Tprotationimage

October 5, 2024

1 TP rotation d'une image.

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.

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

1.1 Principe

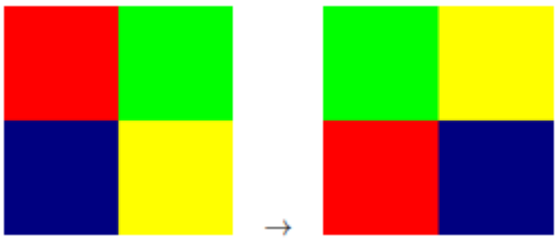
1.1.1 Première approche

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.

1.1.2 Activité 1

Consigne Réflexion commune : Considérons une Résoudre un petit problème image aux dimensions connues. Quelles étapes pourrions-nous imaginer pour répondre à notre problématique ?

Correction *1pixel ne rien faire*

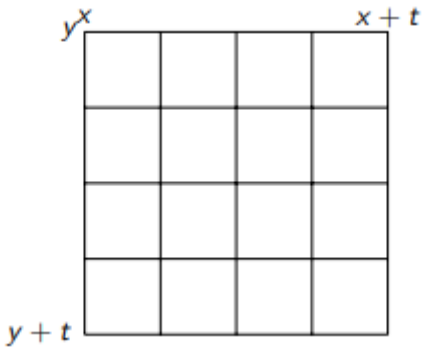


Rotation

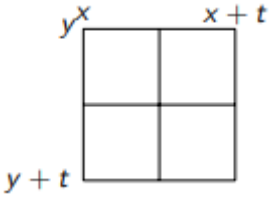
Récursivité : on divise la taille des problèmes par 2.

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

1.2 Construction de l'algorithme

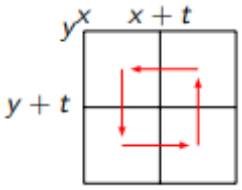


t = 4. On divise la taille par 2 et on applique l'algorithme récursivement sur chaque partie

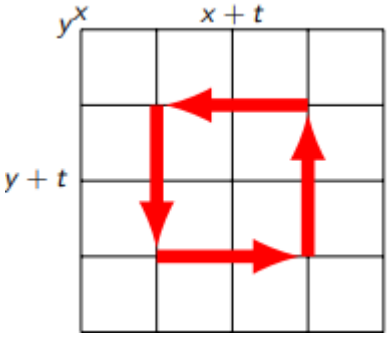


t = 2. On divise encore la taille par 2 et on applique l'algorithme récursivement sur chaque partie.

t = 1 ; On ne fait rien.



On fait tourner et on remonte l'appel



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.

1.3 Algorithme de rotation

1.3.1 - Chargement de l'image

```
from PIL import Image

im = Image.open("image.png")
im.show()

# Récupération des informations
largeur, hauteur = im.size
2 px = im.load()
```

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** ***

1.4 Activité 2

1.4.1 Consigne

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.

1.4.2 Correction

```
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()

    def montrer(self):
        """
         affiche l'image
         """
        self.image.show()

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

1.5 L'algorithme de rotation

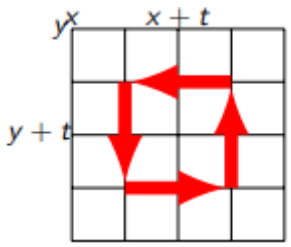
1.5.1 Diviser

1.6 Activité 3

1.6.1 Consigne

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) → None** qui effectue les quatre rotations récursives sur les quatre blocs, puis fait tourner à l'aide de la fonction tourner.

1.6.2 Correction



```
def rotation(self, x: int, y: int, t: int) -> None:
    if t > 1:
        t = t // 2
        self.rotation(x, y, t)
        self.rotation(x+t, y, t)
        self.rotation(x, y+t, t)
        self.rotation(x+t, y+t, t)

        self.tourner_antihoraire(x, y, t)
```

1.7 Activité 4

1.7.1 Consigne

Ecrire 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].

1.8 Activité 5

1.8.1 Consigne

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

1.8.2 Correction

```
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[l, c+t], self.px[l+t, c+t], self.px[l+t, c], self.px[l, c] = \
            self.px[l, c], self.px[l, c + t],  self.px[l+t, c+t], self.px[l+t, c]
```

1.8.3 Correction

```
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+t, l+t], self.px[c+t, 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 c in range(x, x+t):
        for l in range(y, y+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]

        # avec choix de la rotation
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)


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)
# modification de la méthode rotation

im = Image_lib("angry.png")
im.montrer()
im.fait_tourner(True)
im.montrer()
im.fait_tourner(False)
im.montrer()
```