Rotation d'une image

Christophe Viroulaud

Terminale - NSI

Algo 03

La rotation d'une image est une fonctionnalité proposée par n'importe quel logiciel de retouche tel **Gimp**. L'opération n'est cependant pas triviale et peut demander une durée non négligeable.

Construire un algorithme de rotation d'une image en appliquant le principe de *diviser pour régner*.

Sommaire

Principe - Première approche

Diviser pour régner se décompose en trois parties :

- *diviser :* Le problème est partagé en plusieurs petits problèmes identiques.
- *traitement :* Chaque petit problème est résolu.
- *recombinaison :* Les petits problèmes résolus sont assemblés pour remonter au problème principal.

Activité 1 : Réflexion commune : Considérons une image aux dimensions connues. Quelles étapes pourrions-nous imaginer pour répondre à notre problématique ?

Avant de regarder la correction

- Prendre le temps de réfléchir,
- Analyser les messages d'erreur,
- Demander au professeur.

Correction

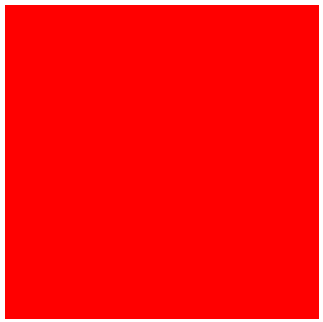


Figure 1 – 1 pixel : rien à faire

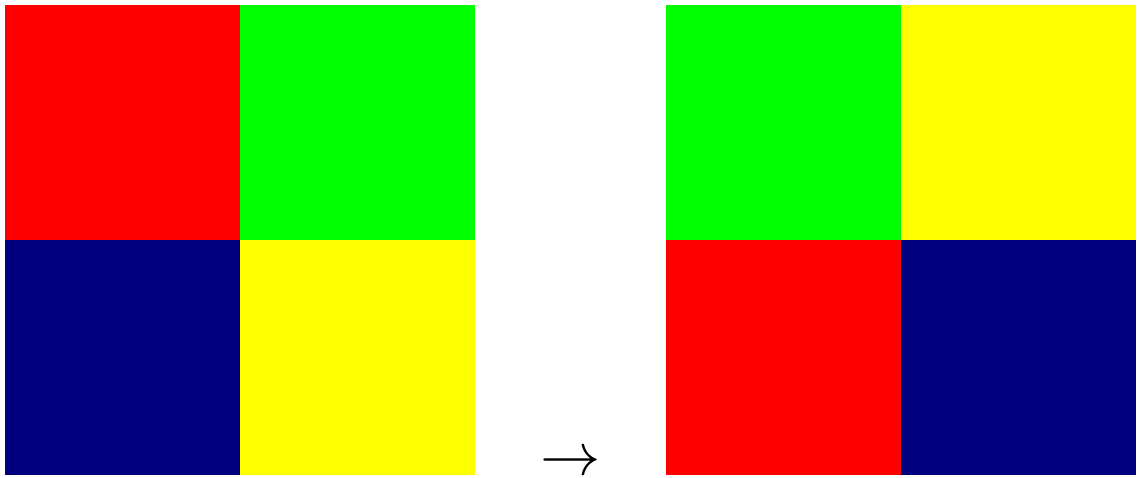


Figure 2 – Rotation

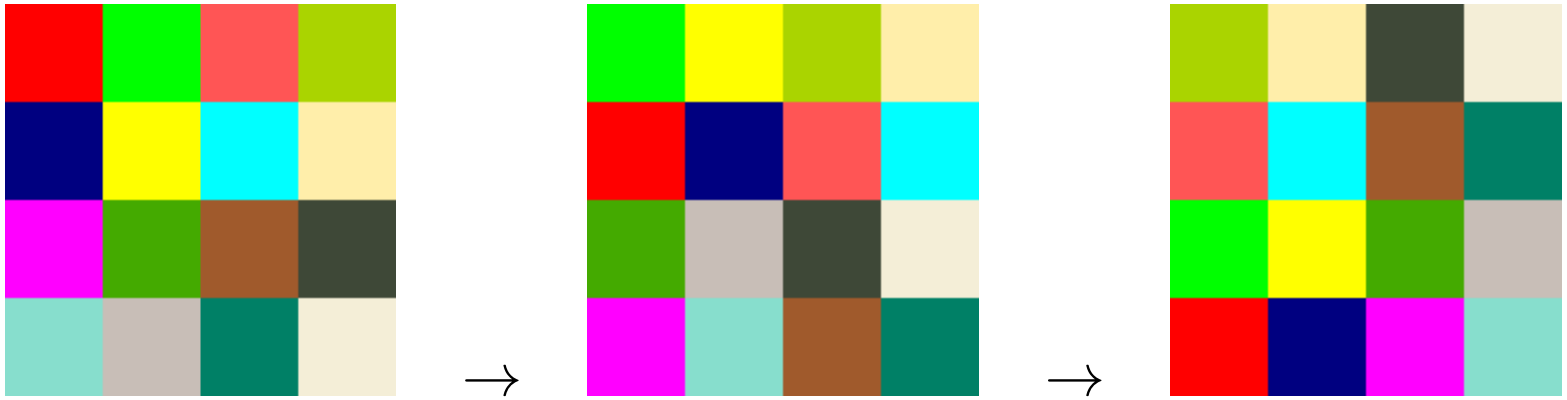


Figure 3 – Récursivité : on divise la taille des problèmes par 2.

Observation

On retrouve un algorithme de type *diviser pour régner*. On divise un gros problème, difficile à résoudre, en petits problèmes simples.

Sommaire

Construction de l'algorithme

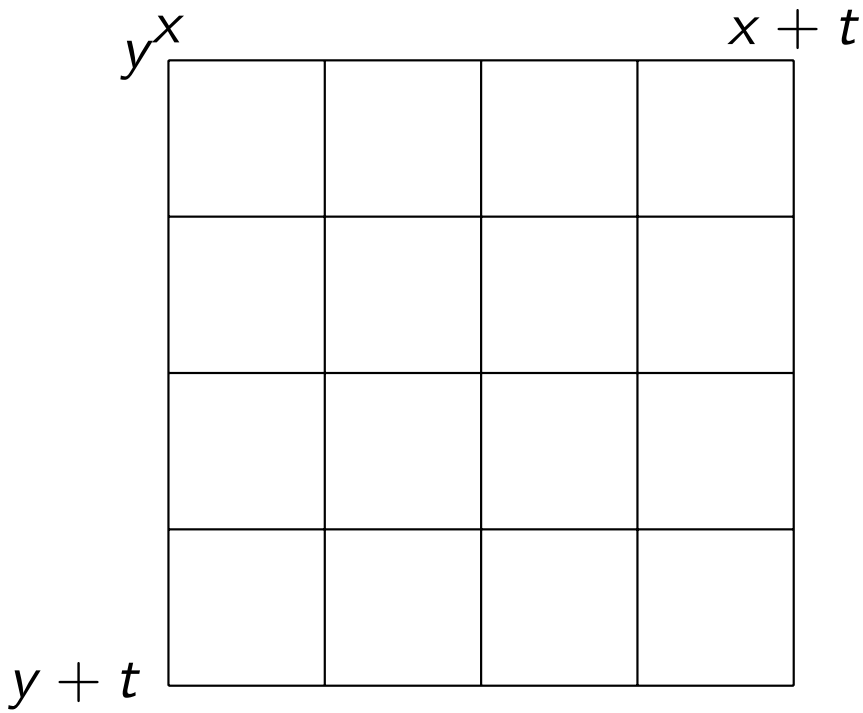


Figure 4 – $t = 4$. On divise la taille par 2 et on applique l'algorithme récursivement sur chaque partie.

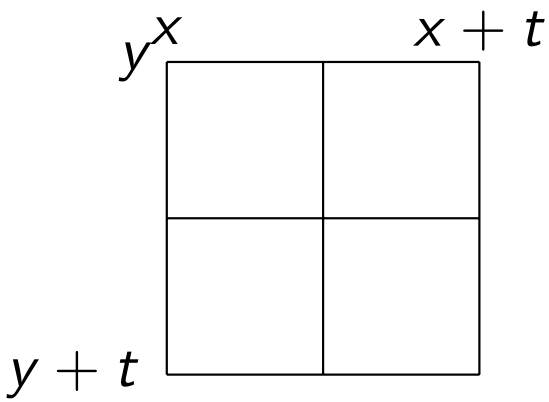


Figure 5 – $t = 2$. On divise encore la taille par 2 et on applique l'algorithme récursivement sur chaque partie.

