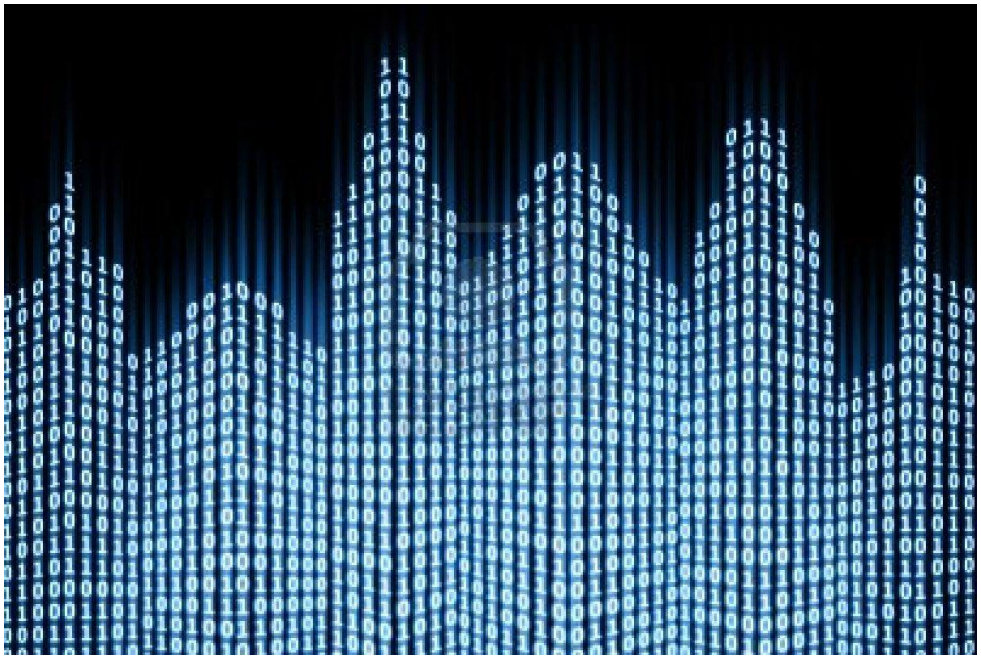
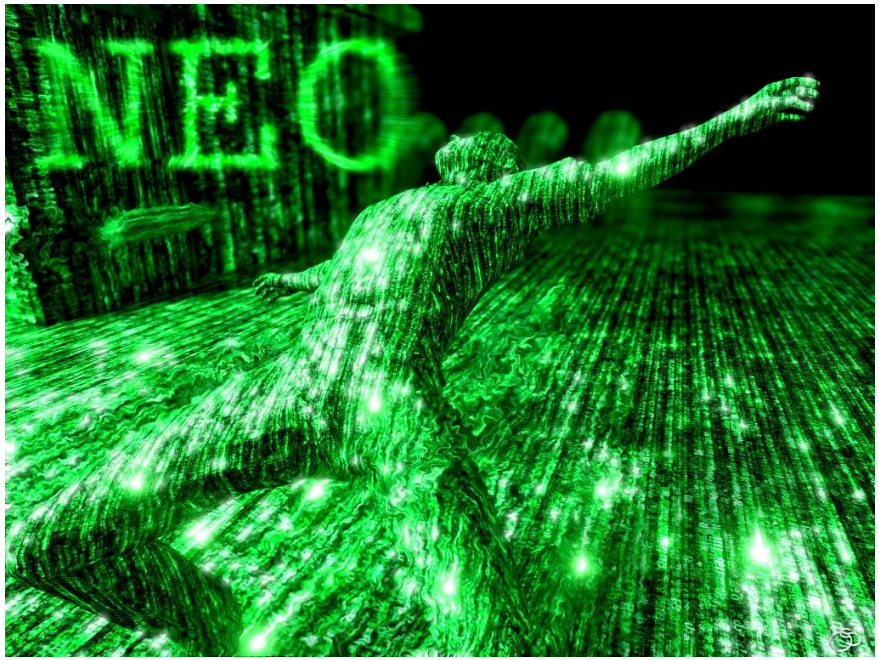
CHAPITRE 2

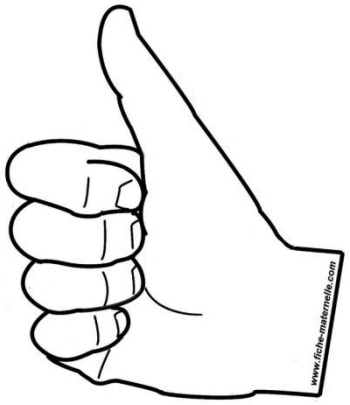
IMAGES NUMÉRIQUES

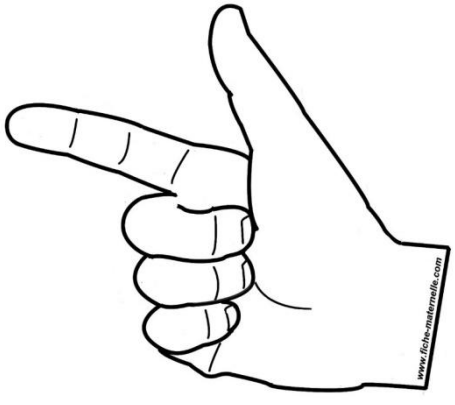


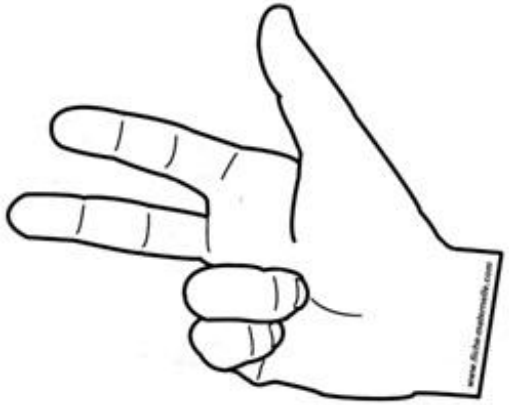


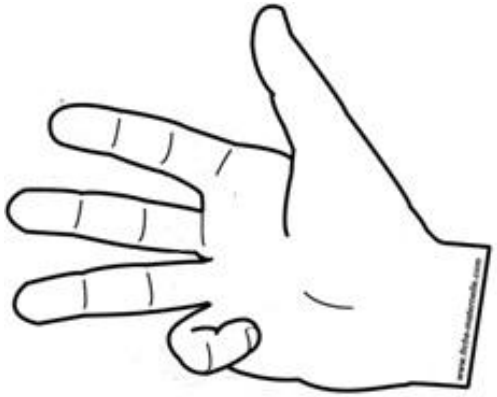
Introduction Rappel sur la base binaire

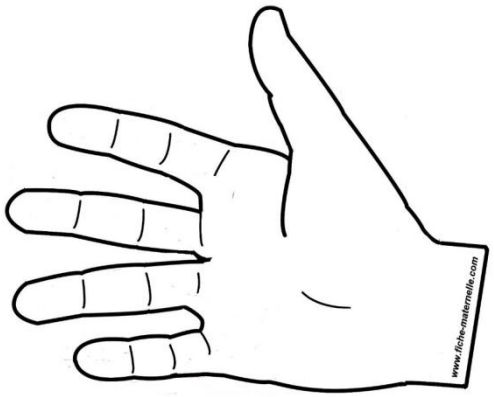
Q: Comment un enfant fait-il généralement pour compter?



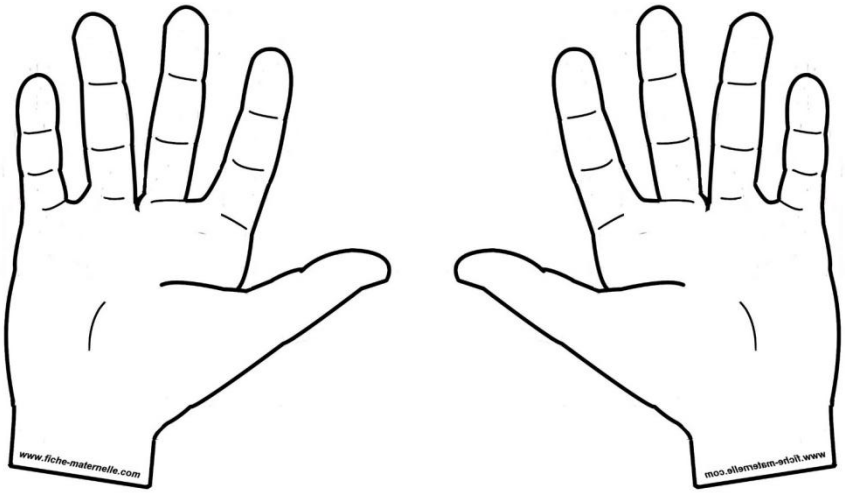








Etc…



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	

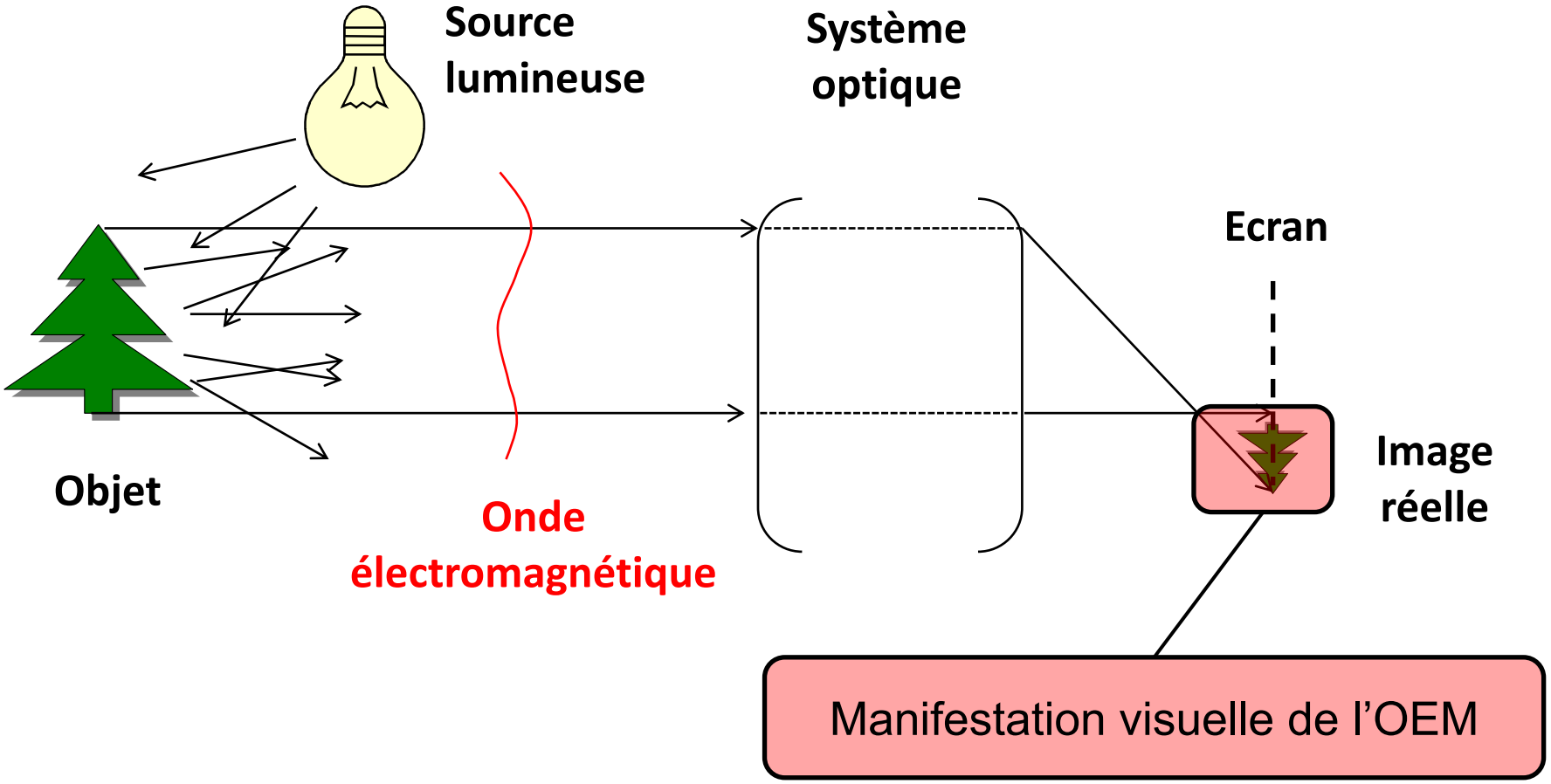
L'octet est l'unité de stockage en mémoire

I. Images en niveaux de gris

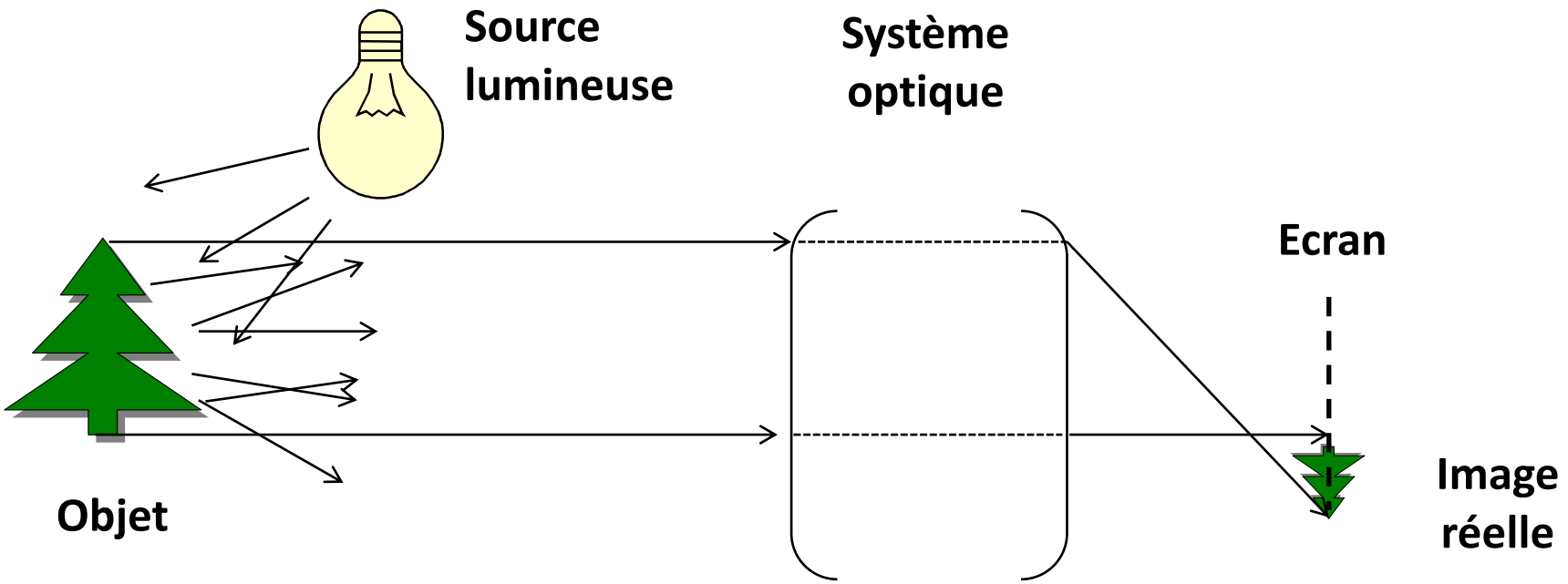
1.1. De l'image Analogique à l'image numérique

Q: comment former une image sur un écran ?

