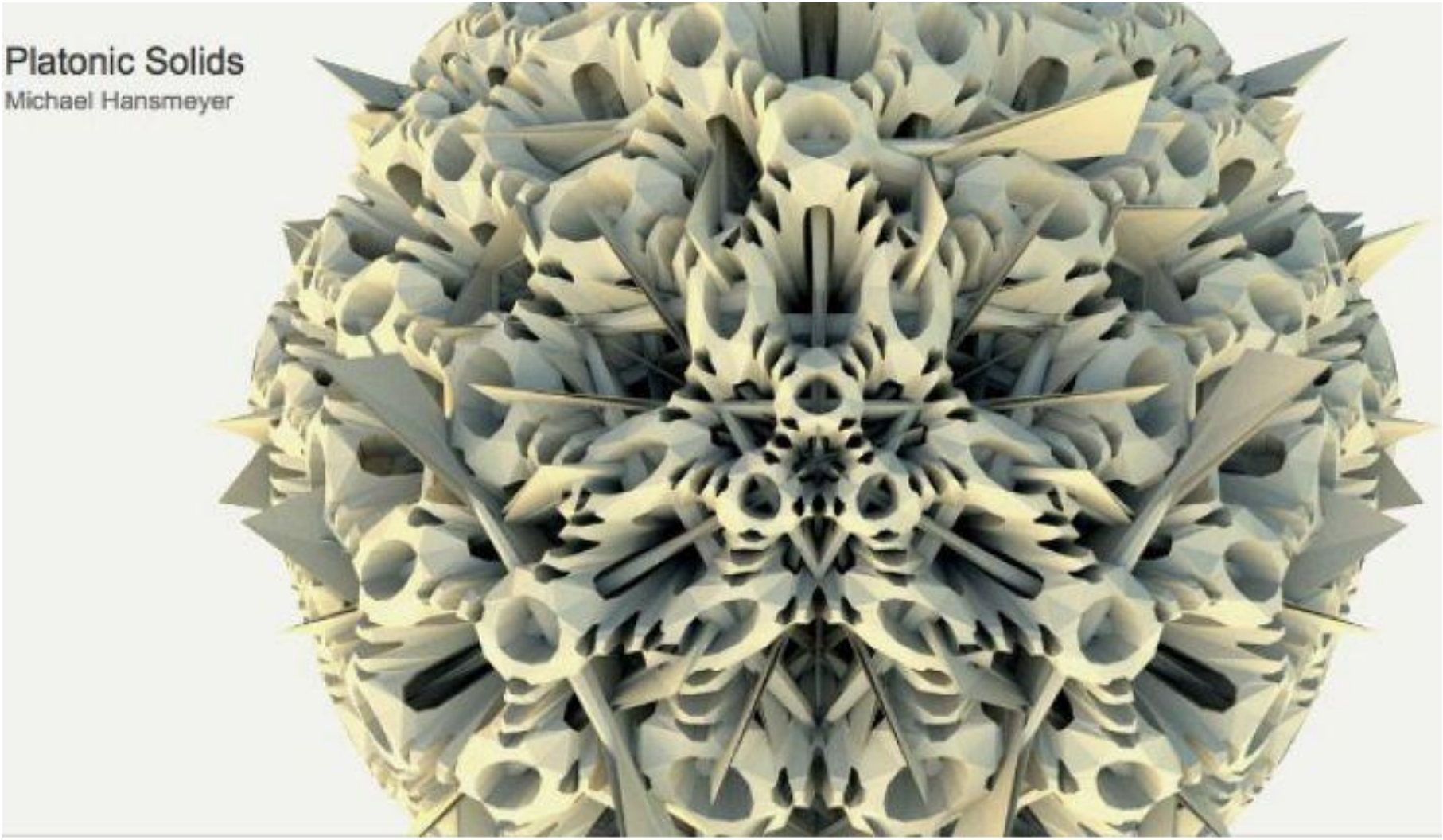
http://blog.blprnt.com/blog/blprnt/7 -days-of-source-day-2-nytimes-36536

Shadow Monsters (Philip Worthington -2005)



http://worthersoriginal.com/viki/# page=shadowmonsters

Platonic solids (Michael Hansmeyer-??)



Dessinateur du XIX^{ieme} siècle Ernst Haekel

http://worthersoriginal.com/viki/# page=shadowmonsters

Champ d'ozone (HeHe - 2007)

http://hehe.org.free.fr/hehe/champsdozone/

COP15 GENERATIVE IDENTITY (Studio okdelux, londre - 2009)

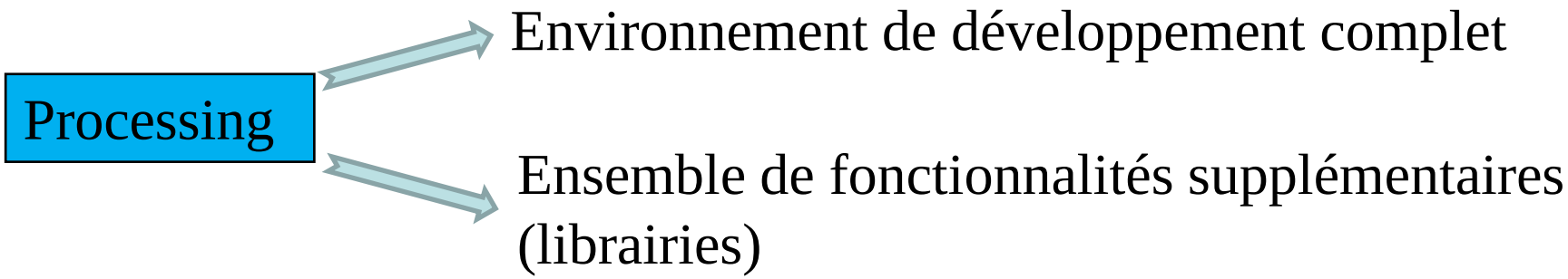
LOGO animé http://www.okdeluxe.co.uk/cop15/

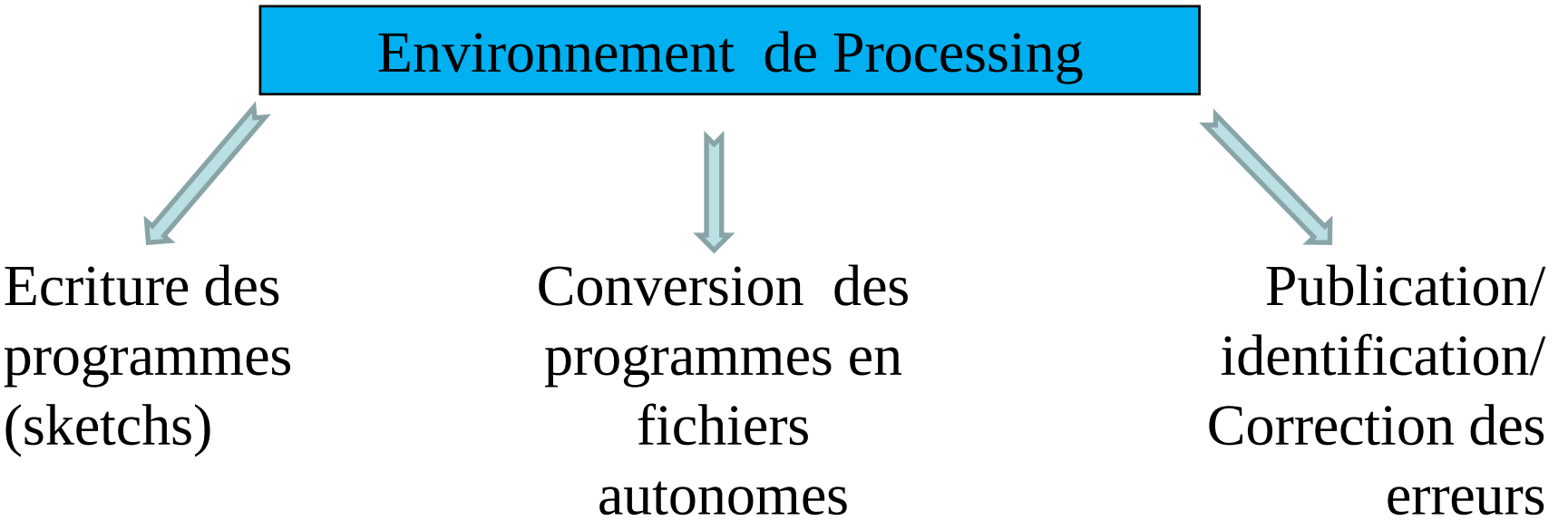
BODY NAVIGATION (Jonas Jongejan & Ole Kristensen - 2008)

http://3xw.ole.kristensen.name/works/body-navigation/

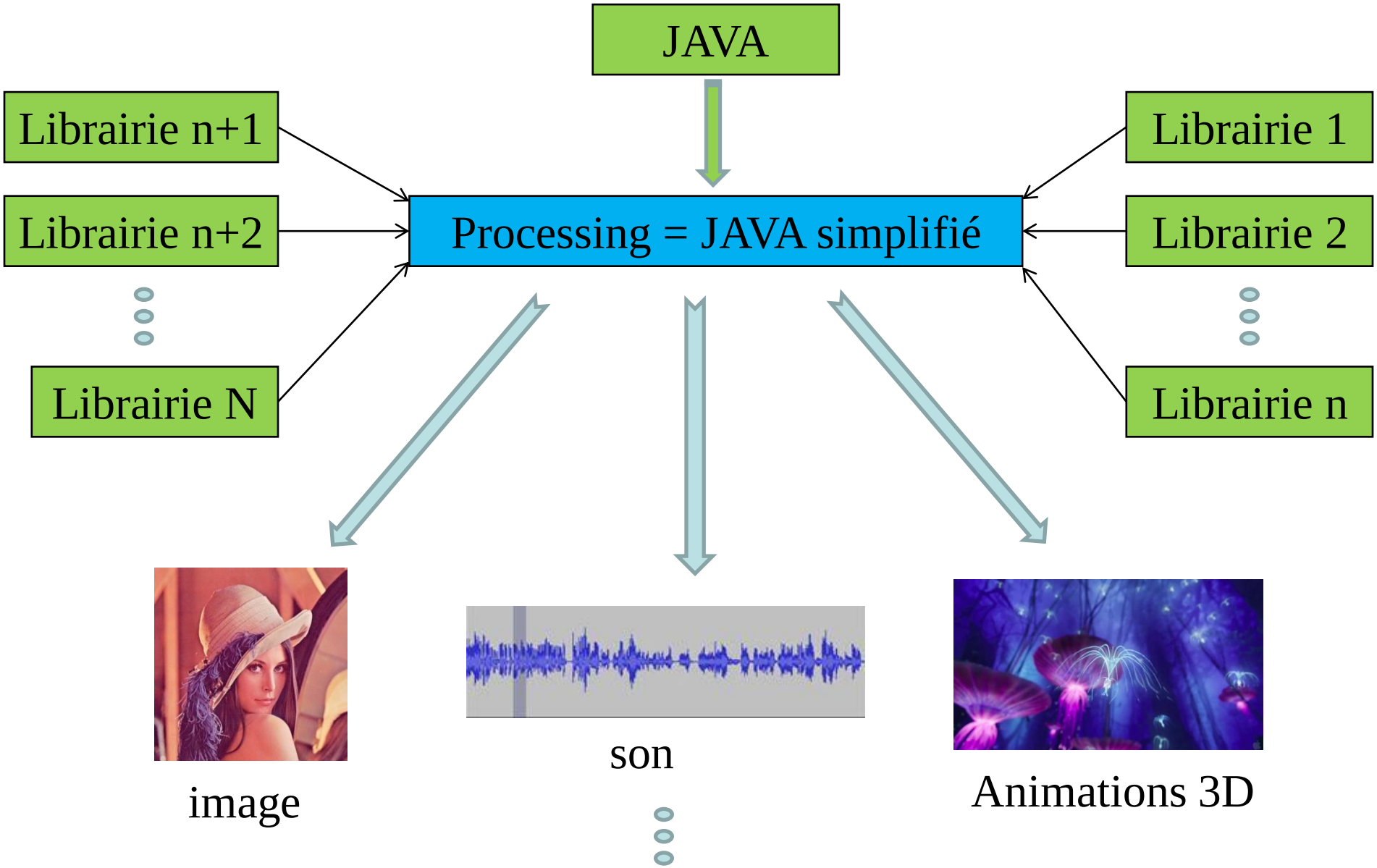
3. INSTALLATION DE PROCESSING

4. LES BASES DE PROCESSING

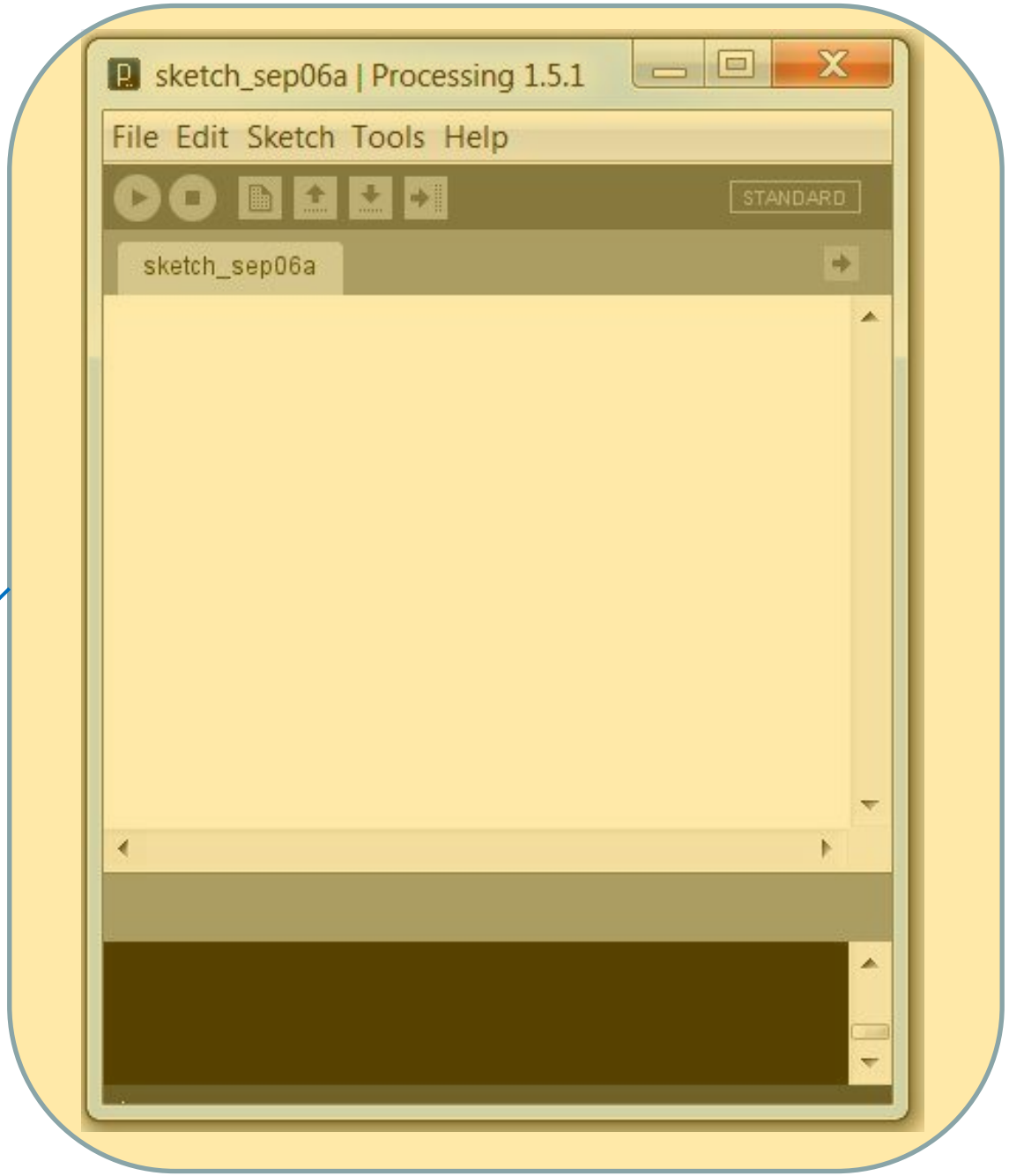




4. LES BASES DE PROCESSING



4.1 L'interface



Interface Processing

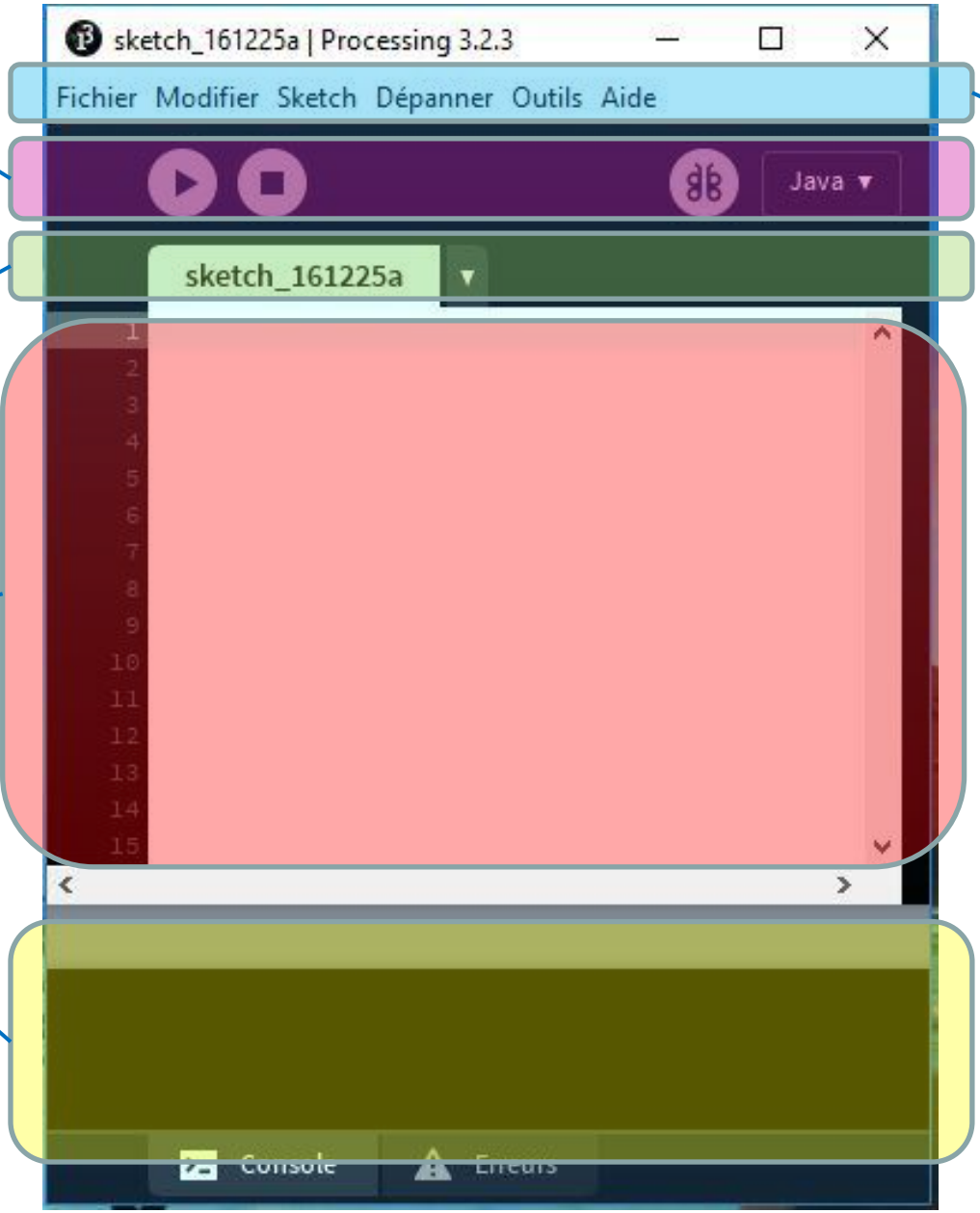
4.1 L'interface

Barre d'action

Barre d'onglet

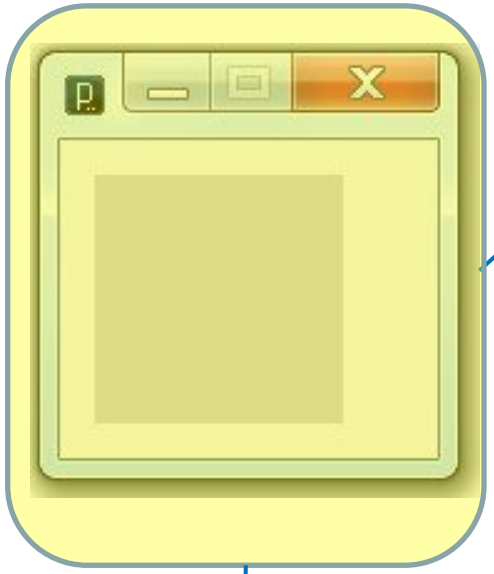
Zone d'édition
(pour taper le programme)

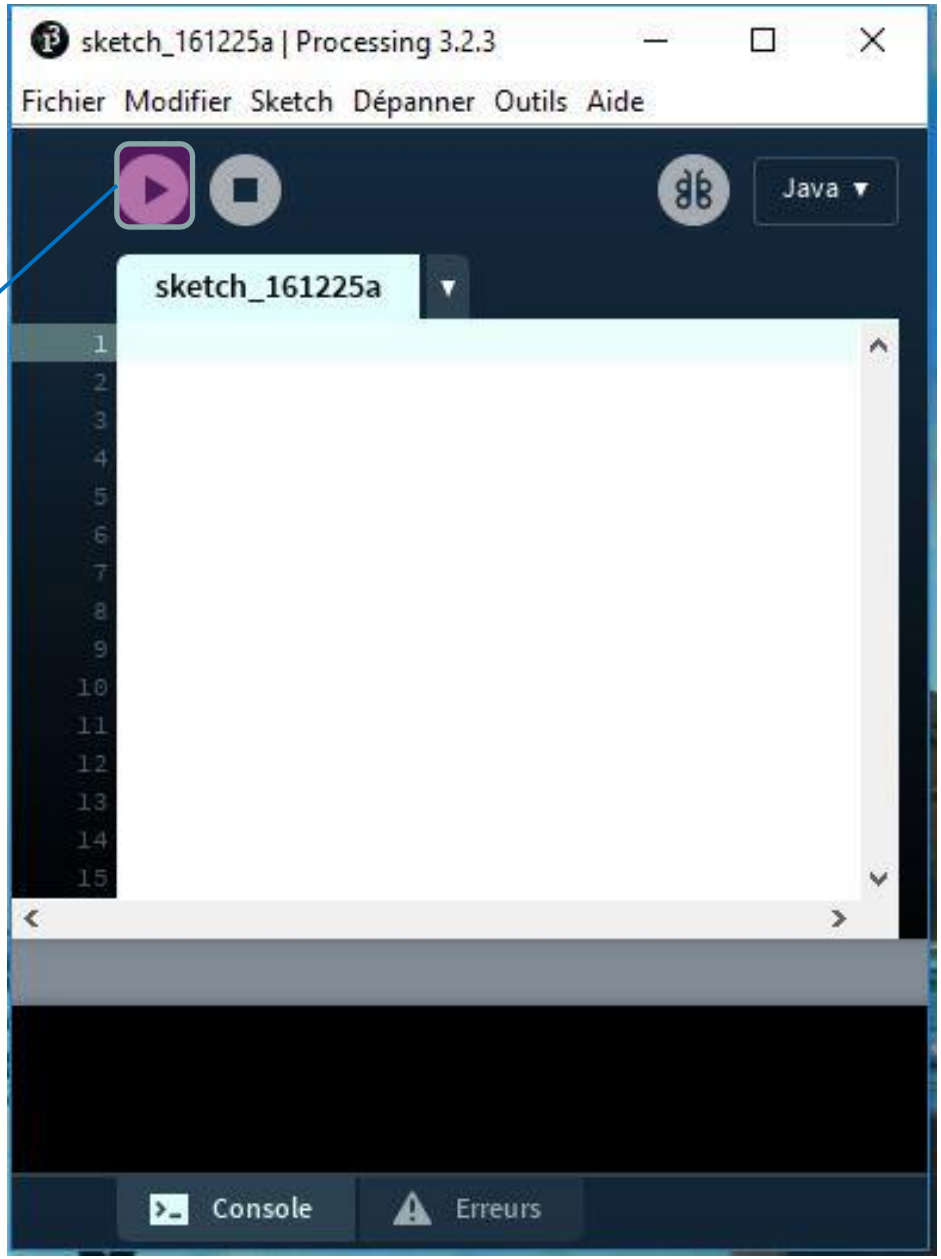
Console
(message d'erreur / affichage d'un texte)



Barre de menu

4.1 L'interface

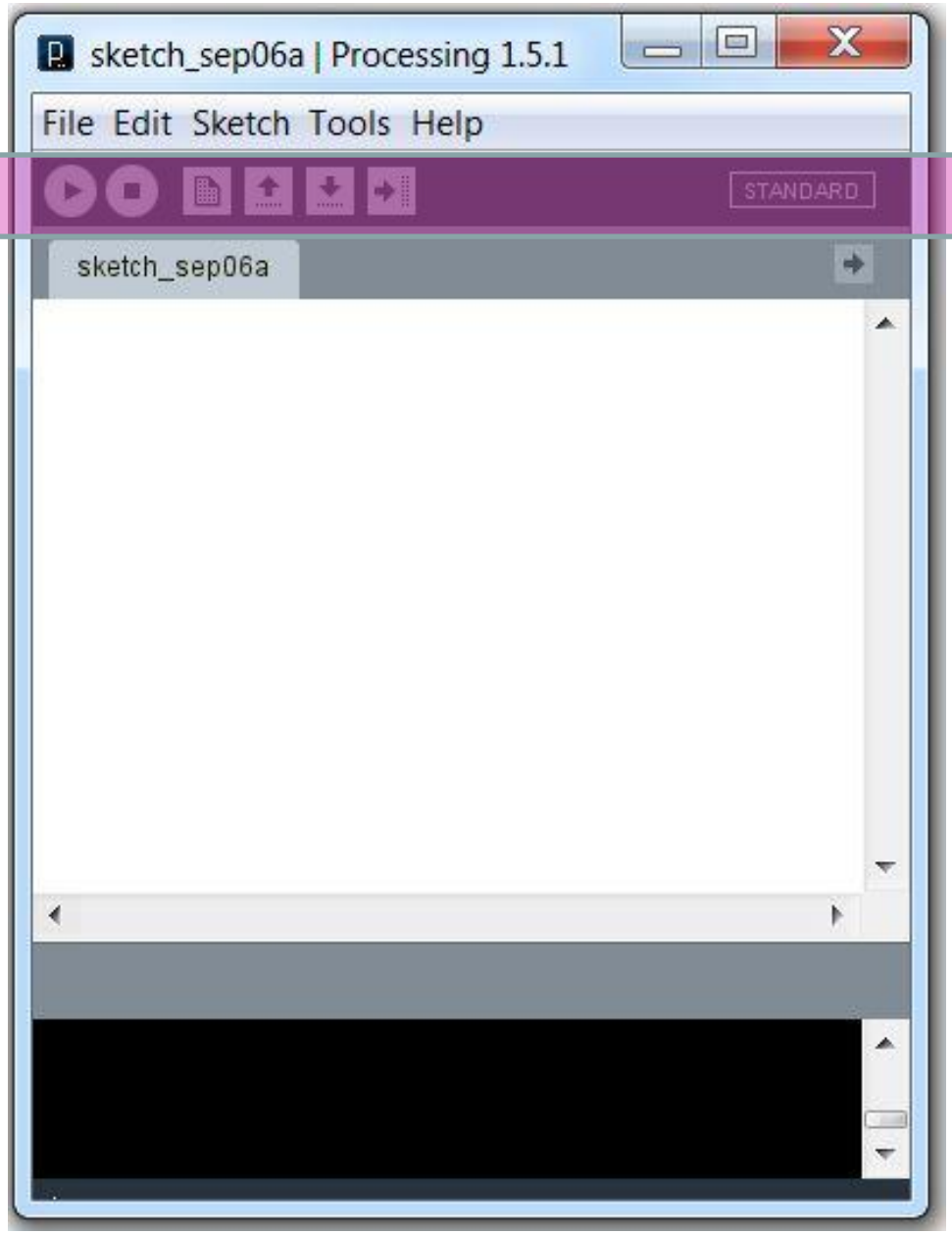




Fenêtre d' affichage
(espace de dessin et d'animation)

4.1 L'interface

Barre d'action



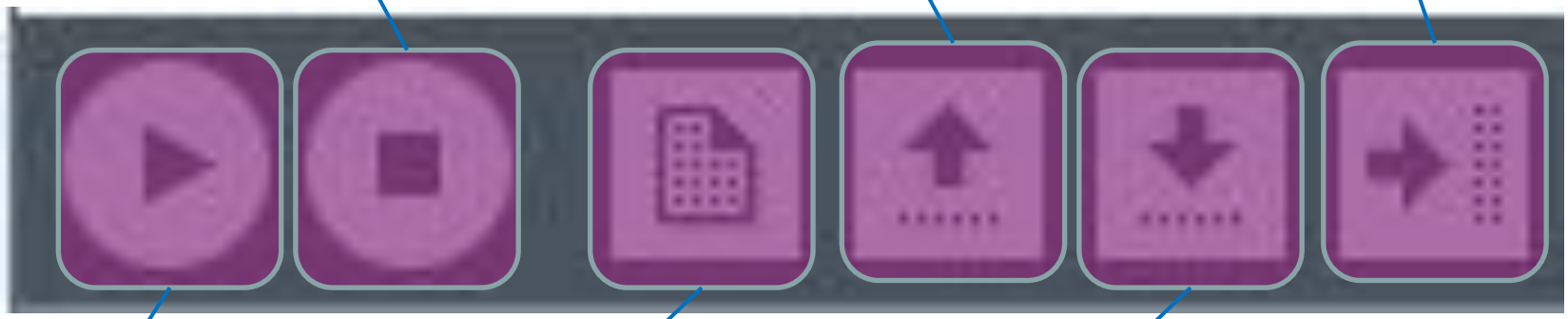
4.1 L'interface

4.1.1 Les barres d'action

Bouton "Stop": arrête l'exécution du sketch .

Bouton "Open" : ouvre un sketch existant.

Bouton "Export" : exporte le sketch pour le web.



Bouton "Run" : exécute du sketch, c-a-d du programme.

Bouton "New" : Crée un nouveau sketch.

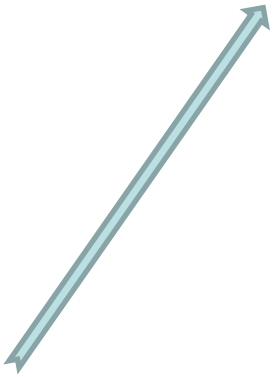
Bouton "Save" : sauvegarde le sketch en cours.

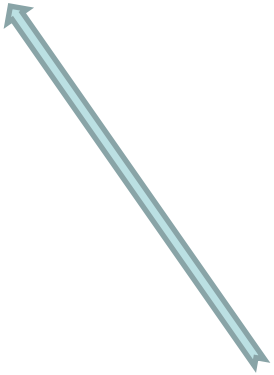
4.1 L'interface

4.1.2 Le dossier de travail

Par défaut

(Windows) Mes Documents/Processing
(Mac) Documents/Processing



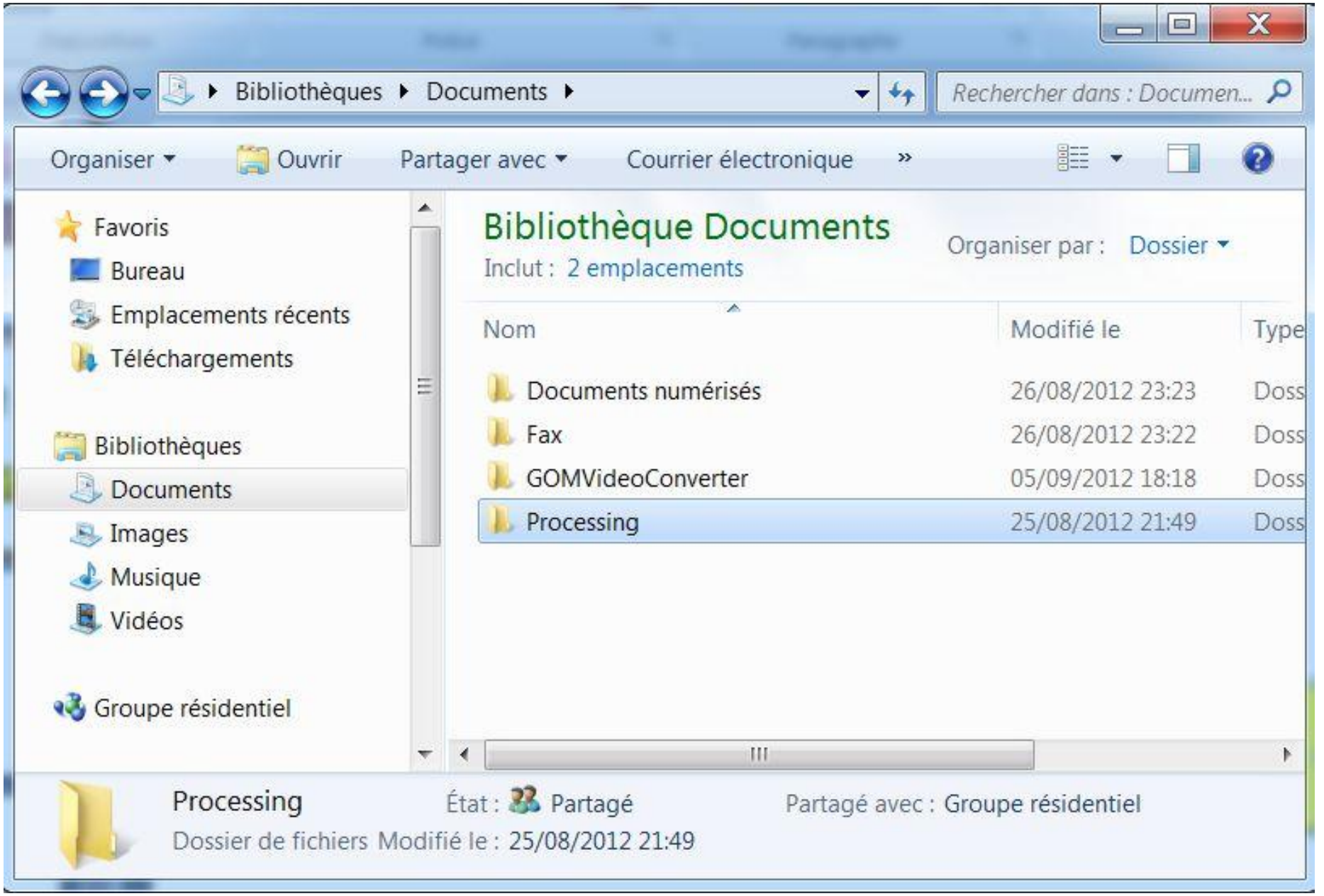


Sketchs

Librairies
(modules externes proposant des fonctionnalités supplémentaires)

4.1 L'interface

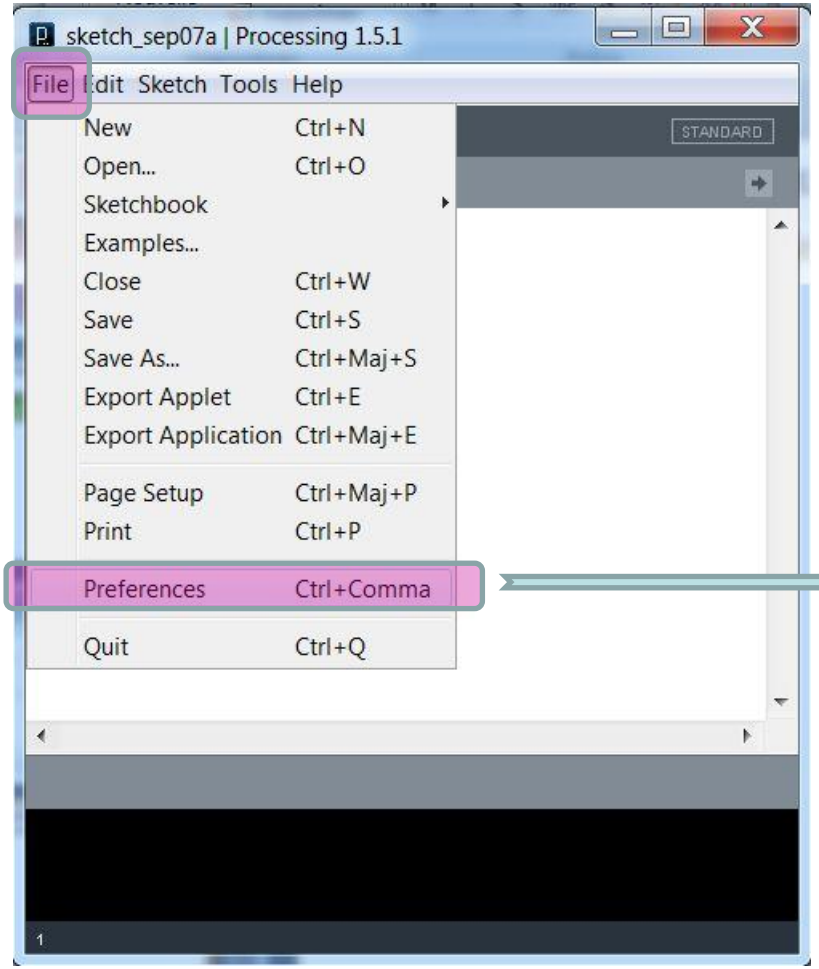
4.1.2 Le dossier de travail

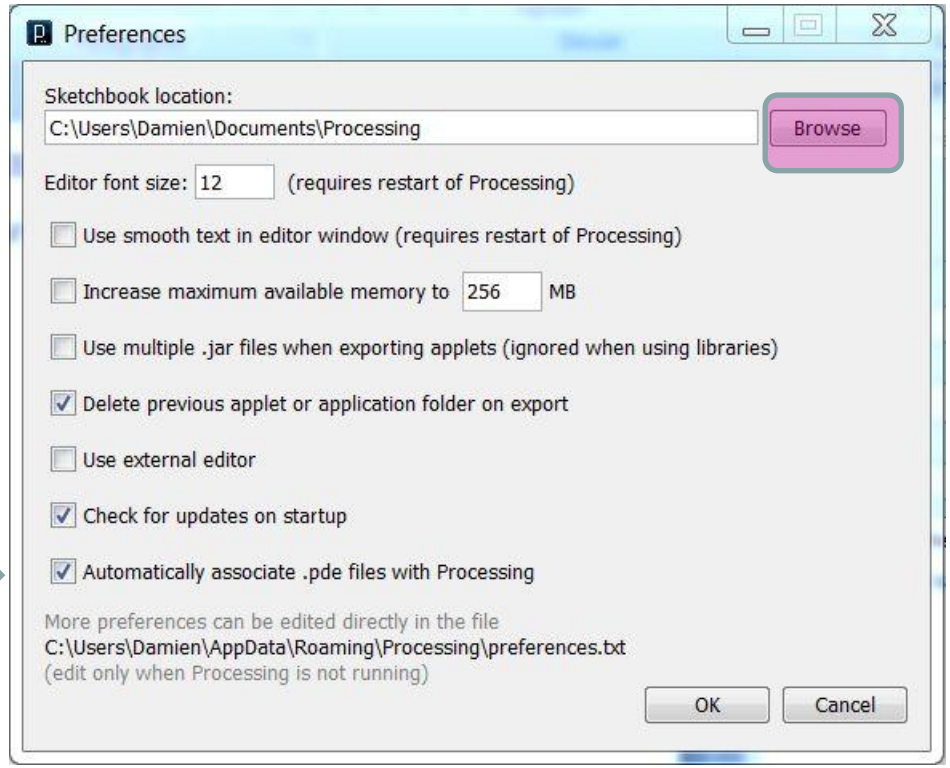


4.1 L'interface

4.1.2 Le dossier de travail

Pour changer le dossier de travail:

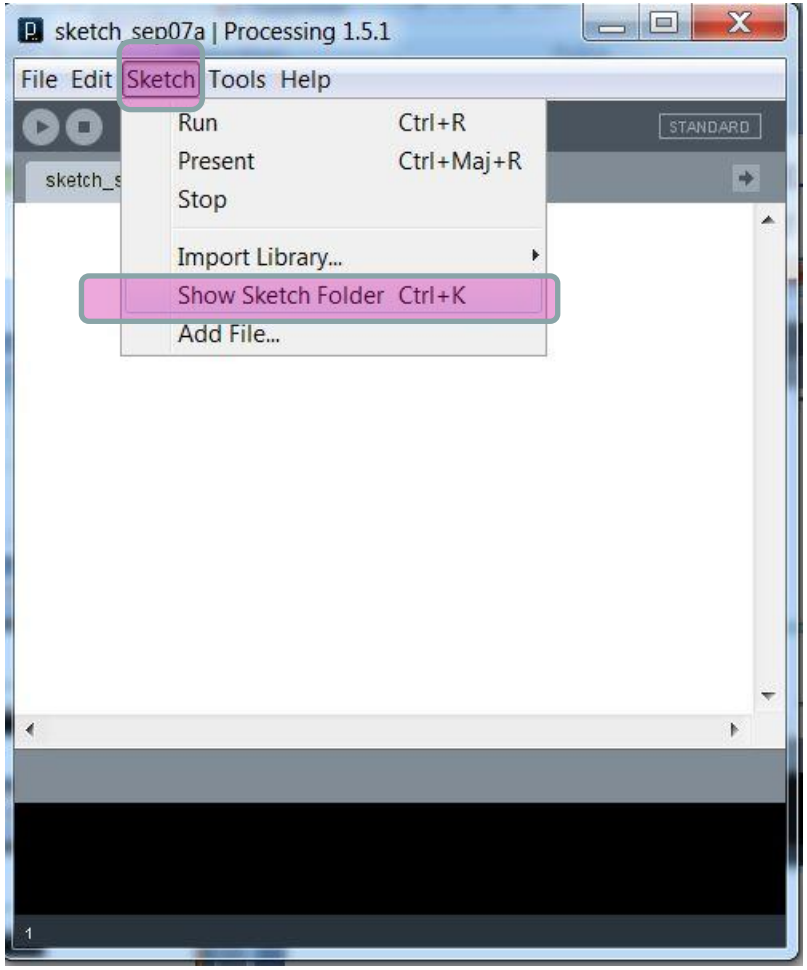




4.1 L'interface

4.1.2 Le dossier de travail

Où se trouve le dossier de travail:



4.2 Les bases du langage de Processing

JAVA ⟹ Règles de syntaxe à respecter pour une exécution correcte

4.2.1 Majuscule et minuscule

size(200,200); ≠ Size(200,200);

4.2.2 Le point virgule

Signale la fin de l'instruction

```
//Dessine un cercle
ellipse(10,10, 10, 10);

//affiche un texte
//dans la console

println(10 + 23);
```

Remarque:
« // » signale le début d'un commentaire ignoré lors de l'exécution

4.2.3 Appel à des méthodes prédéfinies

Méthodes = Fonctionnalités prédéfinies (Librairies)

- Dessiner un rectangle
- Définir une couleur
- Calculer une racine carré
- ...