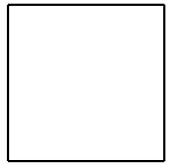


Figure 6 – $t = 1$; On ne fait rien.

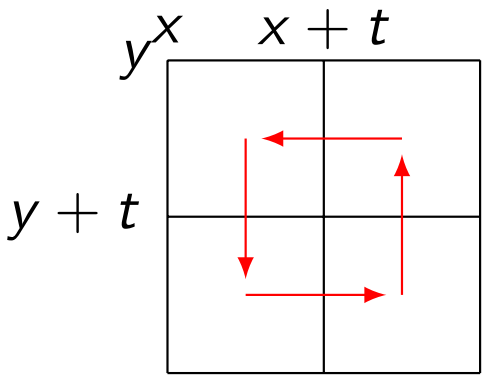


Figure 7 – On fait tourner et on remonte l'appel.

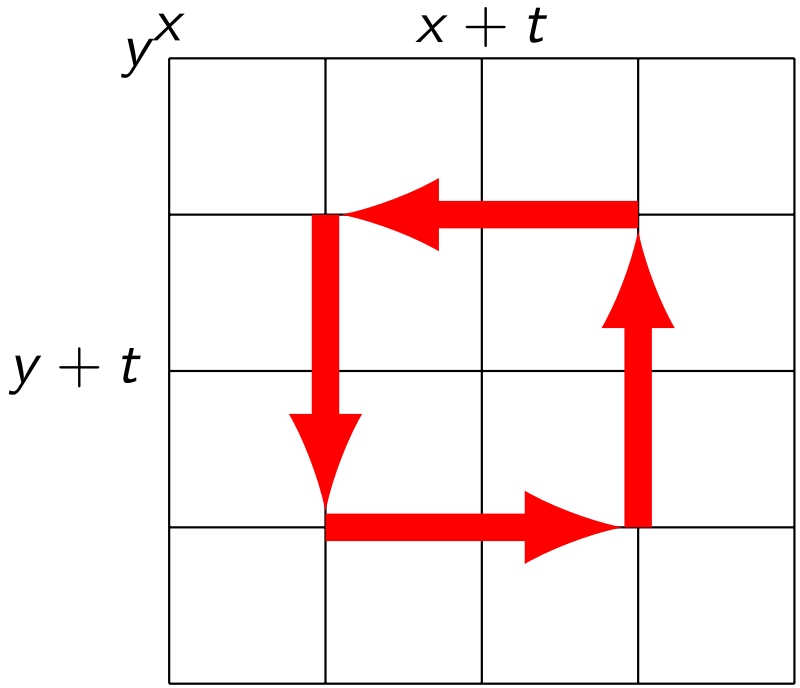


Figure 8 – On fait tourner et on remonte l'appel.

- Si la taille `t` est égal à 1, ne rien faire.
- Sinon : **découper en sous problèmes**
  - diviser la taille `t` en 2,
  - effectuer récursivement la rotation des **quatre parties** de la portion carrée.
  - **résoudre les petits problèmes :** Tourner les pixels.

Sommaire

Algorithme de rotation - Chargement de l'image

PIL (Python Image Library) -anciennement pillow- est une bibliothèque de traitement d'image.

```
1 from PIL import Image
2
3 im = Image.open("image.png")
4 im.show()
```

Code 1 – Charger une image

```
1 largeur, hauteur = im.size
2 px = im.load()
```

Code 2 – Récupérer des informations

Information

La variable `px` contient une matrice représentative des pixels de l'image. La couleur du pixel de coordonnées `(x,y)` est donnée par l'instruction `px[x,y]`. Il est également possible d'affecter une nouvelle `couleur` à un pixel : `px[x,y] = couleur`.