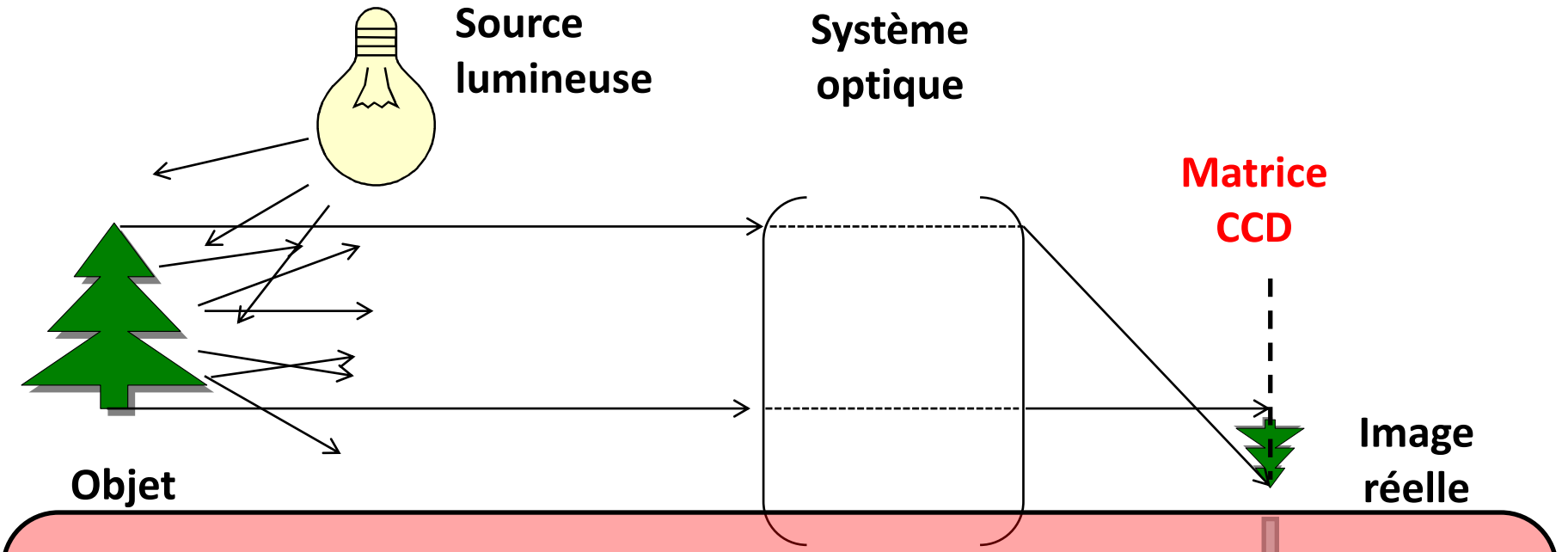
Avec un système optique (objectif, lentille, etc…)



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

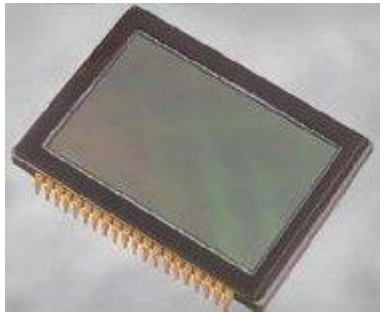
Convertisseur courant-tension

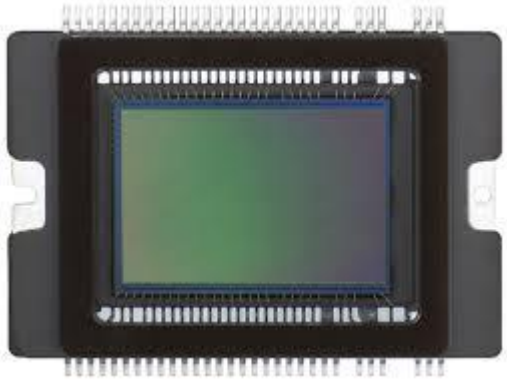
Etape de numérisation de l'image

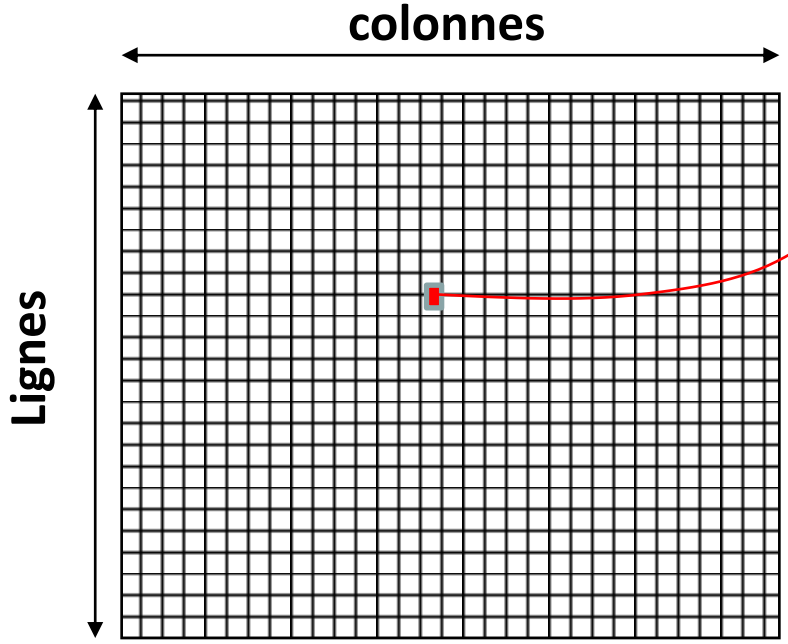
Quantification: Passage d'une tension analogique mesurée à une valeur numérique codée sur un sur un nombre donné de « bit » correspondant à un nombre discret et fini de niveaux de gris entre le noir (intensité lumineuse nulle reçue par le pixel) et le blanc (intensité maximale qu'il est possible d'enregistrer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