Appel à une méthode

- Taper le nom en respectant Majuscule/minuscule
- Préciser les paramètres de la méthode entre parenthèses

Exemple:

```
fill(128);
ellipse(50, 50, 60, 60);
```

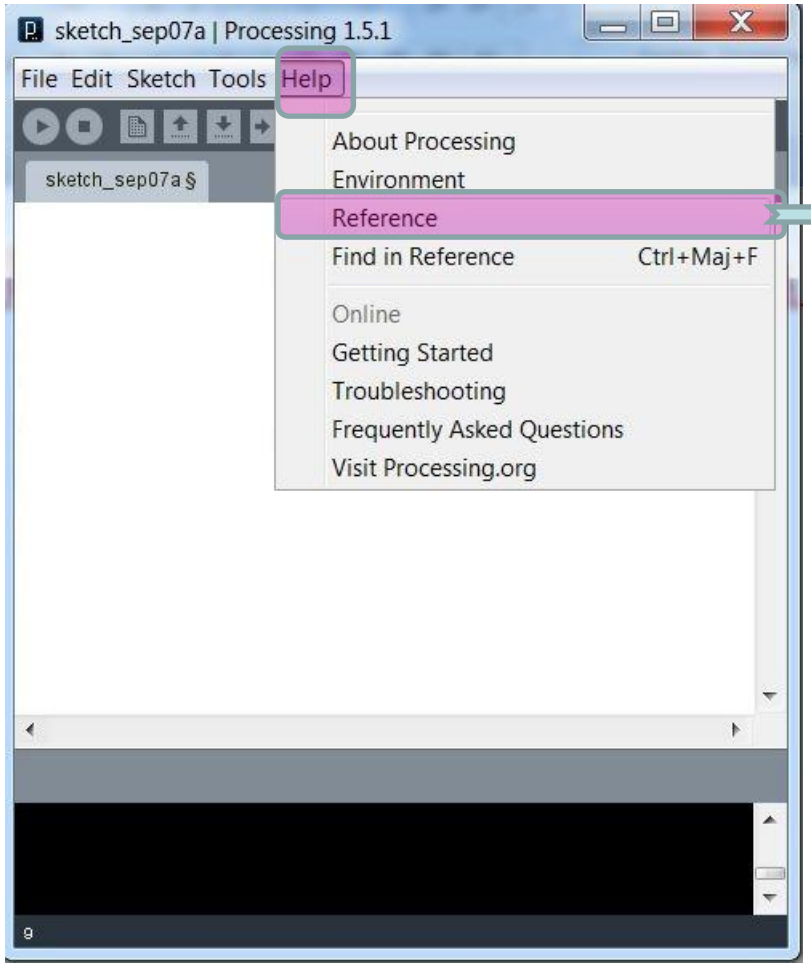
4.2.4 Affichage dans la console

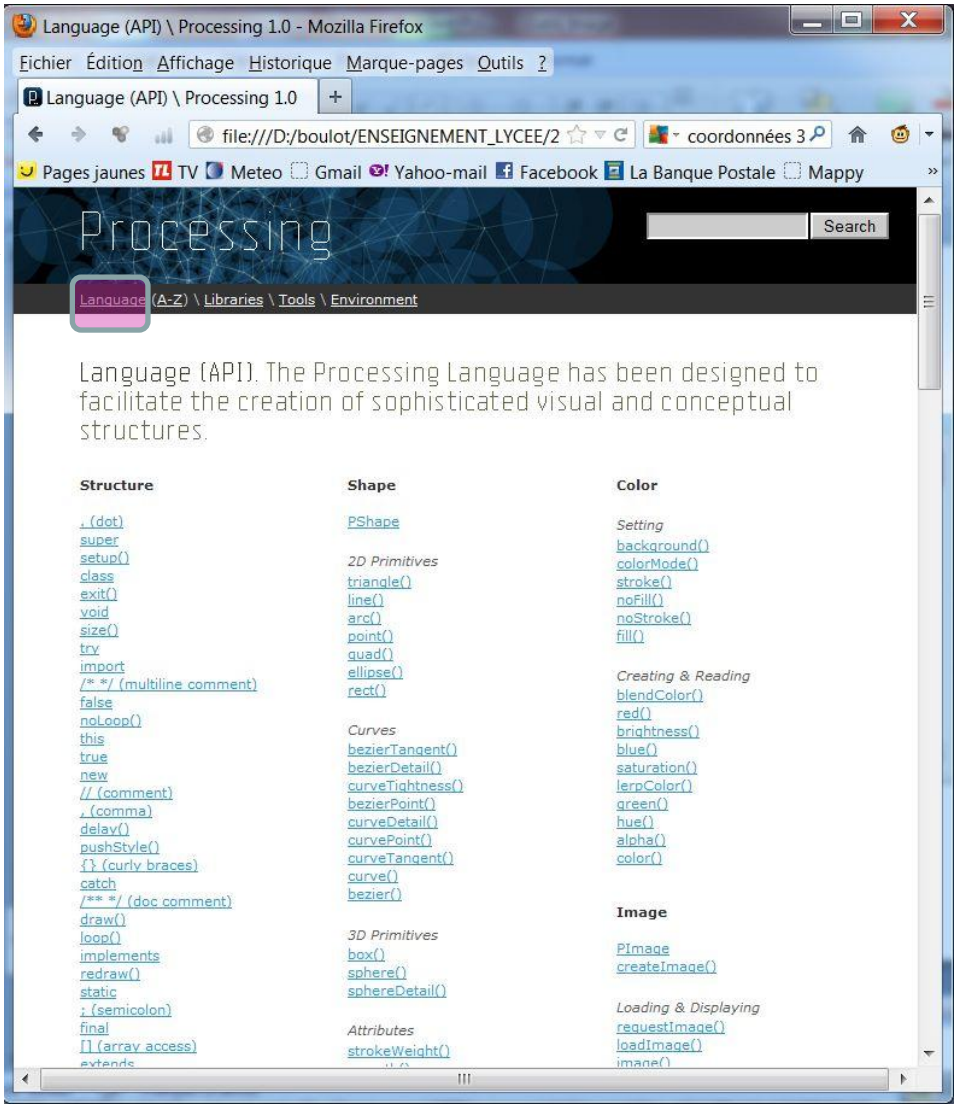
- println("Salut tout le monde!");
- println("1000");

4.2.5 Opérations arithmétiques

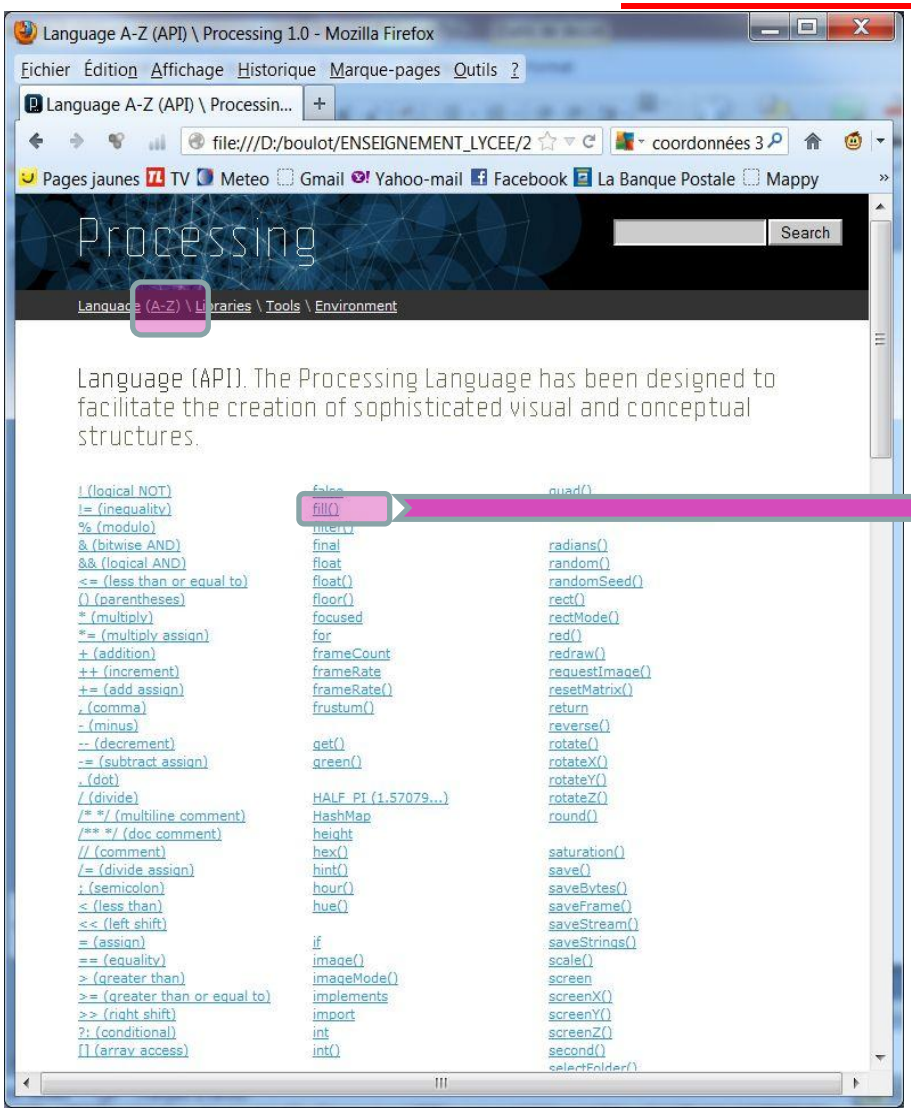
- println(10 + 5);
- println(10 + 5 * 3); // 5*3 (soit 15) puis additionne 10
- println((10 + 5) * 3); // 10+5 (soit 15) puis multiplie 15 par 3
- println(10.4 + 9.2);

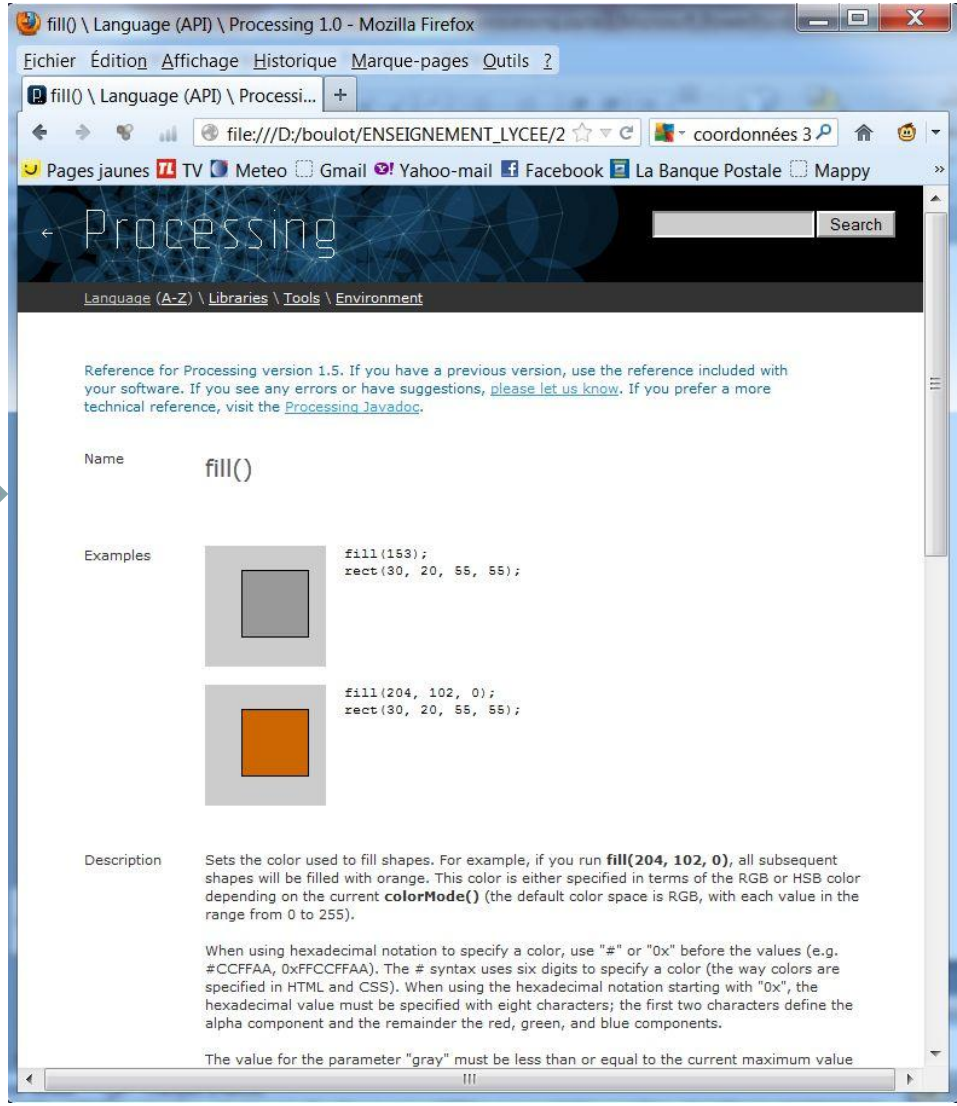
Toutes les méthodes de Processings sont documentées

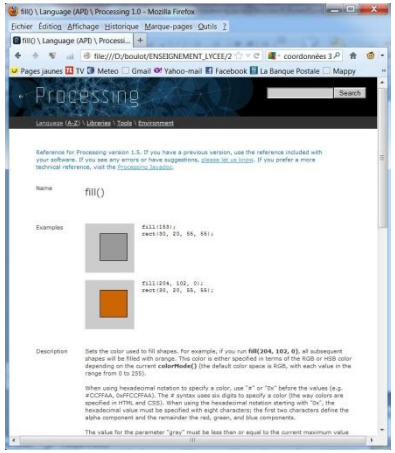




Toutes les méthodes de Processings sont documentées







fill() \ Language (API) \ Processing 1.0 - Mozilla Firefox

Fichier Édition Affichage Historique Marque-pages Outils ?

fill() \ Language (API) \ Processi...

file:///D:/boulot/ENSEIGNEMENT_LYCEE/2

coordonnées 3

Pages jaunes TV Meteo Gmail Yahoo-mail Facebook La Banque Postale Mappy

alpha component and the remainder the red, green, and blue components.

The value for the parameter "gray" must be less than or equal to the current maximum value as specified by **colorMode()**. The default maximum value is 255.

To change the color of an image (or a texture), use tint().

Syntax

```
fill(gray)
fill(gray, alpha)
fill(value1, value2, value3)
fill(value1, value2, value3, alpha)
fill(color)
fill(color, alpha)
fill(hex)
fill(hex, alpha)
```

Parameters		
	gray	int or float: number specifying value between white and black
	alpha	int or float: opacity of the fill
	value1	int or float: red or hue value
	value2	int or float: green or saturation value
	value3	int or float: blue or brightness value
	color	color: any value of the color datatype
	hex	int: color value in hexadecimal notation (i.e. #FFCC00 or 0xFFFFCC00)

Returns None

Usage Web & Application

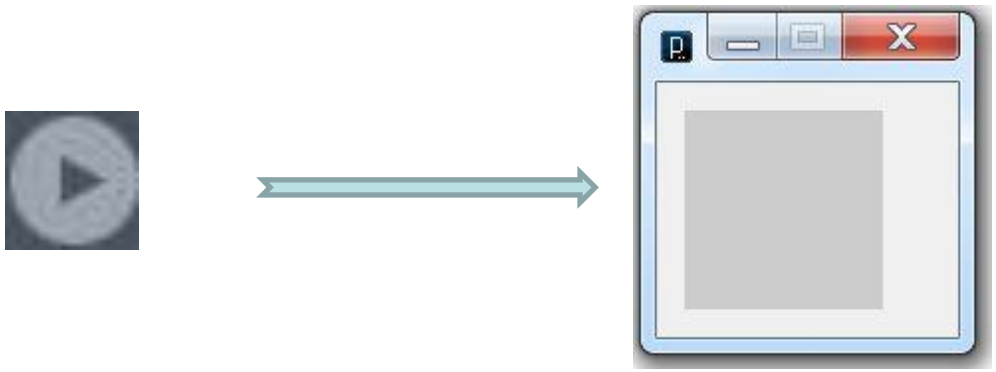
Related
noFill()
stroke()
tint()
background()
colorMode()

Conseil de programmation

SAUVEGARDER REGULIEREMENT LE PROGRAMME!!!

5. L'ESPACE DE DESSIN

- Espace de dessin = espace de représentation graphique 2D, 3D, animation etc.
- Il apparait dans une fenêtre lorsqu'on appuie sur le bouton « run »



- Créé par l'instruction : size(largeur,hauteur)

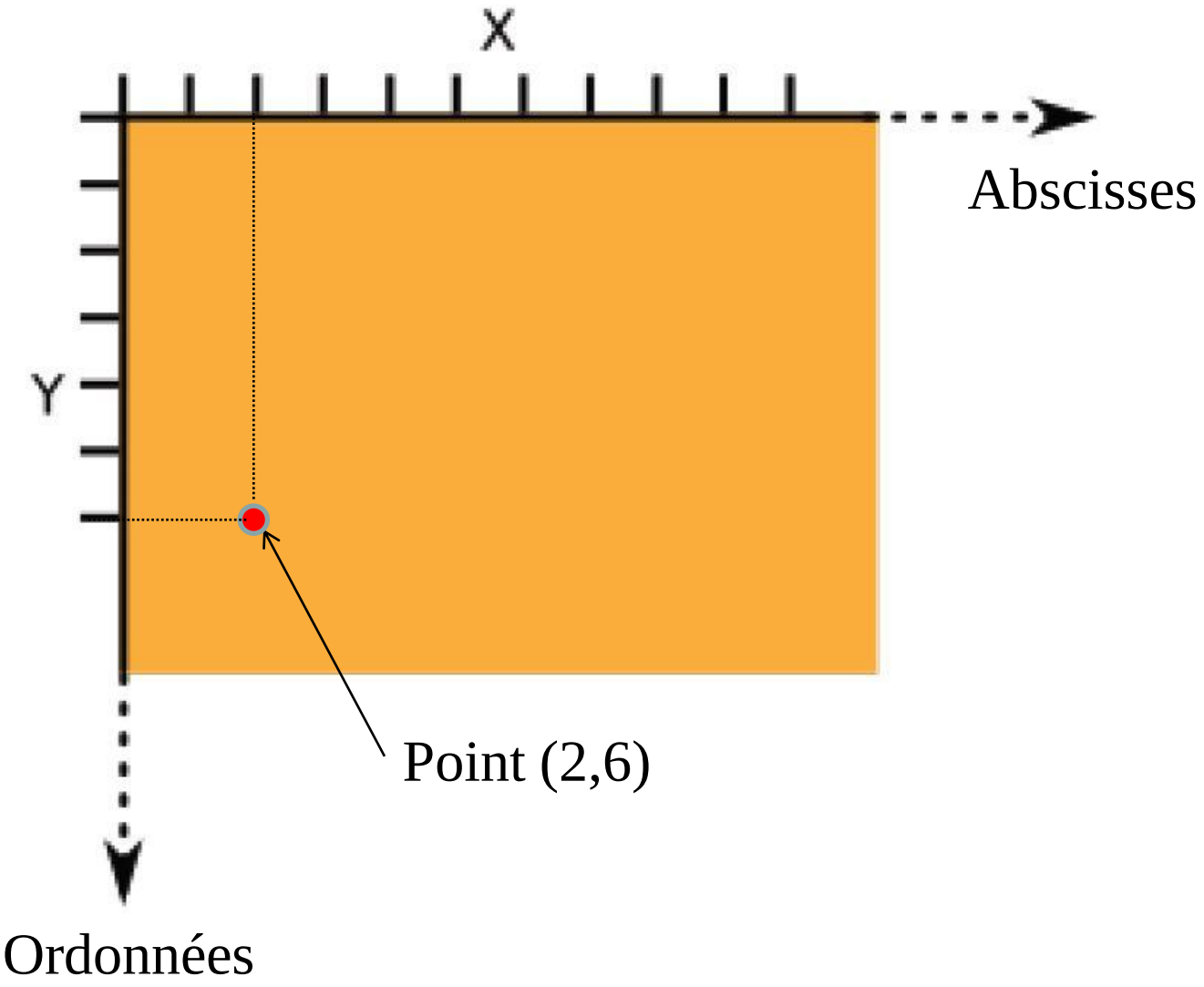
Exemple: size(300,200);

- Par défaut, la taille de l'image est 100×100 pixels

size();

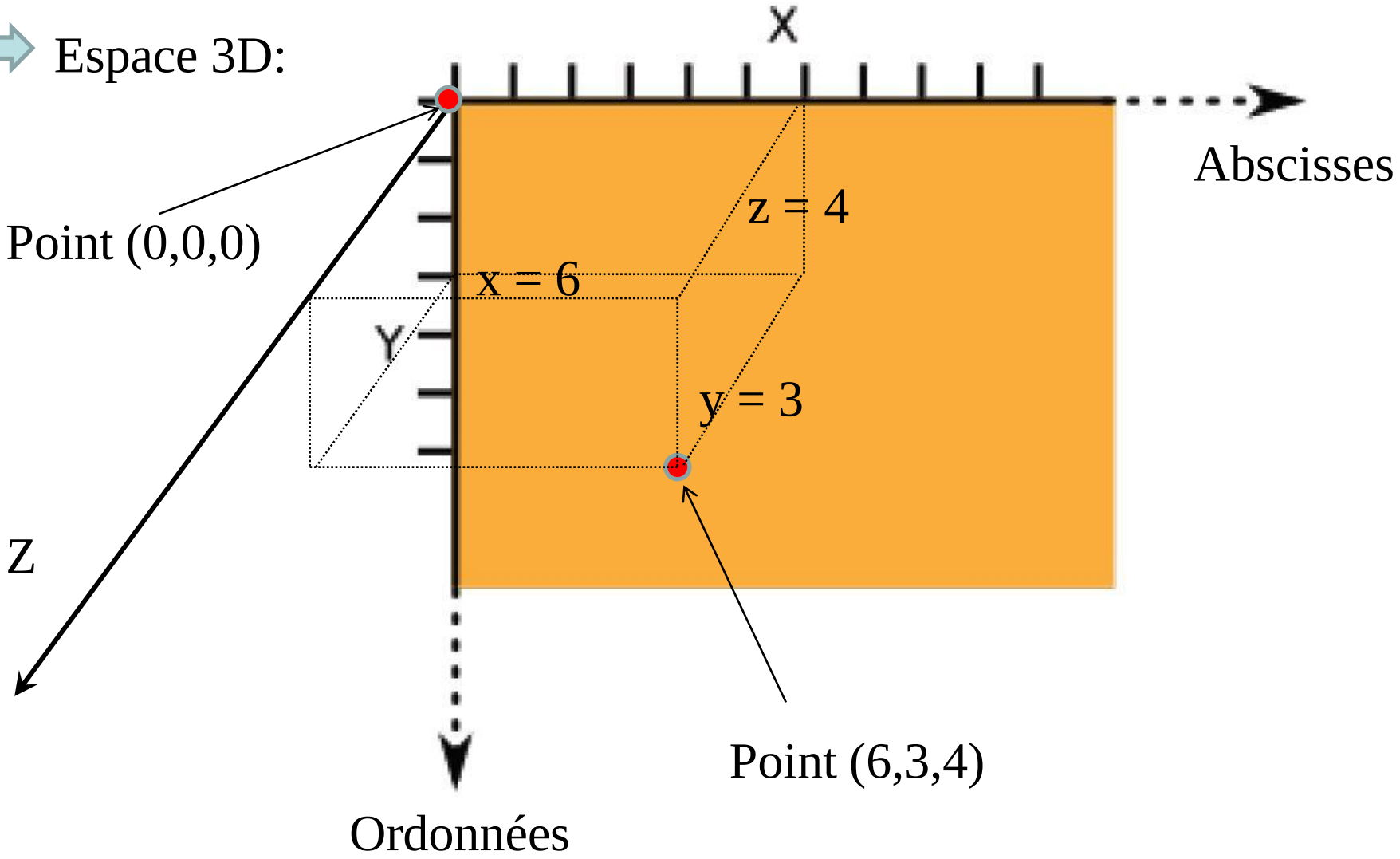
5.1 Les coordonnées d'espace

Espace 2D:



5.1 Les coordonnées d'espace

Espace 3D:



Exemple:

```
size(300,300,P3D);
lights();

translate(50,50,0);
fill(66,150,80);
sphere(25);

translate(150,150,100);
fill(250,0,0);
sphere(25);
```

6. LES FORMES GÉOMÉTRIQUES

6.1 Le point

point(x,y); ou point(x,y,z); (voir documentation)

Exemple:

```
size(100,100);
point(50,50);
```

Exercice:

```
size(100,100);
point(150,150);
```

Commentaire??

Comment régler ce problème?

6.2 La ligne

line(xA,yA,yB,ZB); (voir documentation)

Exemple:

```
line(15,90,95,10);
```

6.3 Le rectangle

rect(x,y, largeur, hauteur); (voir documentation)

Coordonnées du coin supérieur gauche (par défaut)

Exemple:

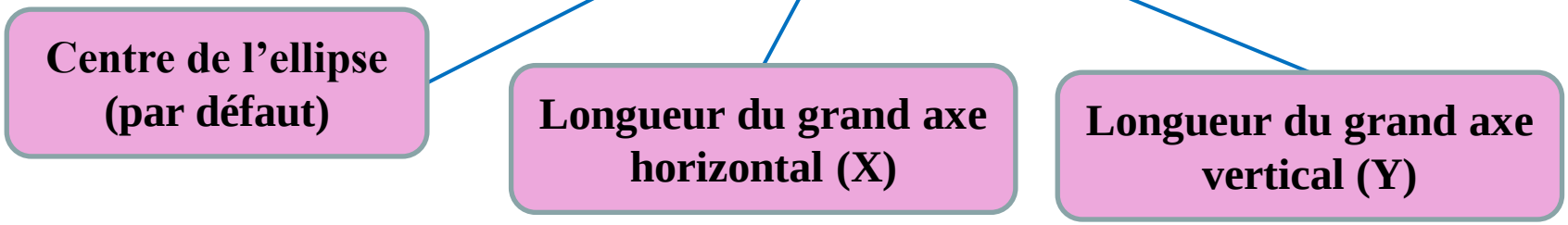
```
Line(10,10,80,80);
```

⇨ Pour que les 1ère coordonnées (x,y) correspondent au centre du rectangle, il faut changer de mode: CENTER

```
rectMode(CENTER);
rect(50, 50, 80, 40);
```

6.4 L'ellipse

ellipse(x,y, largeurX, hauteurY); (voir documentation)



Exemple: ellipse(50, 50, 80, 80);

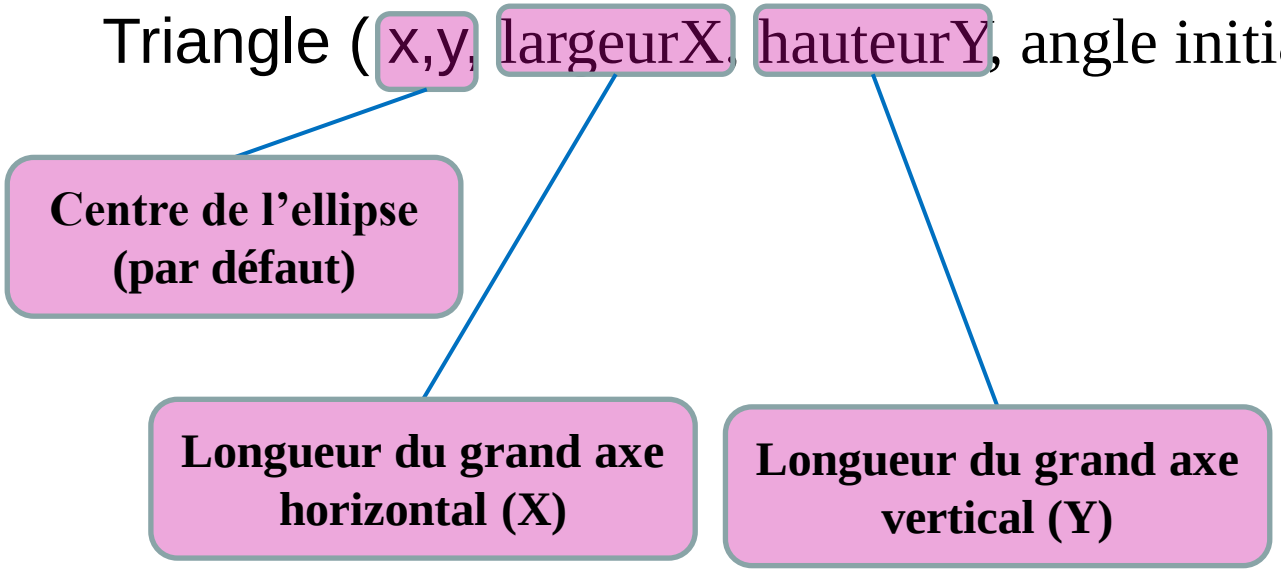
6.5 le triangle

Triangle (x1,y1, x2,y2, x3,y3); (voir documentation)

Exemple: triangle(10, 90, 50, 10, 90, 90);

6.5 L'arc d'ellipse

Triangle (x,y, largeurX, hauteurY, angle initial, angle final);



Exemple: arc(50, 50, 90, 90, 0, PI);

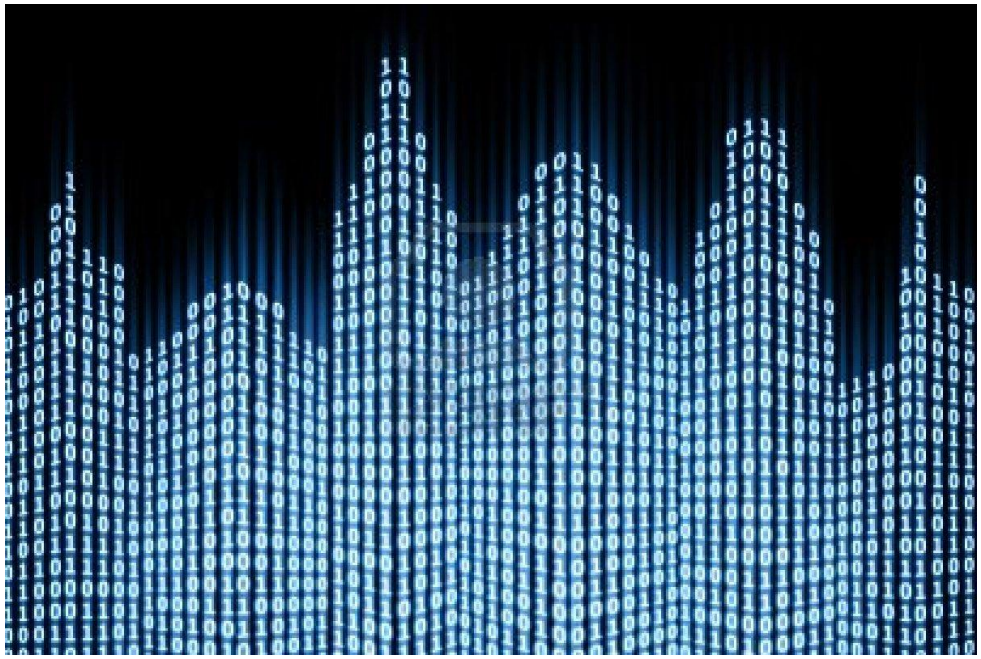
6.13 Les Primitives 3D

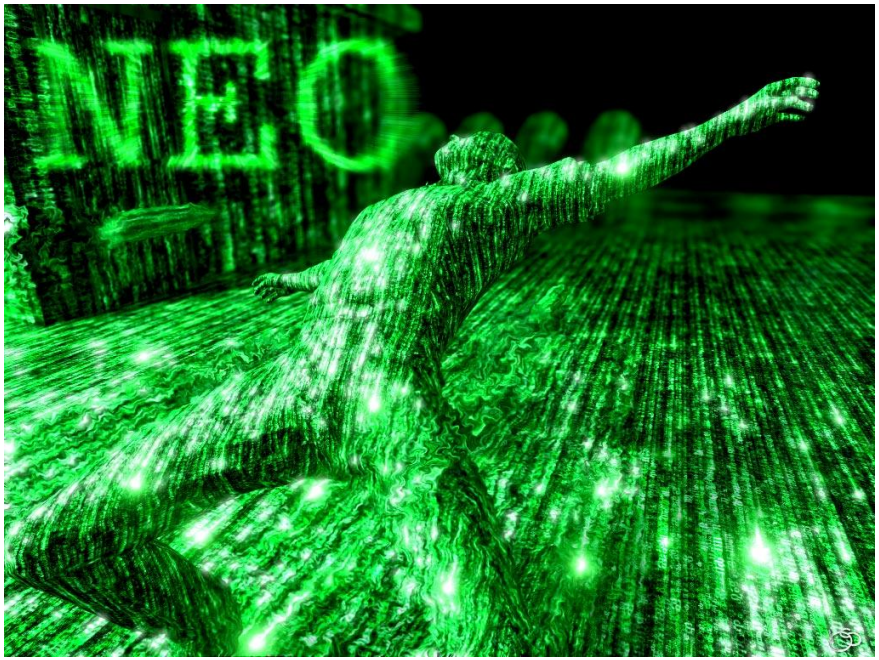
- Appel aux librairies de la 3D: `size(x,y,P3D);`
  - Sphère: `sphere(taille);`
  - Cube: `box(longueur, largeur, hauteur);`
  - Effets d'éclairage de la forme 3D: `lights();`

7. LES REPRÉSENTATION NUMÉRIQUES (À la main au tableau)

- **Comment compte la machine ?**
  - La base décimale
  - La base binaire
  - La base hexadécimale

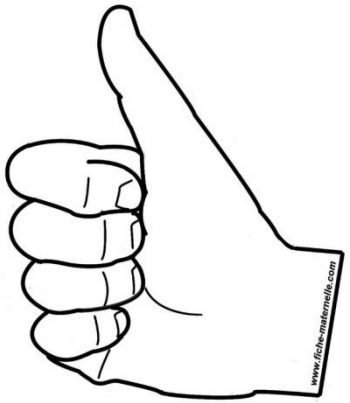
7. LES IMAGES NUMÉRIQUES

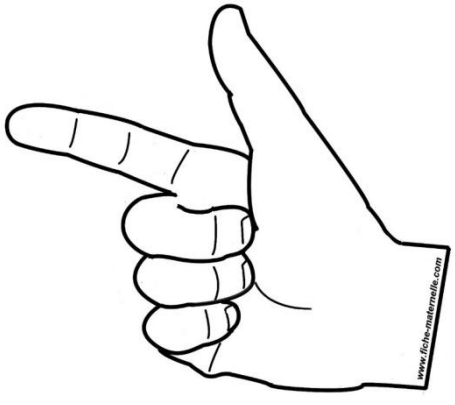


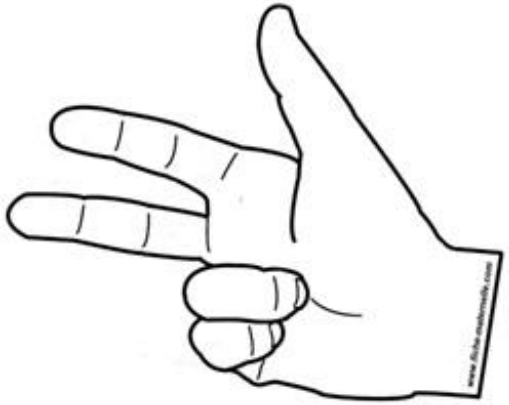


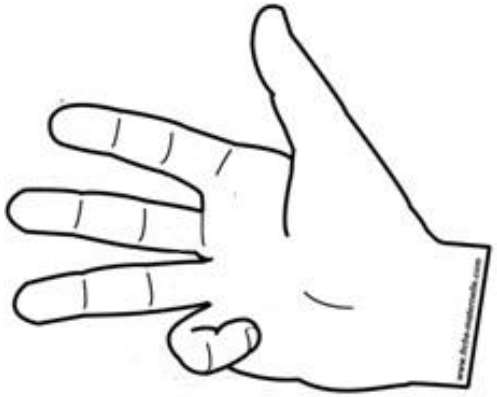
Introduction Rappel sur la base binaire

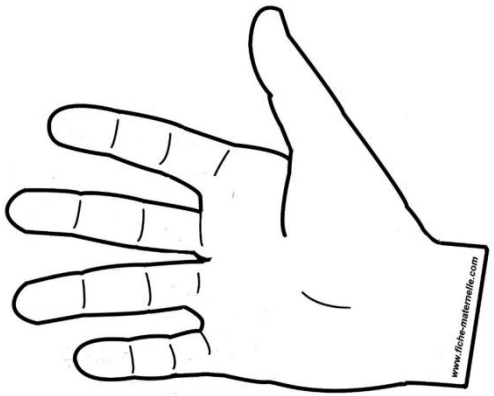
Q: Comment un enfant fait-il généralement pour compter?



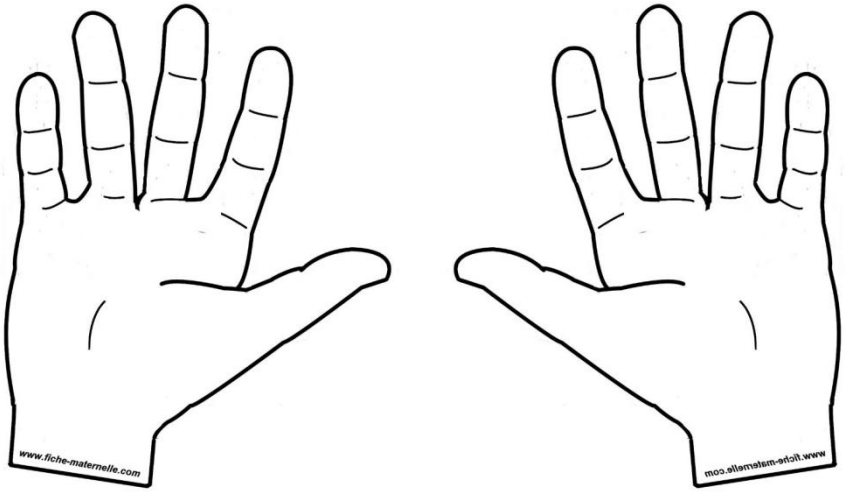








Etc…



10 doigts !!