Activité 2 :

1. Récupérer une image carrée sur https://www.freepng.fr/. Le côté doit être une puissance de 2.
2. Créer une classe **`Image_lib`** et son constructeur qui admet un paramètre : le chemin de l'image. Ce constructeur construira alors un objet **`Image`** de la bibliothèque **`PIL`**.
3. Écrire la méthode **`montrer`**, sans paramètre, qui affiche l'image.
4. Créer une instance de la classe **`Image_lib`** en lui passant pour argument, le chemin de l'image à faire tourner.

Avant de regarder la correction

- Prendre le temps de réfléchir,
- Analyser les messages d'erreur,
- Demander au professeur.

Correction

```python
from PIL import Image

class Image_lib:

    def __init__(self, fichier: str) -> None:
        self.image = Image.open(fichier)
        self.largeur, self.hauteur = self.image.size
        self.px = self.image.load()
```

Code 3 – Constructeur

```
1 def montrer(self):
2     """
3     affiche l'image
4     """
5     self.image.show()
```

Code 4 – Afficher l'image

```
1 im = Image_lib("angry.png")
2 im.montrer()
```

Code 5 – Programme principal

Sommaire

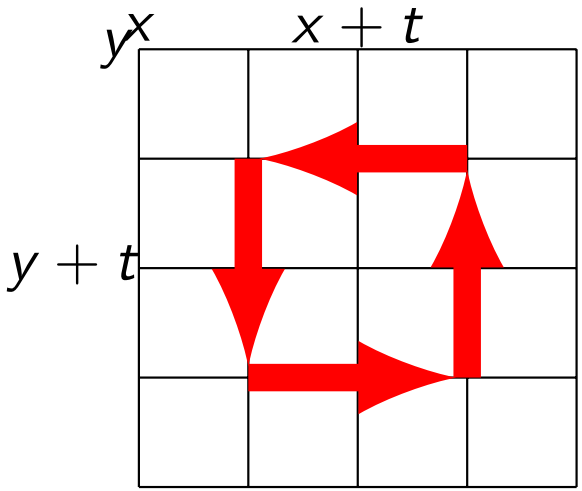
Diviser

Activité 3 : On suppose qu'on dispose de la méthode `tourner_antihoraire(self, x: int, y: int, t: int) -> None` qui fait tourner les 4 blocs dans le sens anti-horaire.
Écrire alors la méthode récursive `rotation(self, x: int, y: int, t: int)` $\rightarrow$ `None` qui effectue les quatre rotations récursives sur les quatre blocs, puis fait tourner à l'aide de la fonction `tourner`.

Avant de regarder la correction

- Prendre le temps de réfléchir,
- Analyser les messages d'erreur,
- Demander au professeur.

Correction

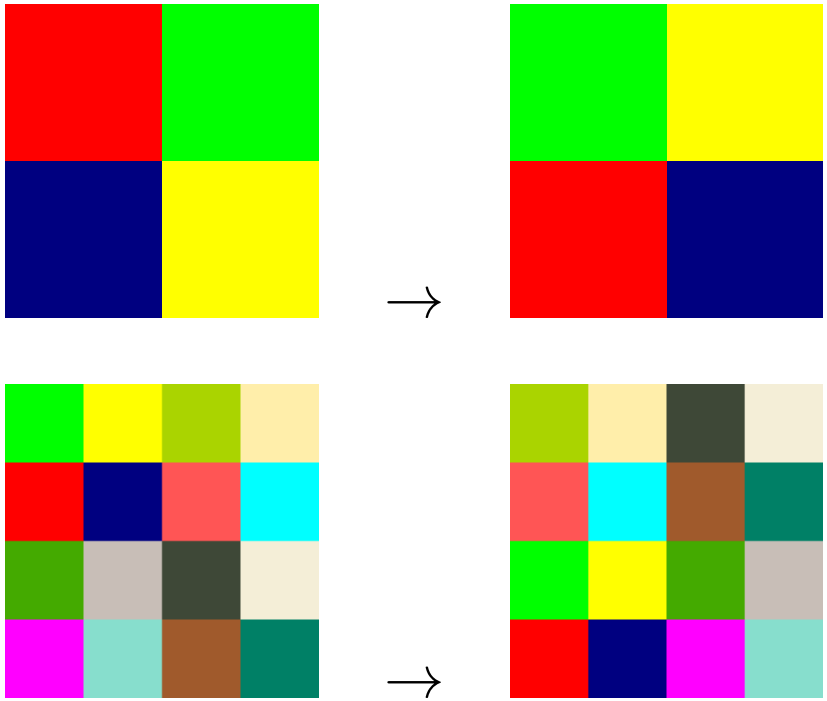



```
1  def rotation(self, x: int, y: int, t: int) -> None:
2      if t > 1:
3          t = t // 2
4          self.rotation(x, y, t)
5          self.rotation(x+t, y, t)
6          self.rotation(x, y+t, t)
7          self.rotation(x+t, y+t, t)
8
9          self.tourner_antihoraire(x, y, t)
```