1.2. Fonctionnement de la matrice CCD

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

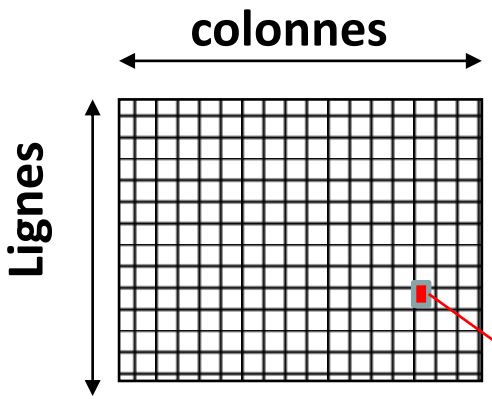


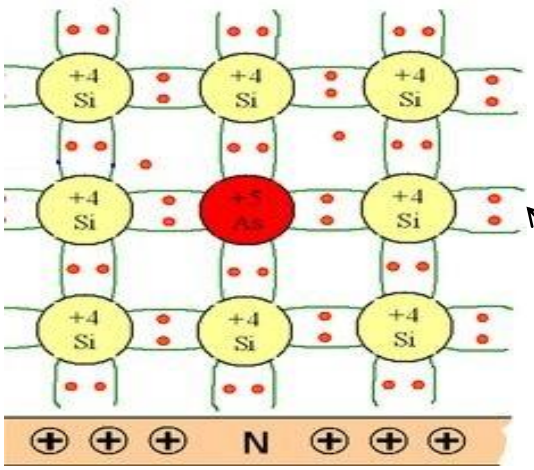


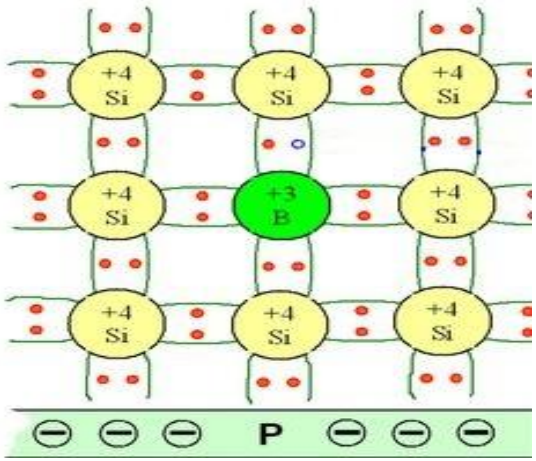


Pixel = cellule photosensible

Matériau semi-conducteur



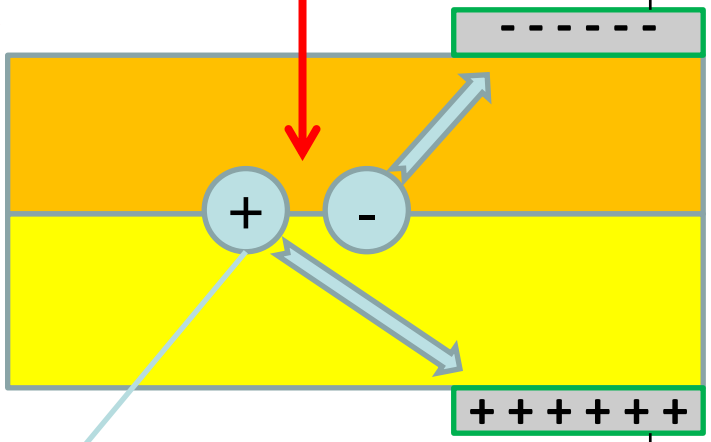




Silicium dopé N

Silicium dopé P

Photon incident



Puits d'électrons

Puits de trous

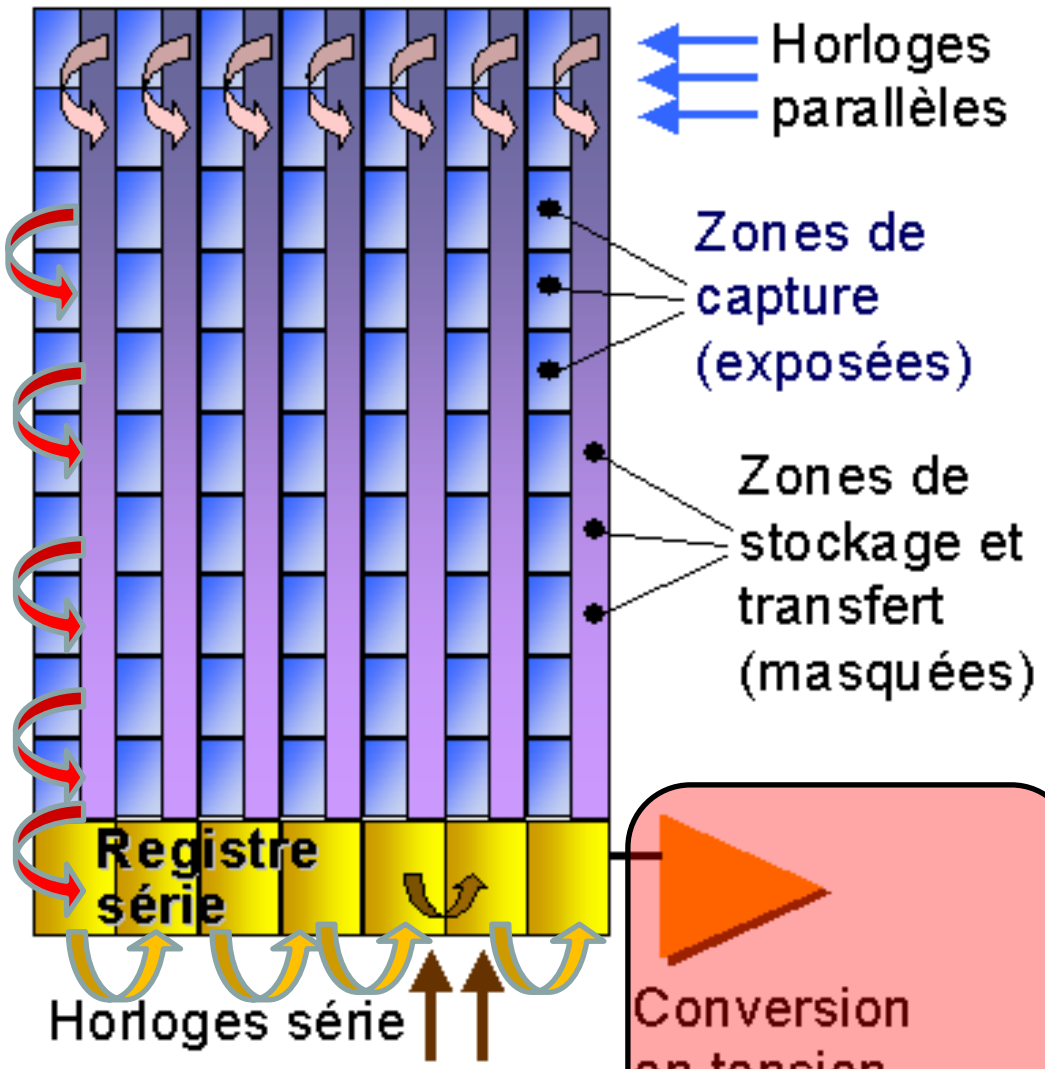
Création d'une paire électron -trou

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

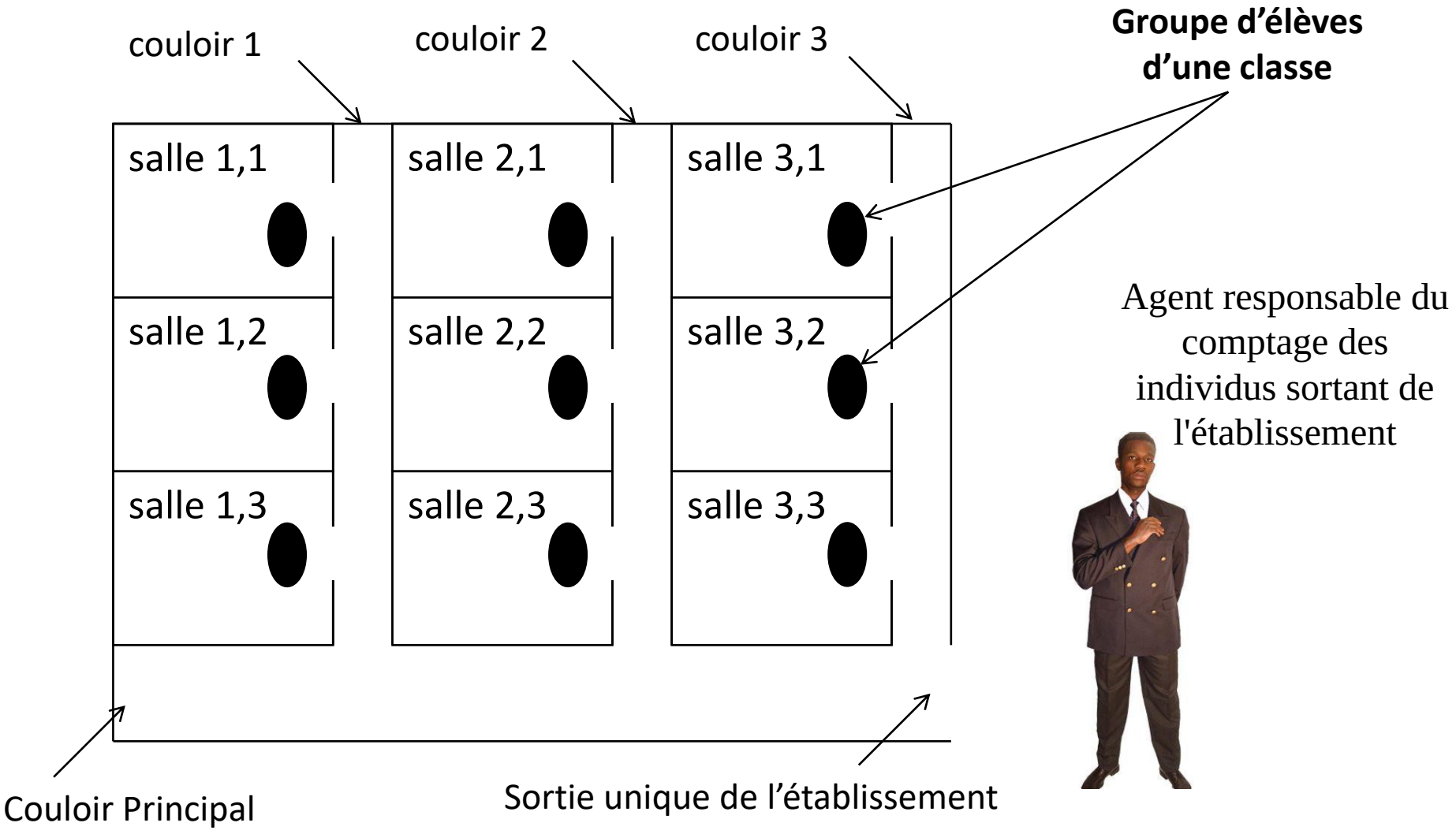


Mouvement de charges = courant électrique

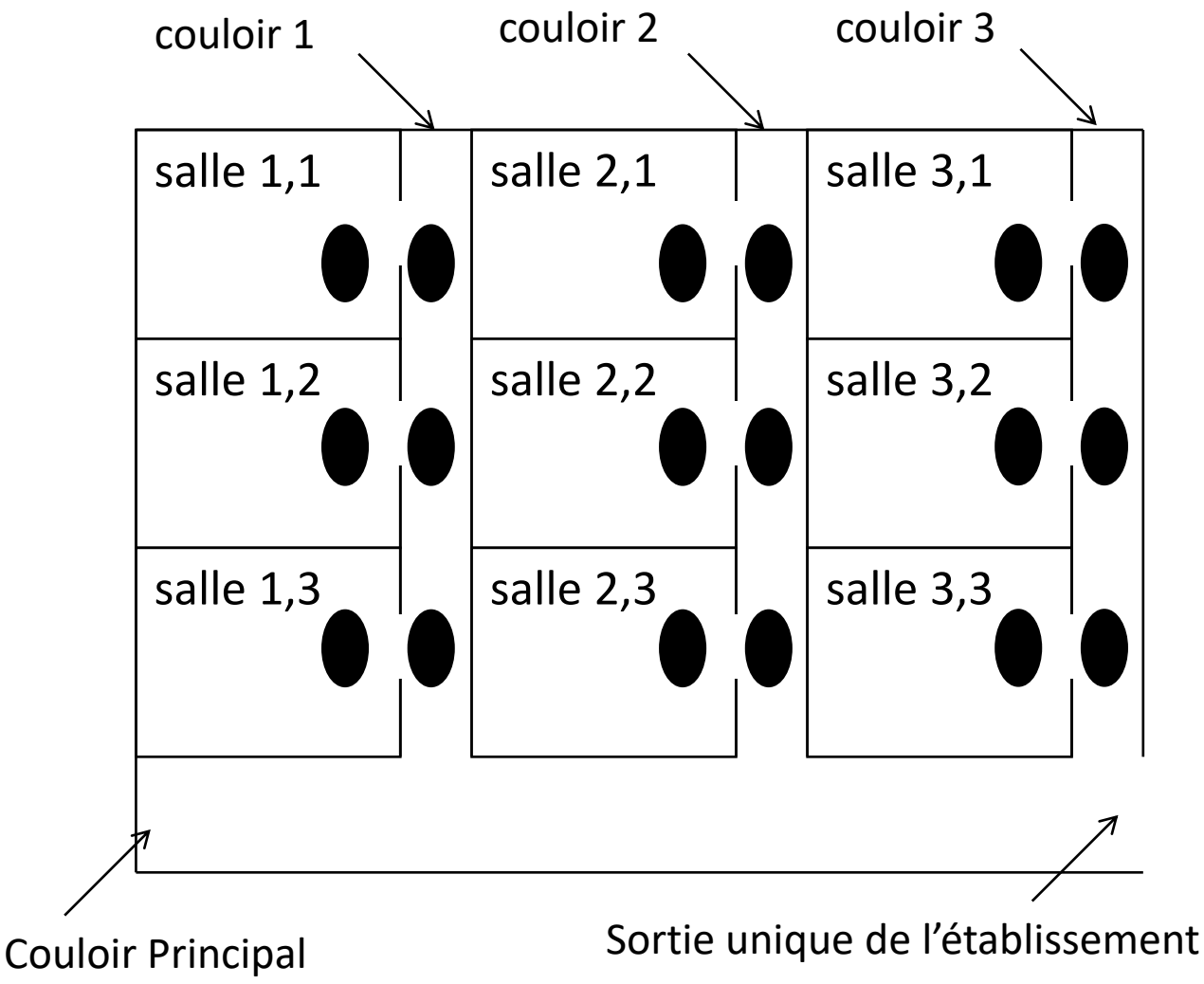
Nécessité de convertir le courant en tension pour la mesure

Processus de comptage

Par analogie: évacuation d'un établissement scolaire



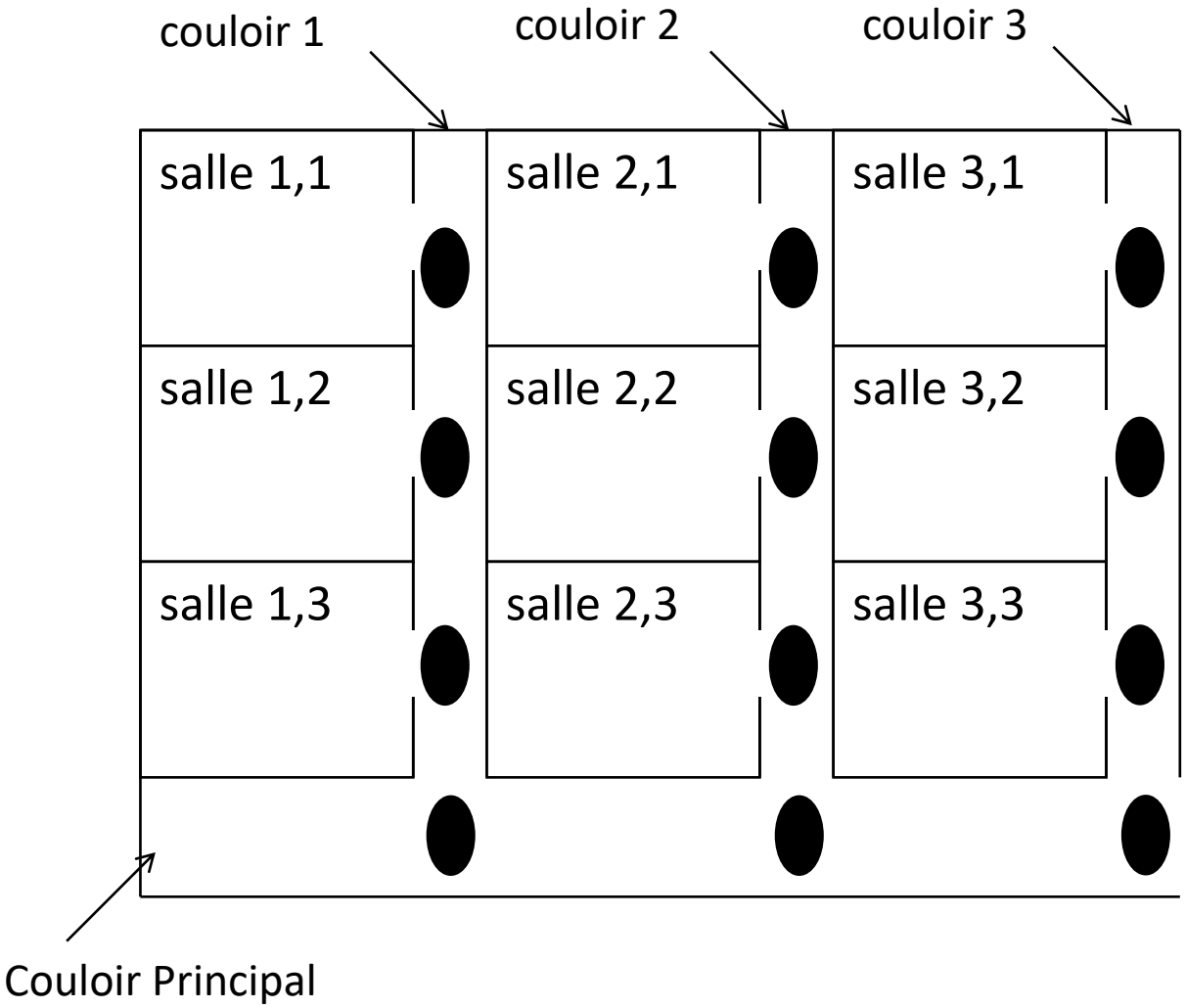
Etape 1: Les élèves de la classe sortent dans le couloir et restent groupés dans le couloir



Agent responsable du comptage des individus sortant de l'établissement

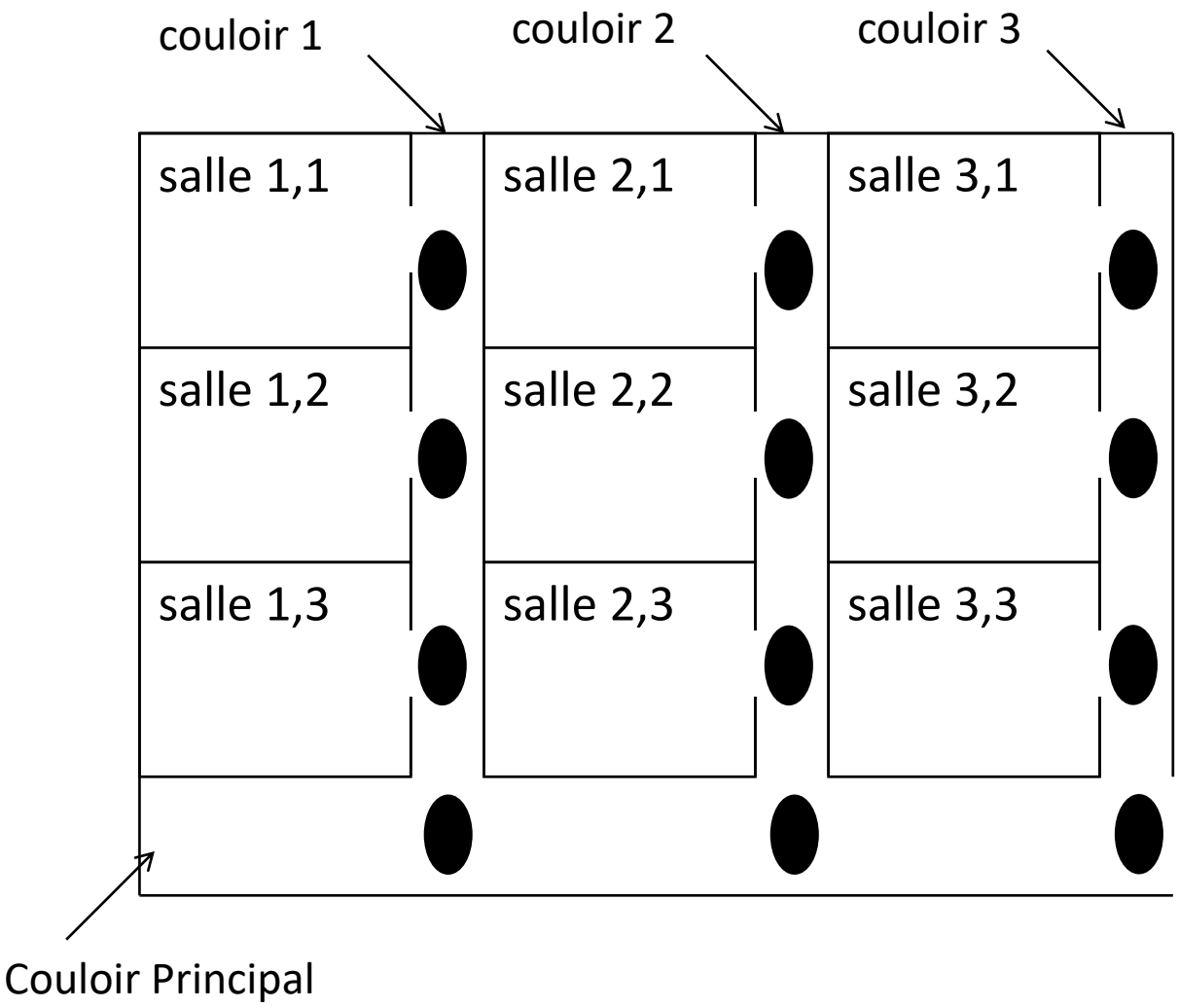
Etape 2: On procède à l'évacuation des élèves dans un ordre bien défini, couloir par couloir.

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



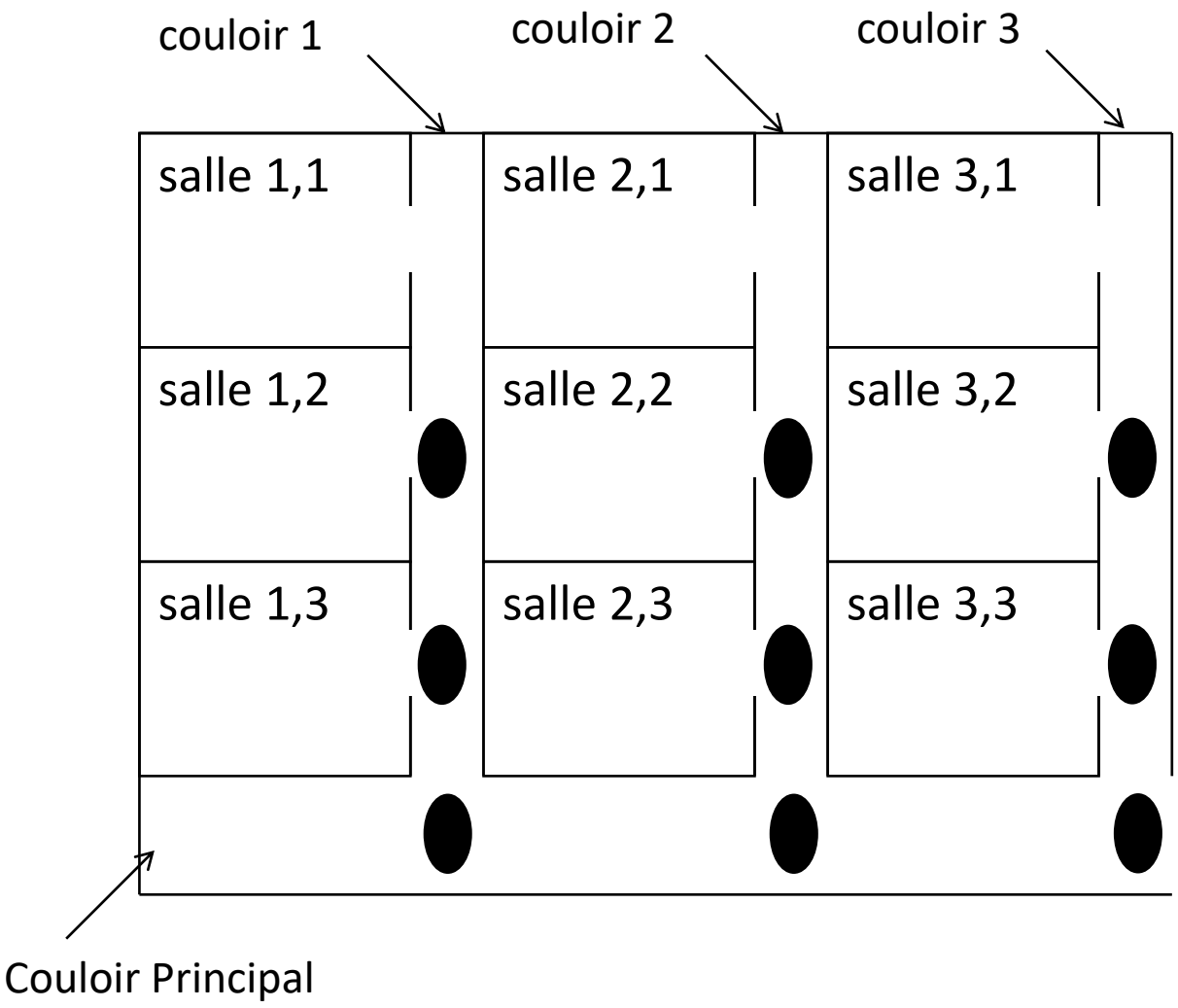
Agent responsable du comptage des individus sortant de l'établissement

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.