- **Monde occidental:** **on compte en base 10**
- **Base:** **0, 1, 2, 3, 4, 5, 6, 7, 8, 9**

10 caractères

- **Tous les nombres en base 10 sont des mots composés de ces 10 caractères (0-9) !!!**

What about the machine?

Ne reconnait que 2 états

Allumé
État 1

Eteint
État 0

Base binaire !!

Tous les nombres en base 2 sont des mots composés de ces 2 caractères (0 ou 1) !!!

1 caractère binaire = 1 bit

Mot de 8 bits = 1 octet

Q: Jusqu'à combien peut-on compter en base 10 avec 1 octet ?

Base binaire		Base 10	
0 0 0 0 0 0 0 0	⟹	000	256 valeurs
1 1 1 1 1 1 1 1	⟹	255	

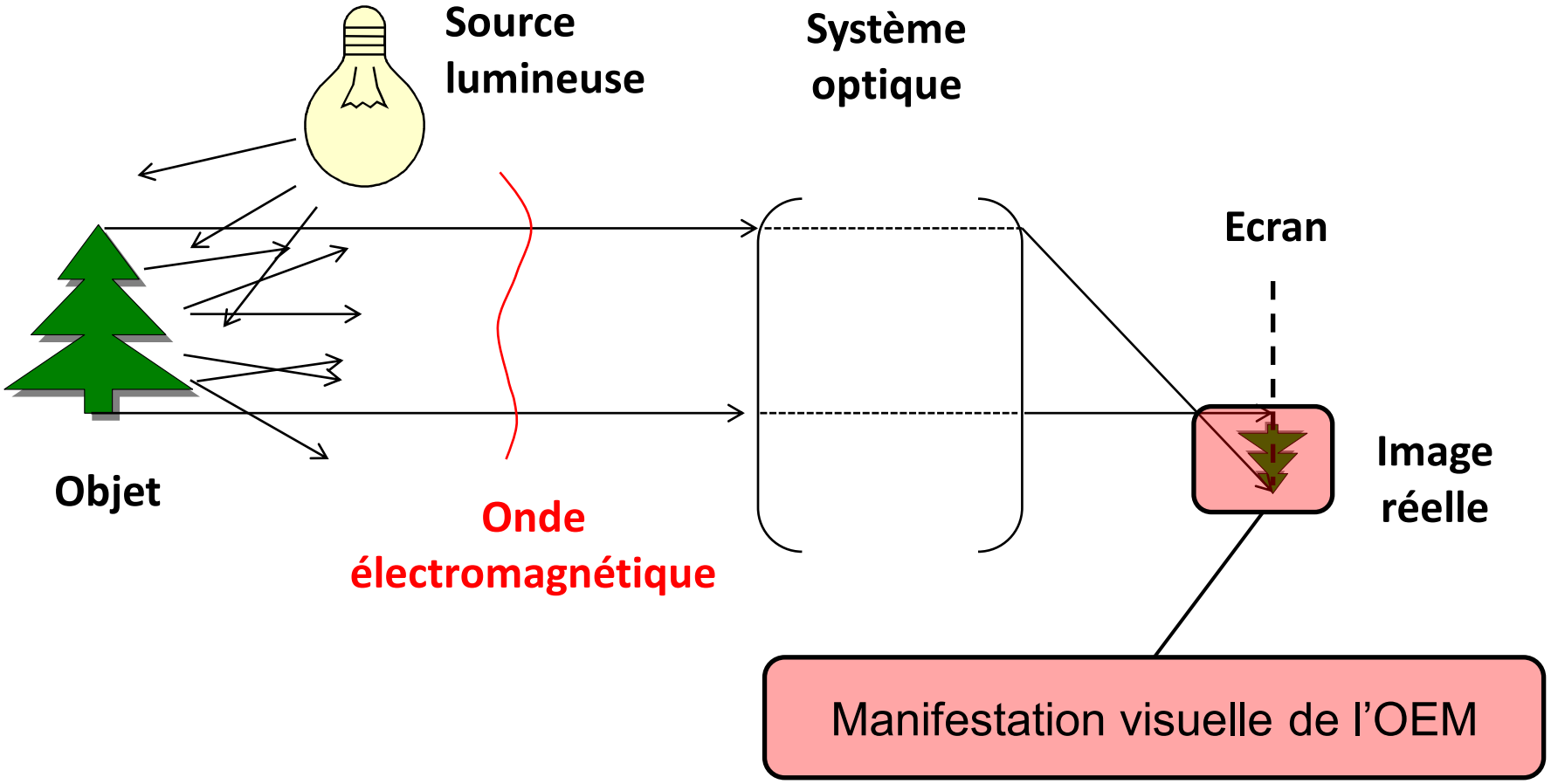
L'octet est l'unité de stockage en mémoire

I. Images en niveaux de gris

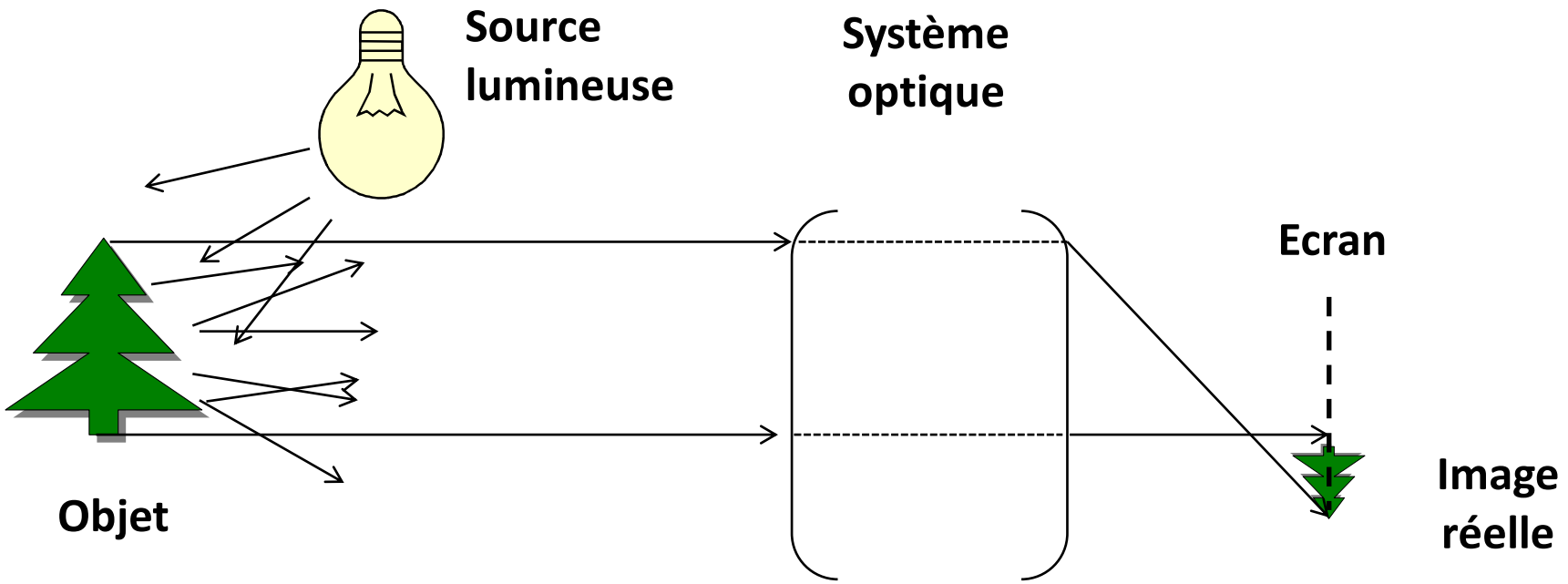
1.1. De l'image physique à l'image numérique

Q: comment former une image sur un écran ?

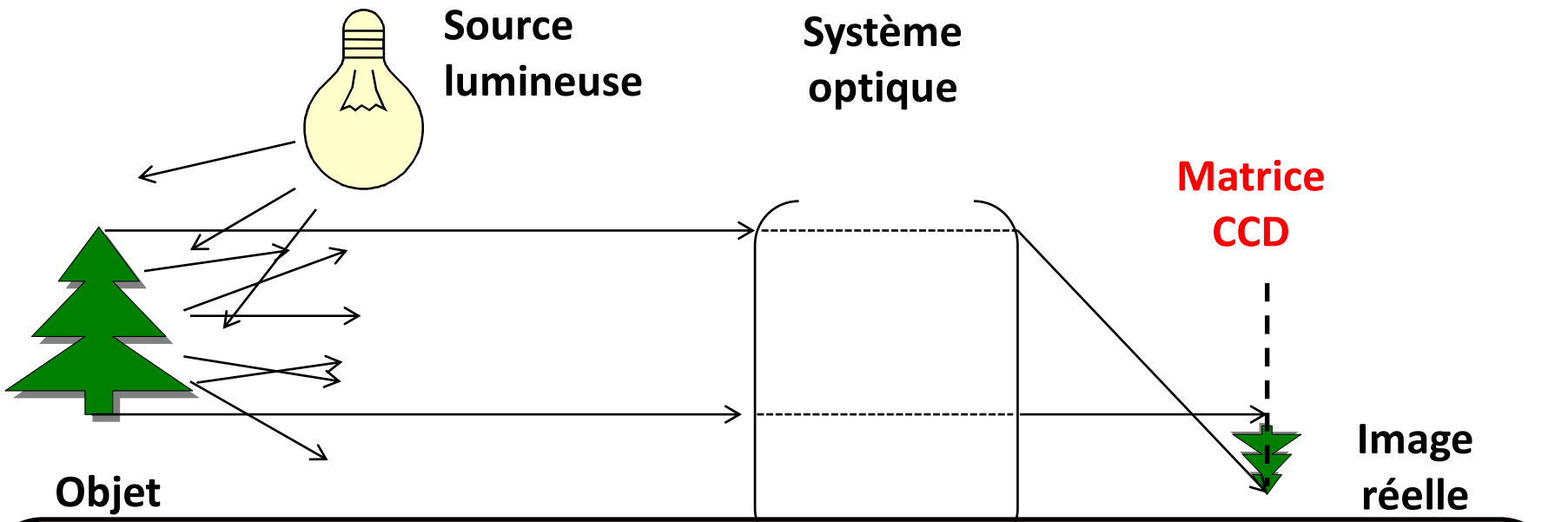
Avec un système optique (objectif, lentille, etc…)



Acquisition d'une image numérique



Acquisition d'une image numérique



Quantification de la tension enregistrée pour chaque pixel sur 8bits, 16bits, etc.… en fonction de la qualité de l'image souhaitée

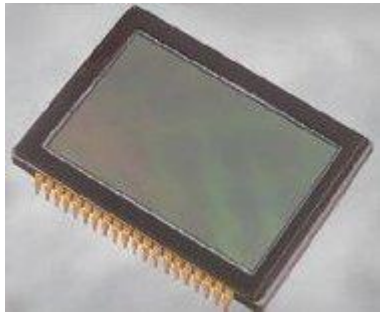
Convertisseur courant-tension

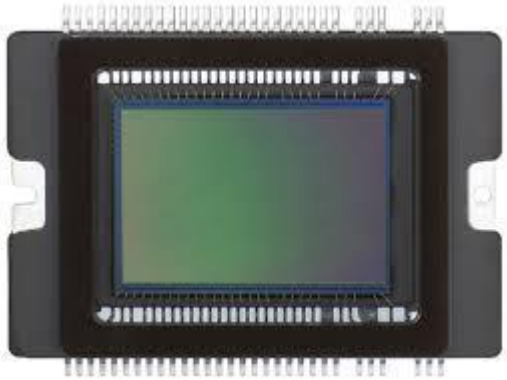
Etape de numérisation de l'image

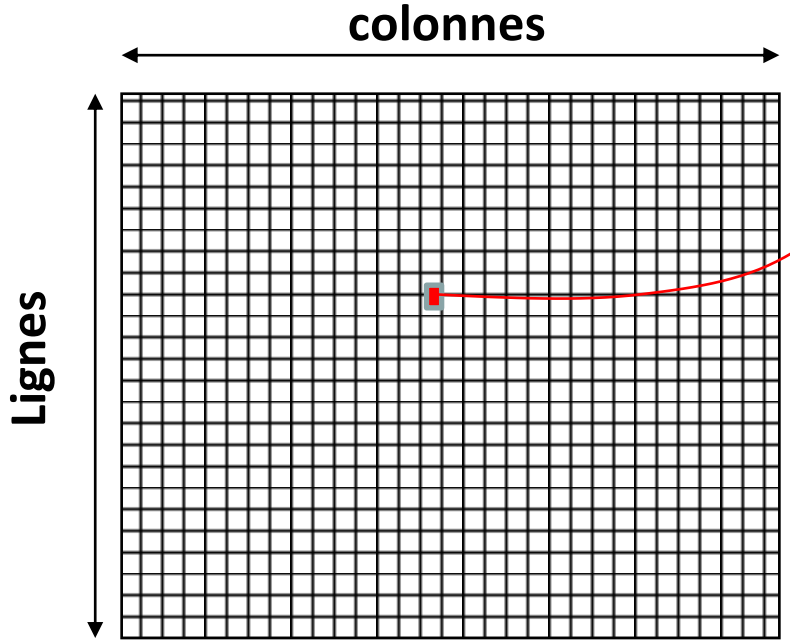
Quantification: Passage d'une tension analogique mesurée à une valeur numérique codée sur un sur un nombre donné de « bit » correspondant à un nombre discret et fini de niveaux de gris entre le noir (intensité lumineuse nulle reçue par le pixel) et le blanc (intensité maximale qu'il est possible d'enregistrer sur un pixel).

1.2. Fonctionnement de la matrice CCD

Matrice CCD: composée de plusieurs pixels (« Picture element ») disposés sous forme de tableau.

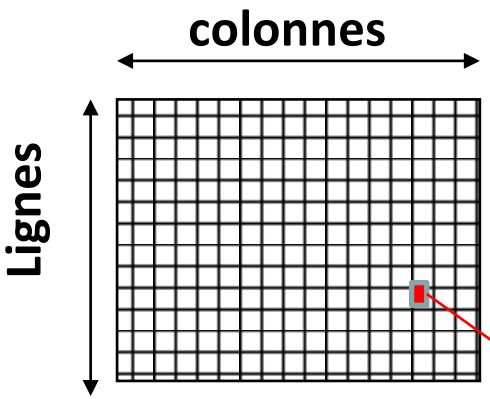


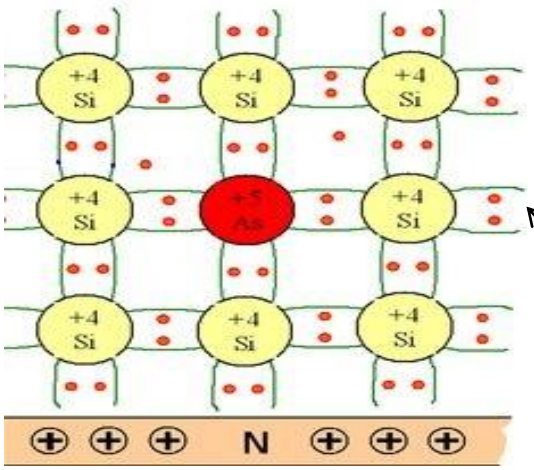


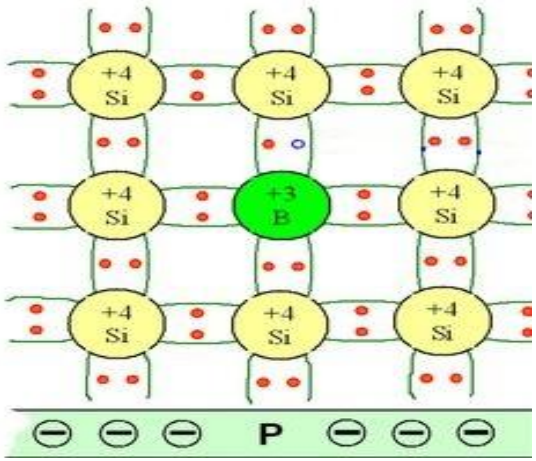


Pixel = cellule photosensible

Matériau semi-conducteur







Silicium dopé N

Silicium dopé P

Photon incident

Puits d'électrons

Puits de trous

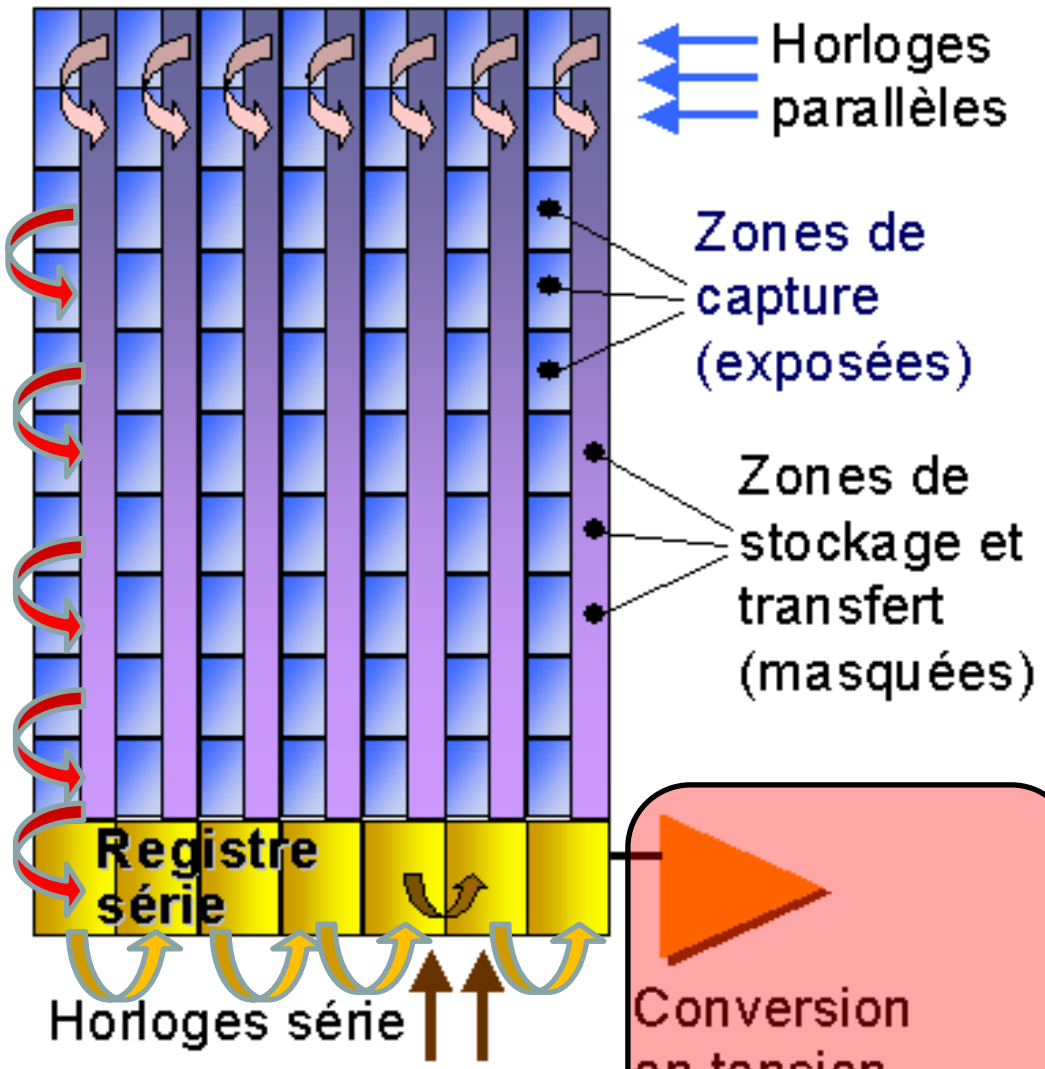
Création d'une paire électron -trou

Nombre d'électrons/trous créés est proportionnel à la quantité de lumière reçue pendant toute la durée de l'exposition lumineuse.

Problème technique :

Comment compter les charge créées sur chaque pixel ?

Transfert des charges de chaque pixel de proche en proche

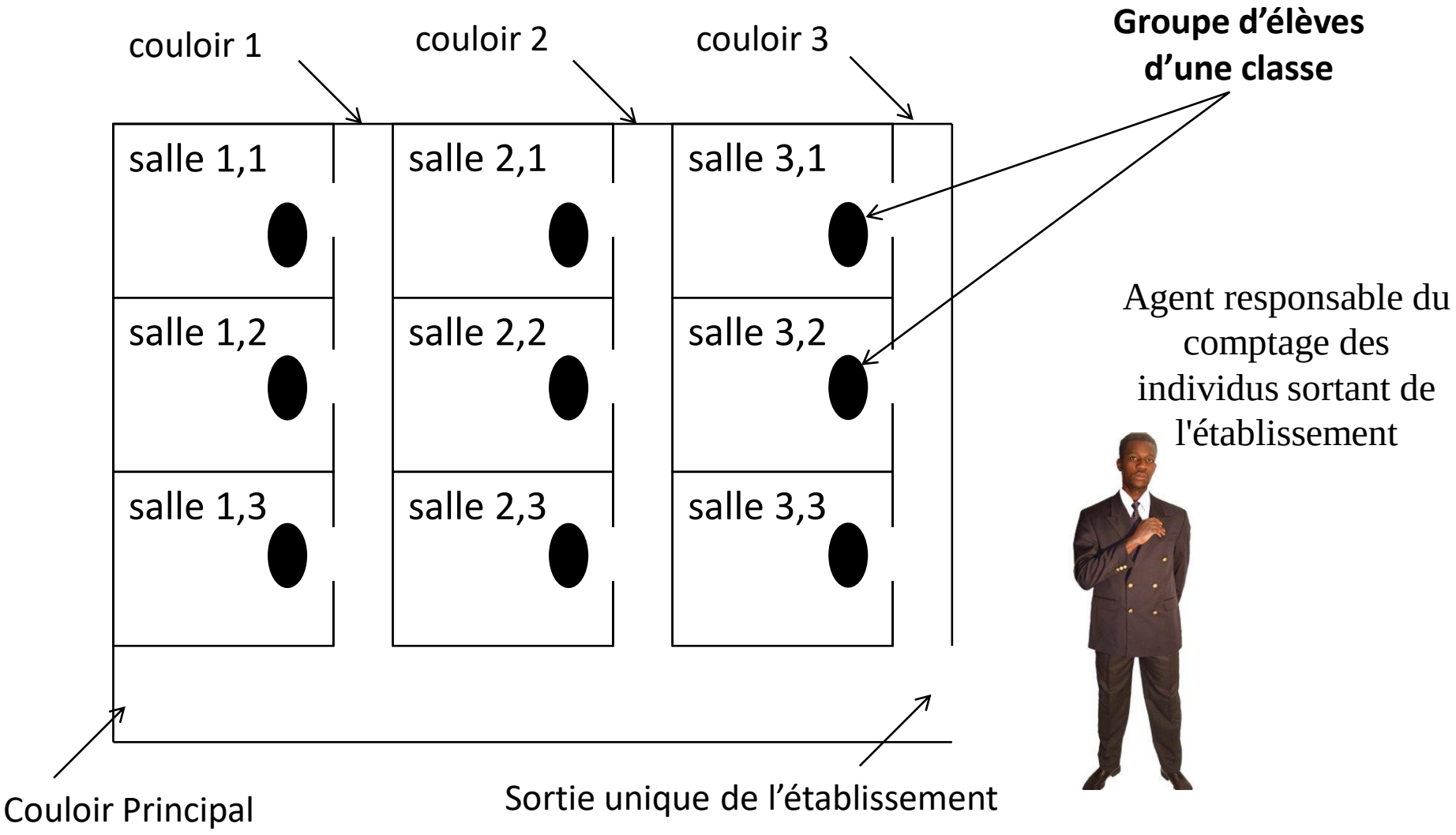


Mouvement de charges = courant électrique

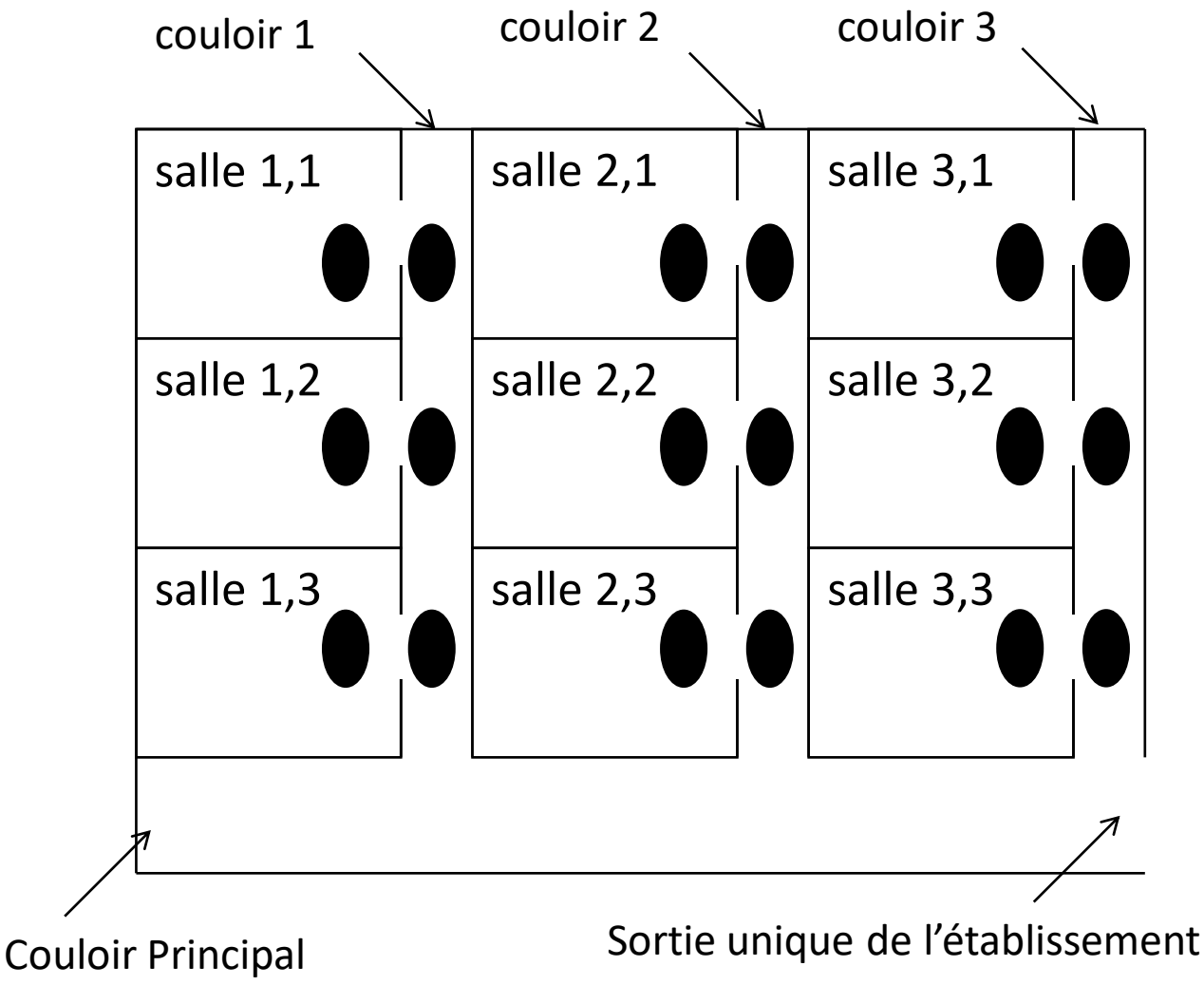
Nécessité de convertir le courant en tension pour la mesure

Processus de comptage

Par analogie: évacuation d'un établissement scolaire



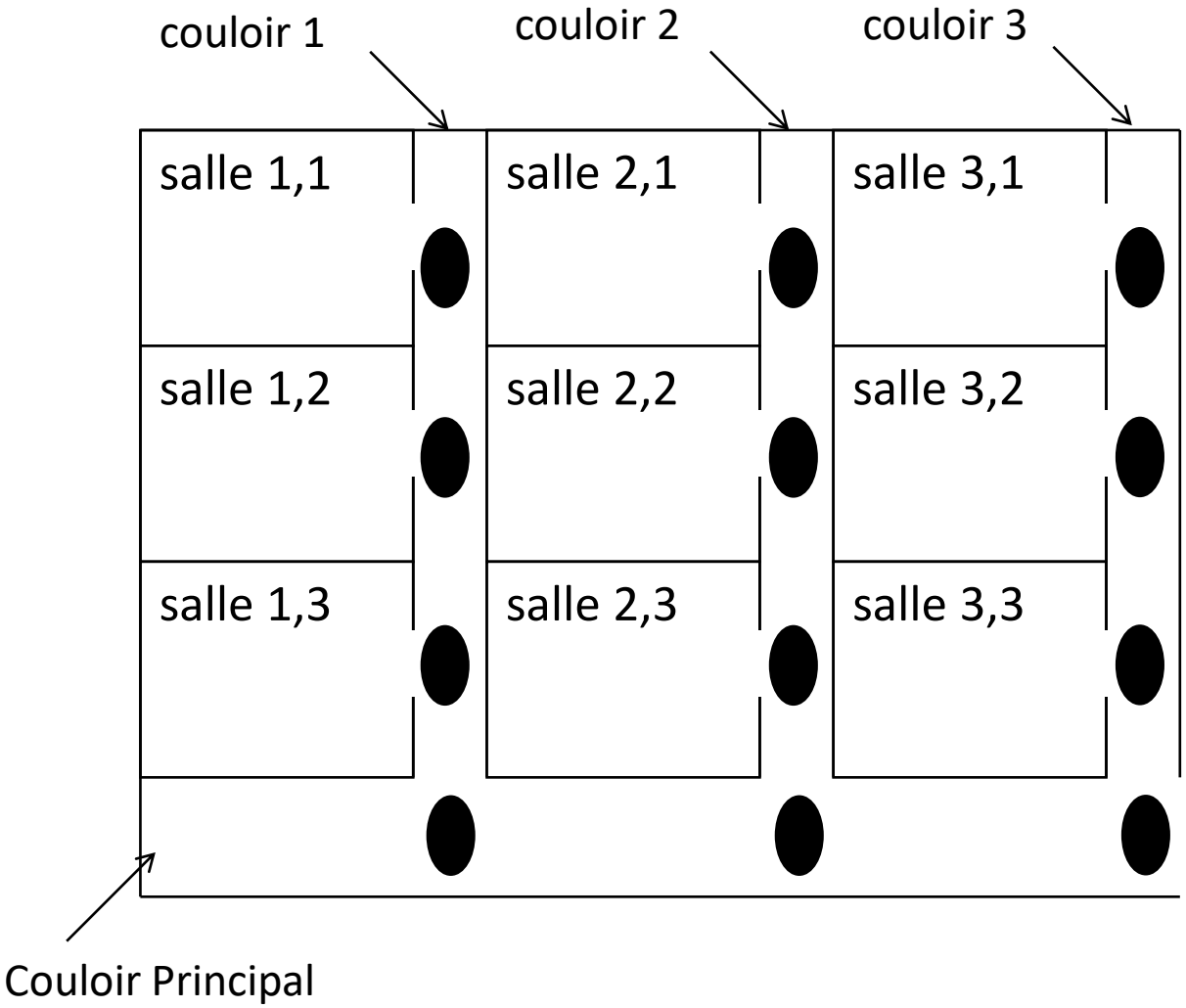
Etape 1: Les élèves de la classe sortent dans le couloir et restent groupés dans le couloir



Agent responsable du comptage des individus sortant de l'établissement

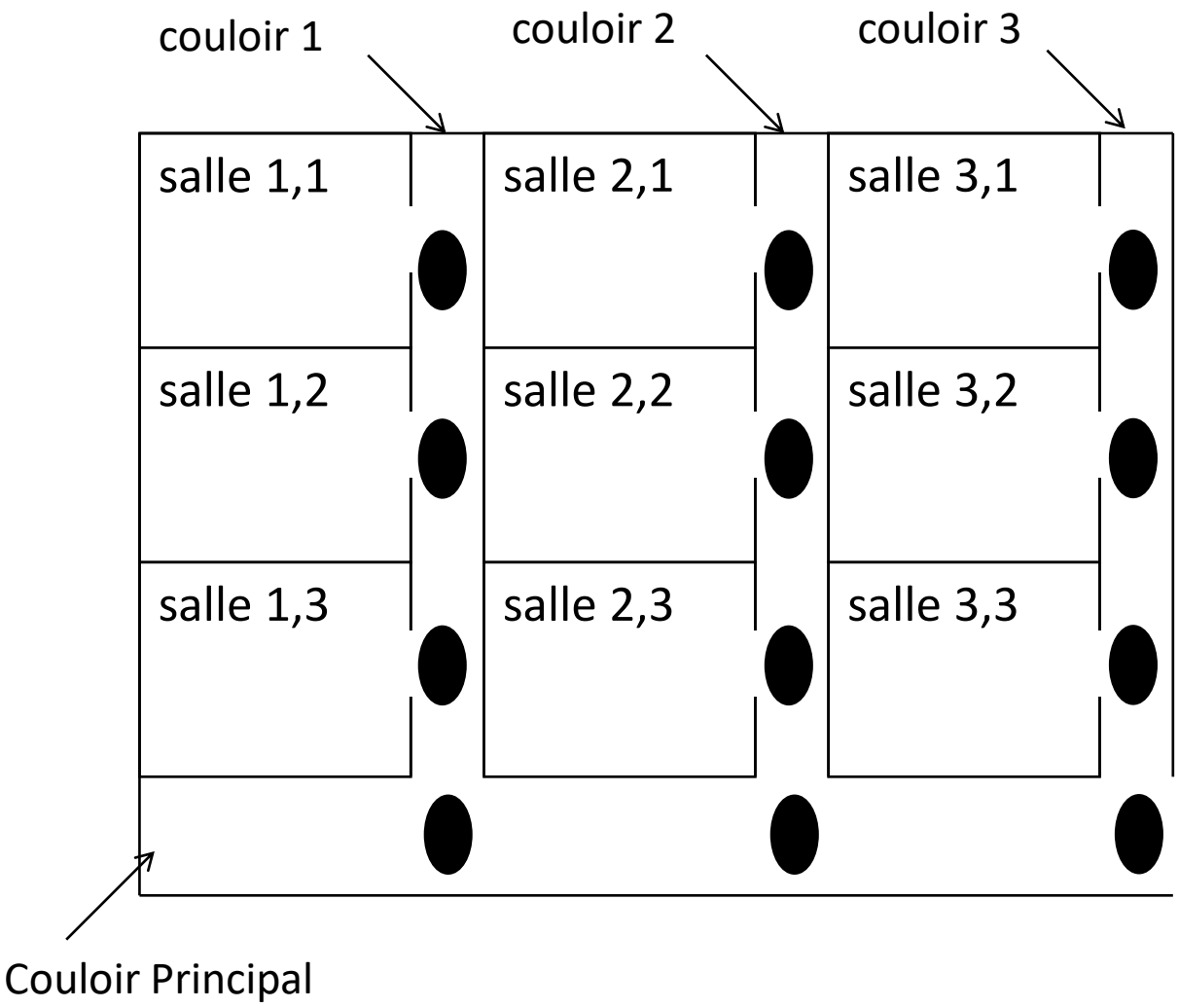
Etape 2: On procède à l'évacuation des élèves dans un ordre bien défini, couloir par couloir.

Dans chaque couloir, les élèves des salles 3 de chaque couloir passent dans le couloir principal



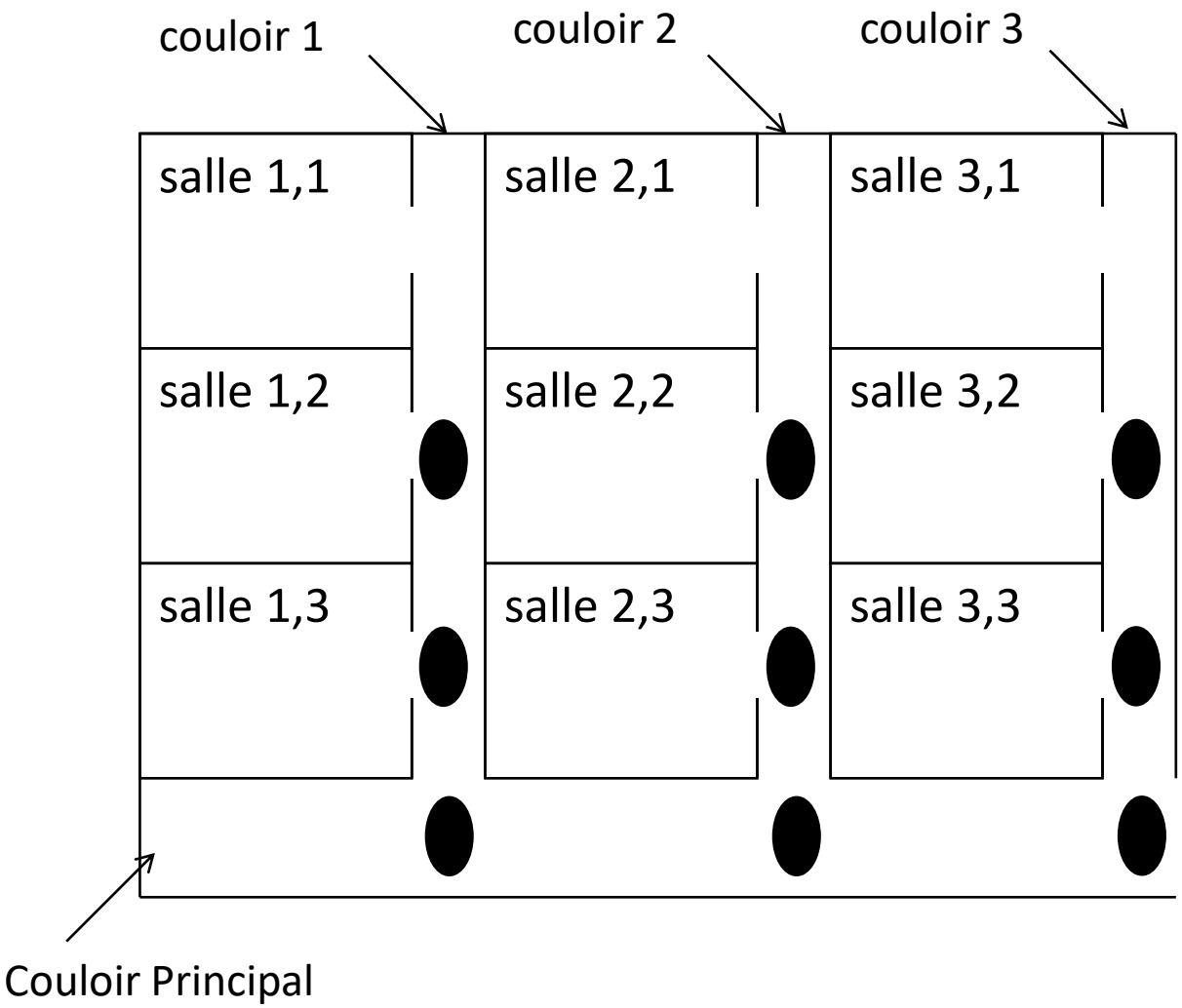
Agent responsable du comptage des individus sortant de l'établissement

Dans chaque couloir, les élèves des salles 2 occupent la place des élèves des salles 3, et les élèves des salles 1 occupent la place des élèves des salles 2.