Code 6 – Le cas limite est atteint quand on a 1 seul pixel.

Sommaire

Résoudre un petit problème



Activité 4 : Écrire la méthode `tourner_antihoraire(self, x: int, y: int, t: int)` → `None` qui fait tourner les pixels compris dans l'intervalle de colonnes $[x; x + t]$ et l'intervalle de lignes $[y; y + t]$.

Avant de regarder la correction

- Prendre le temps de réfléchir,
- Analyser les messages d'erreur,
- Demander au professeur.

Correction


```
1 def tourner_antihoraire(self, x: int, y: int, t: int) -> None:
2     for l in range(y, y+t):
3         for c in range(x, x+t):
4             self.px[l, c+t], self.px[l+t, c+t], \
5             self.px[l+t, c], self.px[l, c] = \
6             self.px[l, c], self.px[l, c + t], \
7             self.px[l+t, c+t], self.px[l+t, c]
```

Code 7 – Chaque *bloc* tourne d'un cran dans le sens anti-horaire

Activité 5 :

1. Écrire la méthode **`tourner_horaire`**, symétrique de **`anti_horaire`**.
2. Écrire la méthode **`fait_tourner`** qui accepte un paramètre de type booléen. Si l'argument passé est **`True`**, la méthode effectuera une rotation dans le sens horaire.
3. Écrire alors un programme principal qui instancie la classe et effectue deux rotations.

Avant de regarder la correction

- Prendre le temps de réfléchir,
- Analyser les messages d'erreur,
- Demander au professeur.

```python
def tourner_antihoraire(self, x: int, y: int, t: int) -> None:
    for l in range(y, y+t):
        for c in range(x, x+t):
            self.px[c, l+t], self.px[c+t, l+t], \
            self.px[c+t, l], self.px[c, l] = \
                self.px[c, l], self.px[c, l+t], \
                self.px[c+t, l+t], self.px[c+t, l]

def tourner_horaire(self, x: int, y: int, t: int) -> None:
    for l in range(y, y+t):
        for c in range(x, x+t):
            self.px[c, l+t], self.px[c+t, l+t], \
            self.px[c+t, l], self.px[c, l] = \
                self.px[c+t, l+t], self.px[c+t, l], \
                self.px[c, l], self.px[c, l+t]
```

Code 8 – Tourner

```python
def fait_tourner(self, horaire: bool = True) -> None:
    """
    tourne l'image de 90°

    Paramètres
    -------
    horaire: booléen; défaut: True
        tourne de 90° dans le sens horaire si True,
        dans le sens anti-horaire sinon
    """
    self.rotation(0, 0, self.largeur, horaire)
```

Code 9 – Avec choix de la rotation

```python
def rotation(self, x: int, y: int, t: int, horaire: bool):
    if t > 1:
        t //= 2
        self.rotation(x, y, t, horaire)
        self.rotation(x+t, y, t, horaire)
        self.rotation(x, y+t, t, horaire)
        self.rotation(x+t, y+t, t, horaire)

        if horaire:
            self.tourner_horaire(x, y, t)
        else:
            self.tourner_antihoraire(x, y, t)
```

Code 10 – Il faut alors modifier la méthode **rotation**

```
1 im = Image_lib("angry.png")
2 im.montrer()
3 im.fait_tourner(True)
4 im.montrer()
5 im.fait_tourner(False)
6 im.montrer()
```

Code 11 – Programme principal