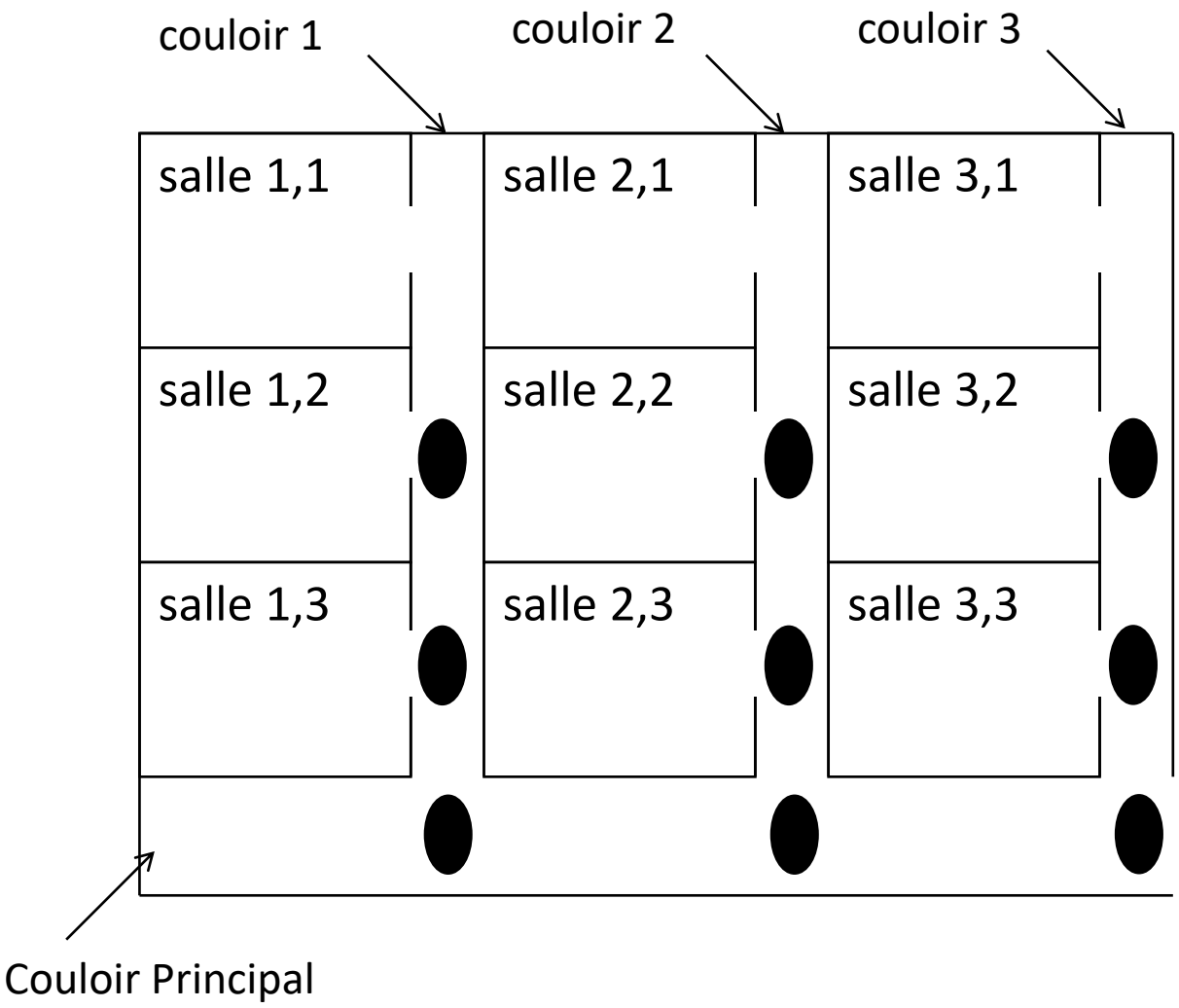
Agent responsable du comptage des individus sortant de l'établissement

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:



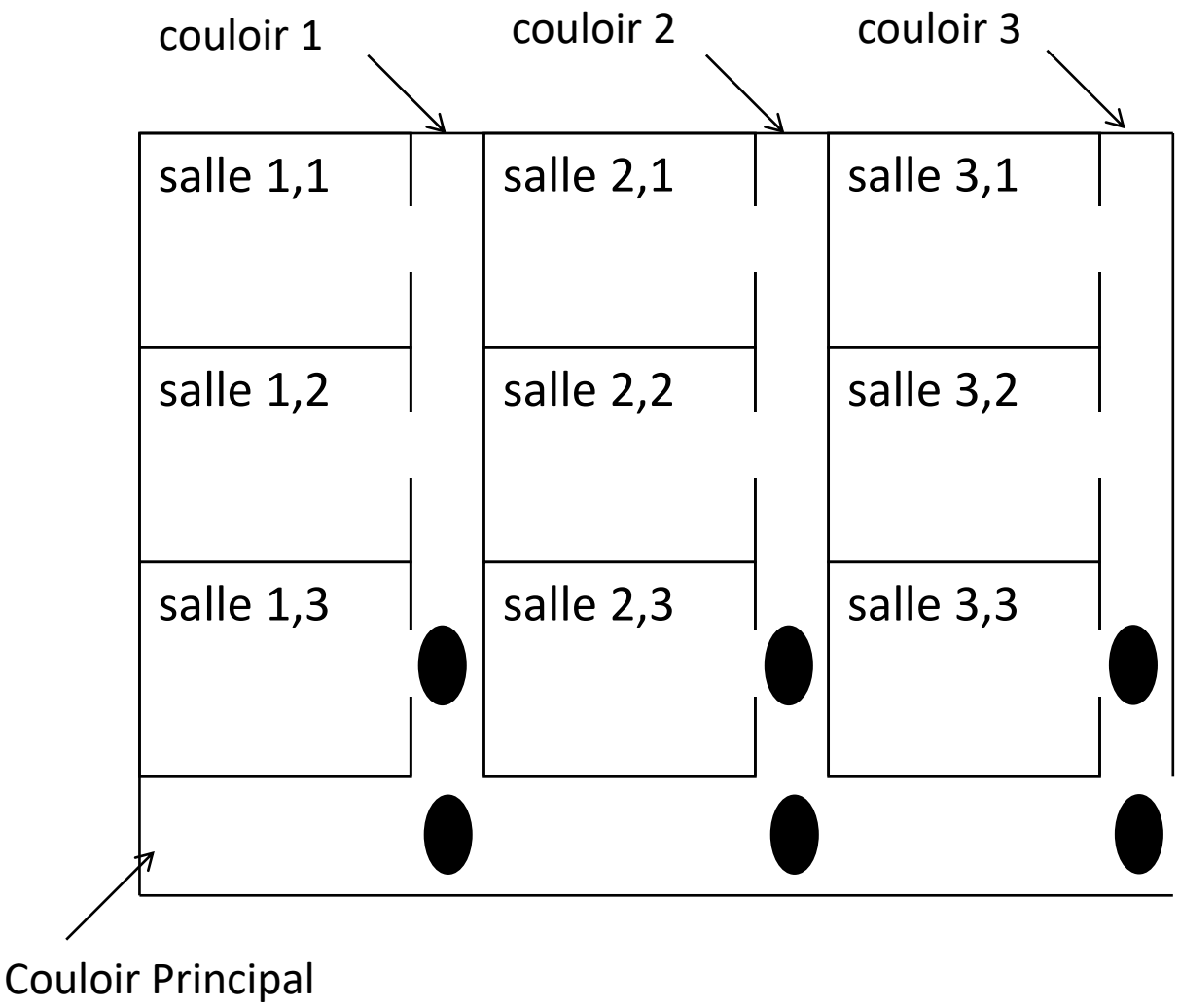
Agent responsable du comptage des individus sortant de l'établissement

Ainsi de suite …



Agent responsable du comptage des individus sortant de l'établissement

Ainsi de suite …



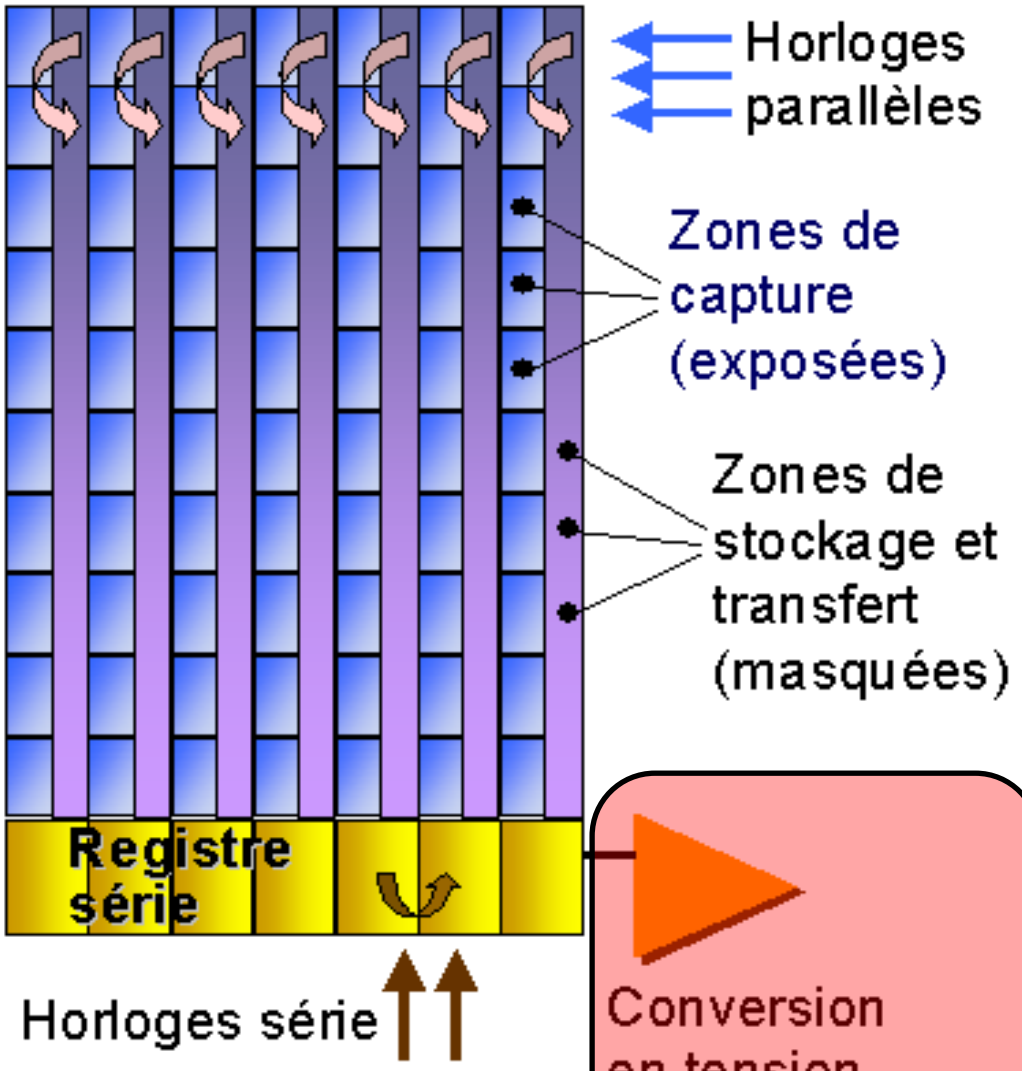
Agent responsable du comptage des individus sortant de l'établissement

- **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



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**

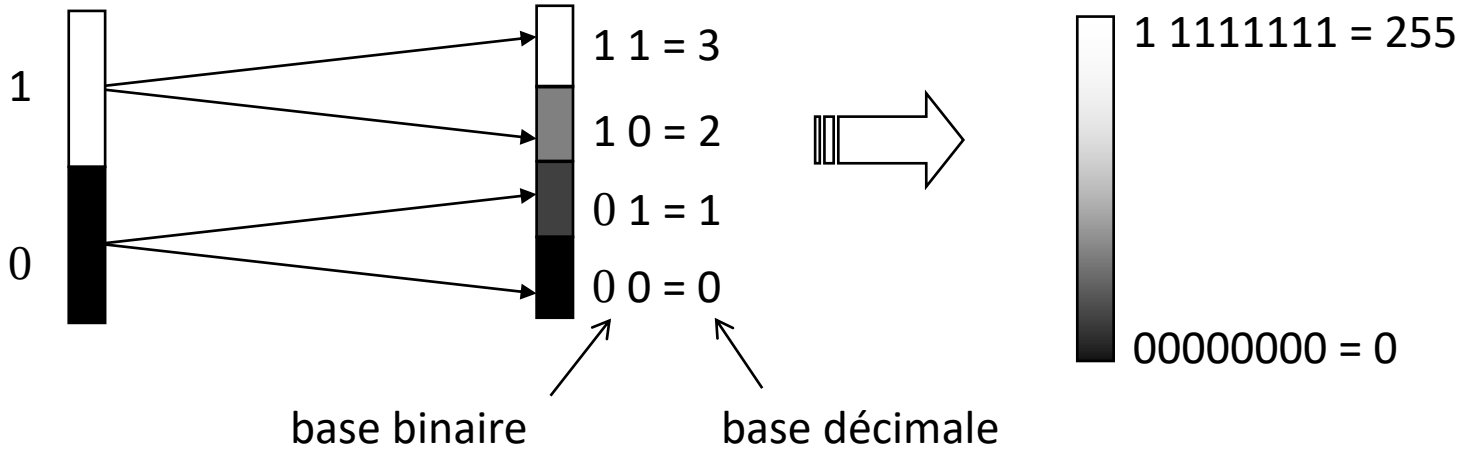
<u>Pour un pixel</u>: Tension de sortie proportionnelle à l'intensité lumineuse

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.

Tableau de valeurs

107	39	98					
100	54	87					
35	200						
76	250	…					

N lignes

M colonnes

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.

Exercice 2:

1) Ouvrir une image en noir et blanc au format PBM dans un logiciel de traitement d'image (Gimp). Réaliser le négatif de cette image.