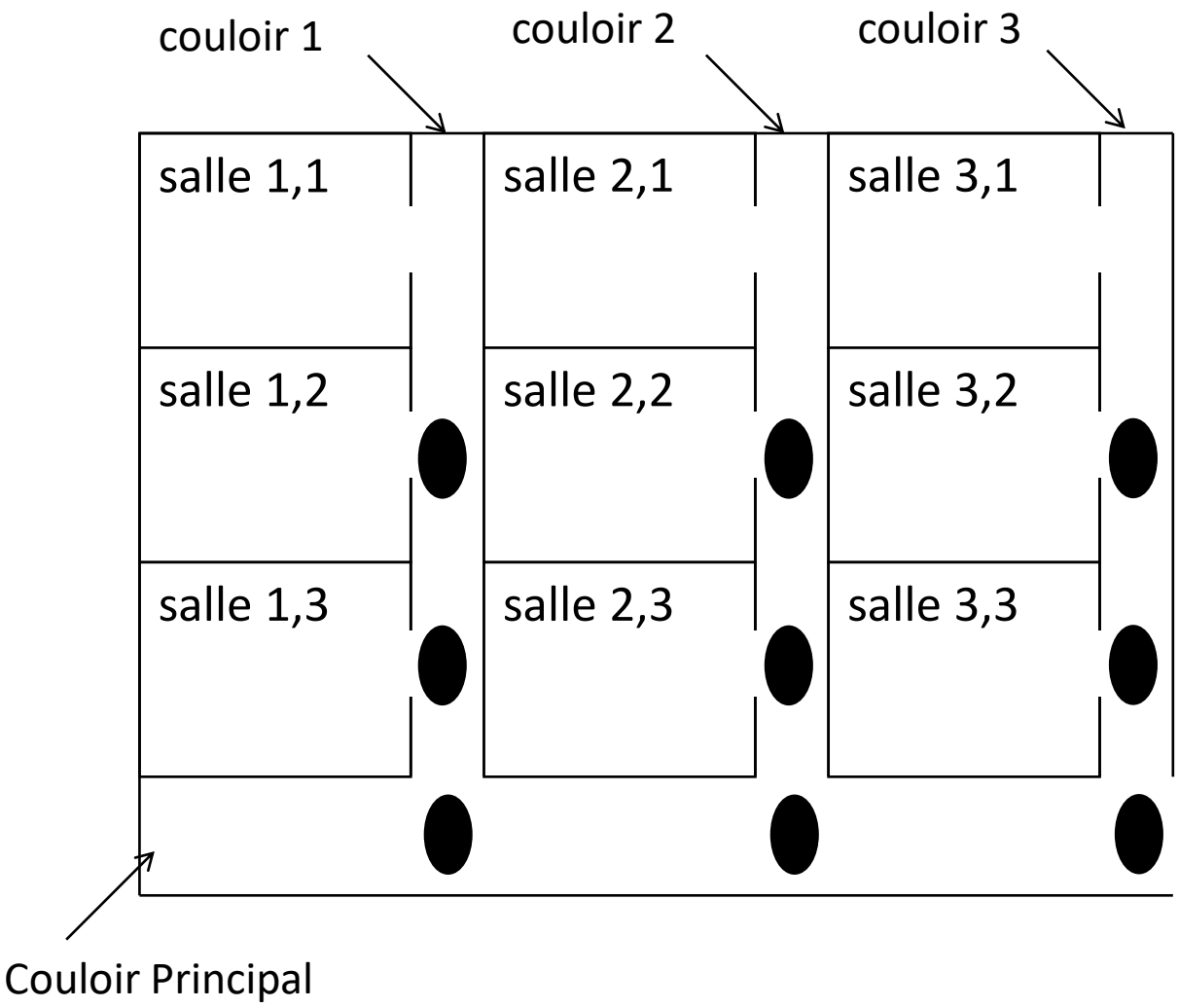
Agent responsable du comptage des individus sortant de l'établissement

Les élèves des salles 3 sont redirigés groupe par groupe, les vers la sortie d'évacuation où il sont comptabilisés dans un ordre bien défini:



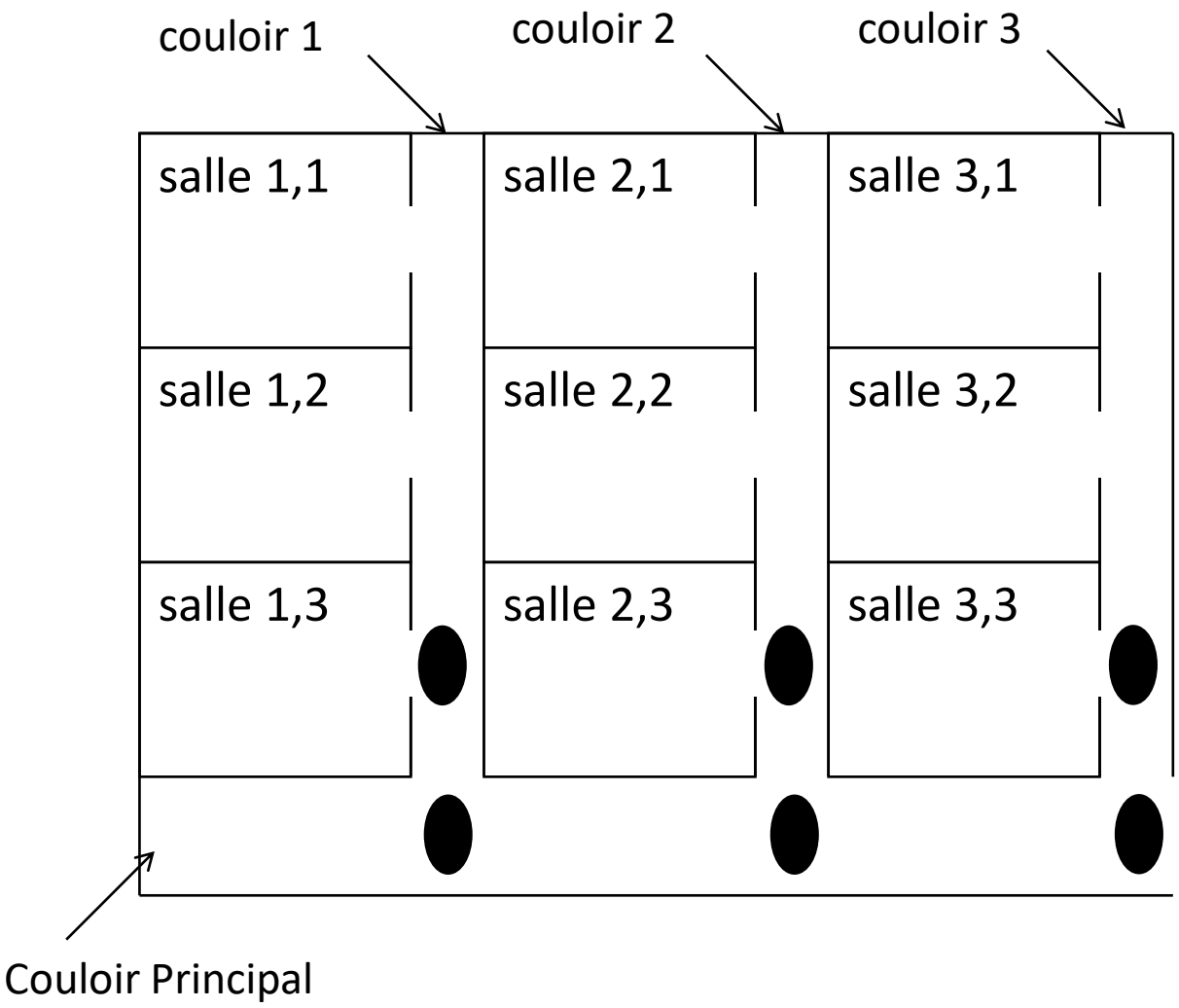
Agent responsable du comptage des individus sortant de l'établissement

Ainsi de suite …



Agent responsable du comptage des individus sortant de l'établissement

Ainsi de suite …



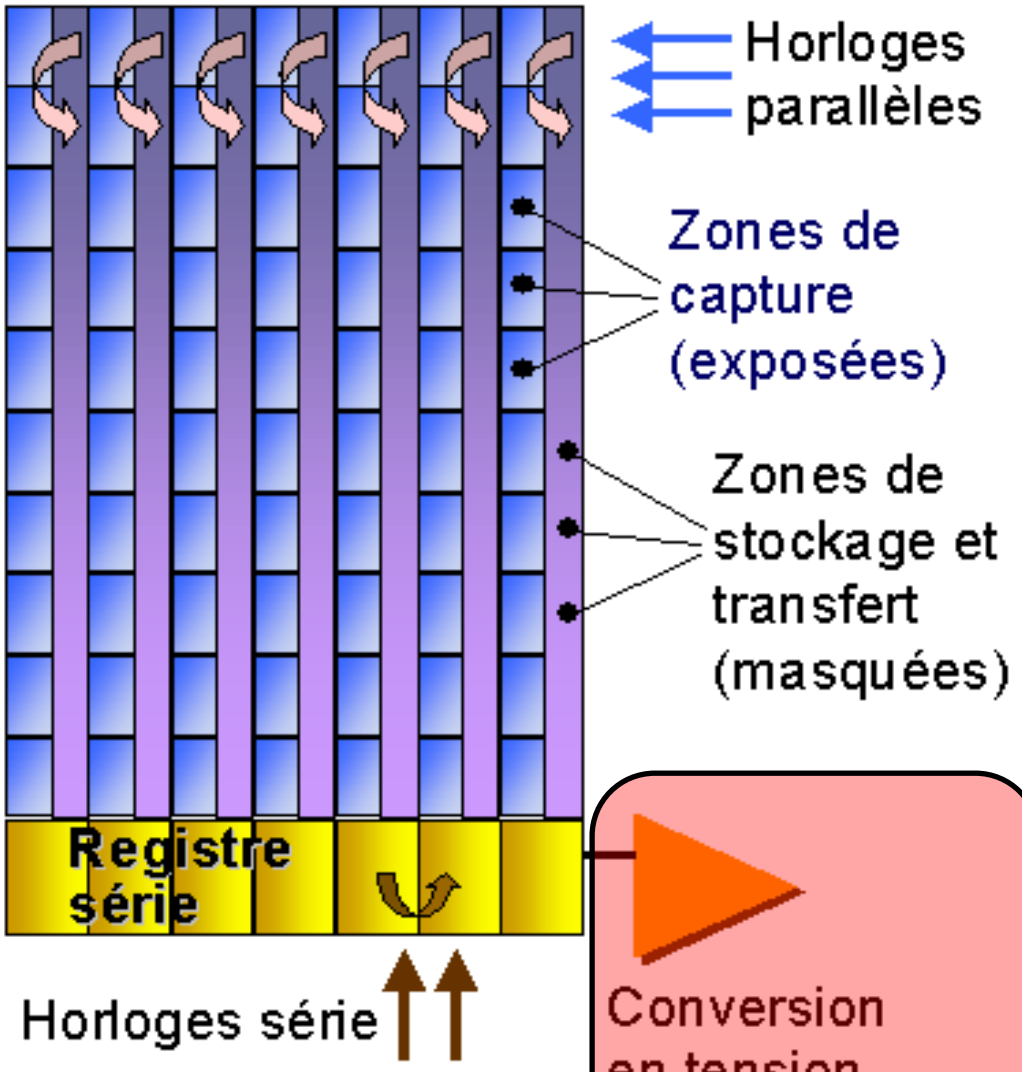
Agent responsable du comptage des individus sortant de l'établissement

- **Ce protocole d'évacuation permet de connaître à chaque instant le nombre d'élève qui se situe dans une classe donnée!!**
- **Généralisable à un nombre quelconque de couloirs et de salles…**

Comptabilisation identique des charges créées sur chaque photosites…

Procédé qui permet de connaitre, l'intensité lumineuse reçue par chaque pixel après chaque exposition !!!

1.3. Quantification de l'image numérique et qualité de l'image



Enregistrement de la valeur sur un nombre de bits correspondant à un nombre discret et fini de niveaux de gris entre le **noir** et le **blanc**

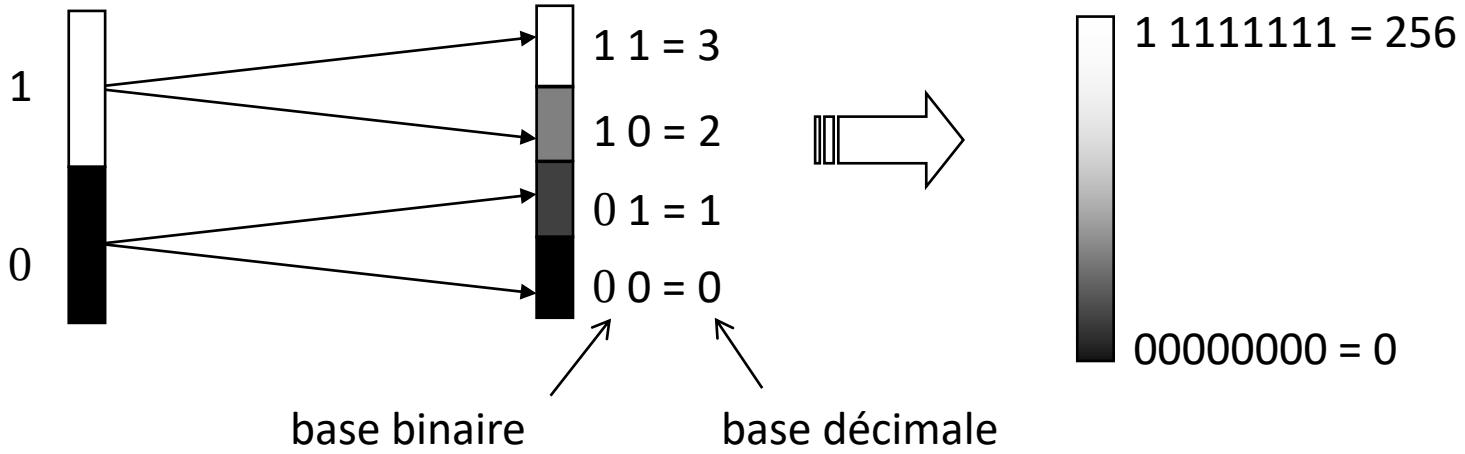
<u>Pour un pixel</u>: Tension de sortie proportionnelle à l'intensité lumineuse

Codage du niveau de gris et qualité d'image

Codage sur 1 bit :
2 nivaux de gris

Codage sur 2 bits :
4 nivaux de gris

Codage sur 8 bits = 1 octet :
256 niveaux de gris



Plus le nombre de bit utilisé est grand, plus le niveau de gris varie de façon continu.

Qualité de l'image d'autant meilleur que le nb de bit alloué par pixel est important !!

Inconvénient: place mémoire importante

Trouver un compromis entre qualité et place mémoire ….

Exemple:

1 bit / pixel :
2 nivaux de gris

2 bits/ pixel :
4 nivaux de gris

8 bits = 1 octet /pixel :
256 niveaux de gris

Remarque et Définition: Le nombre de bit utilisé pour coder le niveau de gris d'une image est appelé « profondeur de l'image ».

<u>1 bit / pixel</u> :

2 nivaux de gris

<u>2 bits/ pixel</u> :
4 nivaux de gris

<u>8 bits = 1 octet /pixel</u> :
256 niveaux de gris

1.4. représentation matricielle de l'image numérique

Classiquement : 1octet/pixel.

Valeur du niveau de gris entre 0 et 255.

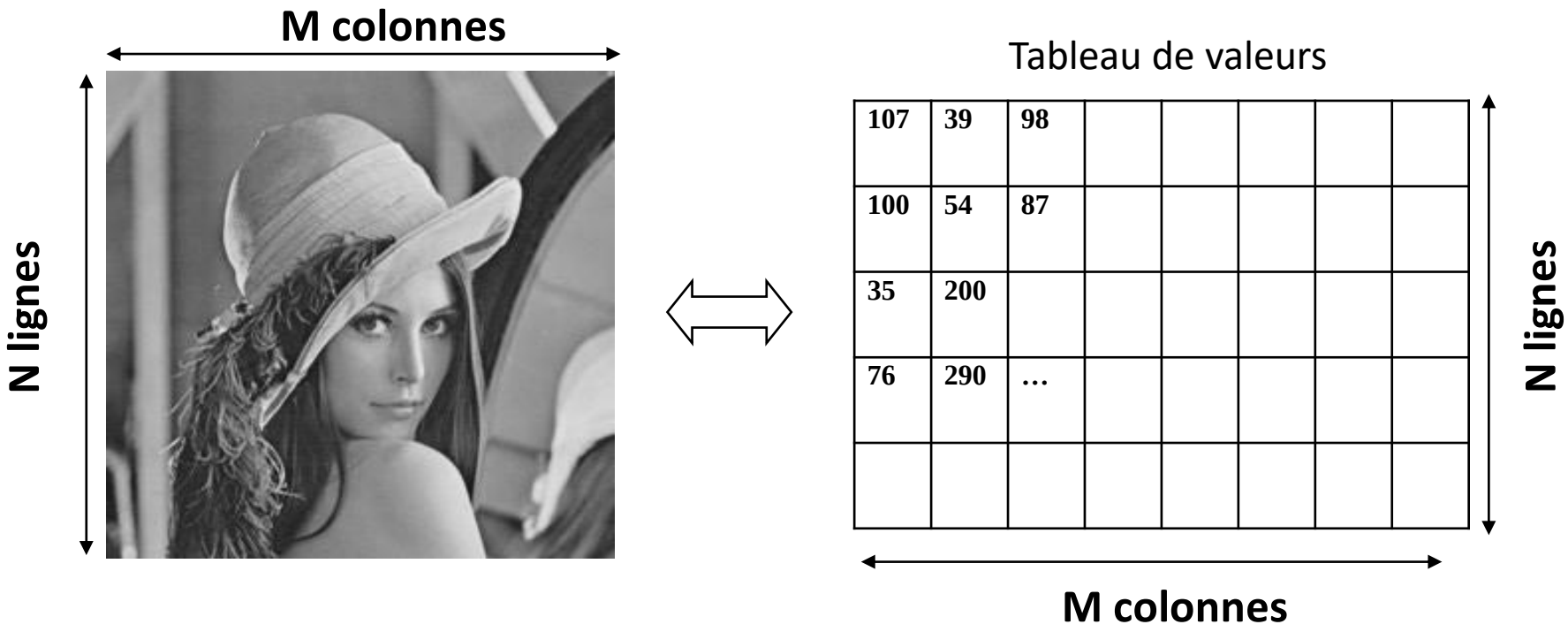
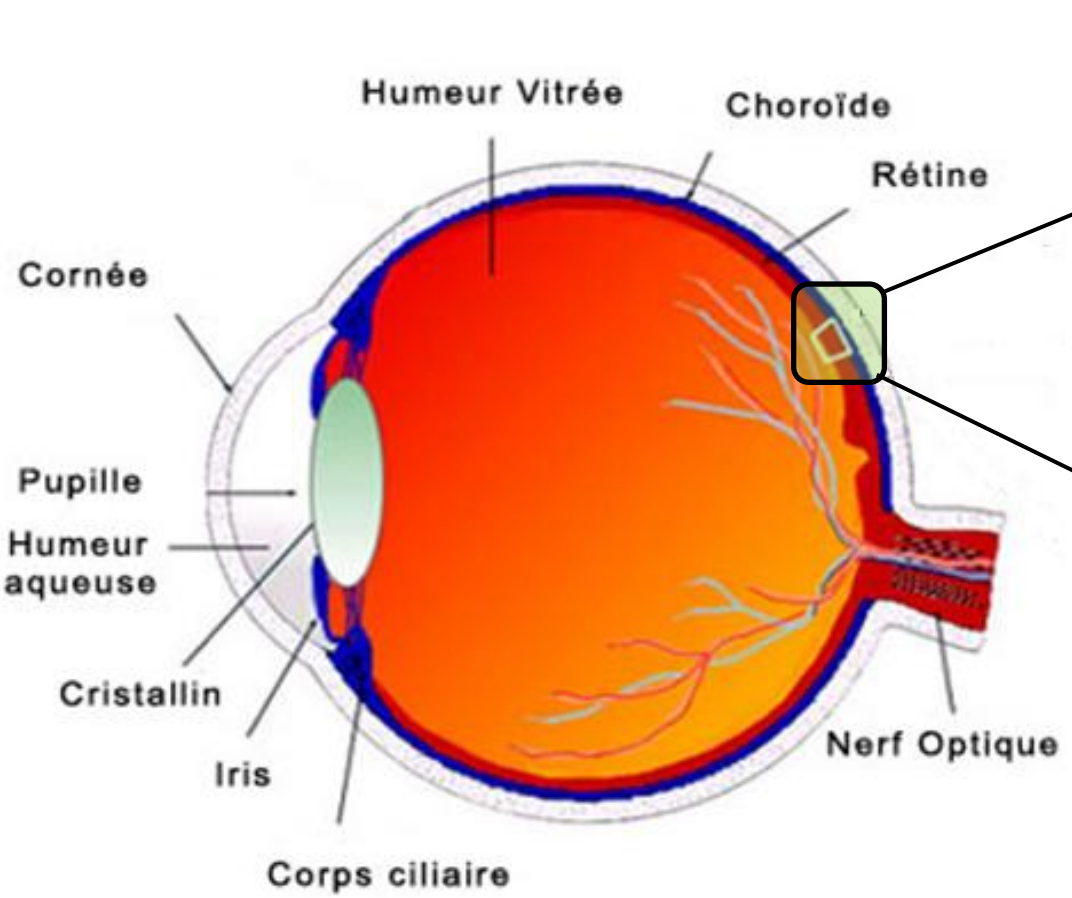


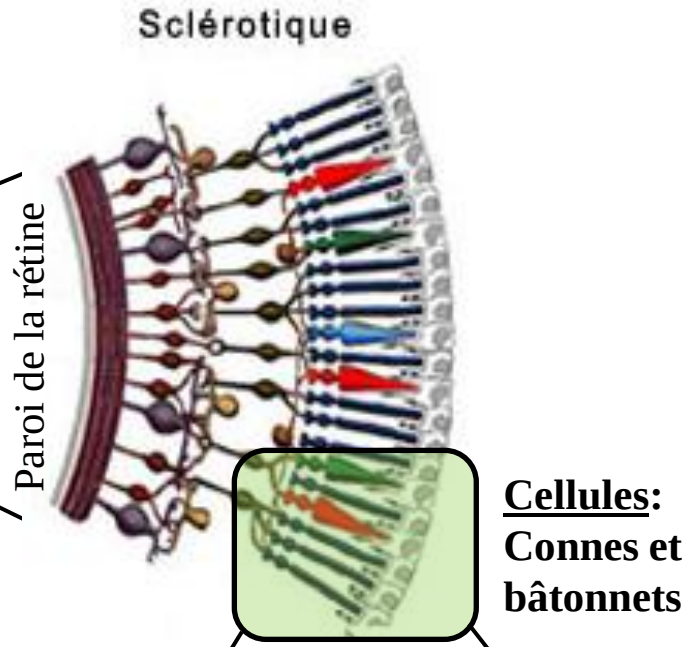
Image numérique = tableau de valeurs comprise entre 0 et 255

Exercice1: Calculer le poids (en octet) d'une image en niveau de gris issue d'un appareil numérique dont la CCD a pour dimension 1024×768 pixels.

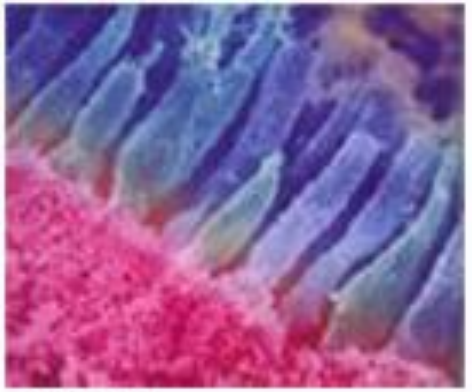
II. Images en couleurs

2.1. Perception des couleurs par l'œil humain





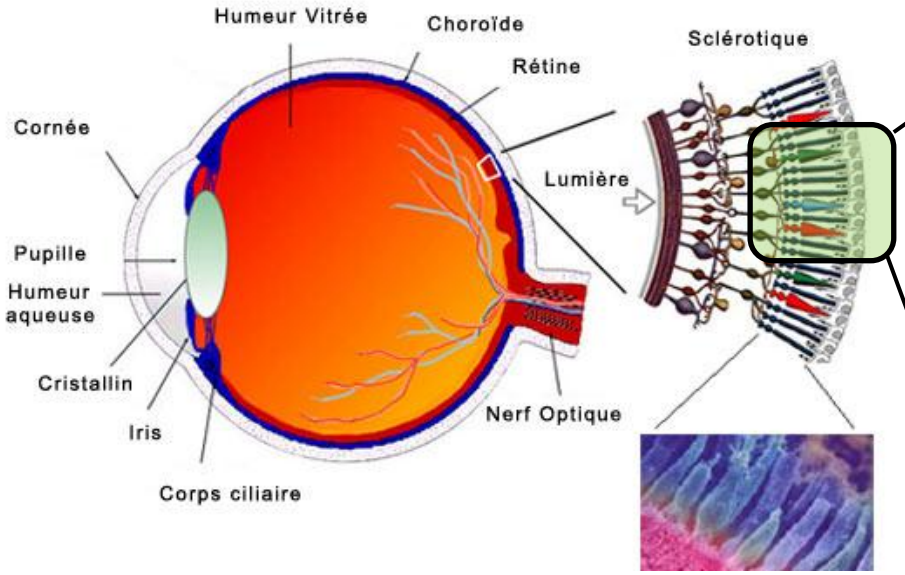
<u>Cellules</u>: Connes et bâtonnets

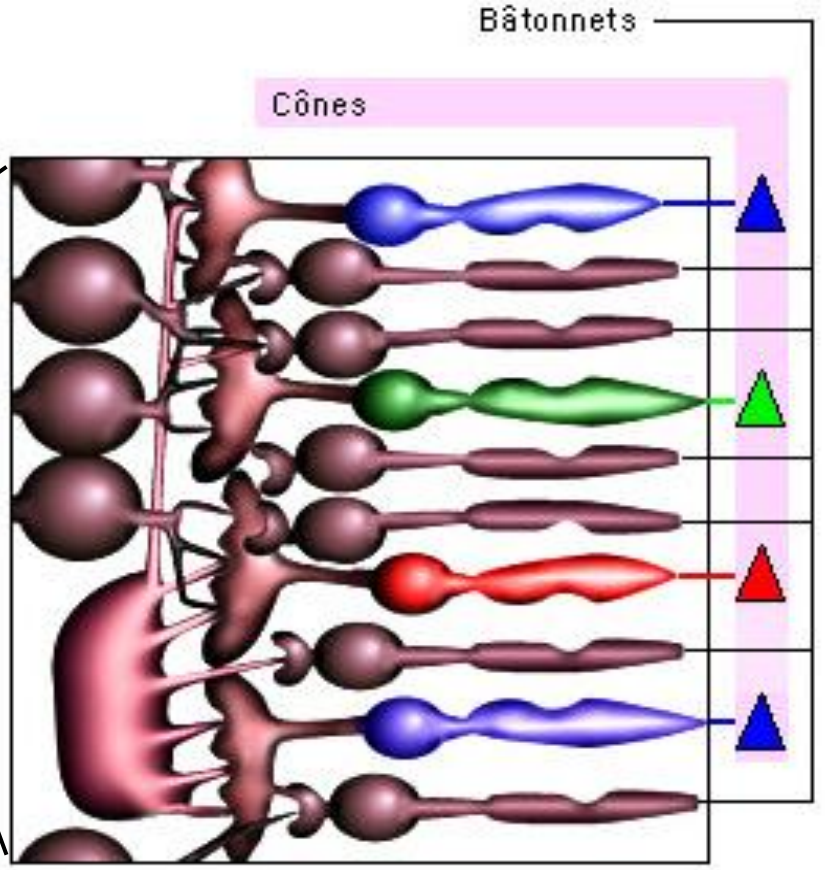


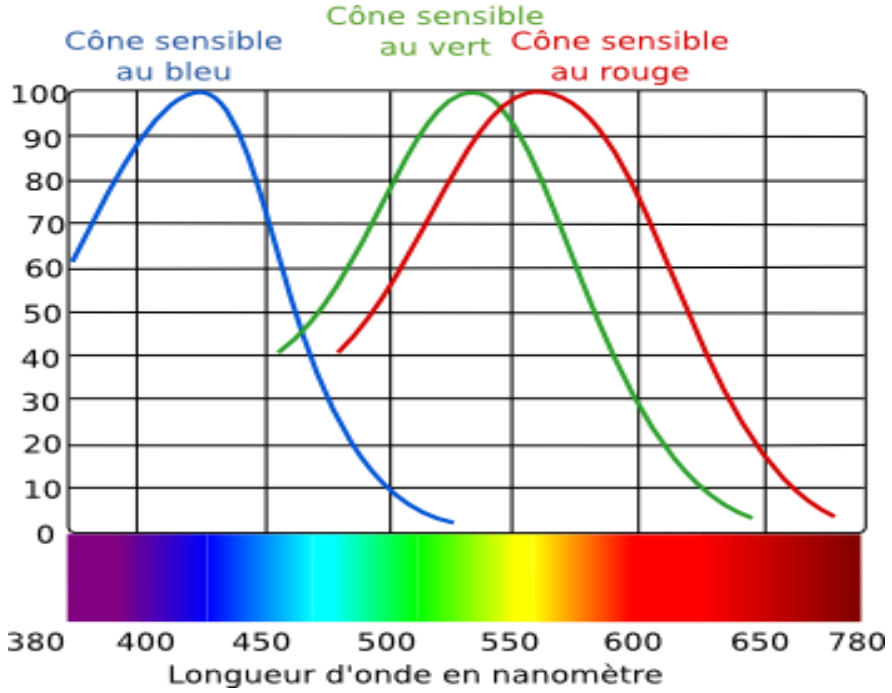
Détail : bâtonnets et cônes

La sclérotique est la partie blanche et opaque de l'oeil. Elle est constituée d'un tissu fibreux solide qui entoure le globe oculaire. La sclérotique contient de fins vaisseaux sanguins. Lorsque l'oeil est irrité par la poussière (ou au cours d'une maladie), les vaisseaux sanguins se dilatent et le blanc de l'oeil apparaît rosé ou injecté de sang.

Sensibilité chromatique des connes





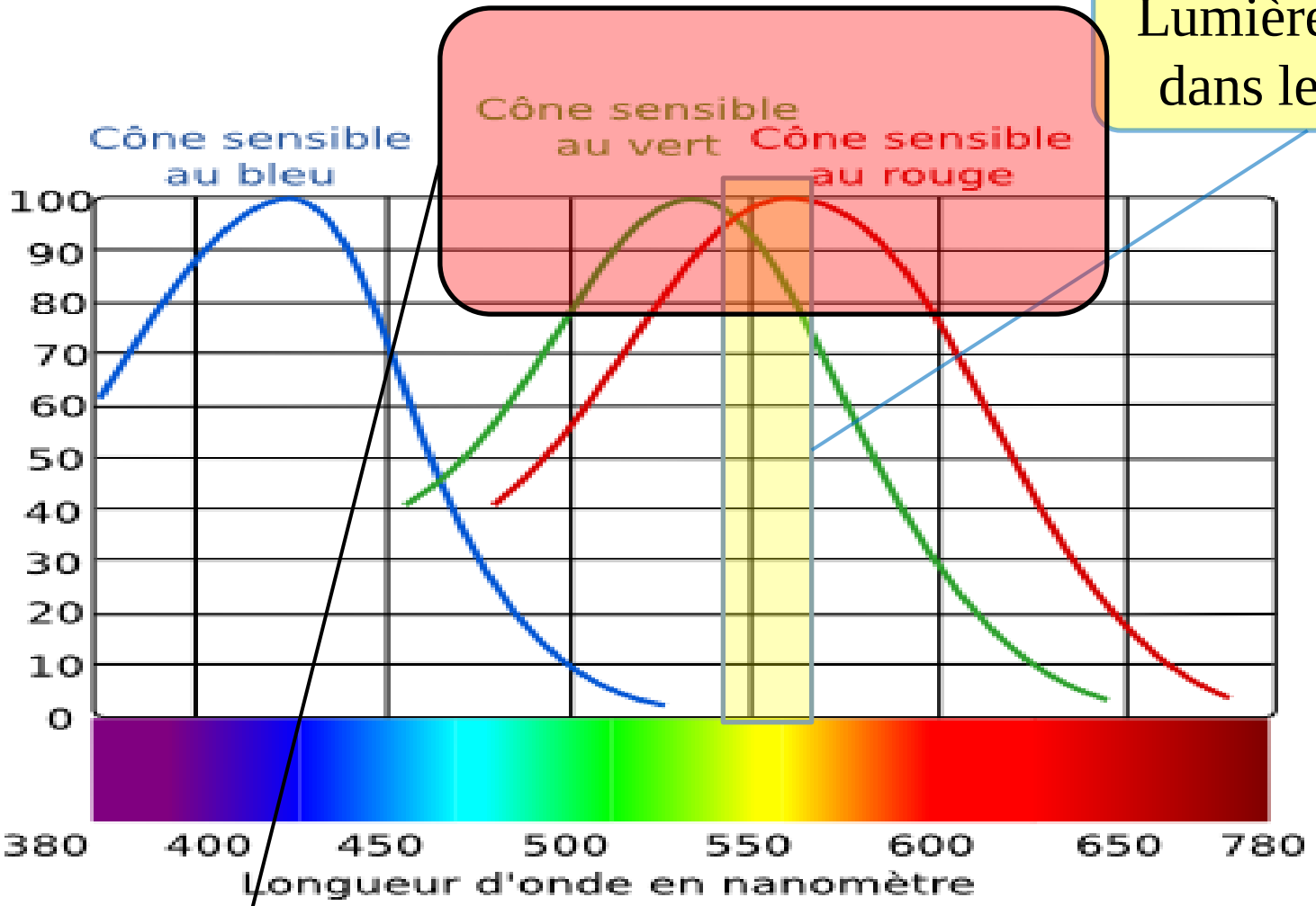


<u>3 sortes de connes</u> dont le maximum de sensibilité est situé dans :

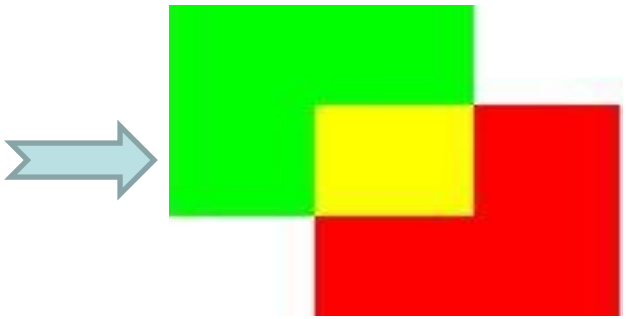
- **le rouge (560nm),**
- **le vert (530nm),**
- **le bleu (424nm).**

Perception des couleurs

Lumière émise dans le jaune



Cellules sensibles au vert et au rouge les plus sensibles (pas au bleu) !!

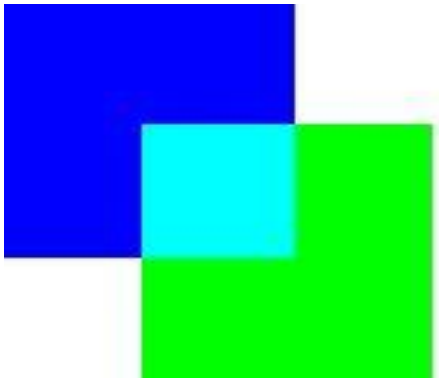


Sensation de couleur jaune identique à celle ressentie par un mélange de de vert et de rouge!

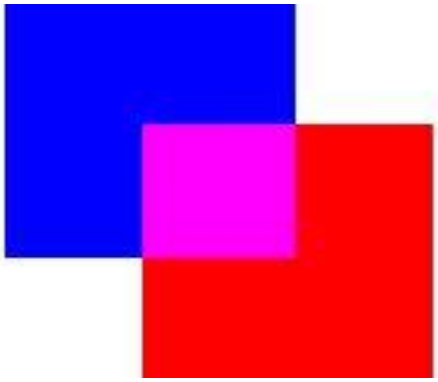
- **Quelle que soit la longueur d'onde reçue, le cerveau reçoit <u>uniquement</u> les intensités du signal reçu par les cellules dans le rouge, le vert et le bleu.**

- **<u>Toutes les teintes</u> peuvent être reconstituées par une combinaison des signaux rouge, vert et bleu.**

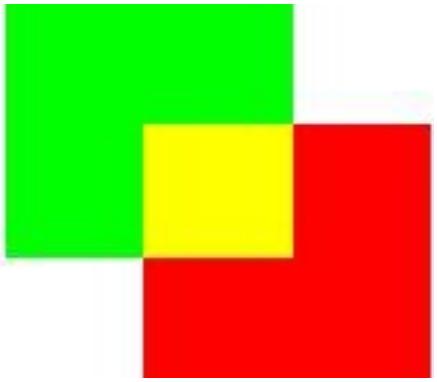
<u>Exemple:</u>



Vert + Bleu → Cyan



Rouge + Bleu → Magenta



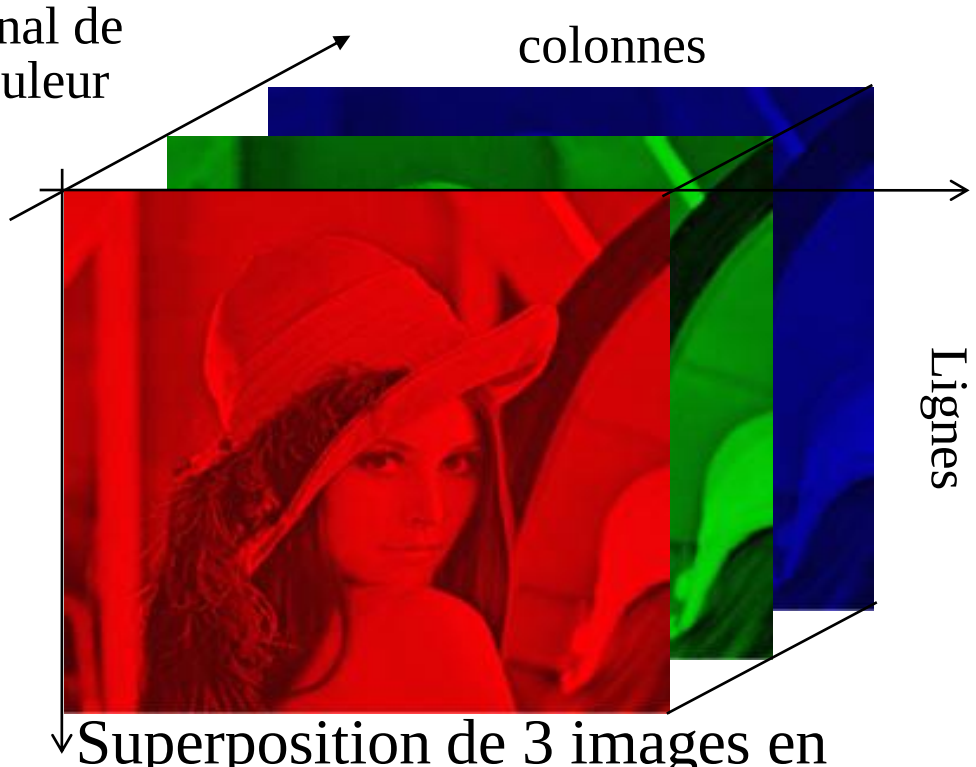
Vert + Rouge → Magenta

Grand intérêt technique!!!

Seulement 3 canaux RVB pour reconstituer la couleur!!!

2.2. Représentation d'une image en couleur

Image en couleur



Superposition de 3 images en niveaux de gris

Modélisation : **Tableau à trois entrées (tenseur d'ordre 3) :**

- Lignes:i
- Colonnes : j
- Couleurs : k

I(i,j,k)

$0 < i < N$

$0 < j < M$

$0 < k < 3$

Exercice: Généralement, dans une image en couleur, comme pour les images en niveaux de gris, **l'intensité des pixels de chaque canal de couleurs est codé sur un octet.**

1) Sur l'image en couleur, combien d'octet attribue-on à chaque pixel ?

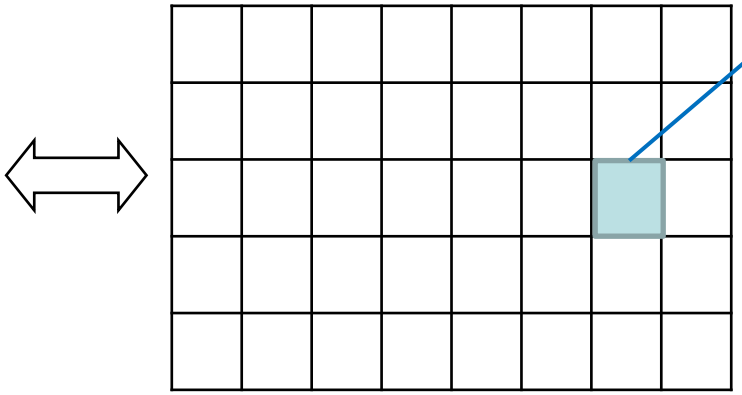
2) Calculer le poids (en octet) d'une image en couleur issue d'un appareil numérique dont la CCD a pour dimension 1024×768 pixels.

