Sup de Vinci
Bachelor 1
Initiation à Python

Dithering d'images

Introduction

Ce TP a pour objectif de vous familiariser avec les structures de données en matrices, et l'utilisation d'algorithmes sur ces données

Récupérer le code de base de ce TP, et le comprendre avant de commencer.

Une image est lue, et est transformée en liste de liste de pixels:

```
# Chaque pixel est une liste de 3 couleurs: Red, Green et Blue (RGB)
image = [
  #   X = 0          X = 1          X = 2
  [ [0, 0, 0], [ 255, 255, 255], [0, 0, 0], ],   # Y = 0
  [ [0, 0, 0], [ 255, 255, 255], [0, 0, 0], ],   # Y = 1
  [ [0, 0, 0], [ 255, 255, 255], [0, 0, 0], ],   # Y = 2
  [ [0, 0, 0], [ 255, 255, 255], [0, 0, 0], ],   # Y = 3
]
```

L'objectif de ce TP est d'implémenter l'algorithme de Dithering de Floyd-Steinberg, permettant d'afficher une version *compressée* d'une image en réduisant ses couleurs.

Dans le code fourni, une image est chargée depuis le disque, vous pouvez charger vos propres images de test.

1 - Créer une classe "Dither"

Créer une classe `Dither` prenant en paramètre une grille que l'on ajoute aux attributs, et qui initialise à `0` 5 coefficiants notés de `a` à `e`.

Ces coefficiants nous serviront plus tard dans le TP

2 - Parcourir l'image

1. Creér une méthode `dither` ne prenant aucun paramètre

Dans cette méthode, parcourir tous les pixels contenus dans la grille, et pour chacun d'entre eux, récupérer ses 3 composantes R, G et B

2. Appeler la fonction `find_closest_color`, en passant en paramètre la valeur du pixel courant, et récupérer la valeur du nouveau pixel dans une variable.

Modifier la valeur du pixel courant, en lui assignant sa nouvelle valeur

3. Appeler la fonction `pixel_error` en passant en paramètre la valeur du nouveau pixel courant, et son ancienne valeur.

Cette fonction retourne la "distance" qui sépare le pixel original avec la couleur par laquelle on l'a remplacé

L'erreur est composée de 3 éléments, une distance pour chaque composante de couleur R, G et B

3 - Propager l'erreur

1. Afin d'avoir un code lisible, nous allons propager l'erreur en appelant une nouvelle méthode que vous créerez à l'étape suivante, appelée `forward_error`

Cette méthode prendra 4 arguments:
- La coordonée `x` et `y` du pixel auquel propager l'erreur,
- La valeur de l'erreur calculée
- Le coefficiant de propagation

2. Appeler la fonction `forward_error` sur les pixels aux alentours du pixel courant:
- Le pixel aux coordonnées `x+1`, `y` aura pour coefficiant `self.a`
- Le pixel aux coordonnées `x-1`, `y+1` aura pour coefficiant `self.b`
- Le pixel aux coordonnées `x`, `y+1` aura pour coefficiant `self.c`
- Le pixel aux coordonnées `x+1`, `y+1` aura pour coefficiant `self.d`
- Le pixel aux coordonnées `x`, `y+2` aura pour coefficiant `self.e`

3. Implémenter la fonction `forward_error`

Cette fonction retourne directement sans aucun effet si le pixel donné en paramètre est hors de l'image

La fonction a pour but d'altérer le pixel donné par les arguments `x` et `y`, en lui additionnant l'erreur multipliée par un coefficiant

Ainsi, la valeur du pixel P sera définie par:

$$P_R = P_R + (E_R * k)$$

Où:
- P_R est la composante rouge du pixel concerné
- E_R est la composante rouge de l'erreur
- k est le coefficiant de propagation de l'erreur

Note: Utiliser la fonction `to_byte` pour obtenir une valeur de composante acceptée, cad un entier positif entre `0` et `255`

4. Tester le fonctionnement en créant une instance de `Dither`, en appelant la méthode `dither` pour appliquer les changements, et utiliser la fonction `show` pour afficher l'attribut `grid` de votre instance.

4 - Floyd-Steinberg

L'algorithme de Floyd-Steinberg est un algorithme de Dither dont les coefficiants de propagations sont définis par:

- `a = 7 / 16`
- `b = 3 / 16`
- `c = 5 / 16`
- `d = 1 / 16`
- `e = 0`

1. Créer une classe `FloydSteinberg` héritant de la classe `Dither`, et y effectuer les changements nécessaire dans les attributs pour obtenir un algorithme de Floyd-Steinberg.

2. Tester le fonctionnement, noter les changements avec la méthode précédente

5 - Atkinson

L'algorithme de Atkinson est un algorithme de Dither dont les coefficiants de propagations sont définis par:

- `a = 1 / 8`
- `b = 1 / 8`
- `c = 1 / 8`
- `d = 1 / 8`
- `e = 1 / 8`

1. Créer une classe `Atkinson` héritant de la classe `Dither`, et y effectuer les changements nécessaire dans les attributs pour obtenir un algorithme de Atkinson.

2. Tester le fonctionnement, noter les changements avec la méthode précédente