2) Techniquement, réaliser le négatif d'une image revient à symétriser le niveau de gris de chaque pixel par rapport à la valeur médiane des niveaux de gris. Par exemple, le noir est transformé en blanc et vice versa. Considérons une image « I » dont le niveau de gris de chaque pixel est codé sur 1 octet. Si « x » est le veau de gris de l'image « I » considérée, et « y » le niveau de gris de la nouvelle image « J » transformée, alors la relation entre y et x est la suivante:

$$\forall x \in \{\text{pixels de l'image I}\},\ \mathbf{y = f(x) = 255 - x}$$

Il s'agit d'une relation affine:

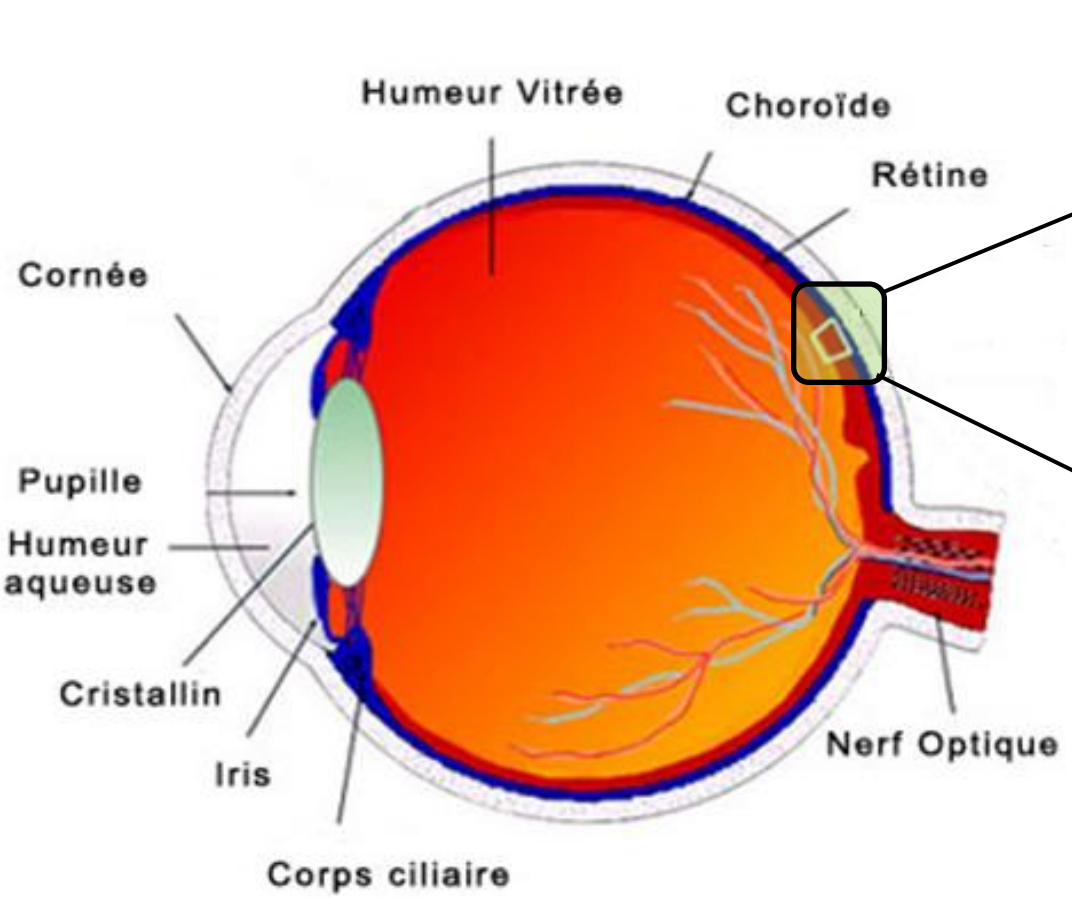
Ecrire l'algorithme permettant de réaliser le négatif d'une image

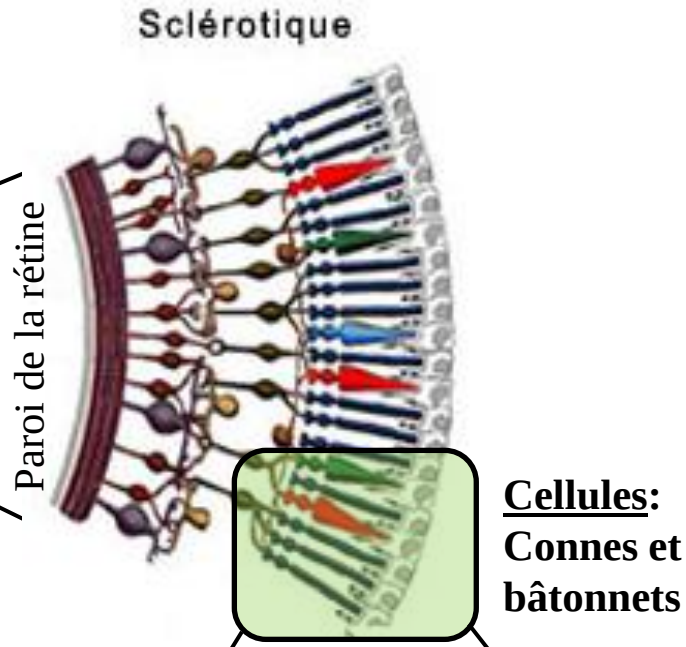
Correction:

Exercice 3: Considérons l'appareil numérique précédent dont la CCD a pour dimension 1024×768 pixels. Cet appareil dispose d'une carte mémoire de 1GO (un gigaoctet). L'utilisateur souhaite réaliser un film en noir et blanc d'une minute, sans compression. Est-ce possible ?

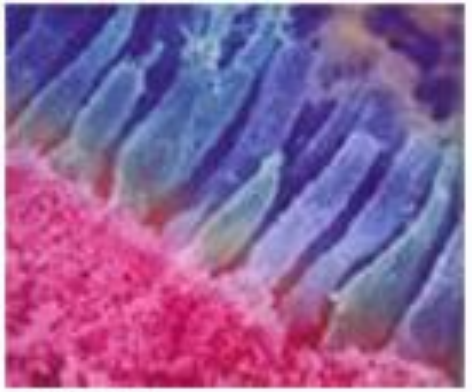
II. Images en couleurs

2.1. Perception des couleurs par l'œil humain





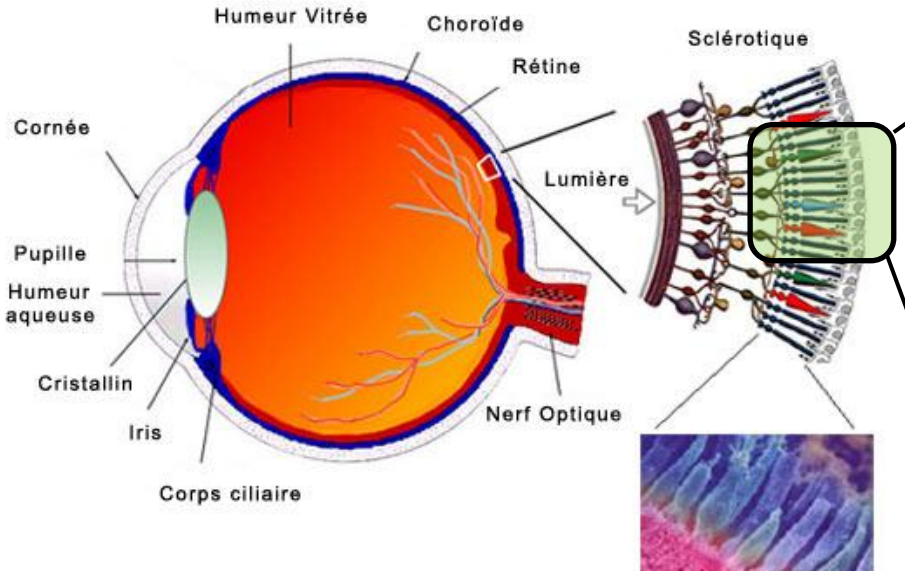
<u>Cellules</u>: Connes et bâtonnets

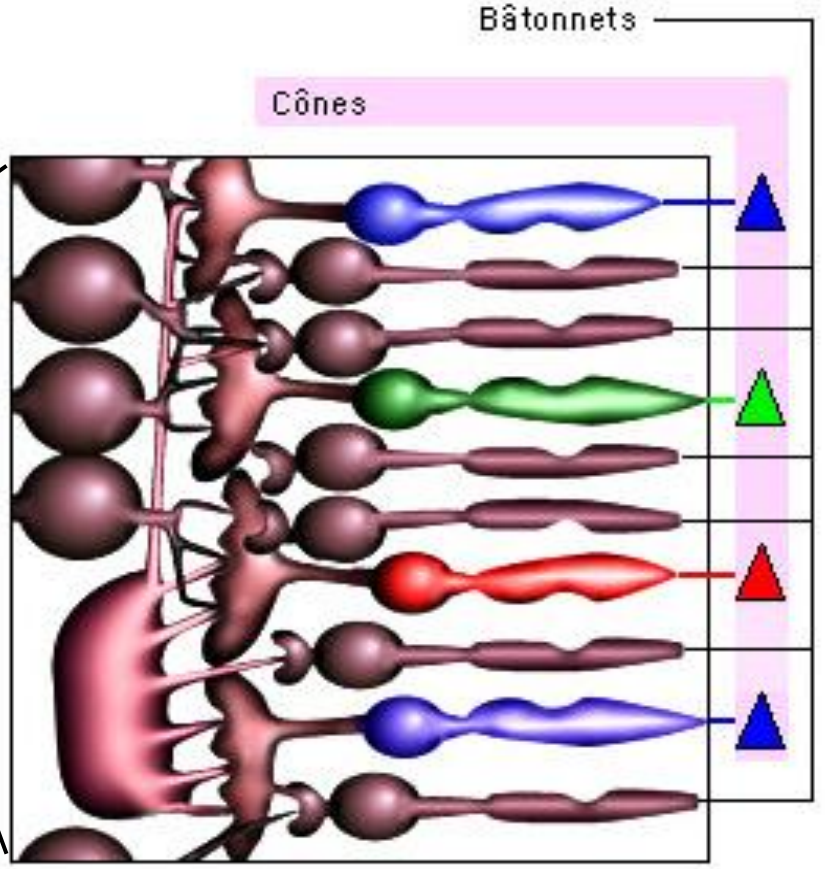


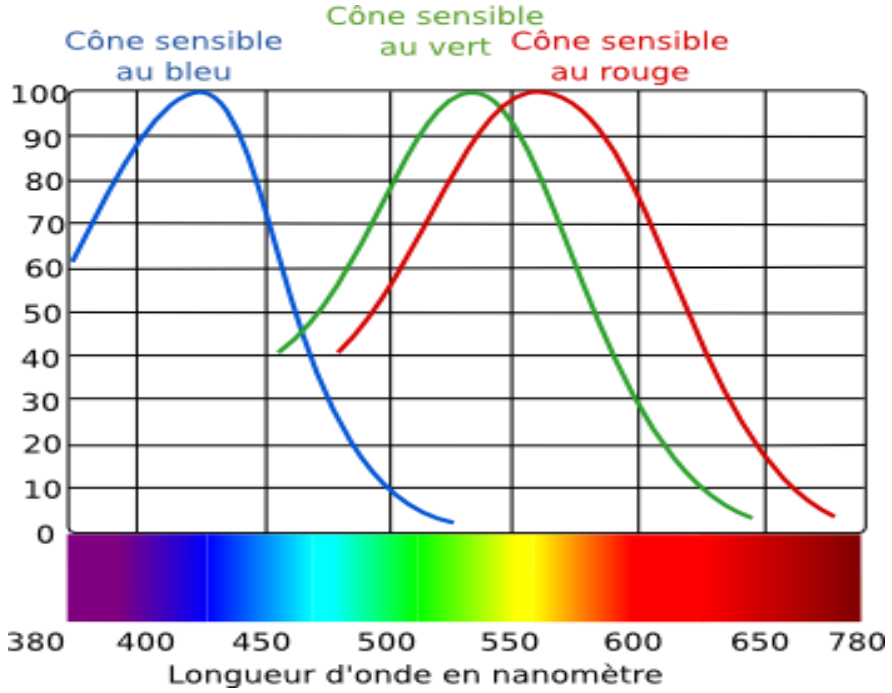
Détail : bâtonnets et cônes

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





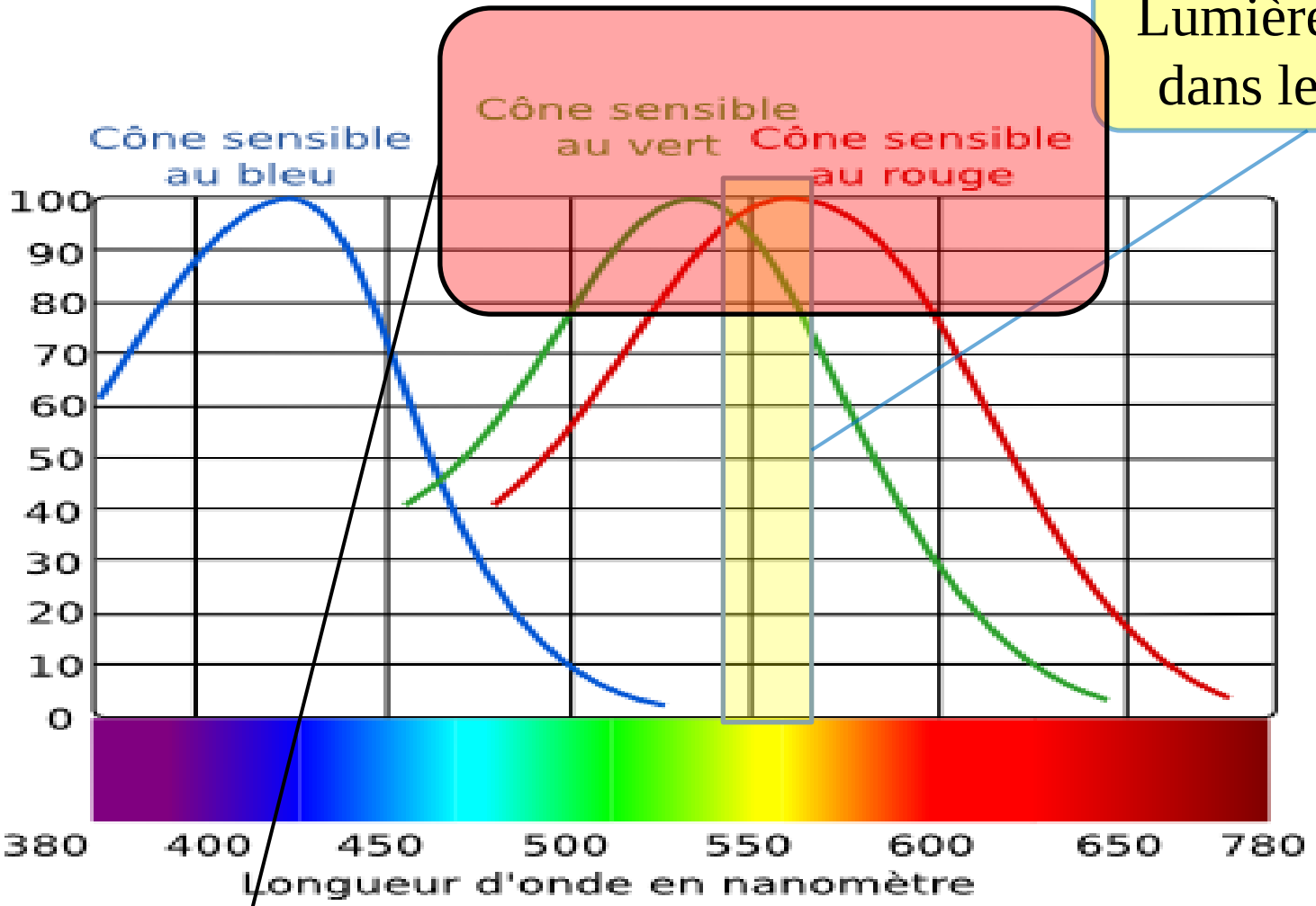


<u>3 sortes de connes</u> dont le maximum de sensibilité est situé dans :

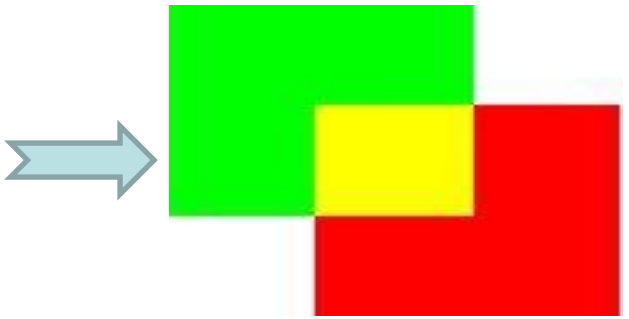
- **le rouge (560nm),**
- **le vert (530nm),**
- **le bleu (424nm).**

Perception des couleurs

Lumière émise dans le jaune



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

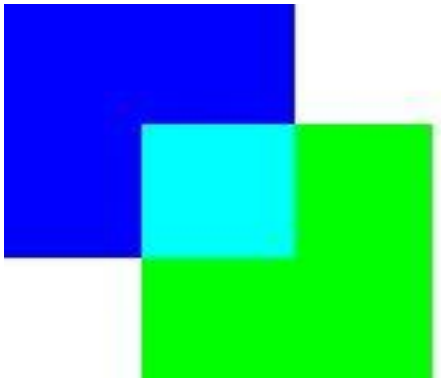


Sensation de couleur jaune identique à celle ressentie par un mélange de de vert et de rouge!