8. LES COULEURS SOUS PROCESSING

RAPPEL

Image en niveau de gris

Tableau de valeurs



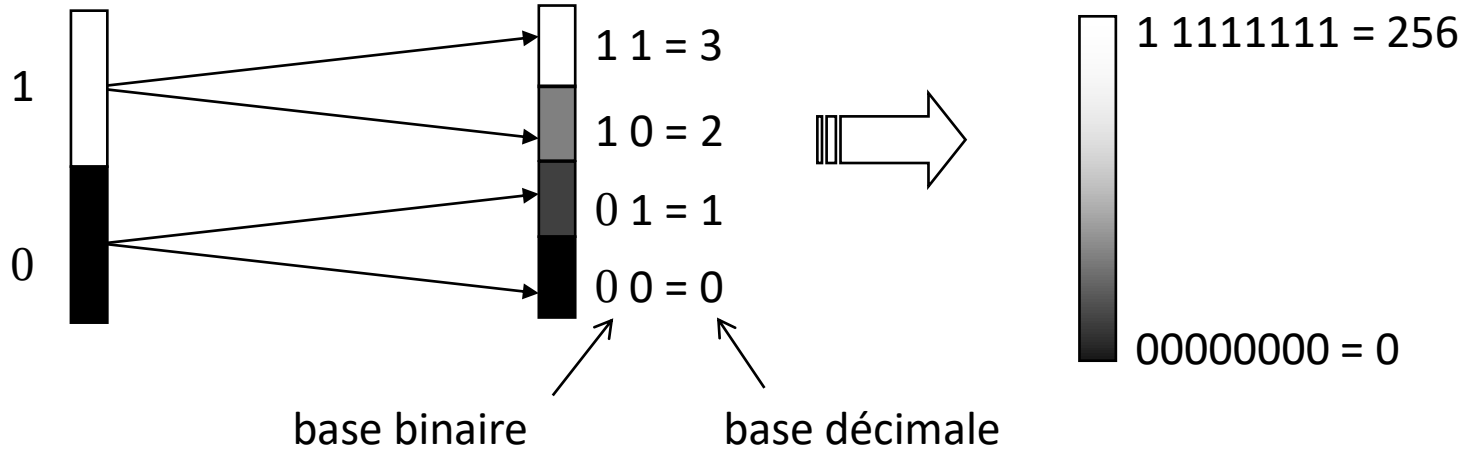
Case = Pixel (picture element)

Défini par un niveau de gris entre le noir le le blanc

Codage sur 1 bit :
2 nivaux de gris

Codage sur 2 bits :
4 nivaux de gris

Codage sur 8 bits = 1 octet :
256 niveaux de gris



Qualité de l'image d'autant meilleur que le nb de bit alloué par pixel est important

Exemple:

1 bit / pixel :
2 nivaux de gris

2 bits/ pixel :
4 nivaux de gris

8 bits = 1 octet /pixel :
256 niveaux de gris

<u>1 bit / pixel</u> :

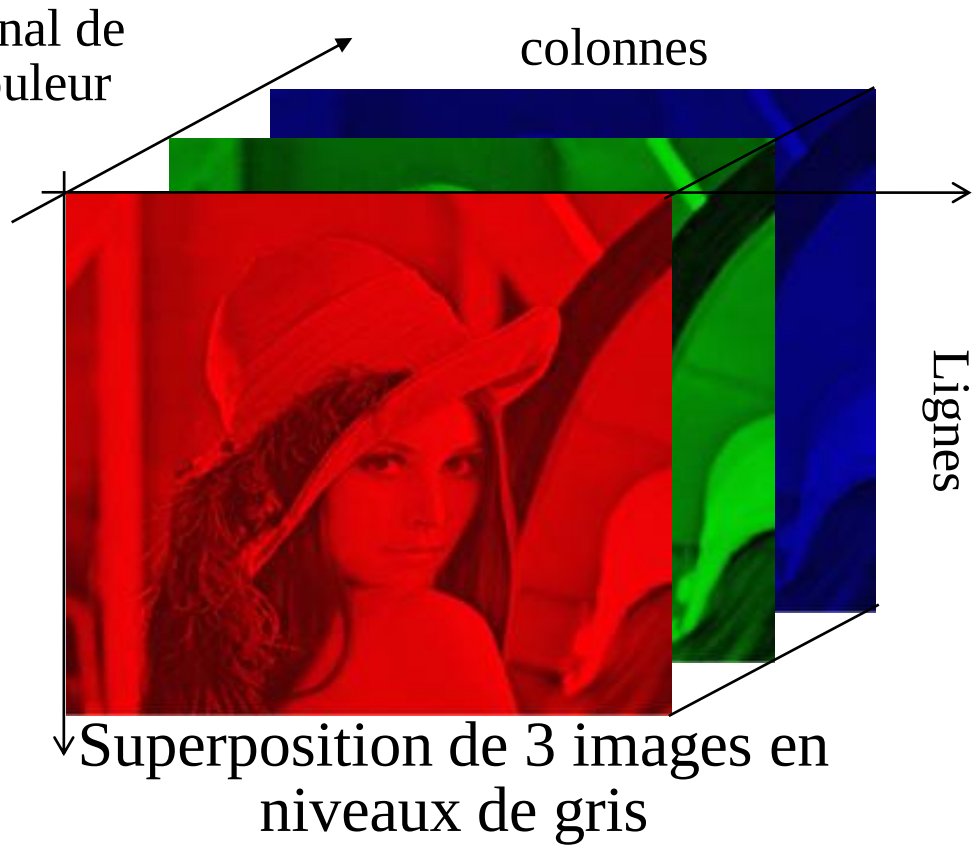
2 nivaux de gris

<u>2 bits/ pixel</u> :
4 nivaux de gris

<u>8 bits = 1 octet /pixel</u> :
256 niveaux de gris

Image en couleur

Image en couleur



Superposition de 3 images en niveaux de gris

Modélisation : Tableau à trois entrées:

- Lignes:i
- Colonnes : j
- Couleurs : k

I(i,j,k)

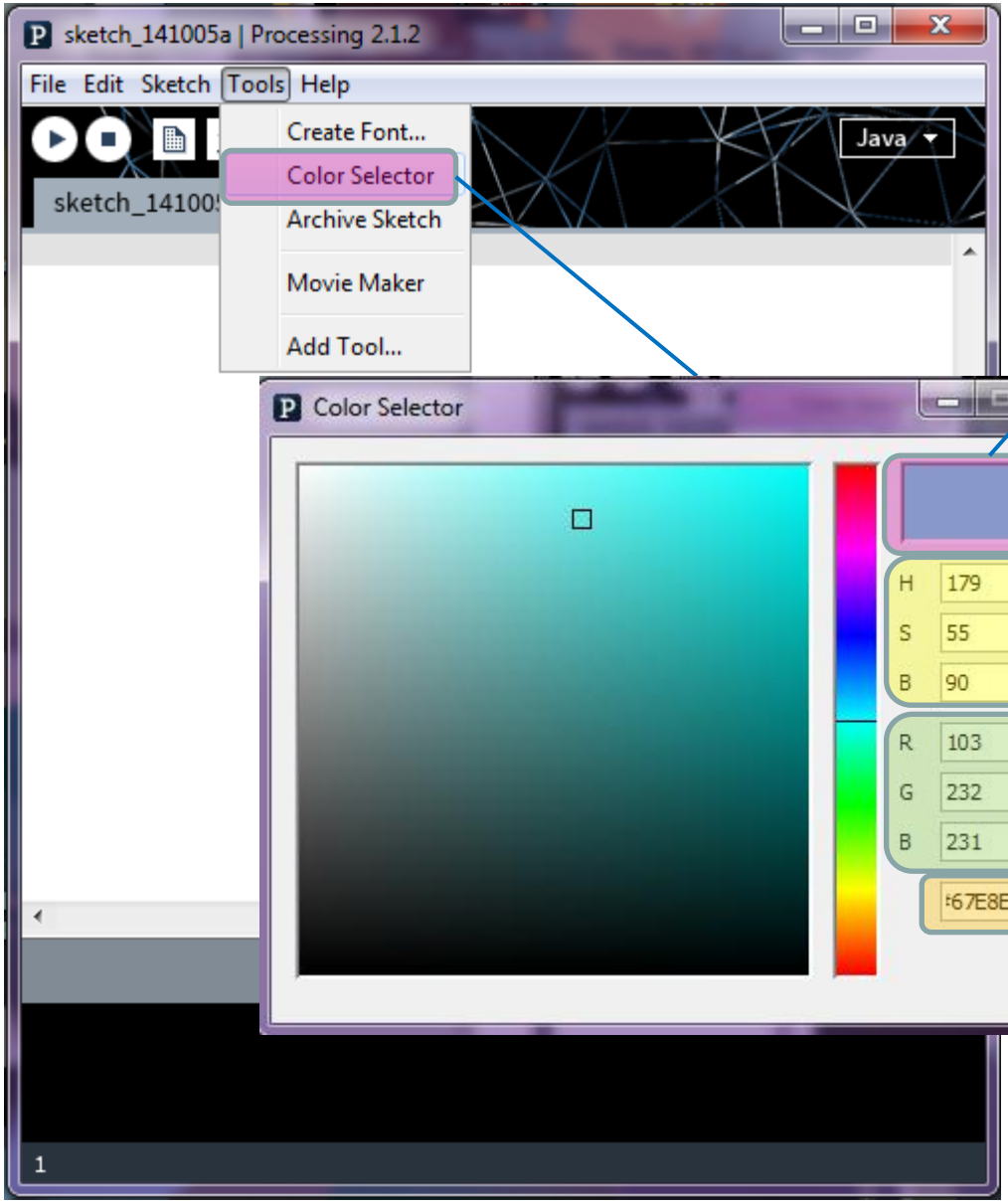
$0 < i < N$

$0 < j < M$

$0 < k < 3$

3 octets / pixels!!

8.0 Un outil majeur : « color selector »



Représentation de la couleur

Teinte/Saturation/Valeur

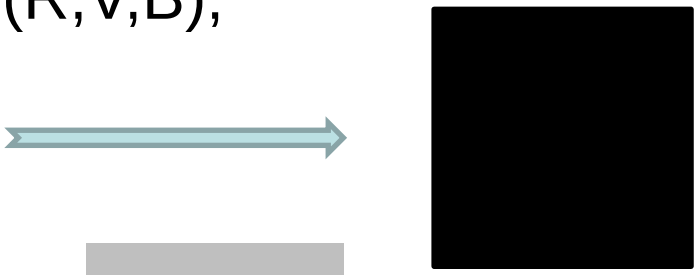
Rouge/Vert/Bleu

Format Hexadécimal

8.1 Couleur de fond

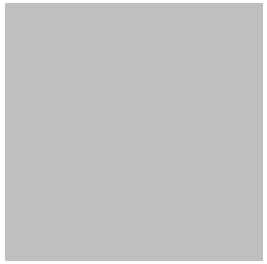
background(R,V,B);

Fond noir: background(0,0,0);



Par défaut la couleur du fond est grise:

background(204, 204, 204);



Rajouter background() à la suite d'une composition déjà existante, l'effacerait !

8.2 Couleur de remplissage d'une forme

fill(R,V,B);

fill(R,V,B); se place avant la description de la forme

Exemple:

```
noStroke();
fill(255, 204, 51);
rect(25, 25, 50, 50);
```

- Format de couleur hexadécimal (WEB):

```
fill(#ffcc33);
rect(25, 25, 50, 50);
```

- Degrés de transparence de la forme (canal alpha) :

```
noStroke();
fill(255, 204, 51); // orange
rect(25, 25, 50, 50);
fill(255, 255, 255, 127); // blanc semi-transparent
rect(35, 35, 50, 50);
```

- Remplissage en niveau de gris: un seul paramètre …

```
fill(127);
rect(25, 25, 50, 50);
```

8.3 Couleur du contour d'une forme

- Dessiner le contour d'une forme: stroke(R,V,B); ou stroke(NG);
- Pas de contour: noStroke();

8.4 Portée des modification de couleurs

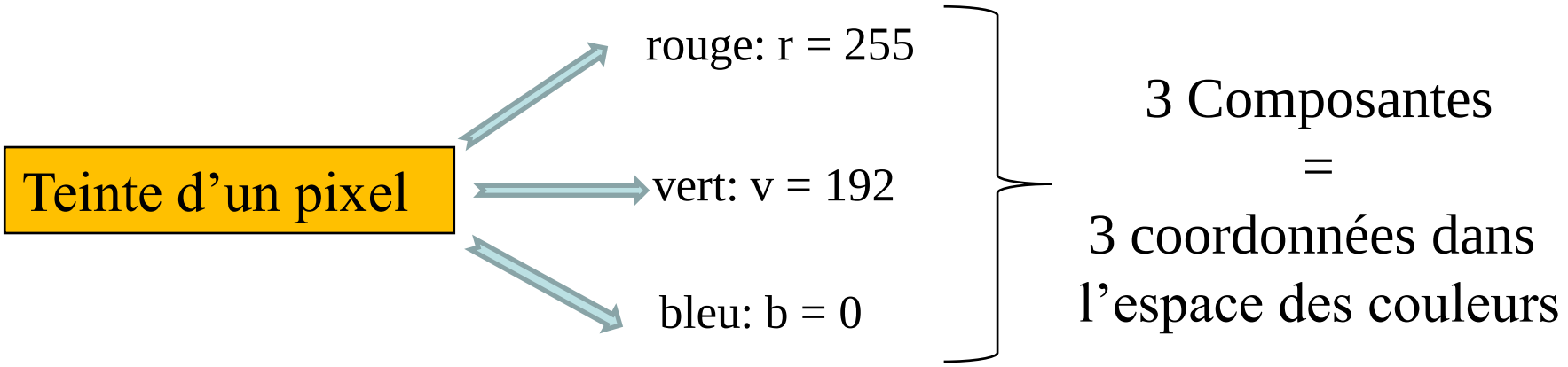
Par défaut, toute modification de style (couleur de remplissage ou de contour, épaisseur ou forme de trait) s'applique à tout ce que vous dessinez ensuite.

- On limite la portée des modifications en encapsulant les lignes de commandes par : pushStyle() et popStyle()

```
size(100, 100);
background(255);
stroke(#000000);
fill(#FFFF00);
strokeWeight(1);
rect(10, 10, 10, 10);
pushStyle(); // On ouvre « une parenthèse » de style
stroke(#FF0000);
fill(#00FF00);
strokeWeight(5);
rect(40, 40, 20, 20);
popStyle(); // On ferme notre parenthèse de style
rect(80, 80, 10, 10);
```

```
size(100, 100);
background(255);
stroke(#000000);
fill(#FFFF00);
strokeWeight(1);
rect(10, 10, 10, 10);
// pushStyle(); // On ouvre « une parenthèse » de style
stroke(#FF0000);
fill(#00FF00);
strokeWeight(5);
rect(40, 40, 20, 20);
// popStyle(); // On ferme notre parenthèse de style
rect(80, 80, 10, 10);
```

8.5 Espace colorimétrique



L'espace (vectoriel) des couleurs est de dimension 3 !!!

Espace 3D des couleurs:

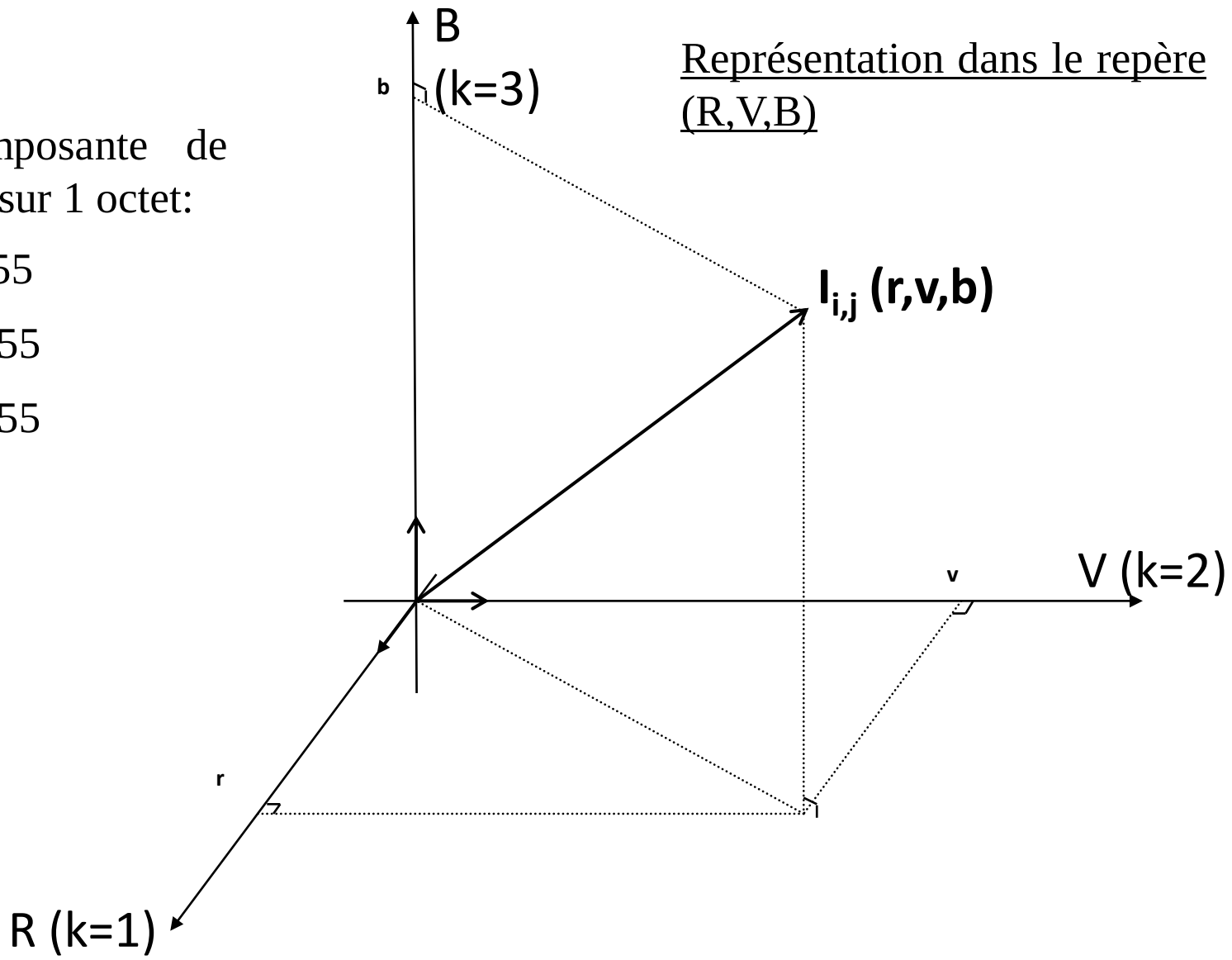
Pour un pixel positionné spatialement en (i,j) dans l'image:

Si chaque composante de couleur est codé sur 1 octet:

$0 \leq r \leq 255$

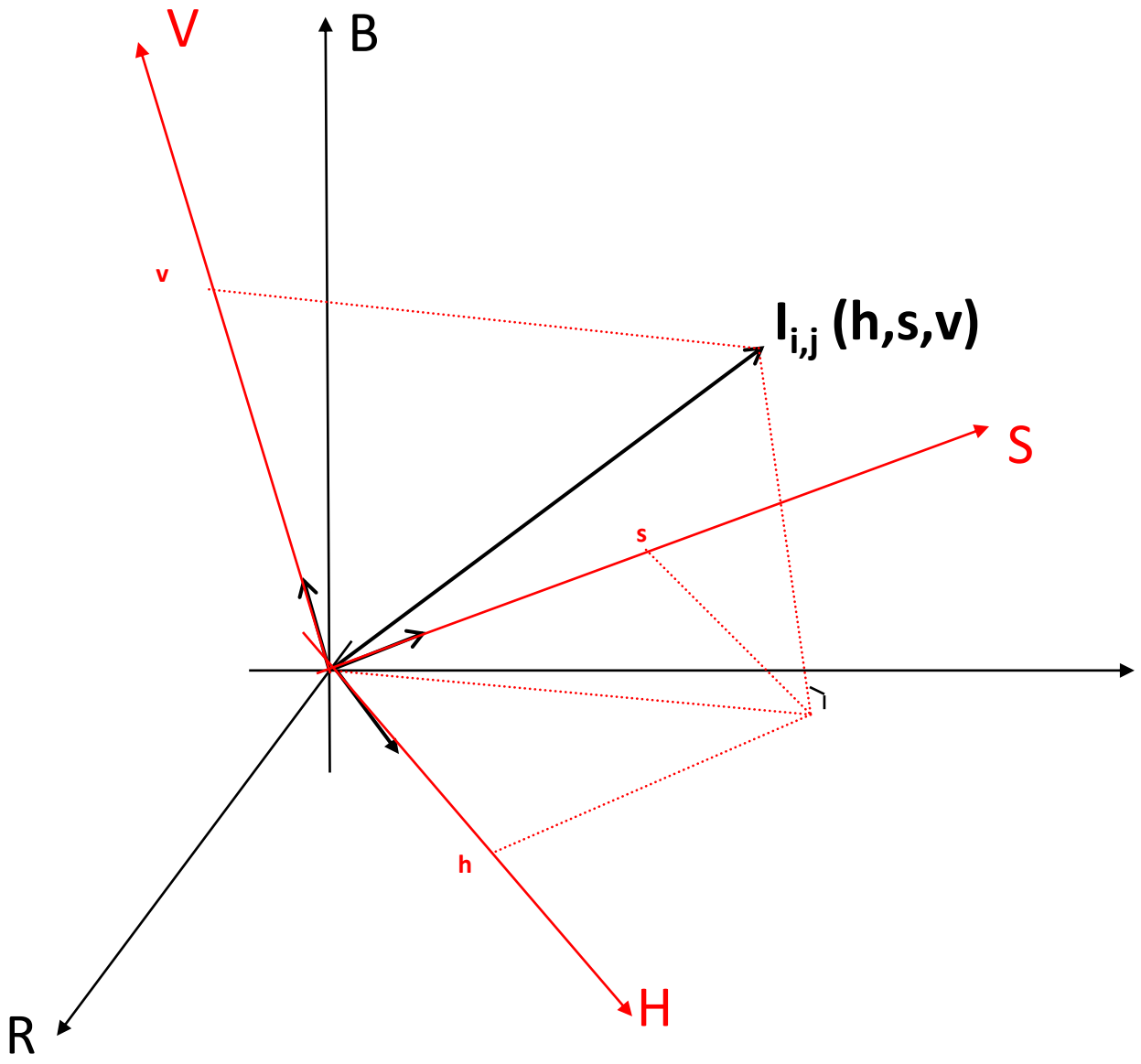
$0 \leq v \leq 255$

$0 \leq b \leq 255$



Représentation dans le repère (HSV) :

On change de repère dans l'espace des couleurs RVB → HSV

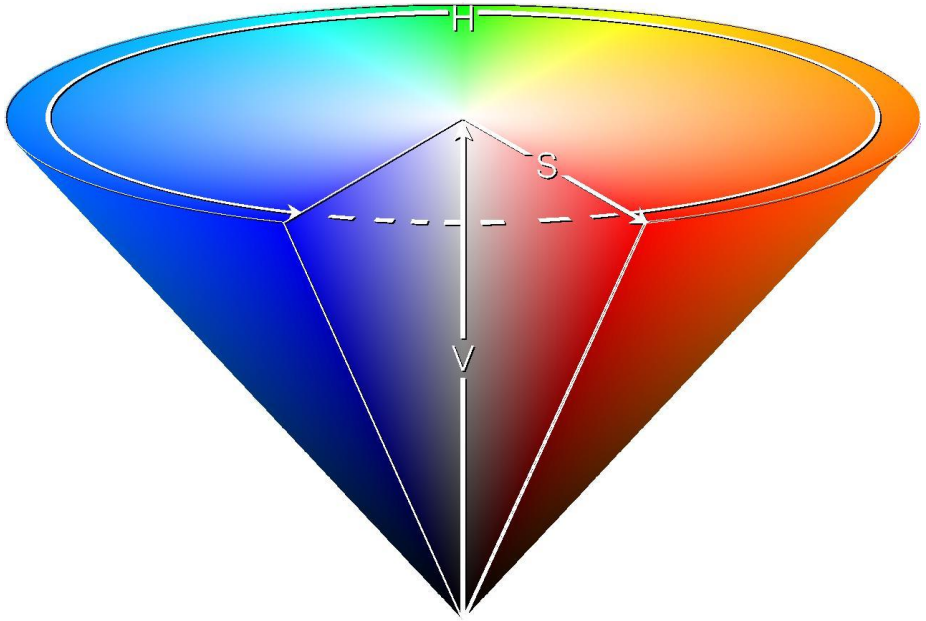


- **La teinte** : le type de couleur (comme rouge, bleu, jaune…). Sa valeur varie entre 0 et 360° ou parfois normalisée en 0–100 %.

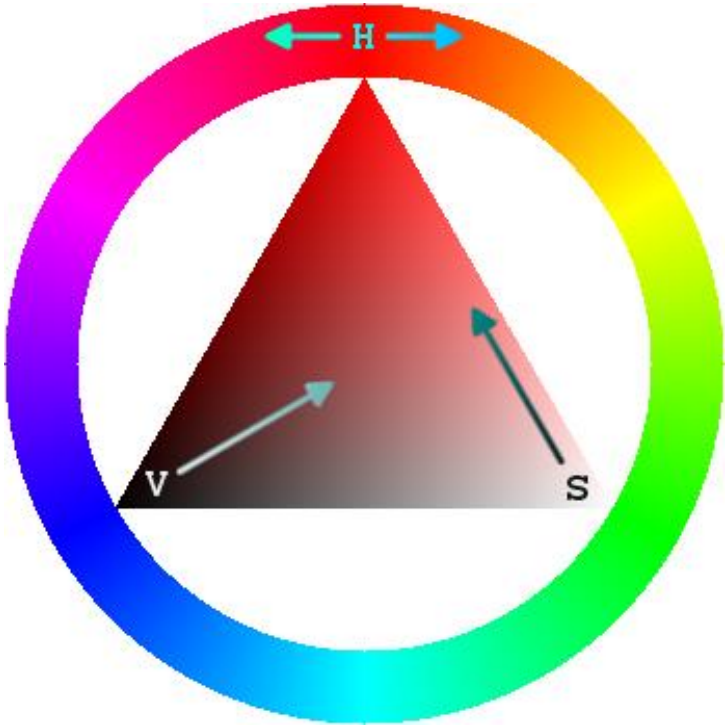
- **La saturation**: l'« intensité » de la couleur qui varie entre 0 et 100 % est parfois appelé « pureté ».
  => plus la saturation d'une couleur est faible, plus l'image sera « grisée » et plus elle apparaitra fade.

- **valeur :** la « brillance » de la couleur : elle varie entre 0 et 100%.

Visualisation pratique du repère (HSV) :



Représentation conique bien adaptée pour représenter tout l'espace HSV en un seul objet.



Roue des couleurs HSV permet à l'utilisateur de rapidement sélectionner une multitude de couleurs.

Conversion d'une image RVB vers HSV

(r, g, b) —transformation→ (h,s,v)

(Combinaison des coordonnées)

$$h = \begin{cases} 0, & \text{si } MAX = MIN \\ 60 \times \frac{g-b}{MAX-MIN} + 0°, & \text{si } MAX = r \text{ et } g \geq b \\ 60 \times \frac{g-b}{MAX-MIN} + 360°, & \text{si } MAX = r \text{ et } g < b \\ 60 \times \frac{b-r}{MAX-MIN} + 120°, & \text{si } MAX = g \\ 60 \times \frac{r-g}{MAX-MIN} + 240°, & \text{si } MAX = b \end{cases}$$

Avec :

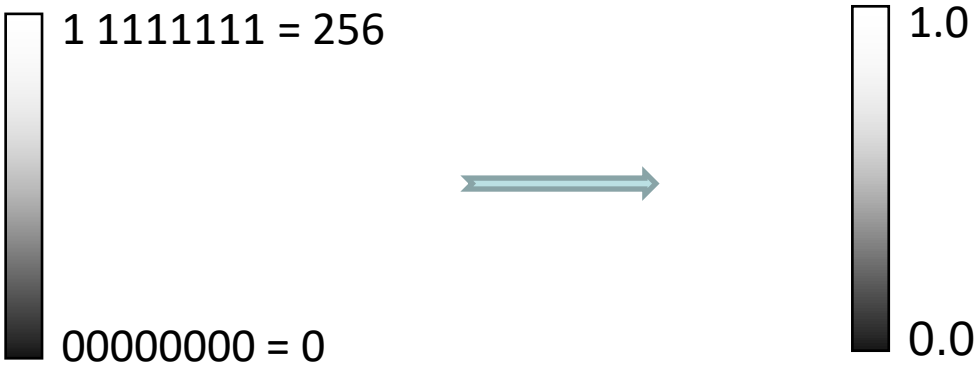
- *MAX* égale au maximum des valeurs (*r*, *g*, *b*),
- *MIN* égal au minimum de ces valeurs

$$s = \begin{cases} 0, & \text{si } MAX = 0 \\ 1 - \frac{MIN}{MAX}, & \text{sinon} \end{cases}$$

$$v = MAX$$

Changer d'espace colorimétrique sous Processing: colorMode();

Changer d'échelle numérique RGB: colorMode(RGB,1.0)



Changer d'espace colorimétrique:

colorMode(HSB,valeurMaxH,valeurMaxS, valeurMaxB)

Exemple:

```
noStroke();
size(400, 128);
// La teinte sera spécifiée avec un chiffre de 0 à 400
colorMode(HSB, 400, 100, 100);
// On effectue quatre cent répétitions
for (int i = 0; i < 400; i++) {
fill(i, 128, 128);
rect(i, 0, 1, 128);
}
```

9. LES VARIABLES

- Les programmes manipulent un certain nombre de **variables**
  - **Données stockées dans l'espace mémoire réutilisable par la suite**

- Chaque variable est caractérisée par :
  - un **nom**
  - un **type**
  - une **place mémoire**

- Le nom:
  - Peut contenir des lettres,
  - chiffres,
  - caractères(-,_)

Les types et leur taille mémoire en JAVA:

Type	*description*	*taille*	*ensemble de valeurs*	*val. init.*
		Nombres entiers		
`byte`	Octet	8 bits	de -128 à 127	`0`
`short`	Entier court	16 bits	de $-32\,768$ à $32\,767$	
`int`	Entier	32 bits	de $-2\,147\,483\,648$ à $2\,147\,483\,647$	
`long`	Entier long	64 bits	de -2^{63} à $2^{63}-1$	
		Nombres flottants		
`float`	Flottant simple précision	32 bits	de $-3,4028235\times10^{+38}$ à $-1,4\times10^{-45}$, 0 et de $1,4\times10^{-45}$ à $3,4028235\times10^{+38}$	`0.0`
`double`	Flottant double précision	64 bits	de $-1,7976931348623157\times10^{+308}$ à $-4,9\times10^{-324}$, 0 et de $4,9\times10^{-324}$ à $1,7976931348623157\times10^{+308}$	
		Autres types		
`char`	Caractère	16 bits	Tous les caractères Unicode	`'\u0000'`
`boolean`	Valeur booléenne	1 bit	`false`, `true`	`false`

Les types de Processing:

int, float, double, boolean, char, string, color

9.1 Les nombres entiers de type « int »

⇨ Codés sur 4 octets = 32 bits

Exemple:

```
int entier;
entier = 3;
print(entier);
```

9.2 Les nombres avec décimales de type « float »

⇨ On parle de nombres à virgule flottante. Ex: PI=3,14159….

⇨ Codés sur 4 octets = 32 bits

Exemple:

```
float decimal;
decimal= PI;
print(decimal);
```

9.3 Les nombres avec décimales de type « double »

- Nombre à virgule flottante comportant une meilleure précision que le type float .
- Codés sur 8 octets = 64 bits

Exemple:

```
double long_decimal;
long_ decimal= PI;
print(long_ decimal);
```

9.4 Vrai ou faux ? Le type « boolean »

- Variable qui ne connait que deux états: Vrai (1) ou faux (0) (1bit)
- Utilisée dans les tests conditionnels
- Utilisée dans les tests conditionnels

Exemple:

```
boolean vraifaux;
vraifaux = true;
println(vraifaux);
```

9.5 Les caractères typographiques de type « char »

- Variable qui permet de stocker une lettre, un symbole, etc.
- Codés sur 2 octets = 16 bits

Exemple:

```
char lettre;
lettre = 'A';
print(lettre);
```

Le caractère est mis entre guillemets simples ‘X ‘

9.6 Une chaine de caractère type « string »

- Variable qui permet de stocker un texte

Exemple:

```
String texte;
texte = "Bonjour!";
print(texte);
```

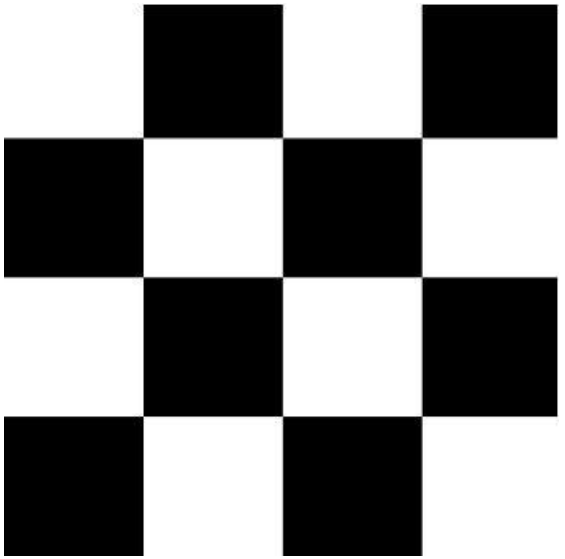
Le texte est mis entre guillemets doubles "X"

9.7 La couleur de type « color »

- Variable qui permet de stocker les caractéristiques d'une couleur: A RVB, A HSV où A est la valeur alpha de transparence.
- Codés sur 4 octets = 32 bits.
  - Chaque composante (A, R,V, B) ou (A, H,S,V) est codée sur 1 octet (0-255).
- Variable utile si on utilise plusieurs fois les mêmes couleurs.

Exemple:

```
noStroke();
color blanc = color(255, 255, 255);
color noir = color(0, 0, 0);
fill(blanc); rect(0, 0, 25, 25);
fill(noir); rect(25, 0, 25, 25);
fill(blanc); rect(50, 0, 25, 25);
fill(noir); rect(75, 0, 25, 25);
fill(noir); rect(0, 25, 25, 25);
fill(blanc); rect(25, 25, 25, 25);
fill(noir); rect(50, 25, 25, 25);
fill(blanc); rect(75, 25, 25, 25);
fill(blanc); rect(0, 50, 25, 25);
fill(noir); rect(25, 50, 50, 25);
fill(blanc); rect(50, 50, 75, 25);
fill(noir); rect(75, 50, 100, 25);
fill(noir); rect(0, 75, 25, 25);
fill(blanc); rect(25, 75, 25, 25);
fill(noir); rect(50, 75, 25, 25);
fill(blanc); rect(75, 75, 25, 25);
```



9.8 Les autres objets

D'autres variables peuvent stoker des objets (image, son, etc…)

On ne parle plus de type, mais d'un objet de la classe correspondante….

Exemple:

```
// Définition d'un objet de la classe PImage
Pimage monImage;

// Chargement du fichier image et remplissage
//de l'objet monImage avec les données
monImage = loadImage("Image_1.pgn");

// Affichage de l'mage
Image(monImage,20,20);
```

9. L'ARCHITECTURE DE BASE D'UN SKETCH

Objectif ?

Gérer l'animation !!!

Animation: initialisation et actualisation de la fenêtre de dessin

Méthode « void setup() » :

- Contient les lignes de codes qui doivent s'exécuter lors du lancement de l'animation:
  - Paramètres d'affichage (taille de la fenêtre de dessin, couleur du background …)
  - Initialisation de toutes les variables

Méthode « void draw() » :

- Se répète sur elle-même par une boucle d'affichage en continu.
- Permet de modifier le contenu de la fenêtre d'affichage à chaque apparition pour créer l'animation…
- Le Nombre d'images par seconde défini par : **frameRate(nbImage)**
- Annuler le rafraichissement des image (draw() ne s'exécute qu'une fois) : **noLoop()**
- Par défaut, draw() est appelé **30 fois par seconde** (i.e. 30 images/s)

Exemple 1:

```
int compteur ;

void setup(){
compteur = 0;
size(400,400);
background(0); // fond noir
frameRate(30); // 30 images/s
}
void draw()
{ println(" La valeur du compteur est: " + compteur )
// cette phrase sera écrite 30 fois/s
}
```

Exemple 2: une seule exécution de la fonction draw()

```
void setup(){
   noLoop(); // on annule le rafraichissement
}
void draw()
{
 println(" il se passe quelque chose " ) // cette phrase sera écrite 1 seule fois
}
```

Modifier la cadence d'affichage: frameRate()

frameRate(nbFoisParSeconde)

L'argument **nbFoisParSeconde** définit le nombre de fois que la fonction draw() est appelé en une seconde

Exemple : affichage d'un point à une position aléatoire 1 fois/s

```
void setup(){
frameRate(1); // 1 images/s
}

void draw()
{
point (random(width), random(heigth));
}
```

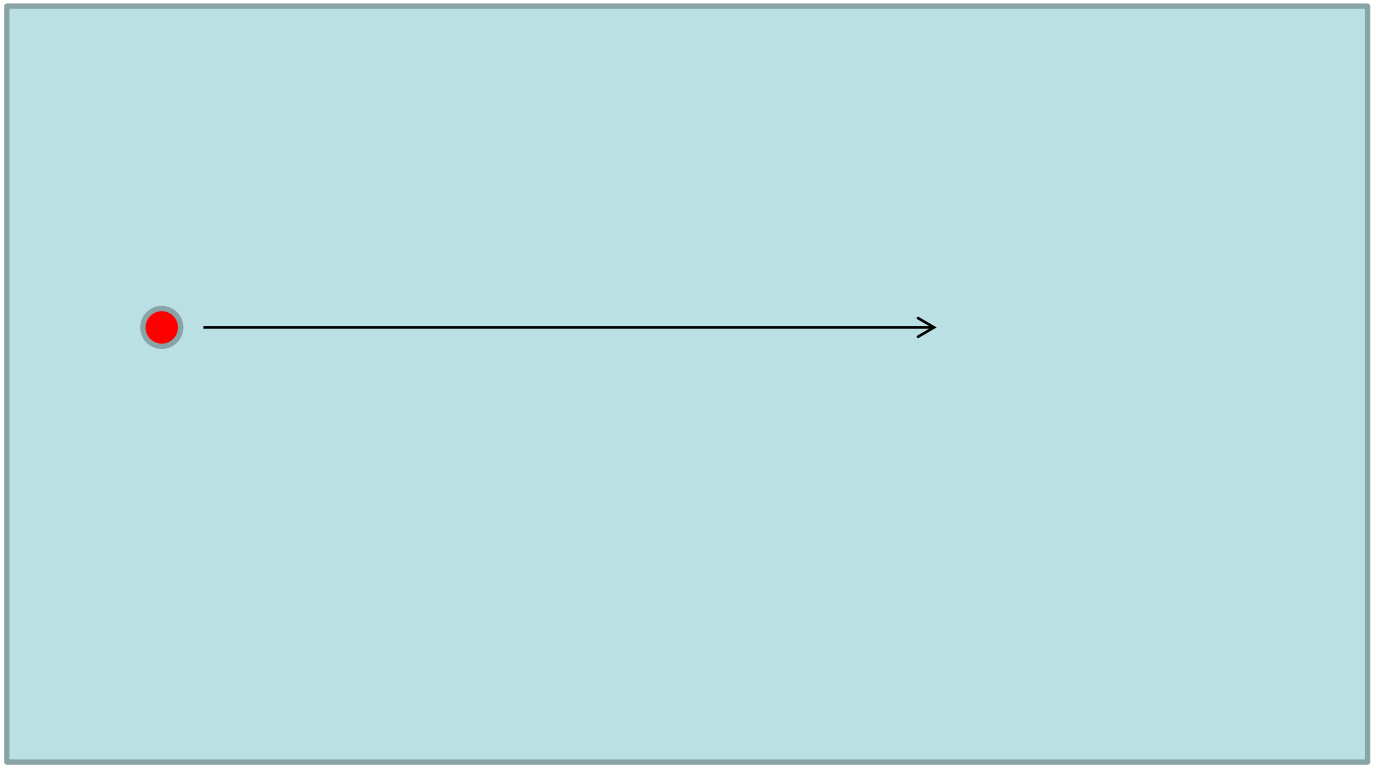
Si on souhaite une cadence plus faible qu'une exécution/seconde, utiliser la méthode **delay():**

```
void draw()
{
point (random(width), random(heigth));
delay(2000);
}
```

Dorénavant: TOUJOURES EMPLOYER LES METHODES setup() ET draw() !!!!!

