

# Changer la taille ou l'orientation

(Semaine du 18 au 22 décembre)

On veut agrandir ou diminuer la taille de l'image. Il y a différentes manières de procéder :

- **Rogner** : on efface des colonnes de pixels à droite ou à gauche pour diminuer la largeur de l'image, ou des lignes de pixels en haut ou en bas pour diminuer la hauteur de l'image
- **Agrandissement/rétrécissement** : on veut modifier la longueur et/ou la largeur de l'image.
- Utiliser des méthodes avancées d'agrandissement sans déformation

On veut aussi transformer l'image sans déformation :

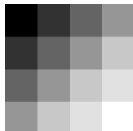

- **Rotation** de 90° : un quart de tour vers la droite ou vers la gauche.
- **Symétrie** selon un axe vertical ou horizontal.

Chacune de ces opérations va être une méthode de la classe `Image`. Il est fortement conseillé de tester sur papier ces transformations avant de commencer à les programmer en C++. Il va falloir pour chaque pixel calculer ses indices dans l'image d'arrivée en fonction de ses indices dans l'image de départ.

1. (★) Ecrire des méthodes `rognerD`, `rognerG`, `rognerH`, `rognerB` de la classe `Image` permettant de créer une nouvelle image à partir de l'image cible qui supprime `nb` des lignes (ou des colonnes) de pixel de l'image respectivement à droite, à gauche, en haut, en bas, où `nb` est un entier passé en paramètre.

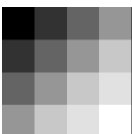
**Exemple** : Si on rogne d'une colonne à gauche l'image  on obtient .

2. (★) Ecrire des méthodes `rotationD`, `rotationG` de la classe `Image` permettant de créer une nouvelle image à partir de l'image cible qui effectue une rotation de l'image de 90° vers la droite (sens horaire) ou vers la gauche (sens trigonométrique). Vous pouvez vous limiter au cas des images carrées (où la longueur est égale à la largeur).

**Exemple** : Si on tourne de 90° vers la droite l'image  on obtient .

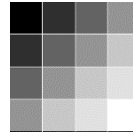
3. (★★) Modifier vos méthodes `rotationD` et `rotationG` afin qu'elles puissent faire la rotation d'images qui ne sont pas carrées.
4. (★) Ecrire des méthodes `retournementH` et `retournementV` de la classe `Image` permettant de créer une nouvelle image à partir de l'image cible qui effectue une symétrie de l'image d'axe vertical ou horizontal.

**Exemple** : Si on effectue un retournement horizontal (symétrie d'axe vertical) à partir de

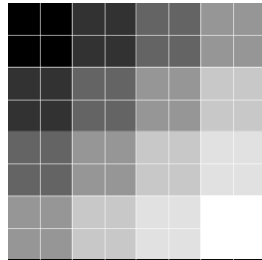
l'image  on obtient .

5. (★★) Ecrire une méthode `agrandissement` de la classe `image` permettant de créer une nouvelle image à partir de l'image cible qui agrandit l'image d'un facteur entier. Une manière de procéder est de dupliquer les pixels : un pixel de l'image d'origine devient un carré de de 4, 9, 16,... pixels.

Exemple :



Si on effectue agrandit d'un facteur 2 l'image



on obtient

6. (★★) Ecrire une méthode `retrecissement` de la classe `image` permettant de créer une nouvelle image à partir de l'image cible et qui rétrécit l'image d'un facteur entier. Une manière de procéder est de fusionner les pixels : un carré de 4, 9... pixels de l'image d'origine devient un pixel qui prend la moyenne des valeurs des pixels d'origine.
7. (★★★) Ecrire une méthode `zoom` de la classe `image` permettant de créer une nouvelle image à partir de l'image cible et qui effectue une [homothétie](#) de l'image d'un facteur flottant  $k$  (agrandissement si  $k > 1$  et rétrécissement si  $0 < k < 1$ ).
8. (★★★★) [Suppression des chemins de plus basse énergie](#).  
 Principe : On associe à chaque pixel de l'image une valeur qu'on appelle énergie.  
 Un chemin est une suite de pixel, un pixel par ligne de l'image tels que les pixels de deux lignes successives « se touchent par au moins un coin ». L'énergie d'un chemin est la somme de l'énergie de chaque pixel qui le compose.  
 Une manière de réduire l'image de 5 colonnes est de supprimer les 5 chemins les moins énergétiques.  
 Cette technique peut être utilisée pour agrandir l'image en dupliquant les chemins de plus basse énergie.