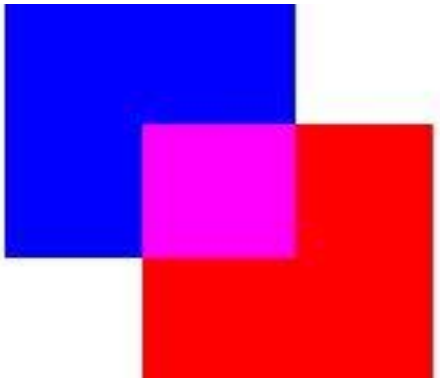
- **Quelle que soit la longueur d'onde reçue, le cerveau reçoit uniquement les intensités du signal reçu par les cellules dans le rouge, le vert et le bleu.**

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

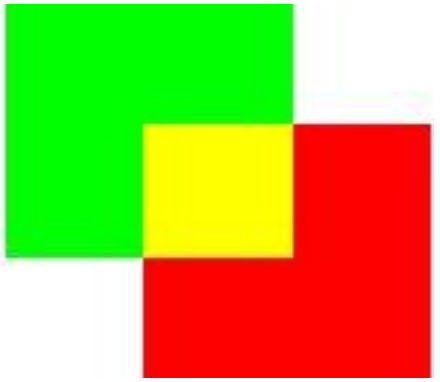
Exemple:



Vert + Bleu → Cyan



Rouge + Bleu → Magenta



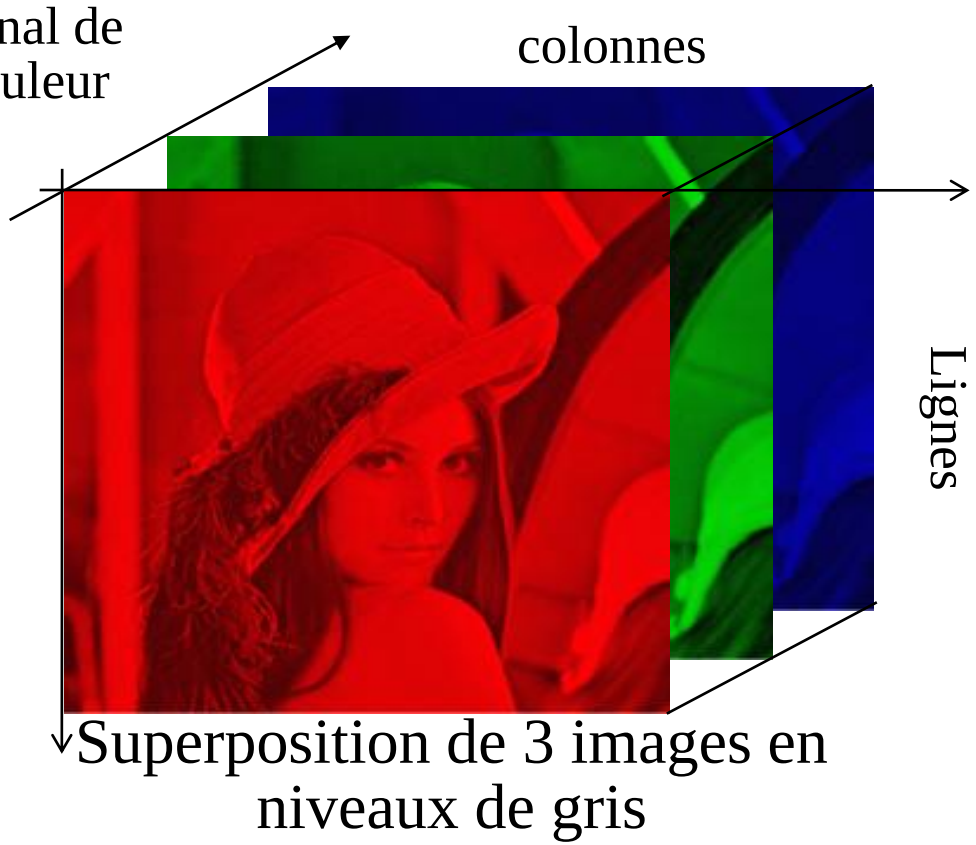
Vert + Rouge → Magenta

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)

$0 < i < N$

$0 < j < M$

$0 < k < 2$

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 ?

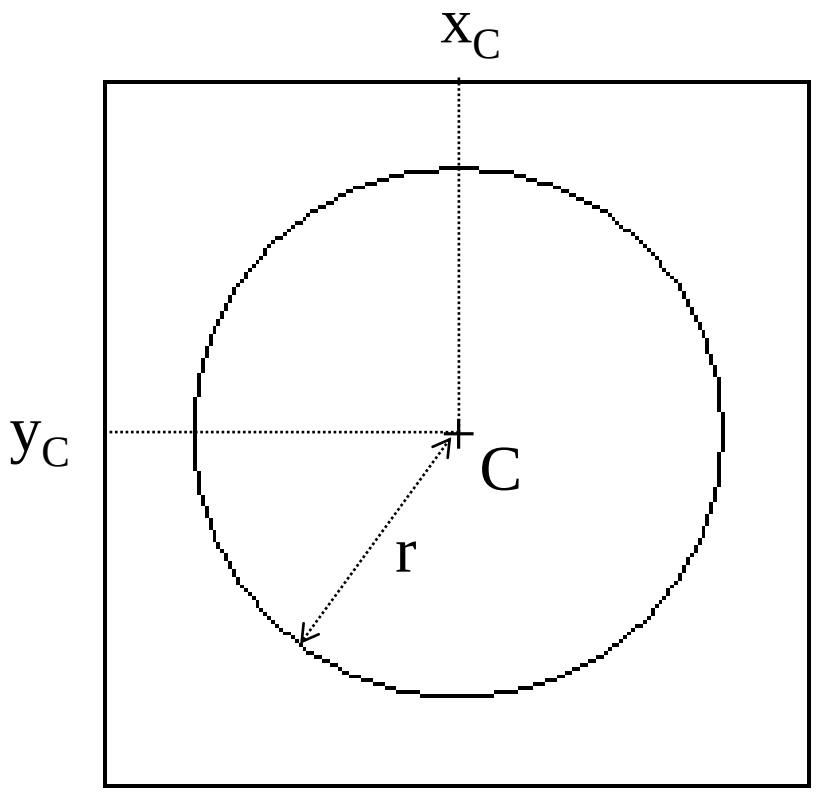
2) Calculer le poids (en octet) d'une image en couleur issue d'un appareil numérique dont la CCD a pour dimension 1024×768 pixels.

Exercice 2: Considérons l'appareil numérique précédent dont la CCD a pour dimension 1024×768 pixels. Cet appareil dispose d'une carte mémoire de 1GO (un gigaoctet). Quelle est la durée maximale que cet appareil est capable de réaliser sans compression ?

III. Quelques formats d'images numériques

3.1. Représentation vectorielle des images

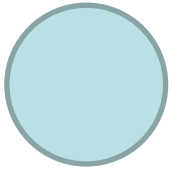
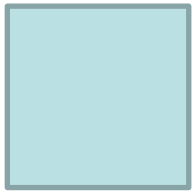
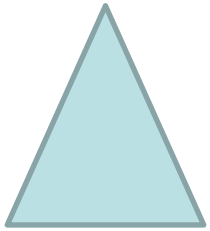
Exemple: image représentant un cercle



Représentation symbolique
=
Image vectorielle

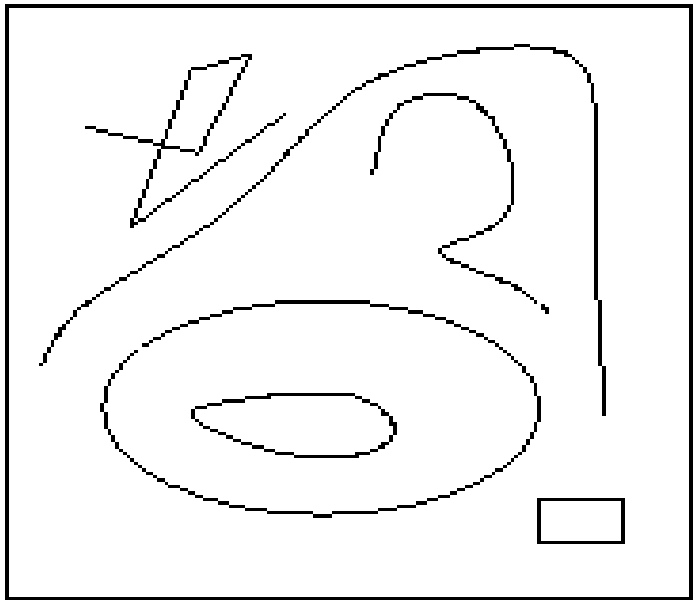
Image intégralement décrite par la donnée de 3 paramètres:
(x_C , y_C , z_C)

Beaucoup de figures géométriques simples/complexes se décrivent par une représentation vectorielle

	indicatif	paramètres	Nb paramètres
	c	(x,y,r)	3
	r	(x,y,a,b)	4
	r	(x,y,a)	3
	t	$(x_1, y_1, x_2, y_2, x_3, y_3)$	6

<u>Avantage</u>: poids mémoriel faible → bonne compression

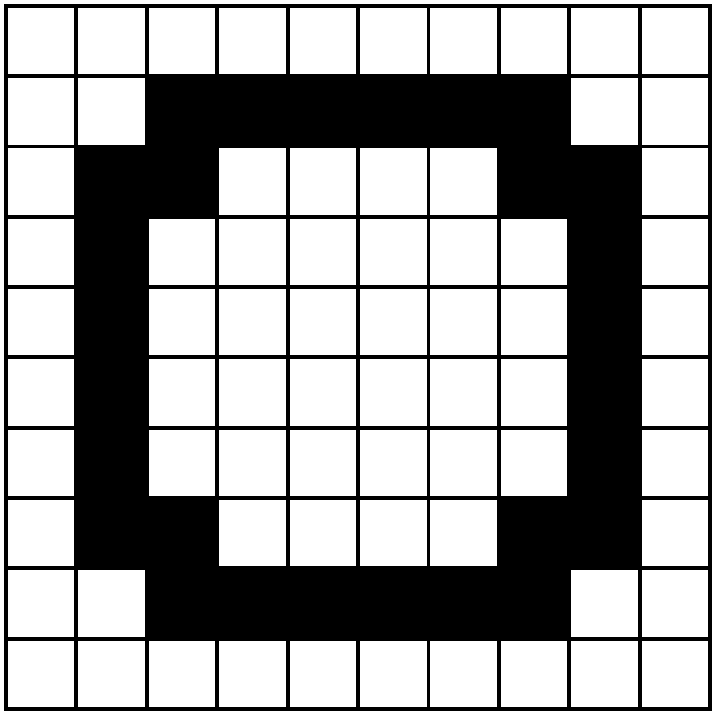
Problème: formes complexes



Non représentable par un nombre fini de paramètres

3.2. Représentation par pixels des images

Exemple: image représentant un cercle



Superposition d'une grille (10 × 10)

1 case = 1 pixel

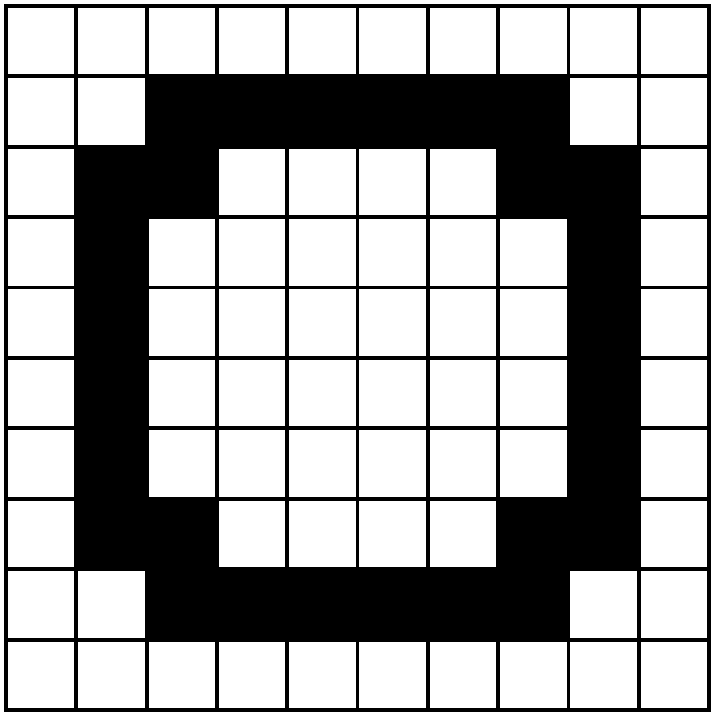
On noircit les pixels qui contiennent une portion de trait

Image = succession de cases noire/blanche

Lecture dans un ordre bien défini :
Gauche → Droite / Haut → Bas

Représentation binaire: **noir = 1 / blanc = 0**

<u>Exercice</u>: Pour une grille (10 × 10)

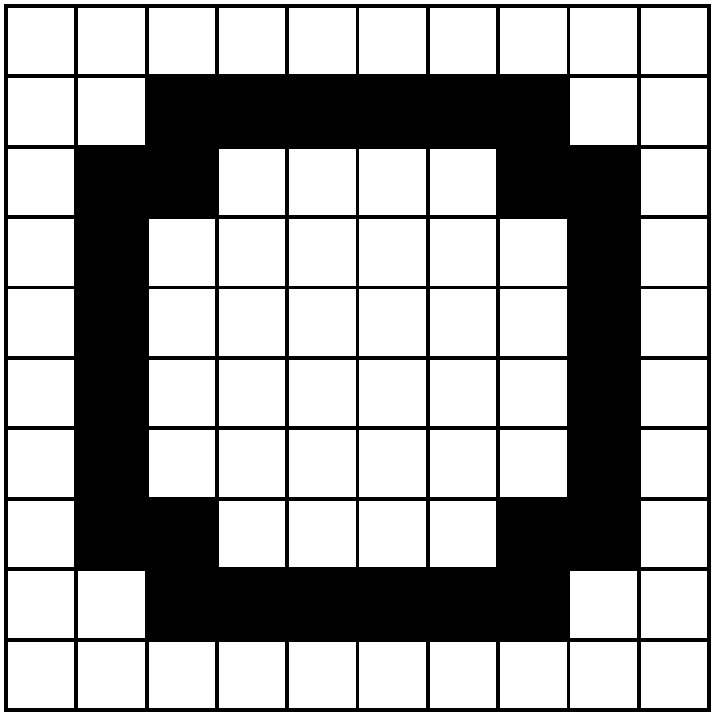


1) Donner la représentation matricielle binaire de l'image en noir et blanc.