Travaux Pratiques : Simuler un mouvement - Annimation

Créer un sckech permettant le mouvement d'une balle de gauche à droite, avec effacement progressif de la trace de la balle;



10. LA PROGRAMMATION ORIENTÉE OBJET

10.1 Rappel sur les classes et l'instanciation: exemple

Classe Chat

Attributs (attributs):

- **nom**
- **couleur**
- **Age**

Méthode membre:

- **Miauler**
- **Marcher**

Etc..

Modèle de chat

INSTANCIATION

INSTANCIATION

Ponpon

Attributs:

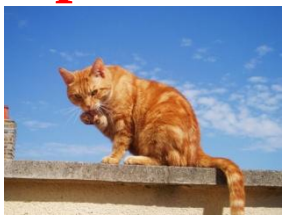
- **nom : pompon**
- **couleur : noir**
- **Age : 6ans**

Méthode membre:

- **Miauler**
- **Marcher**

Etc..

Ponponnette



Attributs:

- **nom : pomponnette**
- **couleur : roux**
- **Age : 3ans**

Méthode membre:

- **Miauler**
- **Marcher**

Etc..

10.2 Comment construire un objet?

- Pour construire un objet il faut d'abord définir son modèle, appelé « **class** ».
  - Définie l'ensemble des variables et méthodes membres que les instances d'une même classe d'objet partagent.

- Lorsqu'on instancie un objet on fait appel à son « **constructeur** ».
  - Constructeur = méthode membre qui porte le même nom que la classe.

Exemple: Classe de l'objet « balle » dans le jeu pong

En s'inspirant du programme déjà écrit (sans POO):

1) Donner les **attributs** (caractéristiques) de la balle.

2) Donner les **méthodes** (actions) de la balle.

3) En respectant l'architecture d'une classe, **écrire la classe « balle »** sous Processing:

   - Vous commencerez par décrire les attributs (variables et leur types)

   - Vous écrierez ensuite le constructeur

   - Vous terminerez par la description des méthodes

Modèle

```
Class balle{
// déclaration des attributs de la classe
          ….
// constructeur de l'objet porte le même nom que la classe
     balle(…){
       // affectation des attributs avec les valeurs du constructeur
             …
      }
/* définitions de toutes les méthodes membres qu'il est possible
d'appeler une fois l'objet créé */
     type méthode1(){
             /*instructions*/
     }
      type méthode2(){
             /*instructions*/
     }
}
```

Correction

```
class Balle {
    //Déclaration attributs de la classe « balle » (paramètre des balles)
    float x,y;
    color couleur;

    //Constructeur de la balle
    Balle (float nouvX, float nouvY, color nouvCouleur) {
    x = nouvX;
    y = nouvY;
    couleur = nouvCouleur;
    }
    //Déclaration des méthodes membres de la classe « balle »
    //Dessiner  la balle
    void display() {
    fill(couleur);
    ellipse(x, y, 40, 40);
    }
}
```

Création d'un objet comme instance d'une classe.

- On fait appel à l'instruction « **new** » qui exécute le constructeur de la classe.

Exemple: Création de 2 balles, « balle1 » et « balle2 »

```
/*Déclaration et création d'une instance de l'objet Balle à la position (100,100)
de couleur blanche*/
Balle maBalle = new Balle(100, 100, color(255));

void setup() {
smooth(); //Lissage des dessins
size(400, 200); //Taille de la fenêtre
}

void draw() {
background(0); //On dessine un fond noir
noStroke(); //On supprime le contour
maBalle.display(); //Affichage de la balle
}
```

Cas général:

```
Class nomObjet{
// déclaration des attributs de la classe
    Type nomVariableMembre1; etc..

// constructeur de l'objet appelé lors de l'instanciation d'un nouvel objet
    nomObjet(arguments){
      // affectation des attributs avec les valeurs du constructeur
            …
     }
/* définitions de toutes les méthodes membres qu'il est possible d'appeler
une fois l'objet créé */
    type méthode1(){
            /*instructions*/
    }
     type méthode2(){
            /*instructions*/
    }
}
```

Remarque: contrairement aux autres méthodes membres, le constructeur ne revoie aucun type !!

10.3 Comment rédiger une méthode membre ?

- Une méthode contient une série d'instruction à exécuter lors de son appel.
- Définie par:
  - **Son nom**
  - **Ses paramètres d'entrée** appelés arguments ou « parameters » sous l'aide en ligne
  - **Ses paramètres de sortie** appelés « returns » sous l'aide en ligne

Exemple:

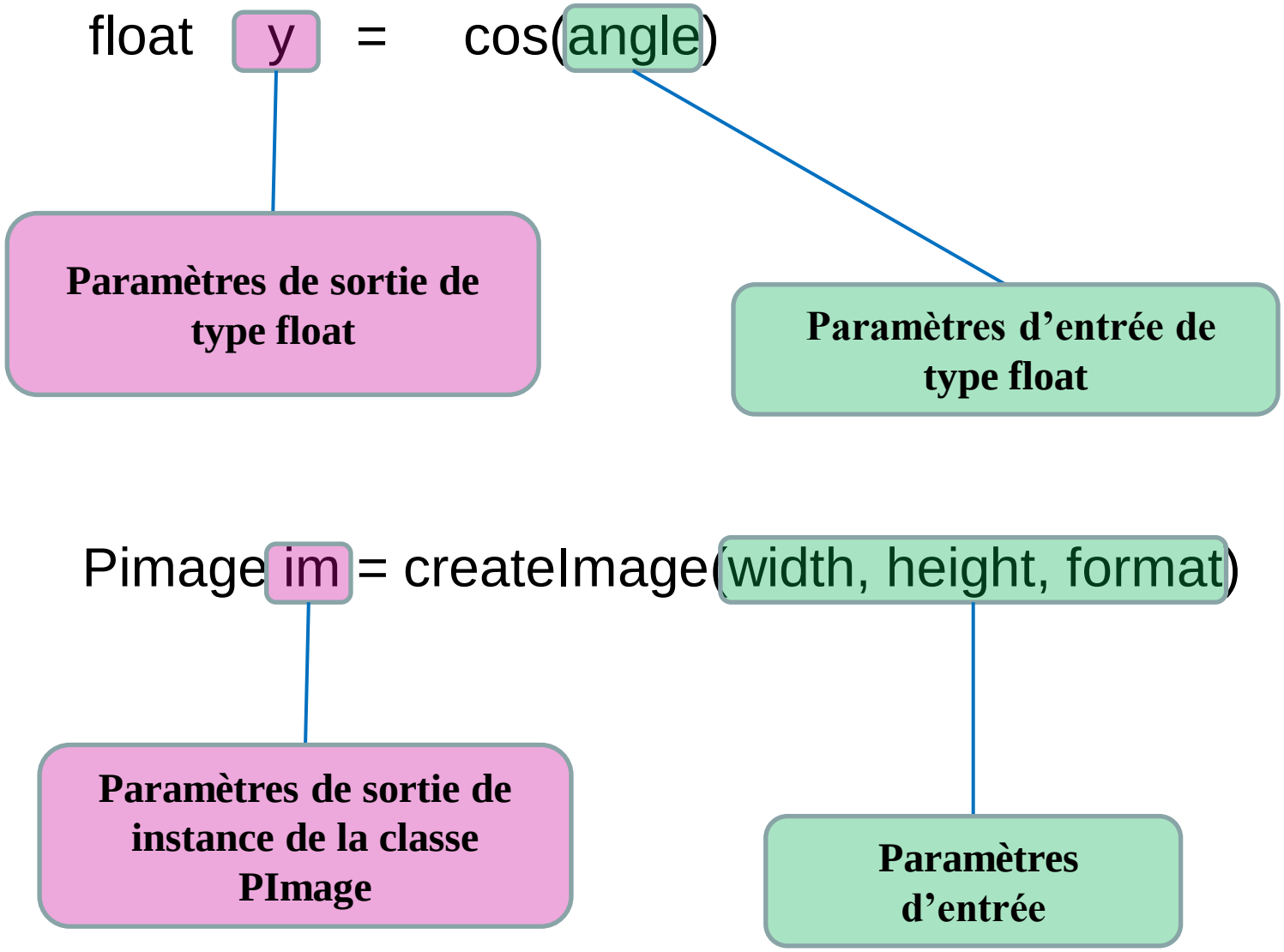
```
void setup () { }
```

Idem pour « draw() »

Pas de paramètres de sortie

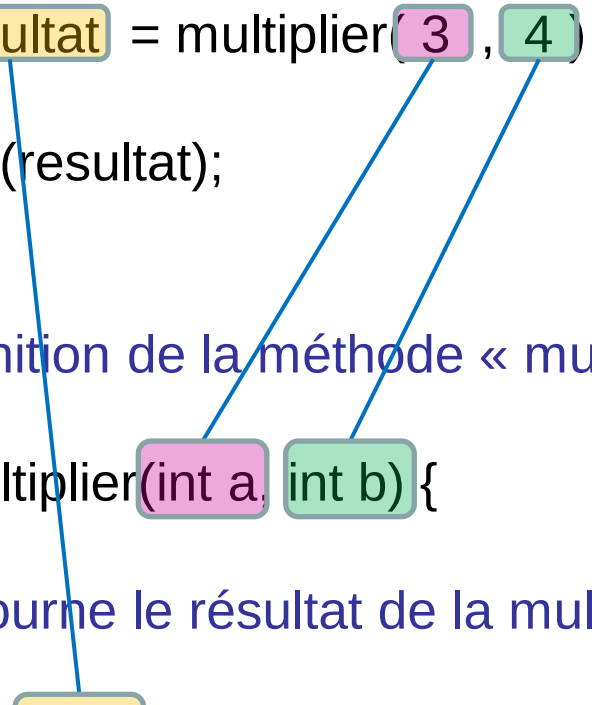
Pas de paramètres d'entrée

Exemple 2:


float y = cos(angle)
Paramètres de sortie de type float
Paramètres d'entrée de type float
Pimage im = createImage(width, height, format)
Paramètres de sortie de instance de la classe PImage
Paramètres d'entrée

Exemple d'écriture d'une méthode: multiplier a et b

```
void setup(){

int resultat = multiplier( 3 , 4 );

println(resultat);
}

// définition de la méthode « multiplier »

int multiplier(int a, int b) {

// Retourne le résultat de la multiplication de ses deux arguments.

return a * b;
}
```



- **Les variables a et b ne sont définie et utilisable qu'à l'intérieur de la méthode « multiplier ».**
- **Pour utiliser une variable dans tout le programme, il faut la déclarer en tout début de programme => utilisable dans setup() et draw()…**

10.4 Architecture générale d'un sketch: A TOUJOUR RESPECTER!

1. **Faire appel aux différentes librairies**

2. **Déclarer toutes les variables globales utilisées et modifiées dans setup() et draw()**

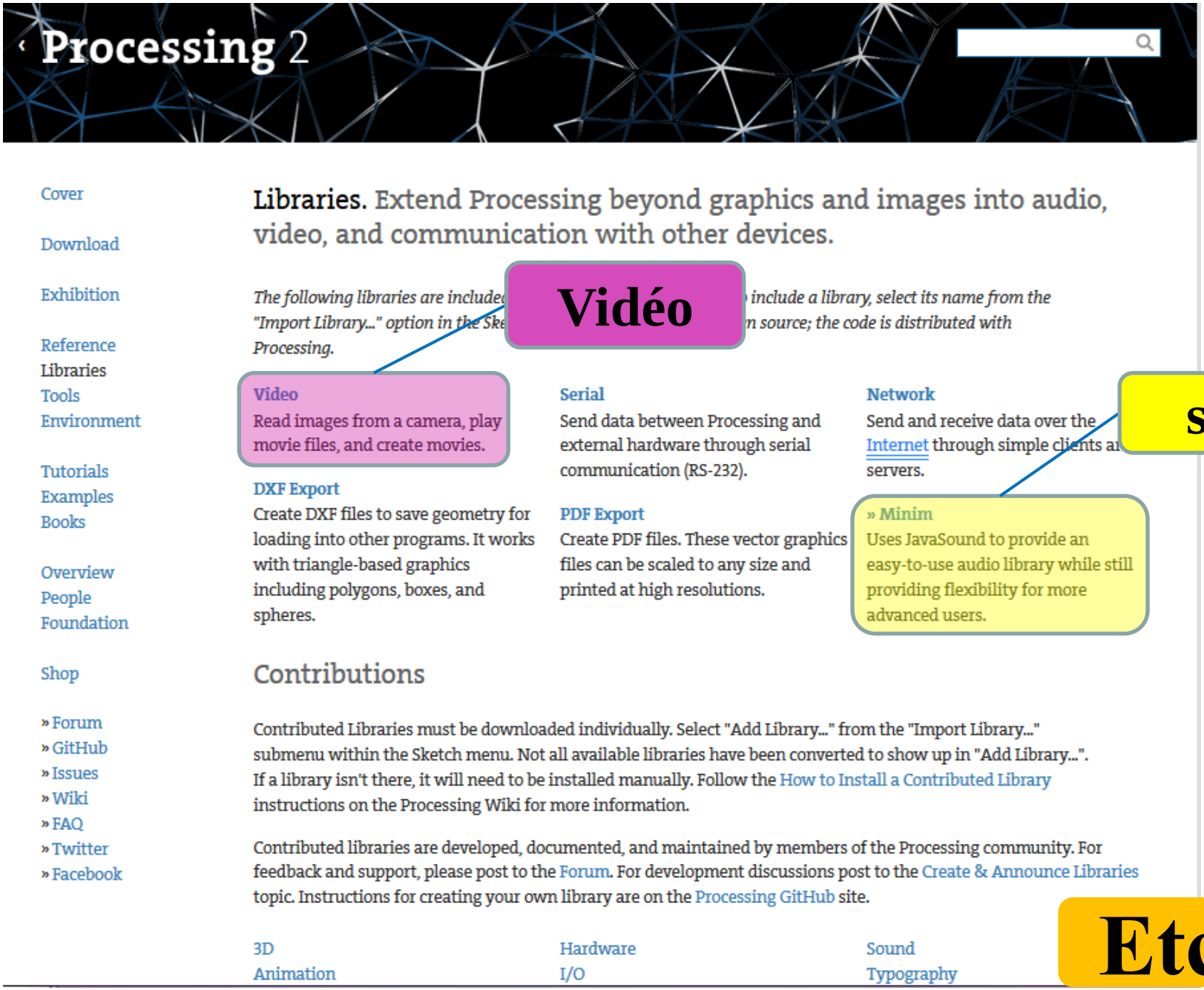
3. **void setup(){} :**
   - **Initialiser toutes les variables**
   - **régler les paramètres d'affichage**

4. **void draw(){} : dessin successif des images réalisant l'animation…**

Exemple:

Ouvrir dans Processing: File > Examples > Video > BackgroundSubtraction

10.5 Exemples de classes : Librairies de Processing !



Processing 2

Cover
Download
Exhibition
Reference
Libraries
Tools
Environment
Tutorials
Examples
Books
Overview
People
Foundation
Shop
» Forum
» GitHub
» Issues
» Wiki
» FAQ
» Twitter
» Facebook

Libraries. Extend Processing beyond graphics and images into audio, video, and communication with other devices.

The following libraries are include… include a library, select its name from the "Import Library..." option in the Sk… n source; the code is distributed with Processing.

Vidéo

Video
Read images from a camera, play movie files, and create movies.

Serial
Send data between Processing and external hardware through serial communication (RS-232).

Network
Send and receive data over the Internet through simple clients a… servers.

son

DXF Export
Create DXF files to save geometry for loading into other programs. It works with triangle-based graphics including polygons, boxes, and spheres.

PDF Export
Create PDF files. These vector graphics files can be scaled to any size and printed at high resolutions.

» Minim
Uses JavaSound to provide an easy-to-use audio library while still providing flexibility for more advanced users.

Contributions

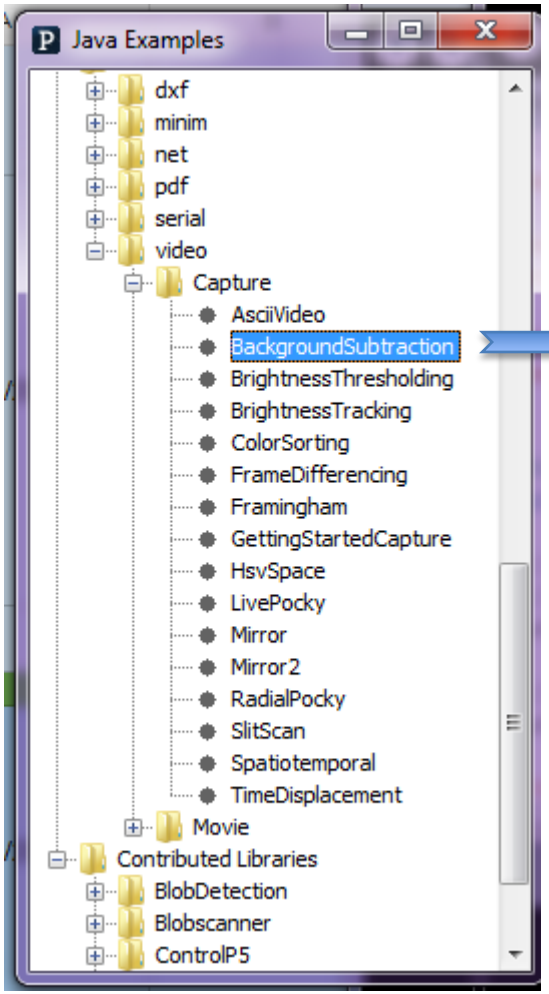
Contributed Libraries must be downloaded individually. Select "Add Library..." from the "Import Library..." submenu within the Sketch menu. Not all available libraries have been converted to show up in "Add Library...". If a library isn't there, it will need to be installed manually. Follow the How to Install a Contributed Library instructions on the Processing Wiki for more information.

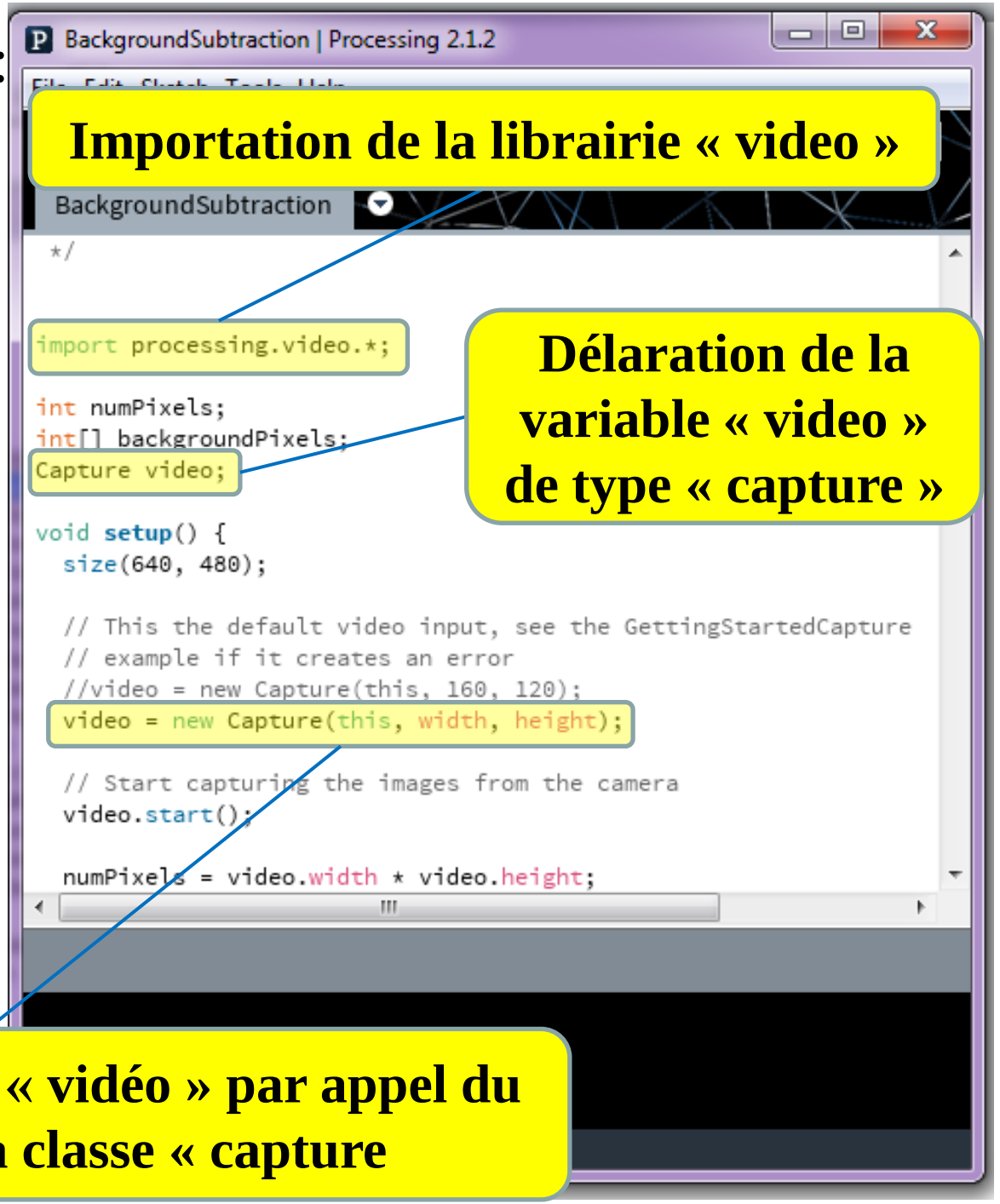
Contributed libraries are developed, documented, and maintained by members of the Processing community. For feedback and support, please post to the Forum. For development discussions post to the Create & Announce Libraries topic. Instructions for creating your own library are on the Processing GitHub site.

3D
Animation
Hardware
I/O
Sound
Typography

Etc…

Classe « **Capture** »:





Importation de la librairie « video »

Délaration de la variable « video » de type « capture »

Instanciation de l'objet « vidéo » par appel du constructeur de la classe « capture

Description de la classe « **Capture** »: (voir dans références)

Name Capture

Examples

```
import processing.video.*;

Capture cam;

void setup() {
  size(640, 480);

  String[] cameras = Capture.list();

  if (cameras.length == 0) {
    println("There are no cameras available for capture.");
    exit();
  } else {
    println("Available cameras:");
    for (int i = 0; i < cameras.length; i++) {
      println(cameras[i]);
    }

    // The camera can be initialized directly using an
    // element from the array returned by list():
    cam = new Capture(this, cameras[0]);
    cam.start();
  }
}

void draw() {
  if (cam.available() == true) {
    cam.read();
  }
  image(cam, 0, 0);
  // The following does the same, and is faster when just drawing the image
  // without any additional resizing, transformations, or tint.
  //set(0, 0, cam);
}
```

Description

Datatype for storing and manipulating video frames from an attached capture device such as a camera. Use `Capture.list()` to show the names of any attached devices. Using the version of the constructor without `name` will attempt to use the last device used by a QuickTime program.

Methods

available()	Returns "true" when a new video frame is available to read
start()	Starts capturing frames from the selected device
stop()	Stops capturing frames from an attached device
read()	Reads the current video frame
list()	Gets a list of all available capture devices such as a camera

Constructor

```
Capture(parent)
Capture(parent, requestConfig)
Capture(parent, requestWidth, requestHeight)
Capture(parent, requestWidth, requestHeight, frameRate)
Capture(parent, requestWidth, requestHeight, cameraName)
Capture(parent, requestWidth, requestHeight, cameraName, frameRate)
```

Parameters

`parent`	PApplet: typically use "this"
`requestWidth`	int: width of the frame
`requestHeight`	int: height of the frame
`frameRate`	int: number of frames to read per second
`cameraName`	String: name of the camera

10.6 Exemple d'utilisation: afficher une ou plusieurs balles…

- Classe « **balle** »:
  - attributs:
    - position suivant x
    - position suivant y
    - couleur
  - Méthodes membres:
    - dessiner la balle

Exercice:

1. **rédiger la classe « balle » (attributs, constructeur, méthodes membres).**
2. **créer une balle blanche appelée « maBalle »à la position (100, 100) en instanciant la classe correspondante et l'afficher en utilisant la méthode draw().**

Correction :

```
/*Déclaration et création d'une instance de l'objet Balle à la position (100,100) de couleur blanche*/
Balle maBalle = new Balle(100, 100, color(255));

void setup() {
smooth(); //Lissage des dessins
size(400, 200); //Taille de la fenêtre
}

void draw() {
background(0); //On dessine un fond noir
noStroke(); //On supprime le contour
maBalle.display(); //Affichage de la balle
}

class Balle {
        //Déclaration des attributs de la classe « balle » (paramètre des balles)
        float x,y;
        color couleur;

        //Constructeur de la balle
        Balle (float nouvX, float nouvY, color nouvCouleur) {
        x = nouvX;
        y = nouvY;
        couleur = nouvCouleur;
        }
        //Déclaration des méthodes membres de la classe « balle »
        //Dessiner  la balle
        void display() {
        fill(couleur);
        ellipse(x, y, 40, 40);
        }
}
```

Exercice: modifier le programme précédents pour afficher deux balles (une blanche et une grise NG=128) aux position (100,100) et (200, 100).

↳ **Appeler ces balles: maBalle1 et maBalle2**

Correction :

```
/*Déclaration et création de 2 instances de l'objet Balle aux positions (100,100) et (200,100) de couleur blanche et grise*/
Balle maBalle1 = new Balle(100, 100, color(255));
Balle maBalle2 = new Balle(200, 100, color(128));

void setup() {
smooth(); //Lissage des dessins
size(400, 200); //Taille de la fenêtre
}
void draw() {
background(0); //On dessine un fond noir
noStroke(); //On supprime le contour
maBalle1.display(); //Affichage maBalle1
maBalle2.display(); //Affichage maBalle2
}
class Balle {
        //Déclaration des attributs de la classe « balle » (paramètre des balles)
        float x,y;
        color couleur;

        //Constructeur de la balle
        Balle (float nouvX, float nouvY, color nouvCouleur) {
        x = nouvX;
        y = nouvY;
        couleur = nouvCouleur;
        }
        //Déclaration des méthodes membres de la classe « balle »
        //Dessiner  la balle
        void display() {
        fill(couleur);
        ellipse(x, y, 40, 40);
        }
}
```

11. OPÉRATEURS ET CONDITIONS

11.1 Opérateurs arithmétiques:

Addition	Soustraction	Multiplication	Division	Modulo (reste d'une division euclidienne)
+	-	*	/	%

Exemple d'utilisation du modulo: Processing – Pearson p 44

```
int a = 4%2 // Renvoie 0 car le reste de la division 4/2 est 0
```

Intérêt de % : Permet de revenir à un point initial sans avoir à recourir à la condition if{…} :

```
float x = 0.0;
void draw() {  // draw() permet d'effectuer des annimations
x = (x+1) % width ; // le point ne dépasse pas la largeur de l'écran
point(x,height/2);
}
```

Exemple:

```
float x = 0.0;

void draw() {  // draw() permet d'effectuer des animations
  x = (x+1) % width ; // le point ne dépasse pas la largeur de l'écran

  rect(x,height/2, 10,10); // dessine un rectangle à l'abscisse x

  if (x==0){ // change de couleur de remplissage lorsqu'on arrive au bout de l'écran
    int r = floor(random(0,255));
    int v = floor(random(0,255));
    int b = floor(random(0,255));
    fill(r,v,b);
    println("r= "+r+", v= "+v+ " b= "+b);
  }

}
```

11.2 Opérateurs d'assignation:

Attribution/ modification d'une valeur	incrémentation	décrémentation	Incrémenter et décrémenter de 1	Multiplication ou division par un facteur
=	+=	-=	++ et --	*= ou /=

Exemple

Processing – Pearson p 44/45

```
int x=100;
x+= 200; // la valeur de x passe de 100 à 300

int x=100;
x-= 10; // la valeur de x passe de 100 à 90

int x=100;
x++ // la valeur de x passe de 100 à 101

int x=100;
x-- // la valeur de x passe de 100 à 99
```

Exemple

```
int x=100;
X*= 10; // la valeur de x passe de 100 à 1000

int x=100;
x/= 2; // la valeur de x passe de 100 à 50
```

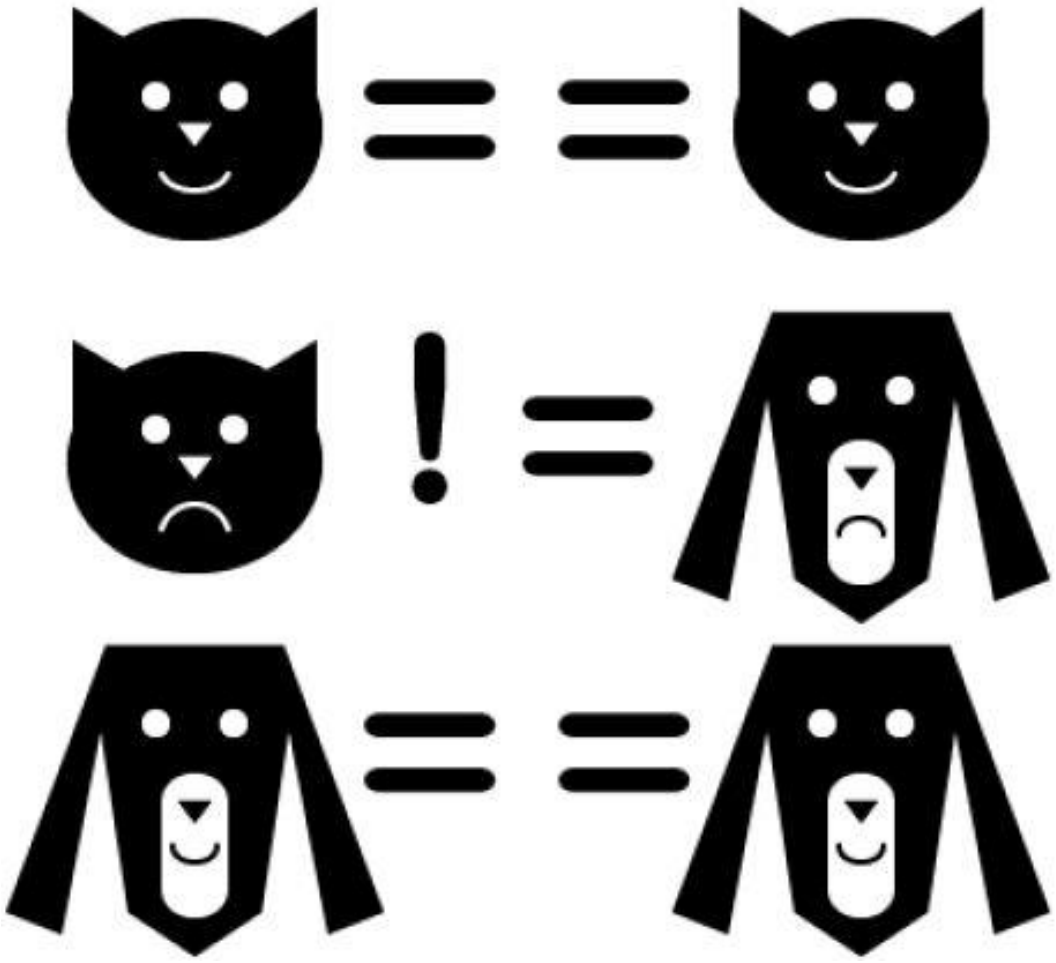
11.3 Opérateurs relationnels (comparaison)

Comparaison entre deux variables de même type (renvoie un booléen)					
supérieur	Inférieur	Supérieur ou égale	Inférieur ou égale	Égalité	Différence
>	<	>=	<=	==	!=

Exemple

```
10 == 1; // renvoie « false » (println(10==1))
36== 6*6; // renvoie la valeur  « true »
36!= 6*6; // renvoie la valeur  « false »
‘a’== ‘a’; // renvoie « true »
‘a’==‘b’; // renvoie « false »
```

Exercice: que renvoie ces comparaisons ?



11.3 Opérateurs logiques (ET / OU)

ET	OU
&&	||

&&	true	false
true	true	false
false	false	false

||	true	false
true	true	true
false	true	false

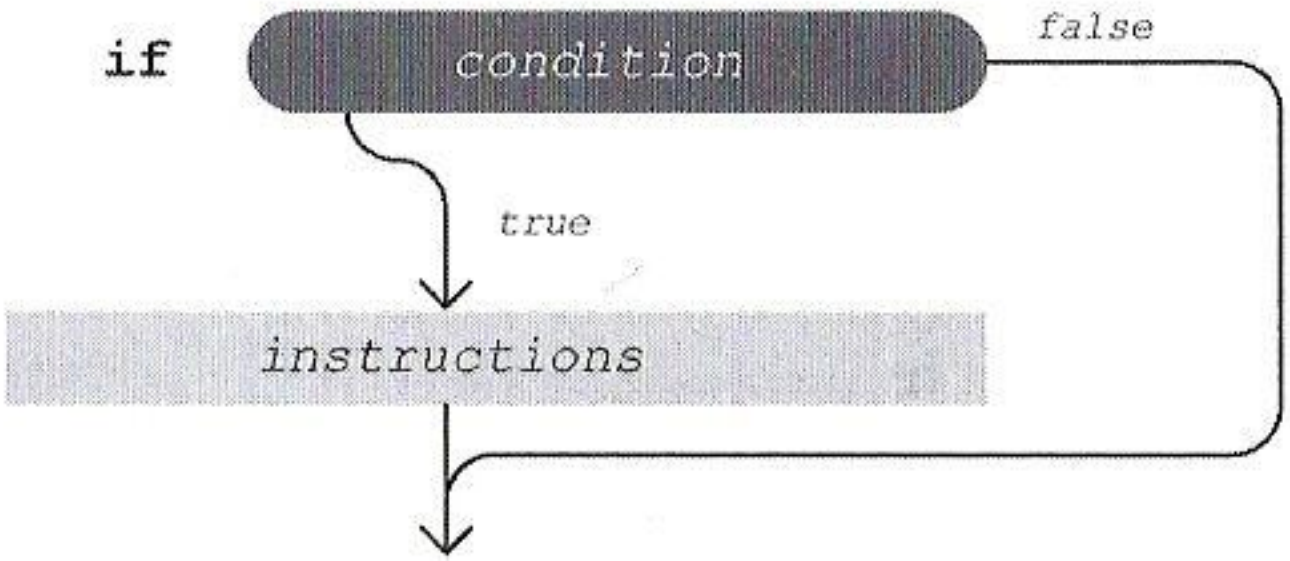
Intérêt : permet le test simultané de plusieurs conditions

12. STRUCTURES CONDITIONNELLE ET ITÉRATIVES

12.1 Structure conditionnelle « if, else if, else »

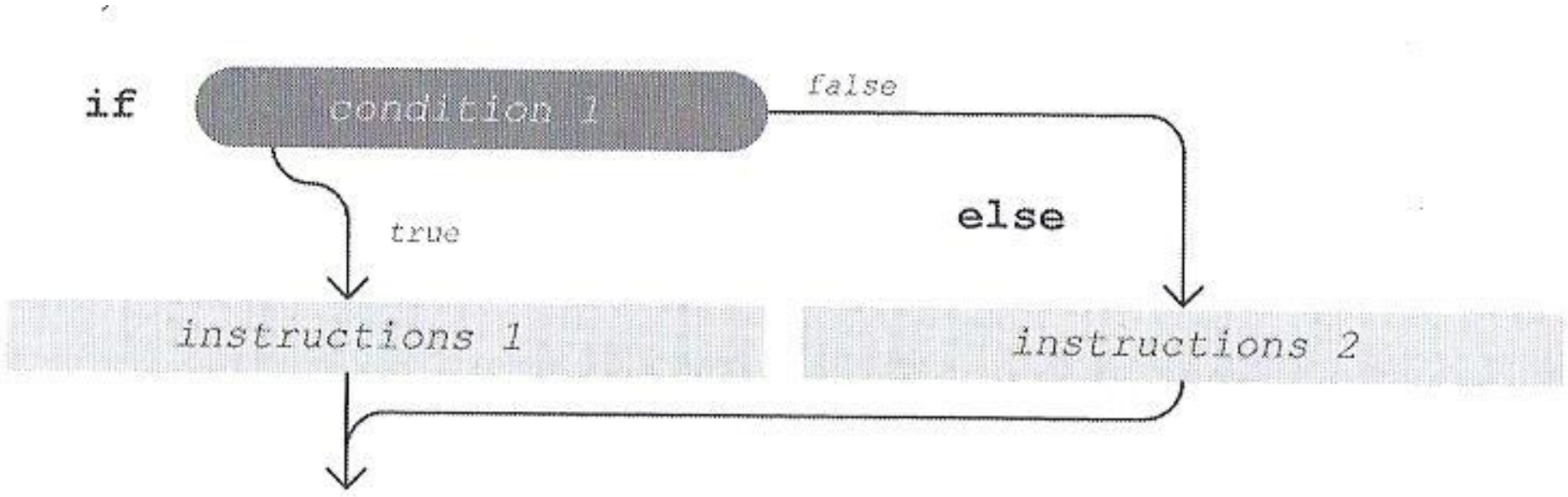
Syntaxe de la structure if ():

```
if (condition){
        // code à exécuter si la condition est vraie
}
```



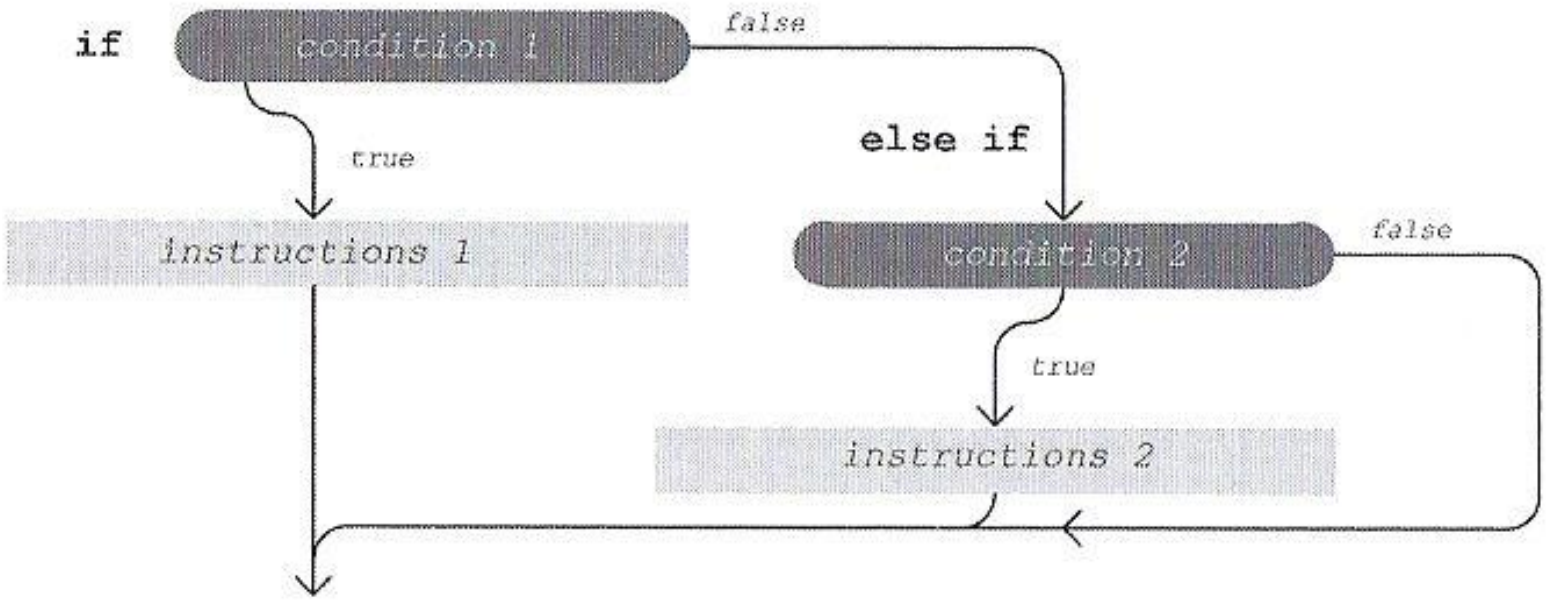
Syntaxe de la structure if () else():

```
if (condition 1){
        // code à exécuter si la condition 1 est vraie
}
else {
        // code à exécuter si la condition 1 est fausse
}
```



Syntaxe de la structure if () else if() else():

```
if (condition 1){
        // code à exécuter si la condition 1 est vraie
}
else if (condition 2) {
        // code à exécuter si la condition 2 est vraie
}
else  {
        // code à exécuter si la condition 2 est fausse
}
```



Cas général : if () else if() else()

```
if (condition 1){
        // code à exécuter si la condition 1 est vraie
}
else if (condition 2) {
        // code à exécuter si la condition 2 est vraie
}
else if (condition 3)  {
        // code à exécuter si la condition 3 est vraie
}
…
else {
// code à exécuter si aucune des conditions n'est vrai
}
```

Exemple

```
if (hour() < 12) {
        println("C'est le matin !");
}
else if (hour() == 12) {
        println("Il est midi !");
}
else {
        println("Ce n'est pas le matin !");
}
```

Test avec plusieurs conditions

```
if (hour() < 6 && hour() > 20) {
        println("Il fait nuit !");
}
else {
        println("Il ne fait pas nuit !");
}
```

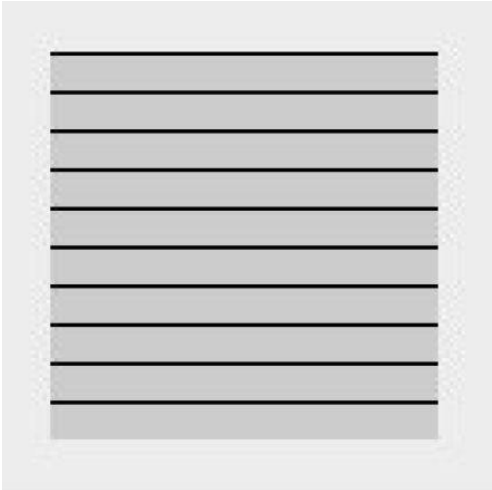
12.2 Structures itératives (répétitions)

Pourquoi ??