2) En déduire le « mot binaire » correspondant à l'image en précisant son nombre de caractères.

3) Quel est ici la profondeur de l'image ?

Correction: 1) Représentation matricielle binaire



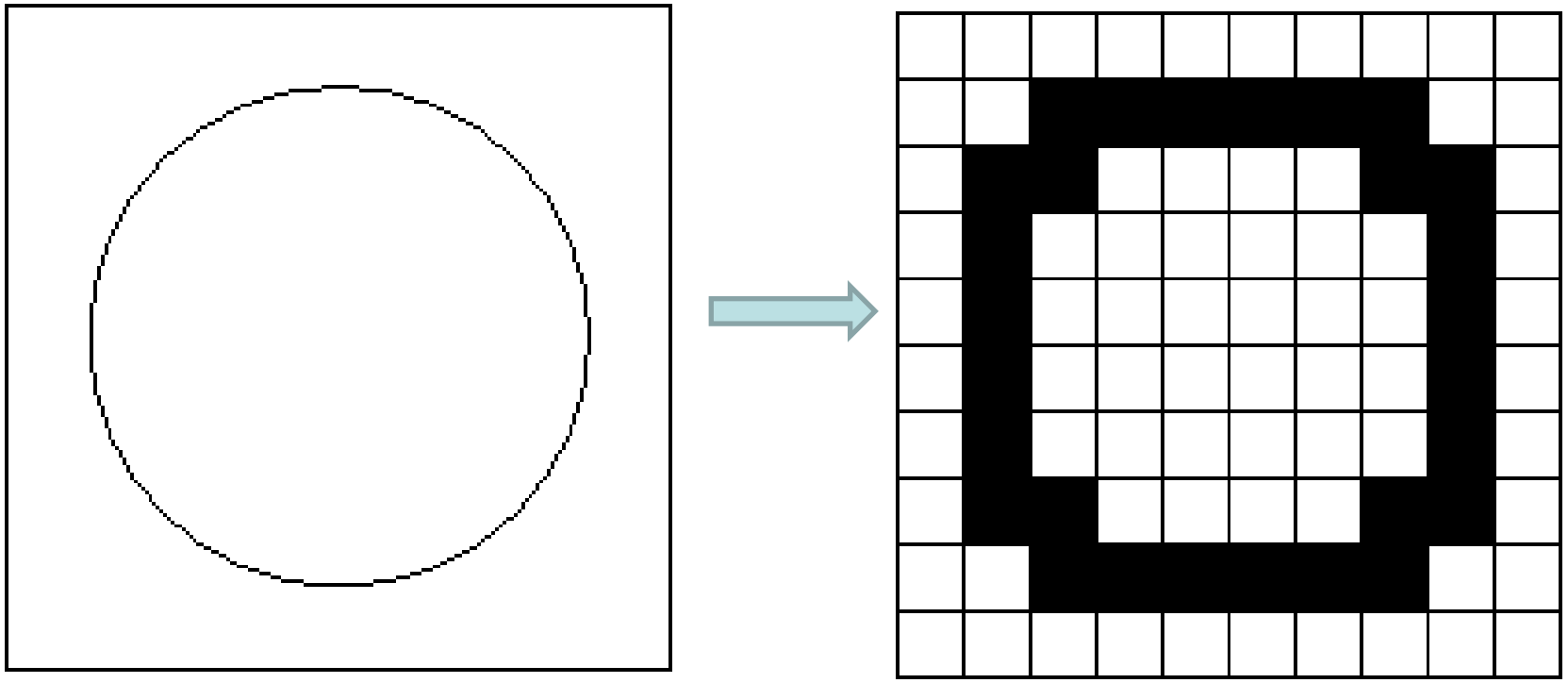
0	0	0	0	0	0	0	0	0	0
0	0	**1**	**1**	**1**	**1**	**1**	**1**	0	0
0	**1**	**1**	0	0	0	0	**1**	**1**	0
0	**1**	0	0	0	0	0	0	**1**	0
0	**1**	0	0	0	0	0	0	**1**	0
0	**1**	0	0	0	0	0	0	**1**	0
0	**1**	1	0	0	0	0	1	**1**	0
0	0	**1**	**1**	**1**	**1**	**1**	**1**	0	0

2) **Image = mot de 100 bits :**

0000000000011111100011000011001000000100100000100100000010010000001001100001100011111100000000000

3) **Ici 1bit/pixel (noir/blanc) ⇒ Profondeur de 1 bit !!**

Q: pourquoi la forme circulaire initiale n'est-elle pas bien représentée ?



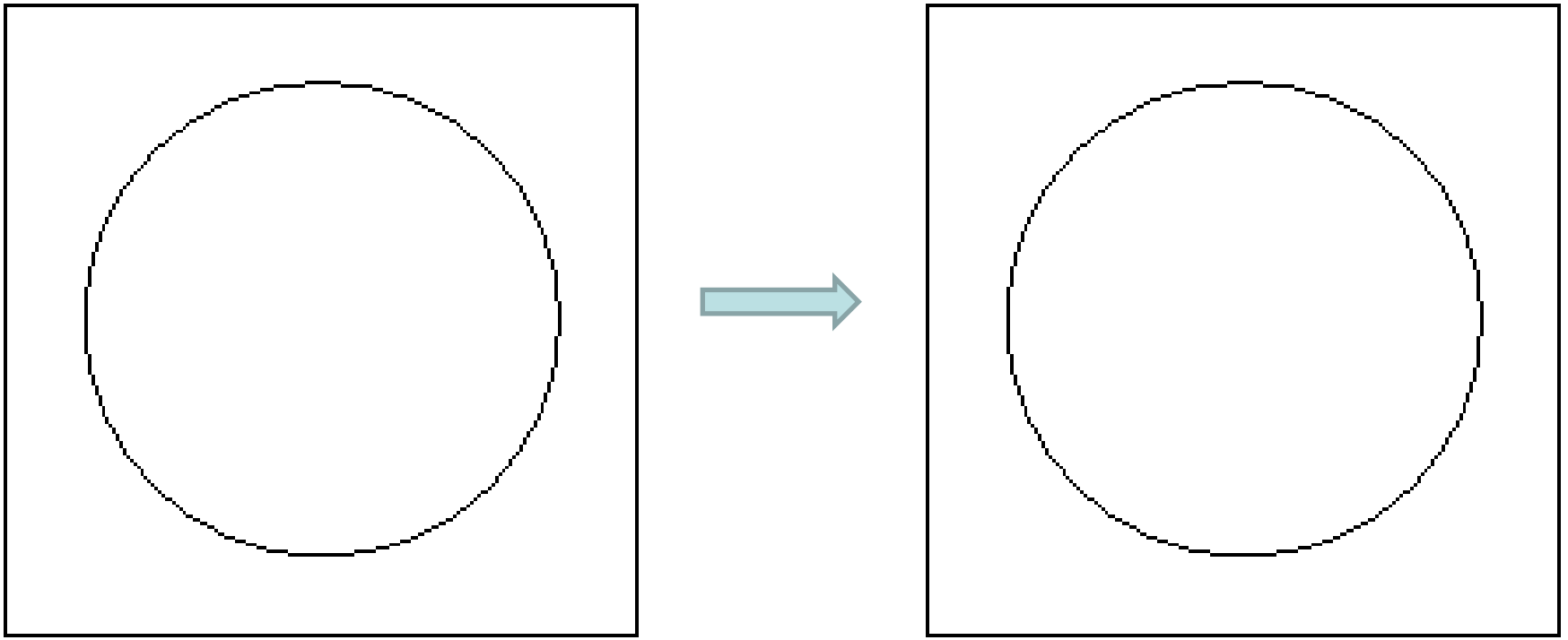
Grille trop grossière ⇒ **Ne comporte pas assez de pixels**

Sous échantillonnage spatial !

Q: Comment remédier à l'effet de pixellisation?

Augmenter le nombre de pixels de la grille

Exemple: grille de (100 x 100) pixels



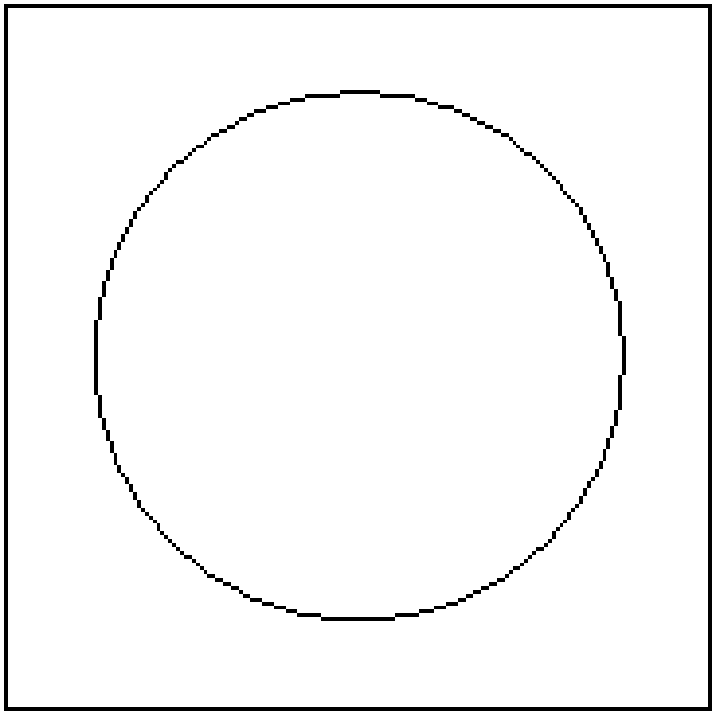
On n'observe plus les effets de la pixellisation

Le sur échantillonnage est inutile:

Pour une grille très fine de quelques millions de pixels l'œil est incapable de discerner deux points voisins.

Augmente inutilement le poids de l'image!!

Bilan et définition:

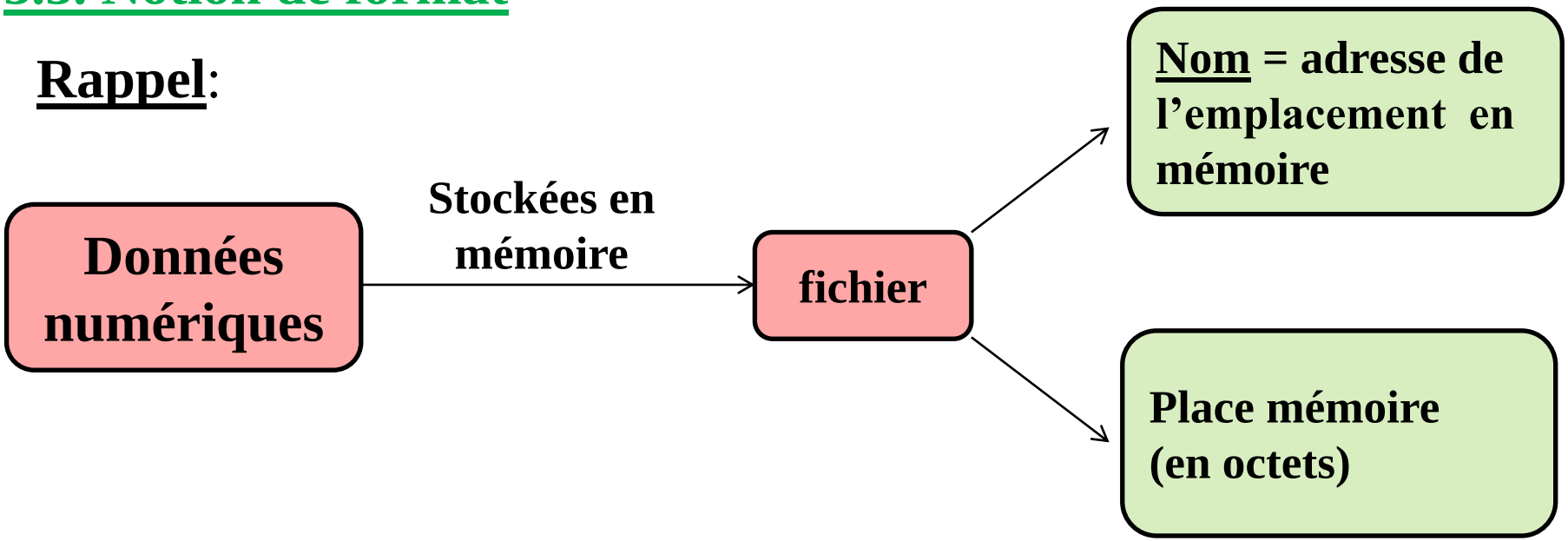


0000000000001111110001100001100100000100100000100100000010010000001001100001100011111100000000000

La représentation d'une image sous la forme d'une suite de pixels chacun codés sur un bit, s'appelle une bitmap. Il s'agit d'une méthode approximative, mais universelle car elle permet de décrire n'importe quelle image en noir et blanc.

3.3. Notion de format

Rappel:



Q: Comment connaitre a priori la nature du fichier ?

Nécessité de définir une extension précédé d'un point

Exemple : pour une image

PBM, PGM, PPM, PNG, JPEG, GIF, PS, PICT, TIF, etc.

Portable BitMap → **Fichier image:** **Cercle.pbm**

3.4. Image en noir et blanc : format PBM (Portable BitMap)

- Le format PBM est l'un des plus simples pour exprimer des images bitmap (en noir et blanc).
- Ecrit en caractères ASCII et vérifie systématiquement l'architecture suivante:

- Le début du fichier : "**P1**" + « **retour à la ligne** » ou « **espace** »
- Largeur de l'image **en base 10**, + « **retour à la ligne** » ou « **espace** »
- Hauteur de l'image **en base 10,** + « **retour à la ligne** » ou « **espace** »
- Liste des pixels codés sur 0 ou 1 ligne par ligne, de haut en bas et de gauche à droite **sans prendre en compte les retours à la lignes et les espaces**.
- Aucune ligne ne doit dépasser 70 caractères et toutes les lignes commençant par le caractère # sont des commentaires.

Exercice: écrire le fichier cercle.pbm de l'image précédente

```
P1 # Mon premier fichier PBM : le cercle
10 10 #taille de l'image
0000000000   # ligne 1
0011111100
0110000110
0100000010
0100000010
0100000010
0100000010
0110000110
0011111100
0000000000   # ligne 10
```

3.5. Image en niveau de gris: format PGM (Portable GreyMap)

- Le format PGM est l'un des plus simples pour exprimer des images en niveaux de gris.

- Généralement l'intensité du niveau de gris est codé sur **un octet**, ce qui autorise **256 niveaux entre le noir (0) et le blanc (255).**

- format PGM très similaire format PBM.

- Ecrit en caractères ASCII et vérifie systématiquement l'architecture suivante:

- Le début du fichier : "**P2**" + « **retour à la ligne** » ou « **espace** »

- Largeur de l'image **en base 10**, + « **retour à la ligne** » ou « **espace** »

- Hauteur de l'image **en base 10,** + « **retour à la ligne** » ou « **espace** »

- Valeur maximale M pour exprimer les niveaux de gris **en base 10**, + « **retour à la ligne** » ou « **espace** ».

Si chaque pixel est codé sur un octet, cette valeur est à 255 et correspond au blanc

- Liste des pixels (entre 0 et M) ligne par ligne, de haut en bas et de gauche à droite **sans prendre en compte les retours à la lignes et les espaces**.

- Aucune ligne ne doit dépasser 70 caractères et toutes les lignes commençant par le caractère # sont des commentaires.

3.6. Image en couleurs: format PPM (Portable PixMap)

- Superposition de 3 images en niveaux de gris dans les trois teintes rouge, verte et bleu.

- Niveau de gris dans une couleur particulière est codé sur 1 octet,
  → **256 niveaux** variant entre le noir (0) et le rouge/vert/bleu pur intense (255).

- format PPM très similaire format PBM et PGM.

- Ecrit en caractères ASCII et vérifie systématiquement l'architecture suivante:

- Le début du fichier : "**P3**" + « **retour à la ligne** » ou « **espace** »

- Largeur de l'image **en base 10**, + « **retour à la ligne** » ou « **espace** »

- Hauteur de l'image **en base 10,** + « **retour à la ligne** » ou « **espace** »

- Valeur maximale M pour exprimer les niveaux de gris **en base 10**, + « **retour à la ligne** » ou « **espace** ».

Si chaque pixel est codé sur un octet, cette valeur est à 255 et correspond au blanc

- Liste des valeurs des couleurs, trois par pixel, dans l'ordre rouge, vert, bleu (entre 0 et M) , ligne par ligne, de haut en bas et de gauche à droite **sans prendre en compte les retours à la lignes et les espaces**.

- Aucune ligne ne doit dépasser 70 caractères et toutes les lignes commençant par le caractère # sont des commentaires.

Exemple: En mélangeant du rouge et du vert en quantités égales, on obtient du jaune en augmentant la quantité de rouge, par exemple, rouge = 237, vert = 127, bleu = 16, on obtient du orange. On peut alors écrire un fichier PPM qui représente un carré orange de 100 pixels sur 100 pixels.

```
P3
# Mon premier fichier PPM : orange
100 100
255
237 127 16
237 127 16
237 127 16
237 127 16
...
```

Exercice 1:

1) Lequel des formats PBM, PGM et PPM est adapté pour représenter un carré noir de 10 pixels sur 10 pixels ?

2) Même question pour un carré rouge de même taille.

3) Comparer les taille des fichiers obtenus

Correction:

1) Pour un carré noir, le format PBM suffit, puisqu'il n'y a ni niveaux de gris ni couleurs, et le fichier est le suivant :

```
P1
# Un carré noir
10 10
1111111111
1111111111
1111111111
1111111111
1111111111
1111111111
1111111111
1111111111
1111111111
1111111111
```

2) Pour un carré rouge, il faut obligatoirement recourir au format PPM, seul capable de représenter de la couleur. Le rouge se représente par les trois nombres 255 0 0 : intensité maximale pour le rouge et nulle pour le vert et le bleu. On obtient le fichier suivant :

```
P3
# Un carré rouge
10 10
255
255 0 0
255 0 0
255 0 0
…
(100 lignes
identiques)
```

3) Le fichier représentant la seconde image est significativement plus gros que celui représentant la première.

Exercice 2:

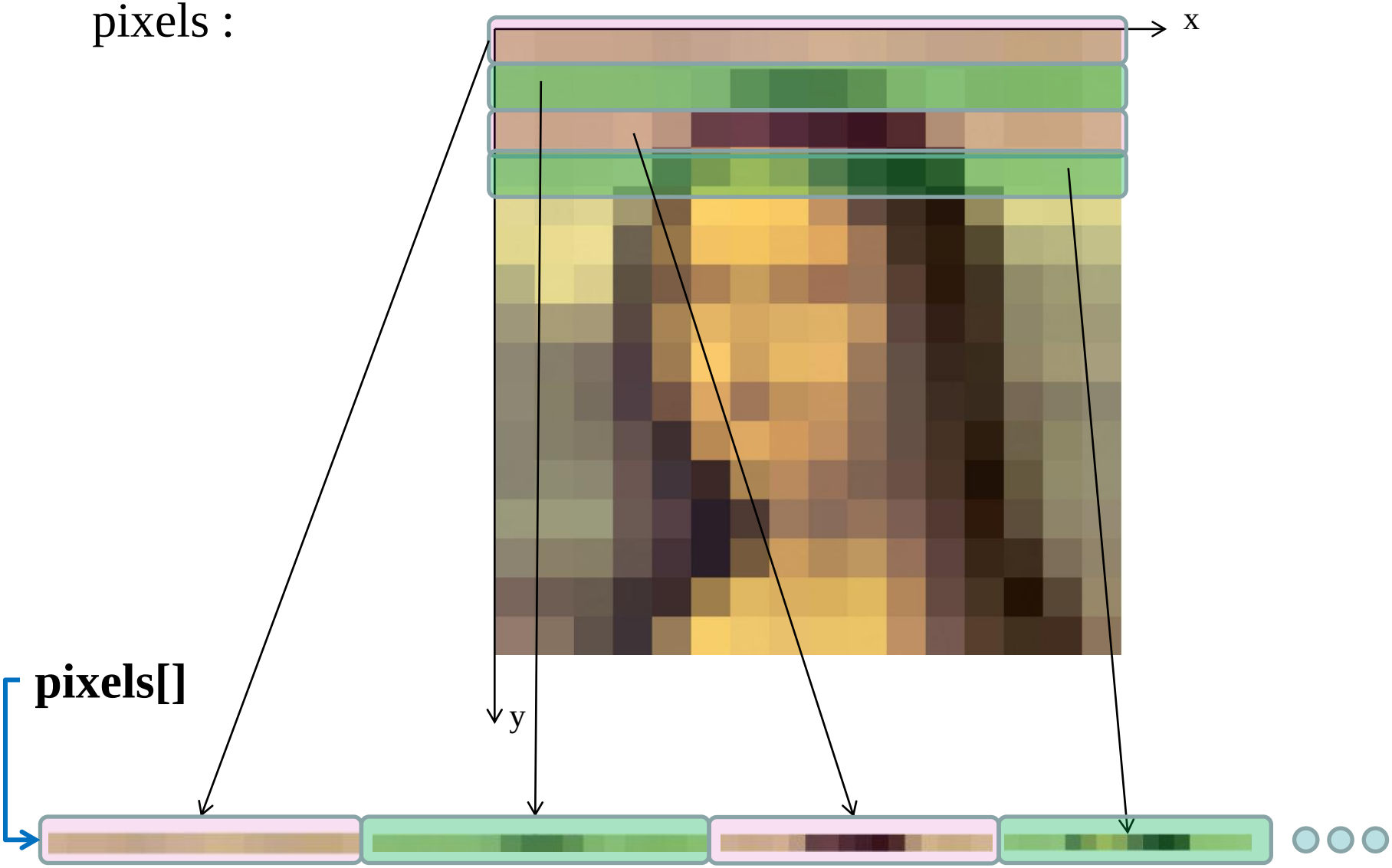
1) Ouvrir une image en noir et blanc au format PBM dans un éditeur de texte (Bloc Note, note pad, etc…). Ecrire un carré noir de 10 par 10 pixel sur le coin en haut à gauche de l'image.

2) Même question pour une image en couleur au format PPM en dessinant un carré rouge de 10 par 10 pixels.

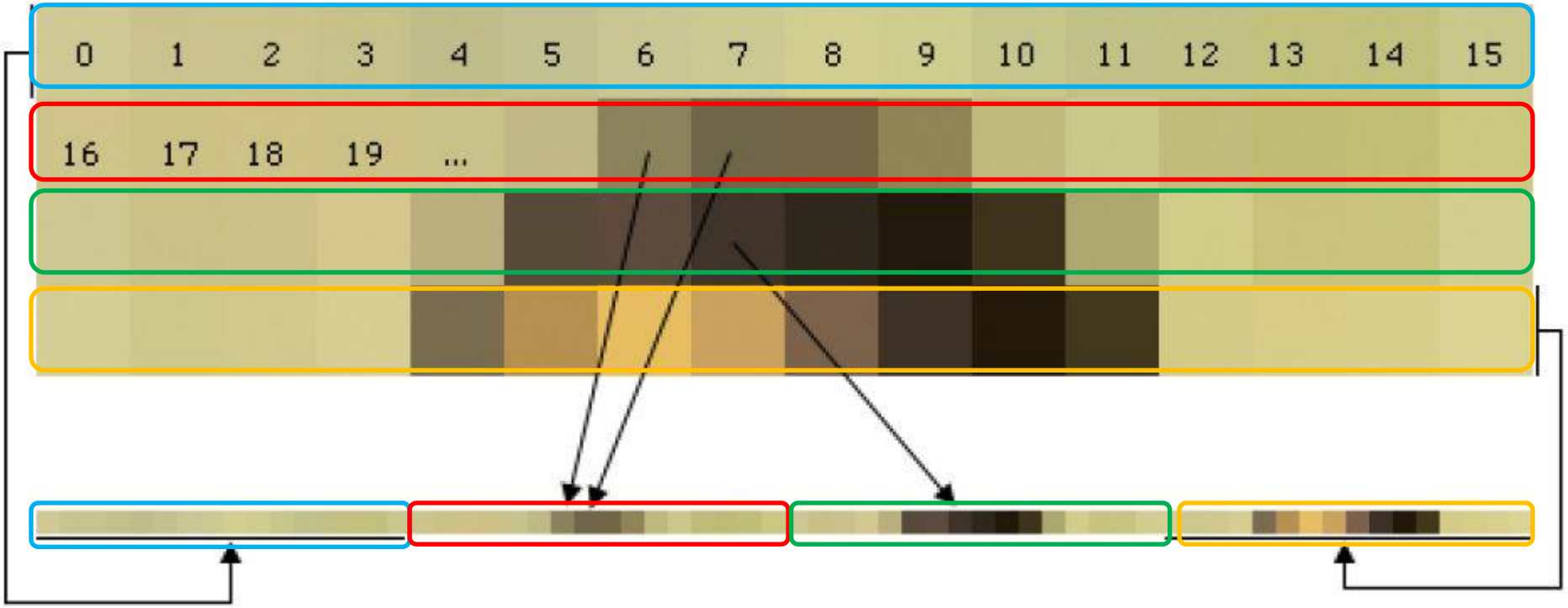
IV. Représentation d'images numériques sous Processing

4.1. Présentation

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



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

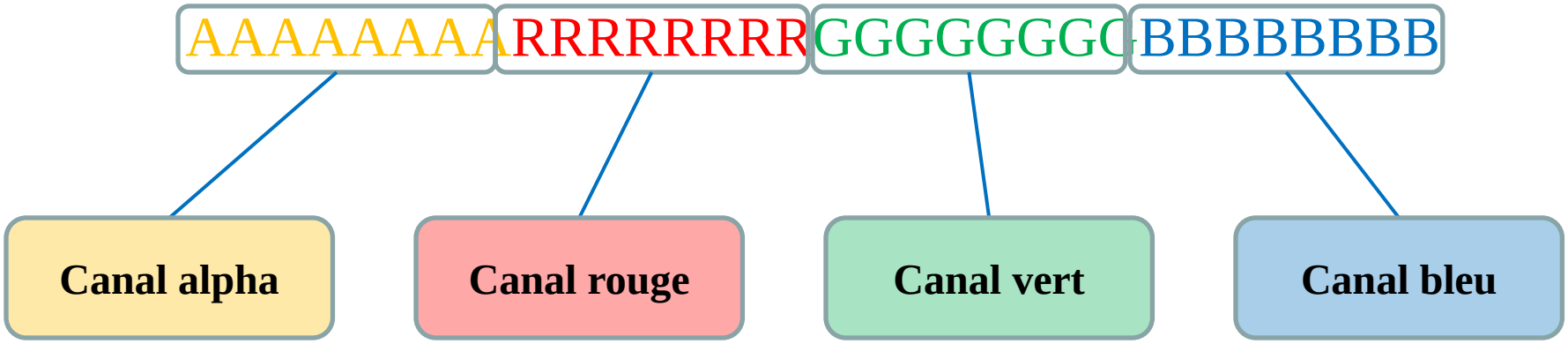


4.2. Représentation d'un pixel

- Chaque **pixel** est mémorisé comme **variable de type « color »** c'est-à-dire comme un nombre au format hexadécimal :

  - ***MonImage.pixels[n]=0xCCFF66A1 ;***

- Type « color » codée sur 4*8bits = **32bits** organisée de la manière suivante :



- Chaque pixel représente un **vecteur à 4 dimensions** dans un espace composé de l'espace tridimensionnel des couleurs complété par une dimension de relative à la transparence :

$$MonImage.pixels[n] = \begin{pmatrix} A \\ R \\ G \\ B \end{pmatrix}$$

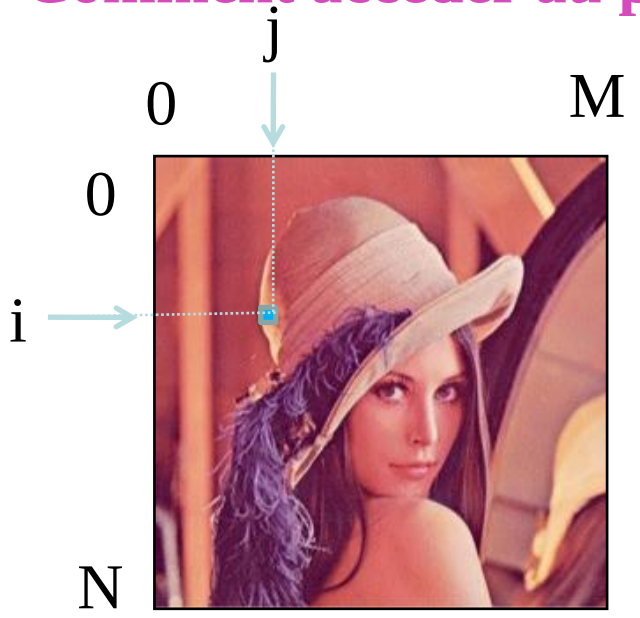
- Pour accéder à un canal particulier, Processing fournit des méthodes spécifiques :

alpha() :
red() :
blue() :
green() :

hue() :
saturation() :
brightness():

4.3. Lien avec la représentation matricielle d'une image

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

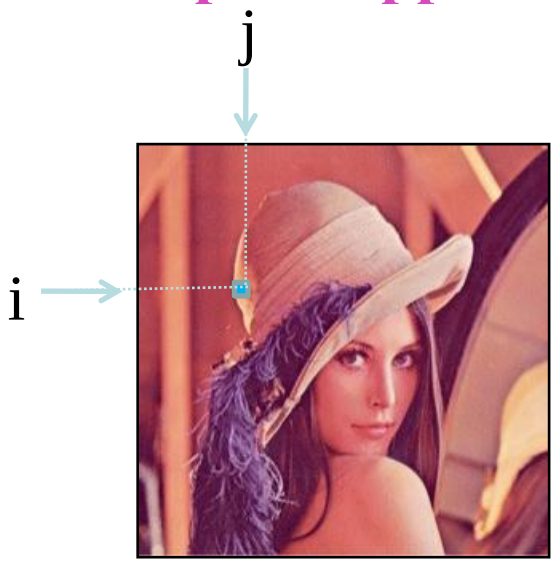


1	2	3	…	…	indice	…	…	N*M-1	N*M

indice = i × image.width + j

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

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);
```