- Permet d'exécuter une exécution plusieurs fois à la suite
- Evite de dupliquer inutilement les lignes de codes

Exemple:

```
line(0, 0, 100, 0);
line(0, 10, 100, 10);
line(0, 20, 100, 20);
line(0, 30, 100, 30);
line(0, 40, 100, 40);
line(0, 50, 100, 50);
line(0, 60, 100, 60);
line(0, 70, 100, 70);
line(0, 80, 100, 80);
line(0, 90, 100, 90);
```



SOLUTION:

Boucle « for »!!

Fastidieux!!

Boucle for()

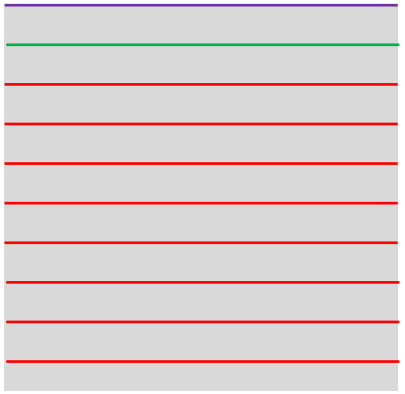
Exemple précédant :

```
for (int i = 0; i < 100; i += 10) {
line(0, i, 100, i);
}
```

Résultat identique!!

Déroulement de la boucle « pas à pas »:

Fenêtre d'affichage



Initialement: i = 0

instruction: line(0, 0, 100, 0);

incrémentation: i = i + 10 => i = 10

test de la condition d'arrêt : i=10 < 100 ? => Condition VRAI !

instruction: line(0, 10, 100, 10);

incrémentation: i = i + 10 => i = 20

test de la condition d'arrêt : i=20 < 100 ? => Condition VRAI !

Etc…

incrémentation: i = i + 10 => i = 100

test de la condition d'arrêt : i=100 < 100 ? => Condition FAUSSE! => Arrêt de la boucle

Syntaxe générale de la boucle for()

```
for (valeur de départ; condition; valeur d'incrémentation){
// instructions à exécuter tant que la condition n'est pas vrai
}
```

- Permet la répétition d'actions dans une dimensions…
- Permet la réalisation d'un compteur…

Imbrication des boucle for()

⇨ Permet la répétition d'actions dans plusieurs dimensions…

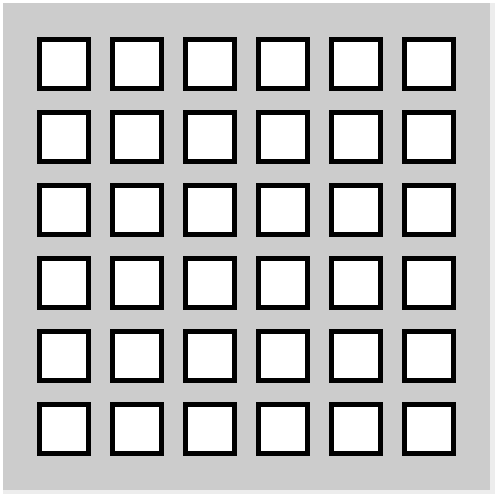
Exemple : 2 boucles

Exercice : Dessiner progressivement la figure donnée par ce programme

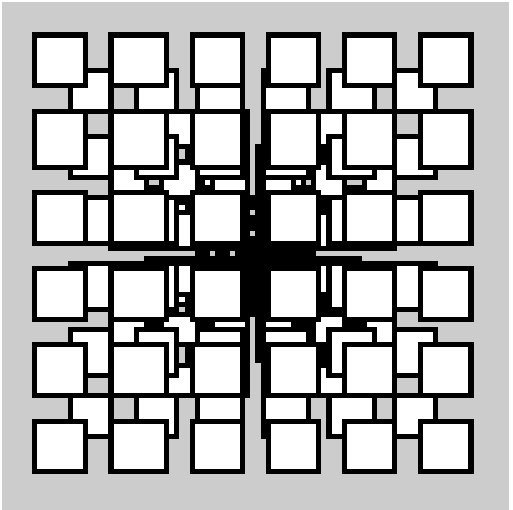
- **pour i = 1 et j=2,3,4, 5, 6**
- **pour i = 2, et j = 2,3,4**

```
translate(7, 7);
//Première boucle (hauteur)
  for (int j = 0; j < 6; j ++) {
     //Seconde boucle (largeur)
       for (int i = 0; i < 6; i++) {
            rect(i * 15, j * 15, 10, 10);
        }
    }
```

Solution :



Exemple : 3 boucles – Ecrire le programme permettant de dessiner l'image suivante:



ATTENTION: vous utiliserez la méthode translate(0, 0, k * -15), pour k=1 à 6 pour dessiner les carrés dans la profondeur…

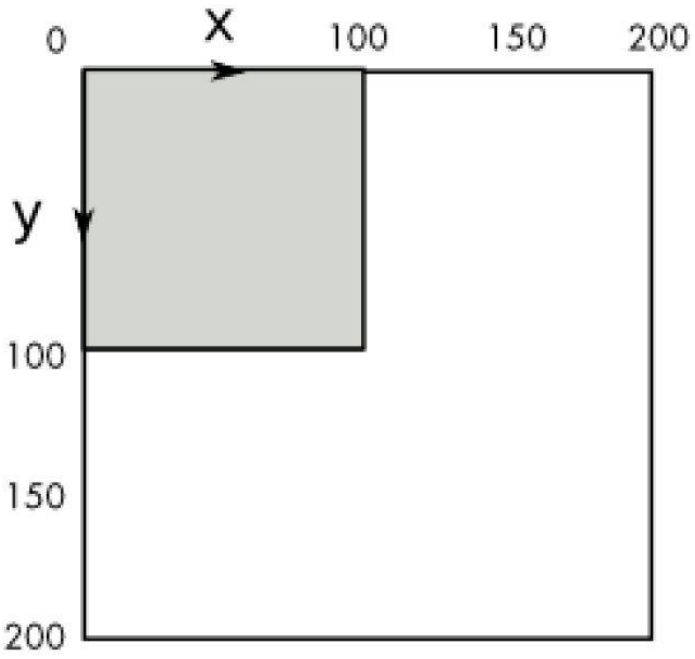
Solution:

```
size(100, 100, P3D);
translate(7, 7);
//Première boucle – profondeur : axe z
for (int z = 0; z < 6; z = z + 1) {
    translate(0, 0, z * -15); //On recule l'objet sur l'axe z
    //Seconde boucle – hauteur : axe y
    for (int y = 0; y < 6; y = y + 1) {
        //Troisième boucle (largeur)
        for (int x = 0; x < 6; x = x + 1) {
        rect(x * 15, y * 15, 10, 10);
        }
    }
}
```

13. LES TRANSFORMATIONS

- Origine par défaut du dessin d'une forme : **coin supérieur gauche de la fenêtre.**

```
size(200, 200);
noStroke();
fill(0);
rect(0, 0, 100, 100);
```



- Les différentes transformations permettent de :
  - Déplacer l'origine
  - Redéfinir l'orientation des axes (rotation)
  - Changer d'échelle (agrandir/rétrécir une forme)

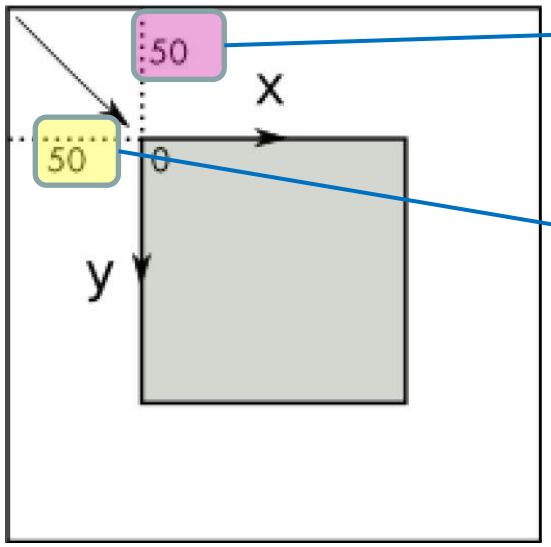
13.1 Translation d'un objet

En mathématiques, une translation est une transformation géométrique qui correspond à l'idée intuitive de « glissement » d'un objet, sans rotation, retournement ni déformation de cet objet.

Une translation est définie par vecteur déplacement $\vec{u} = \begin{pmatrix} u_x \\ u_y \end{pmatrix}$

Tout déplacement dans le plan peut se décomposer en deux déplacement élémentaires suivant les axes (ox) et (oy)

```
size(200, 200);
noStroke();
fill(0);
translate(50, 50);
rect(0, 0, 100, 100);
```



Déplacement suivant x

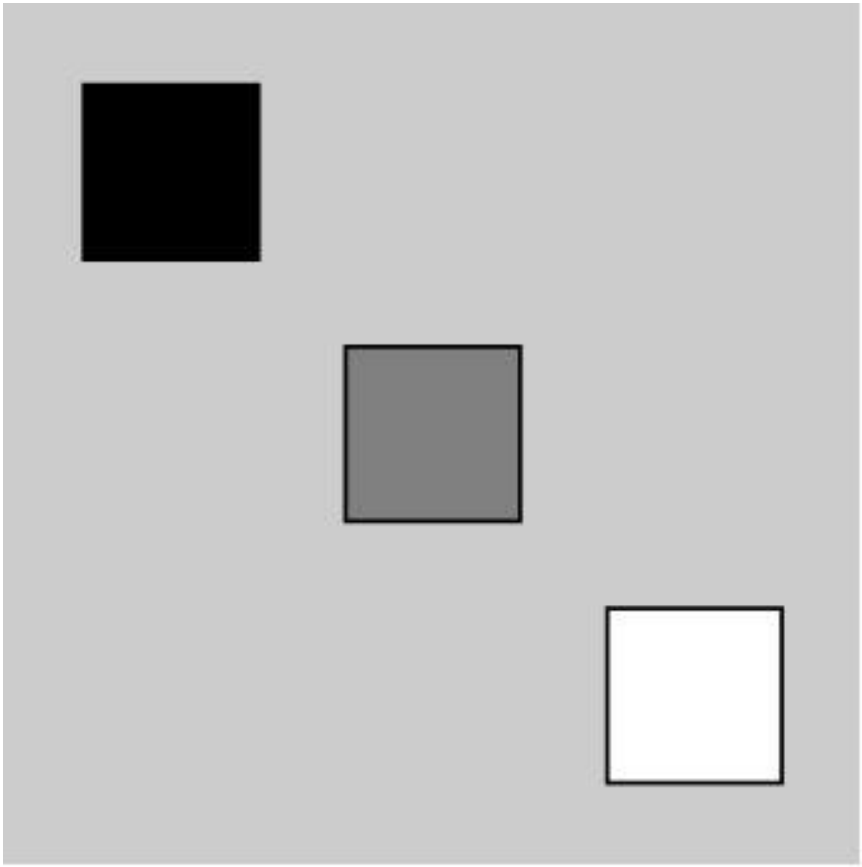
Déplacement suivant y

La nouvelle origine se trouve maintenant au point (50,50)!!

Tout dessin d'une autre forme aura pour origine ce point.

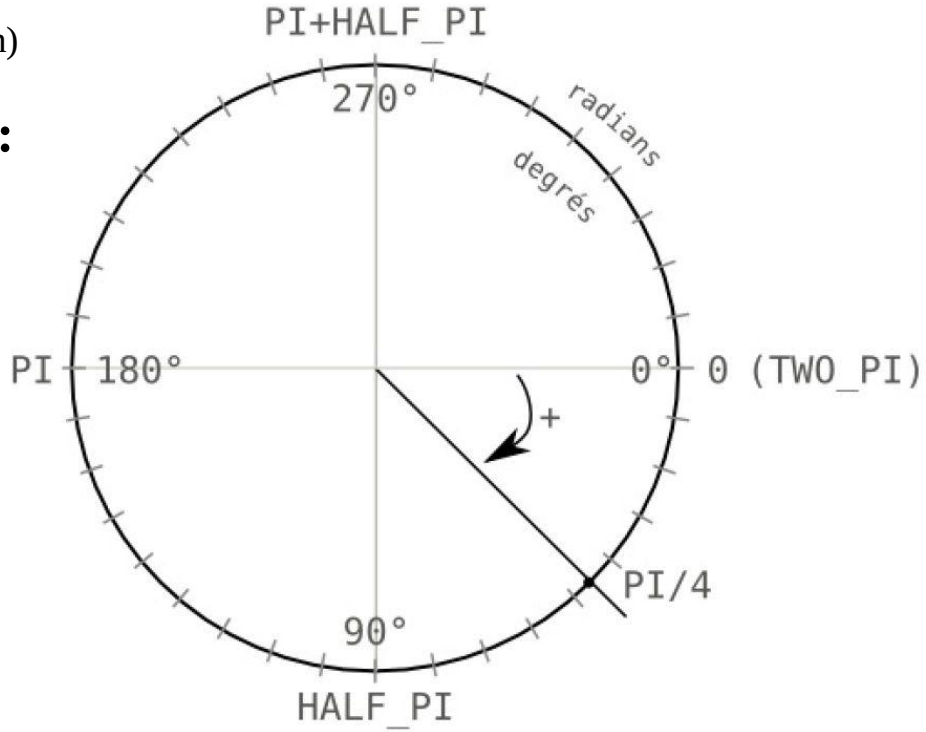
Exemple:

```
size(200,200);

// Noir
fill(0);
translate(20,20);
rect(0,0,40,40);

// Gris
fill(128);
translate(60,60);
rect(0,0,40,40);

// Blanc
fill(255);
translate(60,60);
rect(0,0,40,40);
```



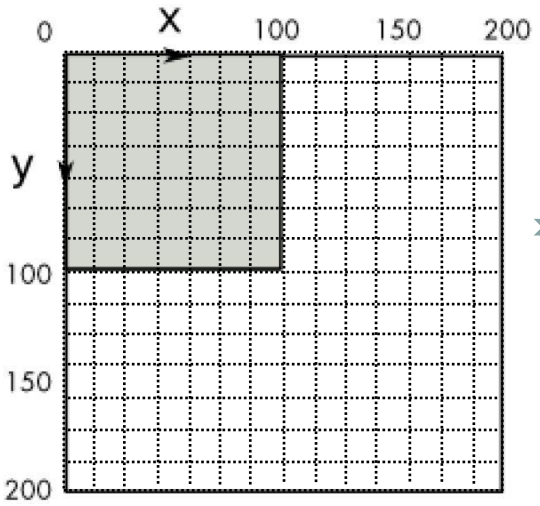
13.2 Rotation d'un objet (Voir p75 –Pearson)

Convention dans la mesure des angles:

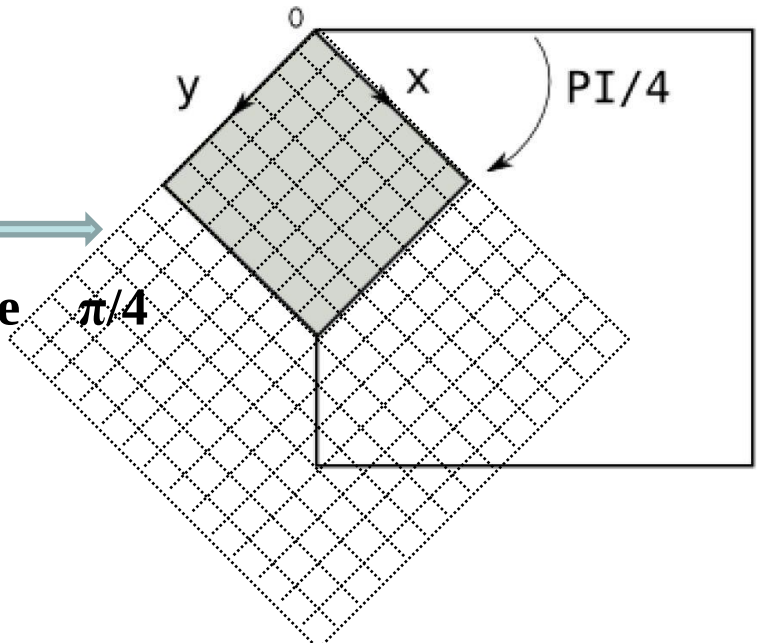


Exemple:

```
size(200, 200);
noStroke();
fill(0);
rotate(PI/4);
rect(0, 0, 100, 100);
```



Rotation de $\pi/4$ radian = 45°



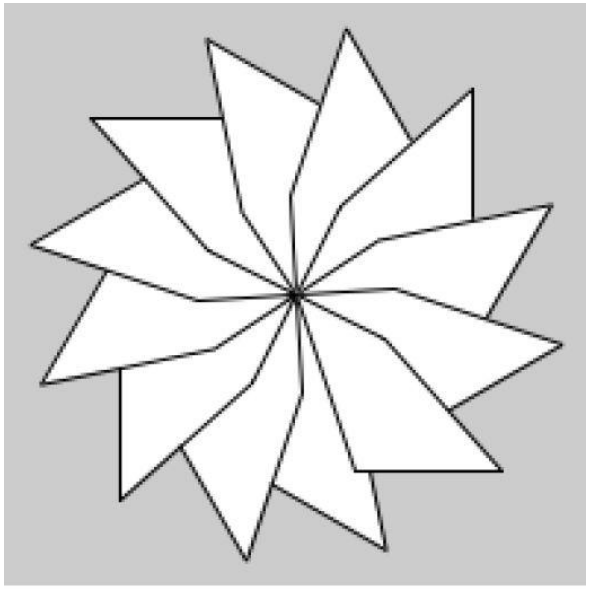
Conversion d'unité d'angles: 2π rad = 360°

Méthode pour la conversion d'angles:

```
float d = degrees(PI/4); // transforme des radians en degrés
float r = radians(180.0); // transforme des degrés en radians
```

Exemple 2:

```
size(200,200);

smooth();
translate(width/2, height/2);

for (int i=0;i<360;i+=30){
rotate(radians(30));
quad(0, 0, 30, 15, 70, 60, 20, 60);
}
```



13.3 homothétie d'un objet (changement d'échelle)

Redimensionnement de la forme des objets: méthode scale()

Exemple:

- **scale(0.5)** ⟹ Divise la taille par 2
- **scale(2)** ⟹ Multiplie la taille par 2
- **scale(1)** ⟹ Aucun effet

Redimensionnement identique suivant x et y

Découplage du redimensionnement suivant x et y:

Exemple: **scale(0.5, 2.0)**

- **0.5** : **Divise par 2 la taille suivant x**
- **2.0** : **Multiplie par 2 la taille suivant y**

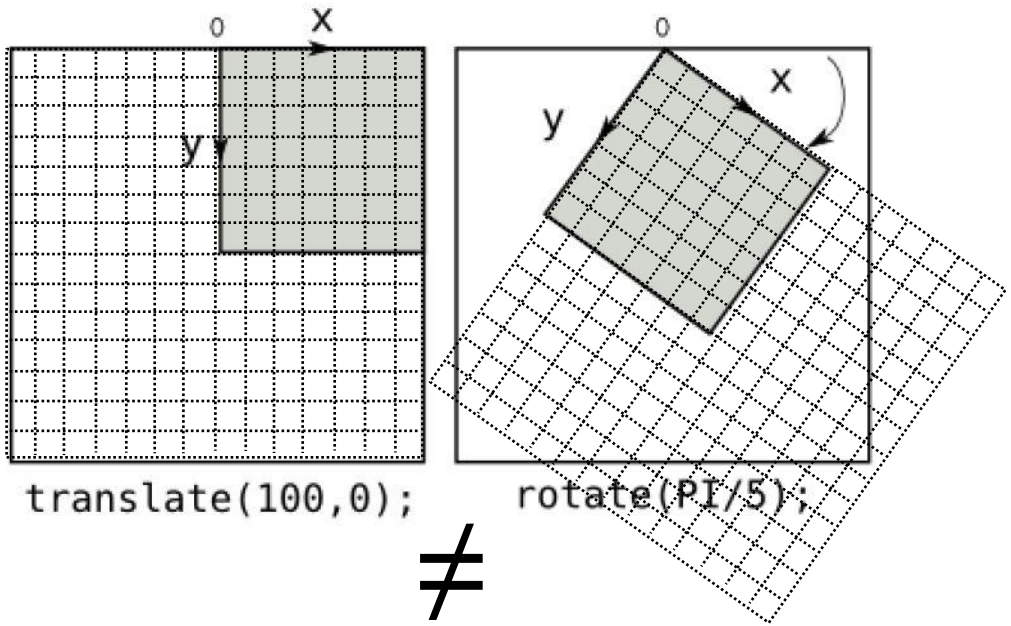
Application:

```
size(200,200);
noStroke();
// Noir
fill(0);
scale(1);
rect(0,0,200,200);
```

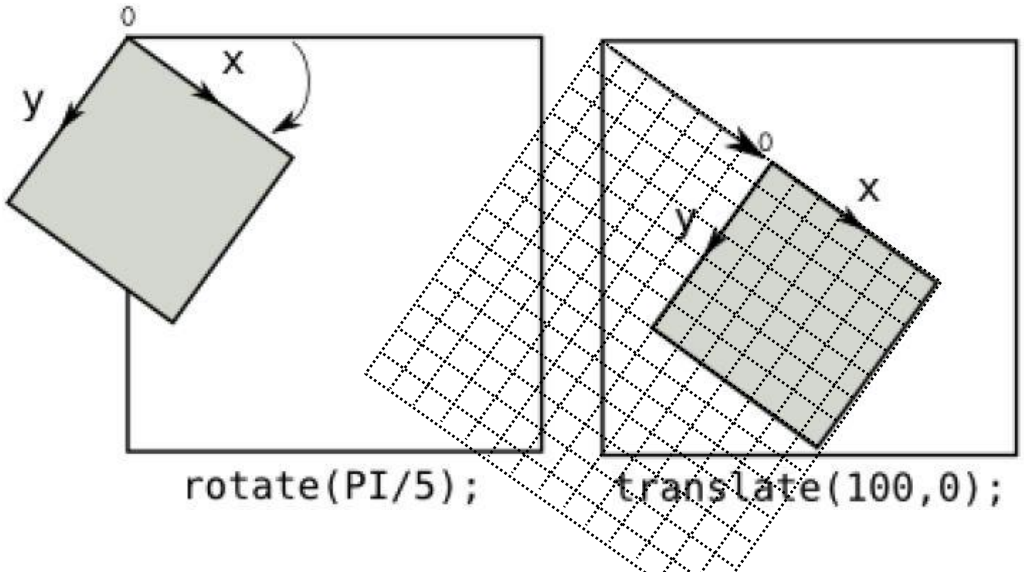
```
// Gris
fill(128);
scale(0.5);
rect(0,0,200,200);
// Blanc
fill(255);
scale(0.5);
rect(0,0,200,200);
```

13.4 Ordre des transformations

L'ordre des transformation est important !!



≠



13.5 Isoler les transformations

- **Les transformations s'accumulent et modifient l'origine, la direction des axes et l'échelle d'une figure…**

- **Pour qu'une nouvelle transformation ne s'applique qu'à une figure donnée, sans prendre en compte les transformations précédente, on place cette nouvelle transformation et l'écriture de la forme entre les méthodes pushMatrix() et popMatrix().**

  - Encapsulation de l'écriture (cf. html)

  - Après **popMatrix()** les anciennes transformations sont rétablies,

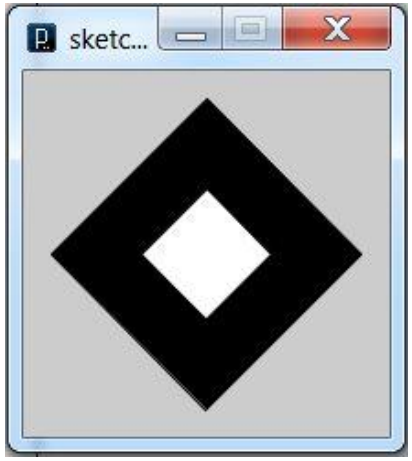
- **pushMatrix() enregistre les coordonnées du point d'origine du dessin, ainsi que la direction des axes et l'échelle, à un instant donné.**

- **popMatrix() rétablit le point d'encrage stockée par la commande pushMatrix()**

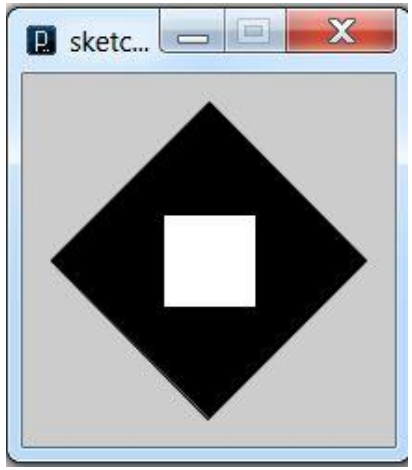
Ces deux fonctions sont utiles en animation (méthode draw())

Exemple

```
size(200,200);
smooth();
rectMode(CENTER);

// Repère au centre de l'écran
translate(width/2,height/2);



// la rotation ne s'applique qu'au
//carré noir
rotate(PI/4);
// Dessin du carré noir
fill(0);
rect(0,0,120,120);



// Dessin du carré blanc qui ne
//tient pas compte de la rotation
fill(255);
rect(0,0,50,50);
```




```
size(200,200);
smooth();
rectMode(CENTER);

// Repère au centre de l'écran
translate(width/2,height/2);

// Sauvegarde des coordonnées
//du repère
//pushMatrix();

// la rotation ne s'applique qu'au
//carré noir
rotate(PI/4);
// Dessin du carré noir
fill(0);
rect(0,0,120,120);

// Restauration des coordonnées
// du repère au centre de l'écran-
//annulation de la rotation des
//axes
//popMatrix();

// Dessin du carré blanc qui ne
//tient pas compte de la rotation
fill(255);
rect(0,0,50,50);
```



13.6 Transformations 3D

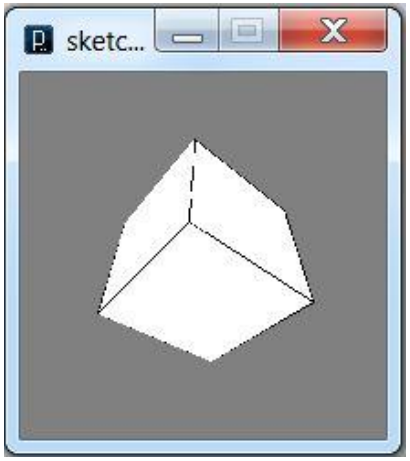
Translation, rotation, changement d'échelle sont applicables en 3D!!

- Appeler la 3ième dimension: **size(300,300,P3D);**
- Rajouter une 3ième coordonnées aux méthodes précédentes

Exemple : **translate(width/2, height/2, -100);**

Exemple 2:

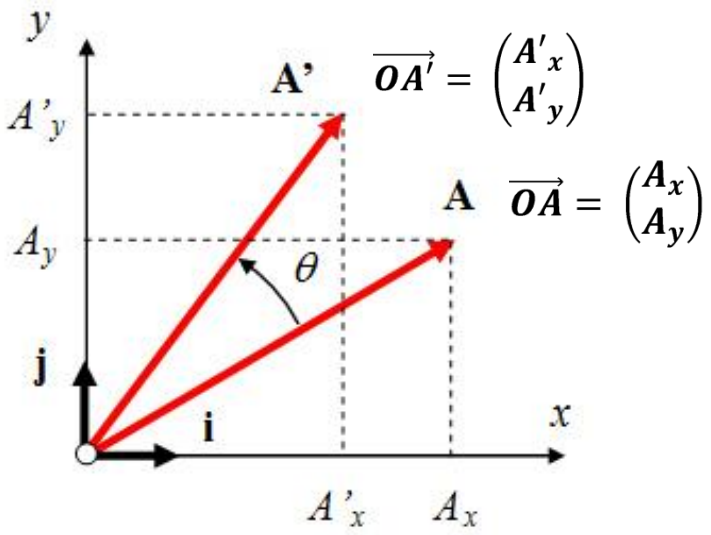
```
float rx = 0;
float ry = 0;
float z = 100;
void setup() {
size(200,200,P3D);
}
void draw() {
background(128);
rx = map(mouseX, 0,width,-PI,PI);
ry = map(mouseY, 0,height,-PI,PI);
translate(width/2,height/2,z);
rotateX(rx);
rotateY(ry);
box(30);
}
```



13.7 Pourquoi pushMatrix et popMatrix ?

Transformations dans le plan (2D) et matrices:

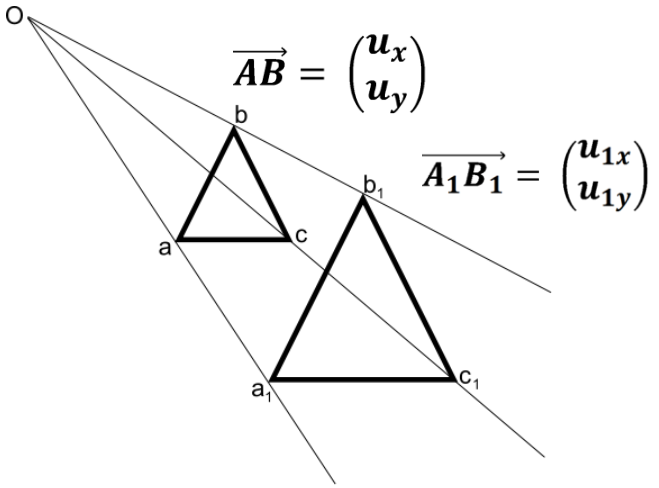
Rotation



$$\begin{pmatrix} A'_x \\ A'_y \end{pmatrix} = \begin{pmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{pmatrix} \begin{pmatrix} A_x \\ A_y \end{pmatrix}$$

Matrice de rotation R(θ)

Changement d'échelle : homothétie de centre O



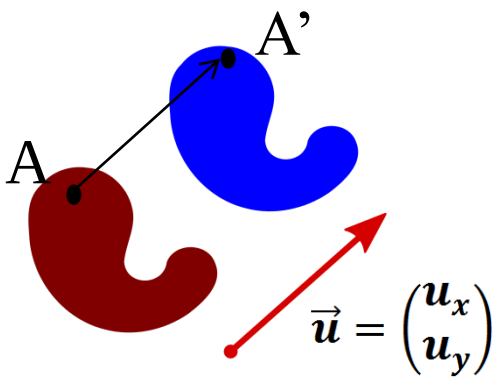
$$\overrightarrow{A_1B_1} = k \cdot \overrightarrow{AB}$$

$$\Leftrightarrow$$

$$\begin{pmatrix} u_{1x} \\ u_{1y} \end{pmatrix} = k \cdot \begin{pmatrix} u_x \\ u_y \end{pmatrix} = \begin{pmatrix} k & 0 \\ 0 & k \end{pmatrix} \begin{pmatrix} u_x \\ u_y \end{pmatrix}$$

Matrice la transformation homothétique

Translation (2D)



$$\overrightarrow{OA'} = \begin{pmatrix} A'_x \\ A'_y \end{pmatrix}$$

$$\overrightarrow{OA} = \begin{pmatrix} A_x \\ A_y \end{pmatrix}$$

$$\overrightarrow{OA} = \overrightarrow{OA'} + \vec{u}$$

$$\Leftrightarrow$$

$$\begin{pmatrix} A'_x \\ A'_y \end{pmatrix} = \begin{pmatrix} A_x + u_x \\ A_y + u_y \end{pmatrix}$$

Expression matricielle:

On se place dans un espace 3D (coordonnées homogènes)

$$\begin{pmatrix} A'_x \\ A'_y \\ 1 \end{pmatrix} = \begin{pmatrix} 1 & 0 & u_x \\ 0 & 1 & u_y \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} A_x \\ A_y \\ 1 \end{pmatrix} = \begin{pmatrix} A_x + u_x \\ A_y + u_y \\ 1 \end{pmatrix}$$

Matrice translation dans l'espace homogène 3D

Transformations dans l'espace (3D) et matrices:

Rotation

$$\begin{pmatrix} x' \\ y' \\ z' \end{pmatrix} = \begin{pmatrix} \cos\varphi & -\sin\varphi & 0 \\ \sin\varphi & \cos\varphi & 0 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} x \\ y \\ z \end{pmatrix}$$

Matrice M de rotation dans le plan (O,x,y)

homothétie de centre O

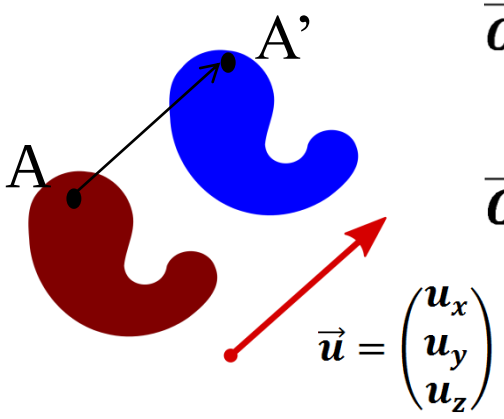
$$\begin{pmatrix} k & 0 & 0 \\ 0 & k & 0 \\ 0 & 0 & k \end{pmatrix}$$

Cas général :

$$M = (\cos\varphi)\begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix} + (1-\cos\varphi)\begin{pmatrix} n_x^2 & n_x n_y & n_x n_z \\ n_x n_y & n_y^2 & n_y n_z \\ n_x n_z & n_y n_z & n_z^2 \end{pmatrix} + (sin\varphi)\begin{pmatrix} 0 & -n_z & n_y \\ n_z & 0 & -n_x \\ -n_y & n_x & 0 \end{pmatrix}$$

Avec (n_x,n_y,n_z) les coordonnées du vecteur unitaire de l'axe de rotation

Translation (3D)



$$\overrightarrow{OA'} = \begin{pmatrix} A'_x \\ A'_y \\ A'_z \end{pmatrix} \qquad \overrightarrow{OA} = \begin{pmatrix} A_x \\ A_y \\ A_z \end{pmatrix}$$

$$\overrightarrow{OA} = \overrightarrow{OA'} + \vec{u} \Leftrightarrow \begin{pmatrix} A'_x \\ A'_y \\ A'_z \end{pmatrix} = \begin{pmatrix} A_x + u_x \\ A_y + u_y \\ A_z + u_z \end{pmatrix}$$

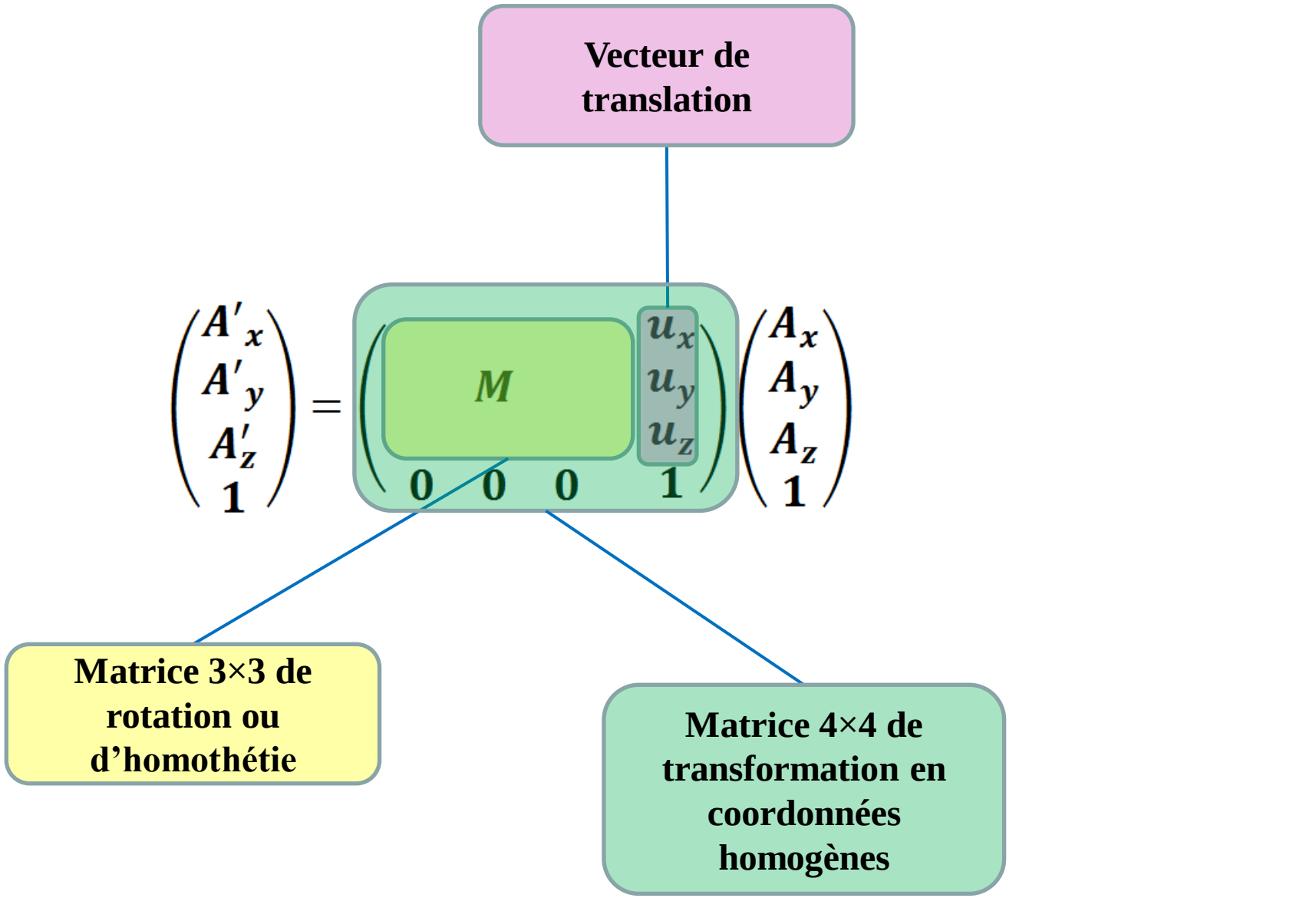
Expression matricielle:

On se place dans un espace 4D (coordonnées homogènes)

$$\begin{pmatrix} A'_x \\ A'_y \\ A'_z \\ 1 \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 & u_x \\ 0 & 1 & 0 & u_y \\ 0 & 0 & 1 & u_z \\ 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} A_x \\ A_y \\ A_z \\ 1 \end{pmatrix} = \begin{pmatrix} A_x + u_x \\ A_y + u_y \\ A_z + u_z \\ 1 \end{pmatrix}$$

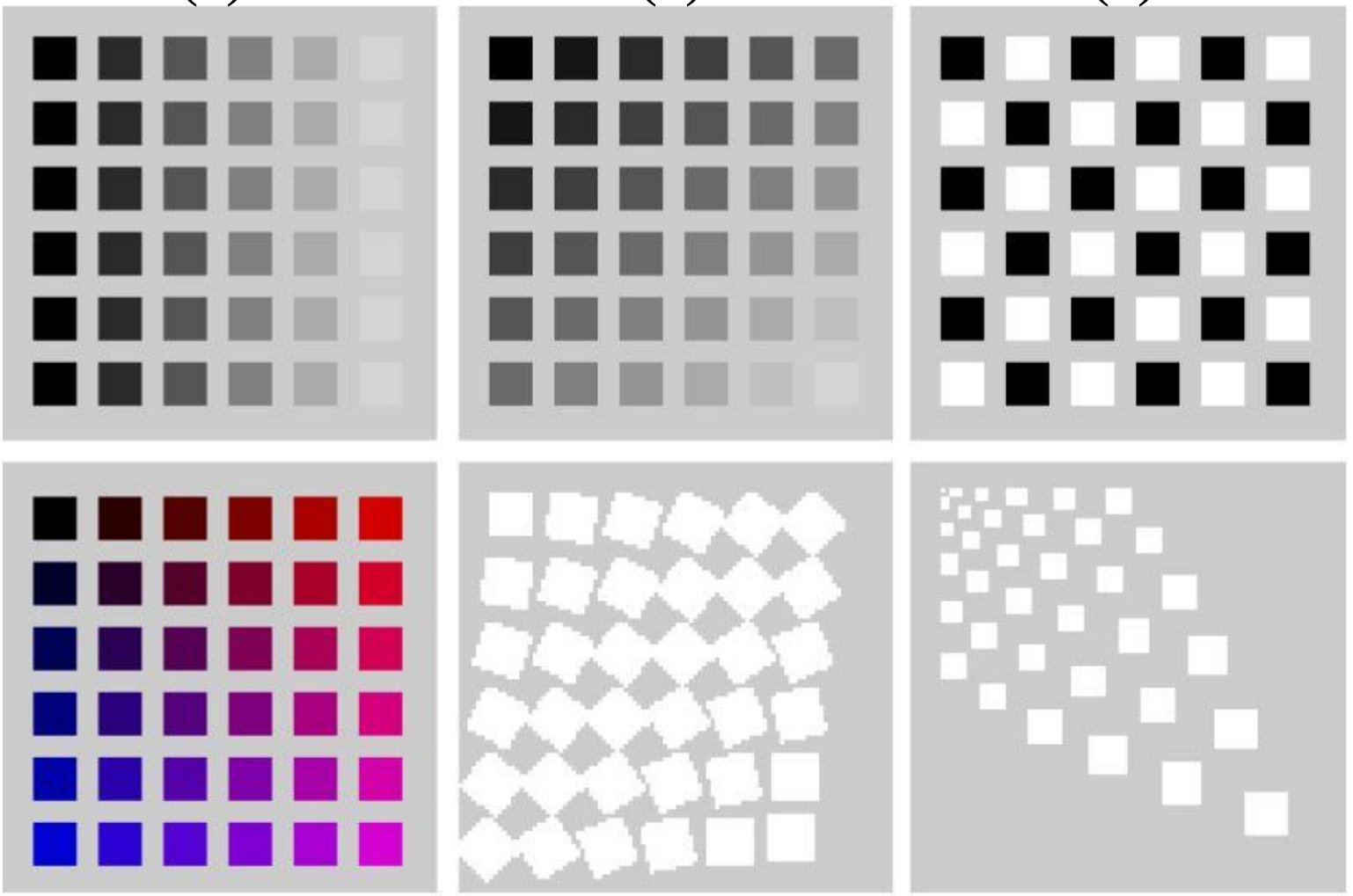
Matrice translation dans l'espace homogène 4D

Cas général en (3D)



Exercice: Réaliser les figures suivantes en utilisant les méthodes **fill(), scale(), rotate()**, et en utilisant deux boucles imbriquées.

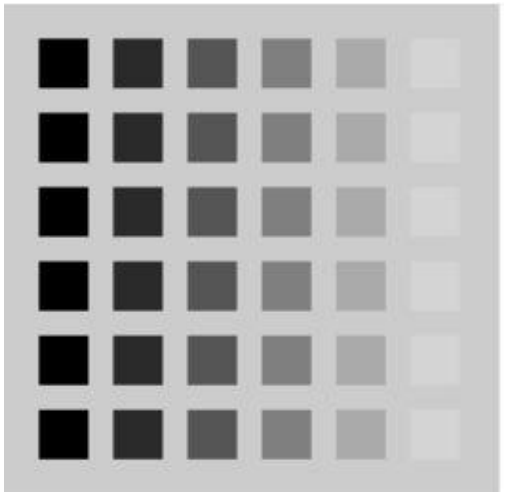
(1) (2) (3)



(4) (5) (6)

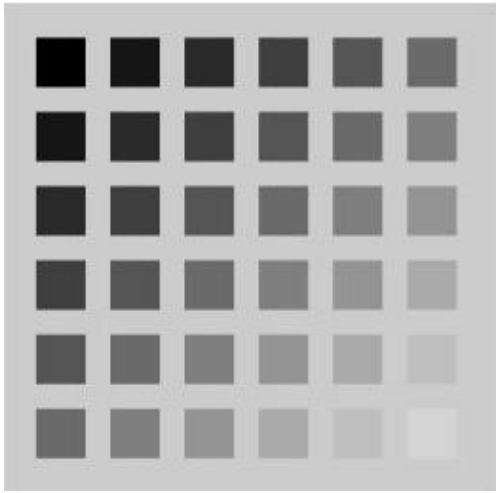
Correction: Exemple 1

```
float gray =0;
translate(7, 7);

//1ère boucle (largeur)
for (int y = 0; y < 6; y ++) {
  //2ième boucle (hauteur)
  for (int x = 0; x < 6; x ++) {
    gray = map(x,0,6,0,255);
    fill(gray);
    rect(x * 15, y * 15, 10, 10);
  }
}
```

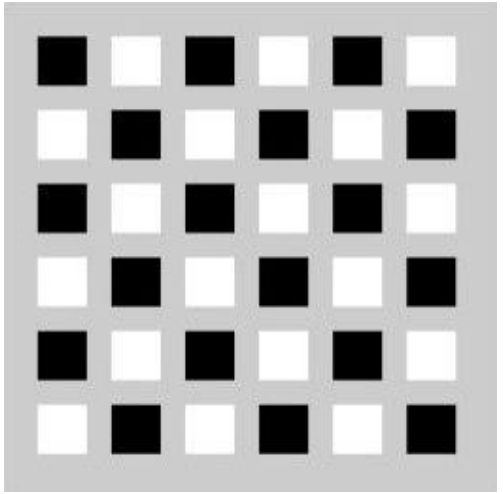


Correction: Exemple 2

```
float gray =0;
translate(7, 7);

//1ère boucle (largeur)
for (int y = 0; y < 6; y ++) {
  //2ième boucle (hauteur)
  for (int x = 0; x < 6; x ++) {
    gray = map(x+y,0,12,0,255);
    fill(gray);
    rect(x * 15, y * 15, 10, 10);
  }
}
```



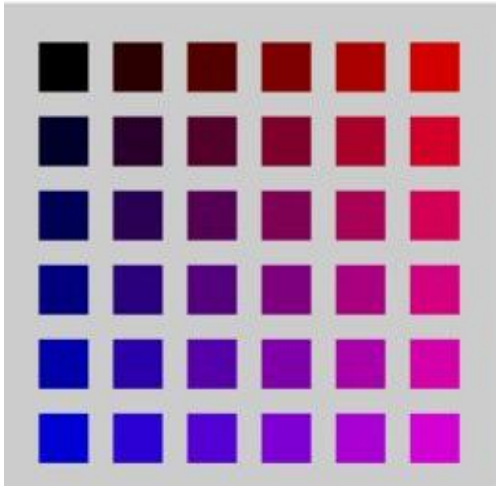
Correction: Exemple 3



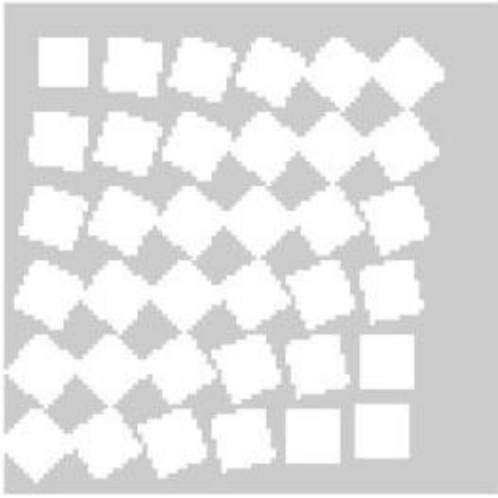
```
float gray =0;
translate(7, 7);

//1ère boucle (largeur)
for (int y = 0; y < 6; y ++) {
  //2ième boucle (hauteur)
  for (int x = 0; x < 6; x ++) {
    /* si x+y est paire: carré noir, gray = 0. (le reste de la division euclidienne de x+y par 2 est nul */
    /* si x+y est impaire: carré blanc gray = 255. (le reste de la division euclidienne de x+y par 2 est égale à 1) */
    gray = (x+y)%2;

    //débug
    float z = x+y;
    println("x=" + x + ", y="+ y + ", x+y=" + z + ",  gray= " + gray);
    // end débug

    noStroke();
    fill(gray*255);
    rect(x * 15, y * 15, 10, 10);
  }
}
```

Correction: Exemple 4

```
float gray1, gray2;
translate(7, 7);

//1ère boucle (largeur)
for (int y = 0; y < 6; y ++) {
  //2ième boucle (hauteur)
  for (int x = 0; x < 6; x ++) {
    gray1 = map(x,0,5,0,255);
    gray2 = map(y,0,5,0,255);
    noStroke();
    fill(gray1,0,gray2);
    rect(x * 15, y * 15, 10, 10);
  }
}
```



Correction: Exemple 5



```
float angle =0;
translate(7, 7);

//1ère boucle (largeur)
for (int y = 0; y < 6; y ++) {
  //2ième boucle (hauteur)
  for (int x = 0; x < 6; x ++) {
    pushMatrix(); // les transformation de translations et rotations ne s'appliquent qu'au carré dessiné dans la boucle
    noStroke();
    fill(255,255,255);
    // 1ère étape: on translate à la position voulue
    translate(x * 15, y * 15);
    // 2ième étape: on on effectue la rotation du carré
    angle = map(x+y,0,12,0,PI/2);
    rotate(angle);
    // on dessine le carré en prenant en compte des transformations précédentes
    rect(0,0, 10, 10);

    popMatrix();
  }
}
```

Correction: Exemple 6

```
float gray =255, echelle;
translate(7, 7);

//1ère boucle (largeur)
for (int y = 0; y < 6; y ++) {
  //2ième boucle (hauteur)
  for (int x = 0; x < 6; x ++) {

    pushMatrix();
    echelle = norm(x+y,0,12);

    //debug
   // println("echelle="+ echelle);


    fill(gray);
    noStroke();
    scale(echelle);
    translate(x * 15, y * 15);
    rect(0,0, 10, 10);
    popMatrix();
  }
}
```

14. APPLICATION DES TESTS CONDITIONNELS À LA PROGRAMMATION DE « PONG »

Objectif ?

Animer la balle

Gérer les collisions

Faire évoluer le sketch: animer une balle…

- Classe « **balle** »:
  - attributs:
    - position suivant x
    - position suivant y
    - couleur
    - deplacementX
    - deplacementY
  - Méthodes membres:
    - dessiner la balle
    - mettre en mouvement la balle (bouge)

Exercice:

1. **Modifier la classe « balle » en incorporant la méthode membre « bouge » qui prend en compte les déplacements suivant X et Y. On utilisera les attributs deplacementX et deplacementY, on ajoutera 2 paramètres d'entrée au constructeur permettant d'initialiser ces 2 variables…**

2. **Activer le déplacement d'une balle dans la méthode draw().**

```
/*Déclaration et création d'une instance de l'objet Balle à la position (100,100) de couleur blanche*/
Balle maBalle = new Balle(100, 100, color(255), 2,2);

void setup() {
smooth(); //Lissage des dessins
size(400, 200); //Taille de la fenêtre
}

void draw() {
background(0); //On dessine un fond noir
noStroke(); //On supprime le contour
maBalle.bouge();
maBalle.display(); //Affichage de la balle
}

class Balle {
        /*Déclaration des attributs de la classe « balle »
        (paramètre des balles)*/
        float x,y;
        color couleur;
        float  deplacementX, deplacementY;

        //Constructeur de la balle
Balle (float nouvX, float nouvY, color nouvCouleur, float depX, float depY) {
        x = nouvX;
        y = nouvY;
        couleur = nouvCouleur;
        deplacementX = depX;
        deplacementY = depY;

        }
```

```
//Déclaration des méthodes membres de la classe « balle »
        //Dessiner  la balle
        void display() {
        fill(couleur);
        ellipse(x, y, 40, 40);
        }


        //déplacer la balle
        void bouge() {
        /* dépX et depY sont les déplacement de la balle entre
        2 frames */
        x += deplacementX;
        y += deplacementY;
        }


}
```

Exercice: A partir du programme précédent:

1. **Modifier la classe « balle » en incorporant la méthode membre « testCollision » qui prend en compte les collisions avec le bord.**
2. **Activer le test des collisions d'une balle dans la méthode draw().**

⇨ Classe « **balle** »:

↳ attributs:

- position suivant x
- position suivant y
- couleur

↳ Méthodes membres:

- dessiner la balle
- mettre en mouvement la balle (bouge)
- prendre en compte les collisions avec le bord (testCollision)

```
/*Déclaration et création d'une instance de l'objet Balle à la position (100,100) de couleur blanche*/
Balle maBalle = new Balle(100, 100, color(255), 2,2);

void setup() {
smooth(); //Lissage des dessins
size(400, 200); //Taille de la fenêtre
}

void draw() {
background(0); //On dessine un fond noir
noStroke(); //On supprime le contour
maBalle.bouge();
maBalle.testCollision();
maBalle.display(); //Affichage de la balle
}

class Balle {
        /*Déclaration des attributs de la classe « balle » (paramètre des balles)*/
        float x,y;
        color couleur;
        float  deplacementX, deplacementY;

        //Constructeur de la balle
Balle (float nouvX, float nouvY, color nouvCouleur, float depX, float depY) {
        x = nouvX;
        y = nouvY;
        couleur = nouvCouleur;
        deplacementX = depX;
        deplacementY = depY;

        }
```

```
//Déclaration des méthodes membres de la classe « balle »
        //Dessiner  la balle
        void display() {
        fill(couleur);
        ellipse(x, y, 40, 40);
        }


        //déplacer la balle
        void bouge() {
        /* dépX et depY sont les déplacement de la balle entre 2 frames */
        x += deplacementX;
        y += deplacementY;
        }

        //prendre en compte les collisions avec les bords
        /*Si la balle touche une mur, elle rebondit, i.e. on inverse le déplacement suivant x et suivant y */
        void testCollision() {
        //si le bord de la balle touche une extrémité verticale
                if (x > width-20 || x < 20) {
                deplacementX *= -1;
                }
        //si le bord de la balle touche une extrémité horizontale
                if (y > height-20 || y < 20) {
                deplacementY *= -1;
                }
        }

}
```

Exercice: modifier le programme précédents pour afficher plusieurs balles de couleurs différentes aux position avec différentes positions initiales et différentes vitesse de déplacement…

Vous pourrez utiliser une liste pour gérer les différentes balles instanciées…

```
// nombre de balles
int nbreBalle = 3;

/* liste des instances de toutes les balles
Balles[]  balles = new Balle[nbreBalle];

void setup() {
        smooth(); //Lissage des dessins
        size(400, 200); //Taille de la fenêtre

        // création de plusieurs balles blanches au centre de
        l'écran
        for(int i=0; i<nbreBalle; i++){
        balles(i) = new Balle(width/2, height/2, color(255));
        }
}
void draw() {
        background(0); //On dessine un fond noir
        noStroke(); //On supprime le contour

        //Affichage de toutes les balles
        for(int i=0; i<nbreBalle; i++){
                balles[i].bouge();
                balles[i].testCollision();
                balles[i].display();
        }
}

class Balle {
        /*Déclaration des attributs de la classe « balle »
        (paramètre des balles)*/
        float x,y;
        color couleur;
        float  deplacementX, deplacementY;
        //Constructeur de la balle
Balle (float nouvX, float nouvY, color nouvCouleur, float
depX, float depY) {
        x = nouvX;
        y = nouvY;
        couleur = nouvCouleur;
        deplacementX = depX;
        deplacementY = depY;
        }
//Déclaration des méthodes membres de la classe « balle »
        //Dessiner  la balle
        void display() {
        fill(couleur);
        ellipse(x, y, 40, 40);
        }

        //déplacer la balle
        void bouge() {
        /* dépX et depY sont les déplacement de la balle entre
        2 frames */
        x += deplacementX;
        y += deplacementY;
        }
        //prendre en compte les collisions avec les bords
        void testCollision() {
        //si le bord de la balle touche une extrémité verticale
                if (x > width-20 || x < 20) {
                deplacementX *= -1;
                }
        //si le bord de la balle touche une extrémité horizontale
                if (y > height-20 || y < 20) {
                deplacementY *= -1;
                }
        }
}
```

Comment gérer plusieurs balles ?

Les listes !!!

15. LES LISTES ET TABLEAUX À PLUSIEURS ENTRÉES

15.1 Les tableaux à une seule entrée: les listes

- Les listes permettent de mémoriser sous une seule variable un ensemble de grandeurs de **même type.**

Exemple:

Dans une liste nommée « tailles », de type « *float* » on mémorise les tailles de 3 personnes dans la salle. Chaque valeur de la liste est de type *float*.

- float taille[] = {1.60, 1.62, 1.85} ;

- L'ordinateur crée un tableau de 3 cases mémoires, chacune ayant la taille d'un nombre flottant (4 octets)

0	1	2
1,60	1.62	1.85

Comment accéder à une valeur de la liste ?

Tableau à **une entrée**, dont chaque élément est repéré par **un indice.**

- taille[0] renvoie la valeur 1.60.
- taille[1] renvoie la valeur 1.62.
- taille[2] renvoie la valeur 1.85.

Le 1er indice est l'indice 0 !!!

Exemple: println(taille[3]);

Autre méthode pour créer une liste:

1. On crée une liste vide en précisant le nombre d'élément et le type des données.

2. On rempli la liste (dans un ordre quelconque si on le souhaite).

Syntaxe:

// création d'une nouvelle liste vide de type float contenant 3 éléments

```
float taille[] = new float[3] ;
```

L'instruction « new » crée un nouvel objet, instance d'une classe « liste » de type float

// remplissage de la liste nouvellement créée

```
taille[0] = 1.60 ;
taille[1] = 1.62 ;
taille[2] = 1.85 ;
```

Comment accéder au nombre total d'éléments dans une liste?

En appelant la variable membre « length » de l'objet « taille »

Syntaxe: taille.length renvoie le nombre 3

Exemple d'application: en utilisant une boucle « for » calculer de la moyenne des tailles contenue dans la liste du même nom, et afficher la résultat.

```
// création d'une nouvelle liste vide de type float contenant 3 éléments
float taille[] = new float[3] ;
// remplissage de la liste nouvellement créée
taille[0] = 1.60 ;
taille[1] = 1.62 ;
taille[2] = 1.85 ;
// déclaration d'une nouvelle variable de type float
float moyenne = 0;
// déclaration d'une nouvelle variable de type int  contenant le nb d'élément de la liste
int nbTaille = taille.length;
for (int i = 0, i<nbTaille , i++){
        moyenne += taille[i];
}
moyenne /= nbTaille;

Println(" la moyenne des taille  = " + moyenne + "m");
```

Comment ajouter des éléments à une liste ?

- Utiliser les fonctions append() et expand()
  - Voir l'aide en ligne
- Voir aussi les fonctions shorten(), subset(), splice(), concat(), arrayCopy()
  - Permet d'extraire et de manipuler une partie de la liste

Comment trier les éléments d'une liste ?

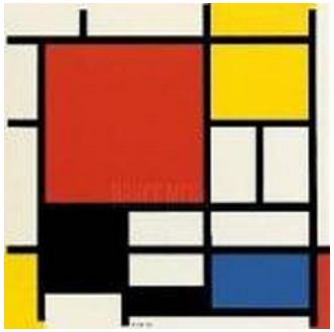
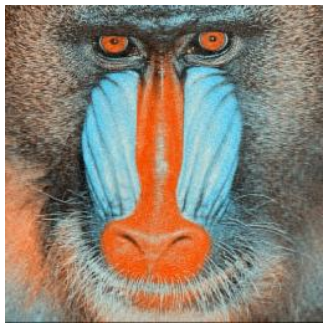
- Utiliser les fonctions sort() et reverse()
  - Voir l'aide en ligne

15.2 Suite (liste) d'images

Exemple: Importation de 5 images

Manipulation préalable:

↳ Placer 5 images dans le dossier « data »


Ecrire le programme suivant:

```
size(500,500);

// création d'une nouvelle liste vide de type PImage contenant 5 éléments
Pimage[] MesImages = new PImage[5] ;

// remplissage de la liste nouvellement créée
MesImages[0] = loadImage("image_0.jpg");
MesImages[1] = loadImage("image_1.jpg");
MesImages[2] = loadImage("image_2.jpg");
MesImages[3] = loadImage("image_3.jpg");
MesImages[4] = loadImage("image_4.jpg");

// Affichage des images
image( images[0], 0, 0);          // méthode:  image(img, x, y)
image( images[1], 50, 0);         // (x,y) : coin supérieur gauche de l'image
image( images[2], 100, 0);
image( images[3], 150, 0);
image( images[4], 200, 0);
```

Méthode plus rapide utilisant la boucle « for »:

```
size(500,500);

// création d'une nouvelle liste vide de type PImage contenant 5 éléments
Pimage[] MesImages = new PImage[5] ;

// remplissage et affichage de la liste nouvellement créée de façon
//automatique

for(int i=0; i<MesImages.length(); i++) {

MesImages[i] = loadImage("image_" + i + ".jpg");

image( MesImages[i], random(width), random(height) );
}
```

Automatisation de l'écriture du nom de chaque image

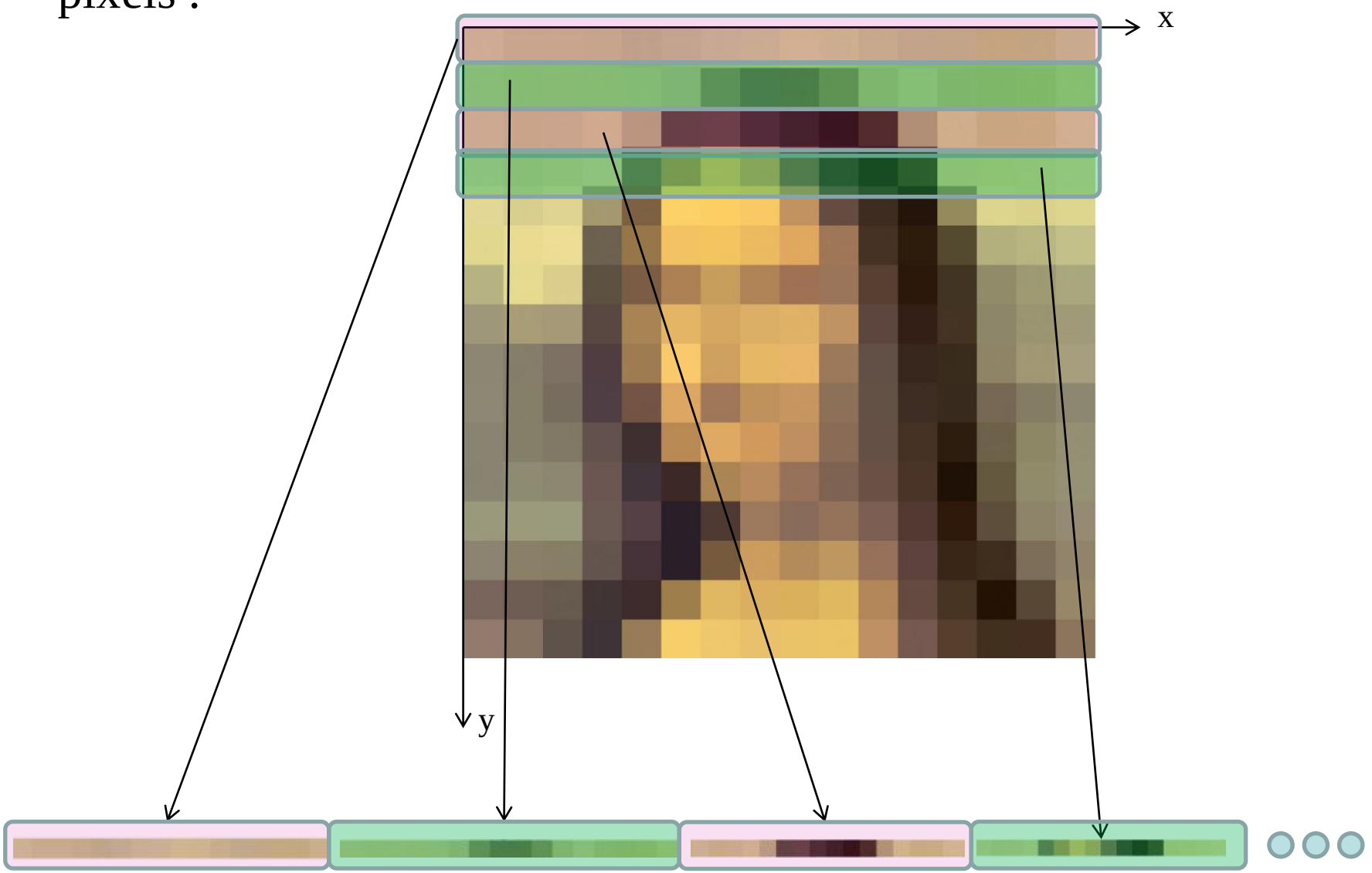
La méthode « length » de l'objet « MesImages », instance de la classe Pimage, renvoie le nombre d'élément de la liste

Positionnement aléatoire des images

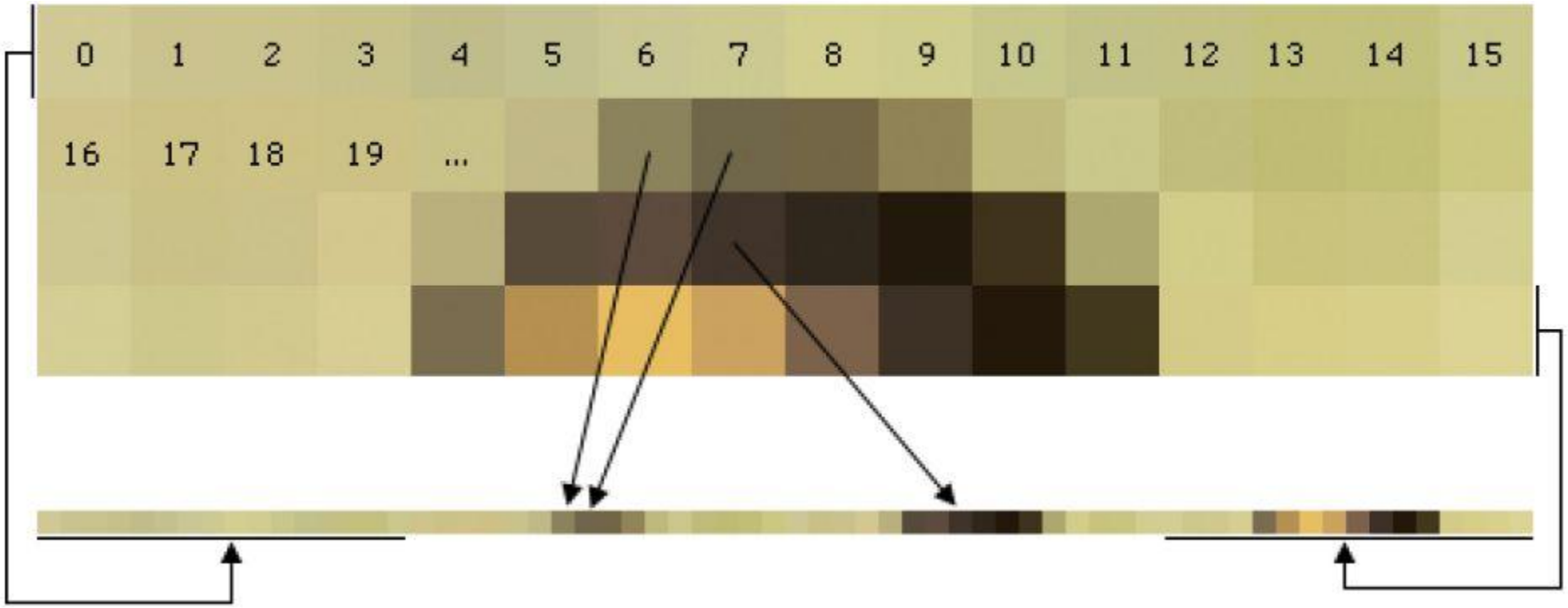
Si « lenght » ne marche pas, remplacer par «size»

15.3 Organisation d'une image comme une suite de pixels

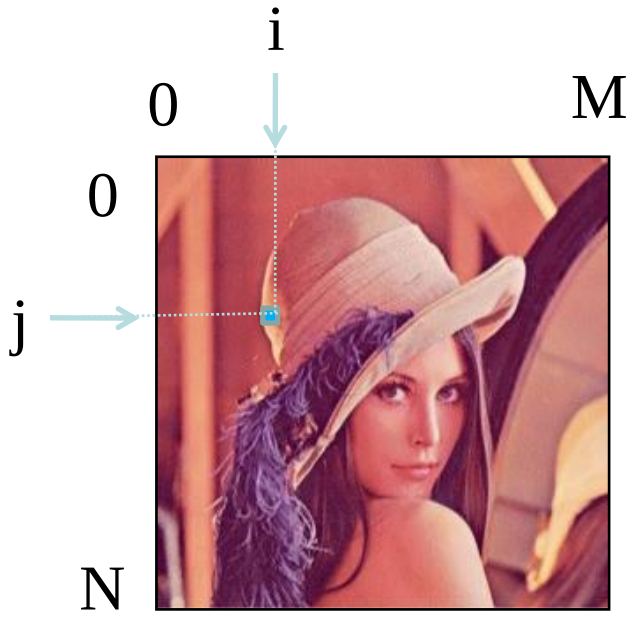
Une image est mémorisée comme une liste (unidimensionnelle) de pixels :



Une image est mémorisée comme une liste (unidimensionnelle) de pixels :



Comment accéder au pixel (i,j) d'une image (pour le modifier…)?

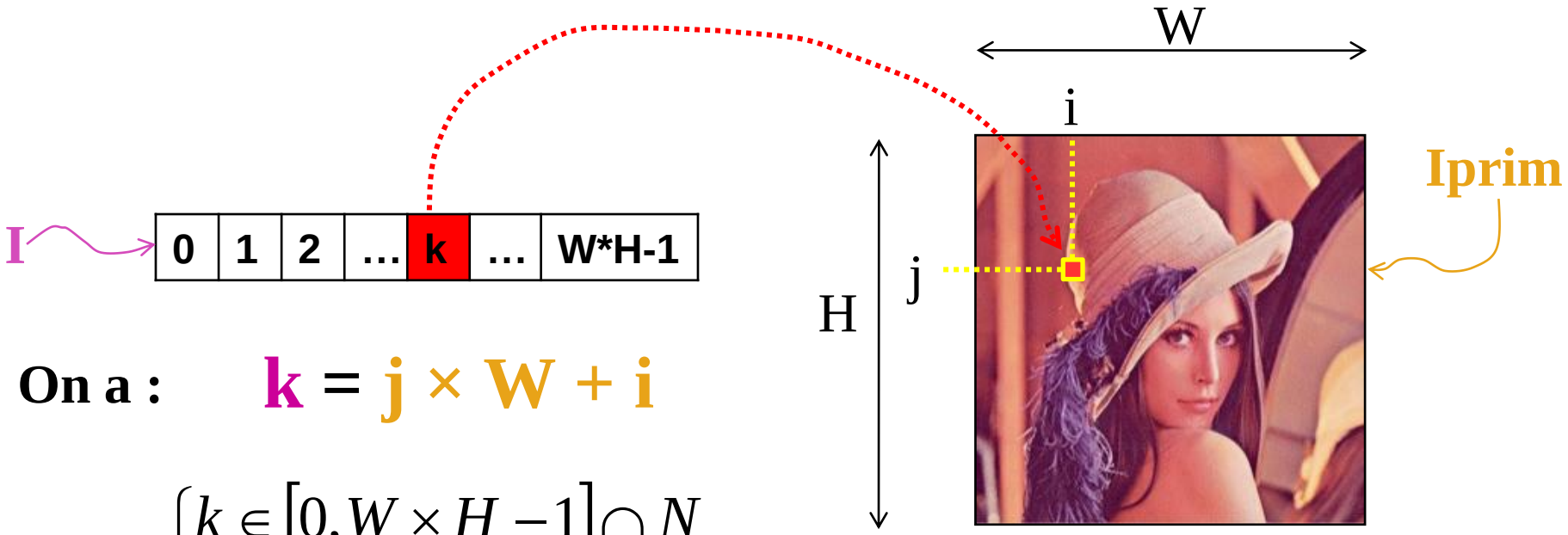


0	1	2	…	…	indice	…	…	N*M-1

indice = j × image.width + i

- **j** : Nombre de lignes
- **image.width** : Largeur de l'image (nombre de pixels dans une ligne)
- **i** : N° de la colonne

Passer d'une image linéaire I à une image matricielle Iprim



On a : $k = j \times W + i$

avec :
$$\begin{cases} k \in [0, W \times H - 1] \cap N \\ i \in [0, W - 1] \cap N \\ j \in [0, H - 1] \cap N \end{cases}$$

avec :
$$\begin{cases} i = k \% W \\ j = E(k / W) \end{cases}$$

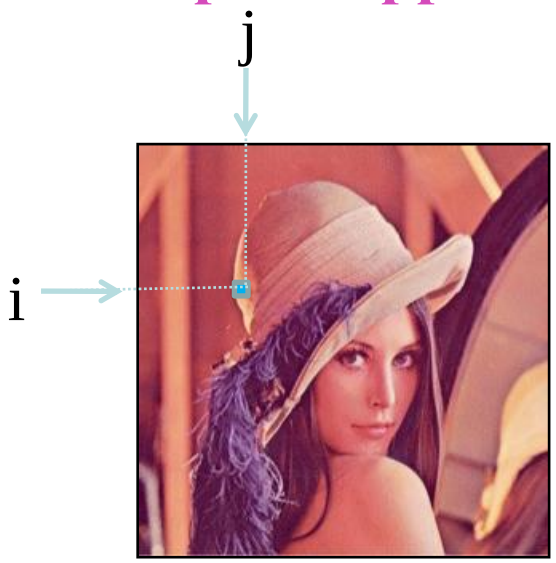
On a donc :

Iprim(i,j) = I(k) = I(j × W + i)

% : reste de la division euclidienne

***E( )*: partie entière**

Exemple d'application: colorier les pixels d'une image




```
size(500,500);
// création d'une variable Lena de type PImage
Pimage Lena ;
// remplissage de la variable Lena
Lena = loadImage("lena.jpg");
// Permission d'accéder à la liste de pixels
Lena.loadPixels();
```

```
// coloriage d'un carré vert autour de l'œil (124,129) (146,141)
for (int i= 124, i<146,i++){
        for (int j= 129, j<141,j++){
        Lena.pixels[i*Lena.width+j] = color( 0,255,0);
        }
}
// Mise à jour des pixels modifiés dans l'image
Lena.updatePixels();

// Affichage de l'image
image(Lena);
```

Exemple d'application: colorier les pixels d'une image

```
// création d'une variable img de type PImage
PImage img = createImage(3, 3, ARG B);
// Permission d'accéder à la liste de pixels
img.loadPixels();
// Coloriage des pixels concernés
img.pixels[0] = color(255, 0, 0);
img.pixels[1] = color(0, 255, 0);
img.pixels[2] = color(0, 0, 255);
img.pixels[3] = color(255, 0, 255);
img.pixels[4] = color(255, 255, 0);
img.pixels[5] = color(0, 255, 255);
img.pixels[6] = color(0, 0, 0);
img.pixels[7] = color(127, 127, 127, 255);
img.pixels[8] = color(255, 255, 255, 0);
// Mise à jour des pixels modifiés dans l'image
img.updatePixels();

// Affichage de l'image ainsi créée avec une talle de 80x80 pixels
image(img, 10, 10, 80, 80);
```

14. APPLICATION DES LISTES À LA PROGRAMMATION DE « PONG »

Objectif ?

- Animer plusieurs balles
- Gérer les collisions de toules les balles

14.1. Exercice:

modifier le programme précédents pour afficher plusieurs balles de couleurs différentes aux position avec différentes positions initiales et différentes vitesse de déplacement…

- **Vous pourrez utiliser une liste pour gérer les différentes balles instanciées…**

```
// nombre de balles
int nbreBalle = 3;

/* liste des instances de toutes les balles
Balles[]  balles = new Balle[nbreBalle];

void setup() {
        smooth(); //Lissage des dessins
        size(400, 200); //Taille de la fenêtre

        // création de plusieurs balles blanches au centre de
        l'écran
        for(int i=0; i<nbreBalle; i++){
        balles(i) = new Balle(width/2, height/2, color(255));
        }
}
void draw() {
        background(0); //On dessine un fond noir
        noStroke(); //On supprime le contour

        //Affichage de toutes les balles
        for(int i=0; i<nbreBalle; i++){
                balles[i].bouge();
                balles[i].testCollision();
                balles[i].display();
        }
}

class Balle {
        /*Déclaration des attributs de la classe « balle »
        (paramètre des balles)*/
        float x,y;
        color couleur;
        float  deplacementX, deplacementY;
        //Constructeur de la balle
Balle (float nouvX, float nouvY, color nouvCouleur, float
depX, float depY) {
        x = nouvX;
        y = nouvY;
        couleur = nouvCouleur;
        deplacementX = depX;
        deplacementY = depY;
        }
//Déclaration des méthodes membres de la classe « balle »
        //Dessiner  la balle
        void display() {
        fill(couleur);
        ellipse(x, y, 40, 40);
        }

        //déplacer la balle
        void bouge() {
        /* dépX et depY sont les déplacement de la balle entre
        2 frames */
        x += deplacementX;
        y += deplacementY;
        }
        //prendre en compte les collisions avec les bords
        void testCollision() {
        //si le bord de la balle touche une extrémité verticale
                if (x > width-20 || x < 20) {
                deplacementX *= -1;
                }
        //si le bord de la balle touche une extrémité horizontale
                if (y > height-20 || y < 20) {
                deplacementY *= -1;
                }
        }
}
```

14.3 Un Objet élégant de gestion des listes dynamiques : ArrayList

Name ArrayList (voir Référence)

Examples

```
// This is a code fragment that shows how to use an ArrayList.
// It won't compile because it's missing the Ball class.

// Declaring the ArrayList, note the use of the syntax "<Ball>" to indicate
// our intention to fill this ArrayList with Ball objects
ArrayList<Ball> balls;

void setup() {
  size(200, 200);
  balls = new ArrayList<Ball>();  // Create an empty ArrayList
  balls.add(new Ball(width/2, 0, 48));  // Start by adding one element
}

void draw() {
  background(255);

  // With an array, we say balls.length. With an ArrayList,
  // we say balls.size(). The length of an ArrayList is dynamic.
  // Notice how we are looping through the ArrayList backwards.
  // This is because we are deleting elements from the list.
  for (int i = balls.size()-1; i >= 0; i--) {
    Ball ball = balls.get(i);
    ball.move();
    ball.display();
    if (ball.finished()) {
      // Items can be deleted with remove().
      balls.remove(i);
    }
  }
}

void mousePressed() {
  // A new ball object is added to the ArrayList, by default to the end.
  balls.add(new Ball(mouseX, mouseY, ballWidth));
}
```

Description

An `ArrayList` stores a variable number of objects. This is similar to making an array of objects, but with an `ArrayList`, items can be easily added and removed from the ArrayList and it is resized dynamically. This can be very convenient, but it's slower than making an array of objects when using many elements. Note that for resizable lists of integers, floats, and Strings, you can use the Processing classes IntList, FloatList, and StringList.

An ArrayList is a resizable-array implementation of the Java List interface. It has many methods used to control and search its contents. For example, the length of the `ArrayList` is returned by its `size()` method, which is an integer value for the total number of elements in the list. An element is added to an `ArrayList` with the `add()` method and is deleted with the `remove()` method. The `get()` method returns the element at the specified position in the list. (See the above example for context.)

For a list of the numerous `ArrayList` features, please read the Java reference description.

Constructor

```
ArrayList<Type>()
ArrayList<Type>(initialCapacity)
```

Parameters

`Type`	Class Name: the data type for the objects to be placed in the ArrayList.
`initialCapacity`	int: defines the initial capacity of the list; it's empty by default

Related

`IntList`
`FloatList`
`StringList`