Rapport Exercice 4

Sanna Thomas, L3STI

September 16, 2024

Introduction de l'exercice 4

Les couleurs sont souvent décomposées en composantes rouge, verte et bleue en infor- matique, et représentées par une série de réels compris entre 0 et 1 (inclus).

Question 1 — Représentation des couleurs avec double

En utilisant des doubles pour représenter les couleurs, chaque composante de couleur est représentée par un double de 64 bits (8 octets).

Donc, pour une couleur, nous avons 3 doubles $\times$ 8 octets = 24 octets.

Question 2 — Représentation des couleurs avec des entiers

En utilisant des entiers pour représenter les couleurs, chaque composante de couleur est représentée par un entier compris entre 0 et 255, soit 8 bits (1 octet).

Donc, pour une couleur, nous avons 3 entiers $\times$ 1 octet = 3 octets, soit 8 fois moins que la représentation avec des doubles. Cependant, la précision est moindre.

Question 3 — Fonction de conversion et empaquetage dans un entier

L'objectif est de convertir les composantes de couleur en entiers et de les empaqueter dans un seul entier.

Variables

`r`, `v`, `b`: les composantes de couleur en entier, comprises entre 0 et 255.

Fonction de paquetage

La fonction de paquetage prend les composantes de couleur en entier et les empaquette dans un seul entier. Elle vérifie tout d'abord que les composantes sont bien comprises entre 0 et 255, puis les empaquette dans un seul entier en utilisant des opérateurs bitwise tels que $\ll$ (Décalage à gauche), pour mettre côte-à-côte les différents octets, et | (OU inclusif logique).

```
1
2  def empaquetage(r:int, v:int, b:int)->int:
3  """
4  Empaquetage de couleurs dans un entier
5
6  Args:
7      r (int): la couleur rouge, comprise entre 0 et 255.
8      v (int): la couleur verte, comprise entre 0 et 255.
9      b (int): la couleur bleue, comprise entre 0 et 255.
10
11 Returns:
12     int: Empaquetage sous forme d'entier
13 """
14 assert 0 <= r <= 255 and 0 <= v <= 255 and 0 <= b <= 255,
       ↪ "Les couleurs doivent être comprises entre 0 et 255"
15
16 return b | (v << 8) | (r << 16)
```

Listing 1: Fonction de Paquetage

Fonction de dépaquetage

La fonction de dépaquetage prend un entier empaqueté et le décompose en ses composantes de couleur en entier. Elle utilise des opérateurs bitwise tels que ≫ (Décalage à droite) et & (ET logique) pour extraire les différents octets de l'entier empaqueté.

```
1
2  def depaquetage(paquet:int)->tuple[int, int, int]:
3  """
4  Dépaquetage
5
6  Args:
7      paquet (int): La couleur sous forme de paquet à
       ↪ convertir
8
9  Returns:
10     tuple[int, int, int]: Retourne les entiers des couleurs
       ↪ rouge, bleue et verte
11 """
12 r = (paquet >> 16) & 0b11111111 # On réduit le nombre à 8
   ↪ bits pour obtenir la couleur rouge
13 v = (paquet >> 8) & 0b11111111
14 b = (paquet) & 0b11111111
15
16 return r, v, b
```

Listing 2: Fonction de Paquetage

Exemple et Output

Explication étape par étape

Supposons que nous avons les composantes de couleur suivantes:

- Rouge: 122
- Vert: 22
- Bleu: 244

Étape 1: définition des couleurs

- Rouge: 122 en binaire (1 octet): '01111010'
- Vert: 22 en binaire (1 octet): '00010110'
- Bleu: 244 en binaire (1 octet): '11110100'

Étape 2: Empaquetage des couleurs dans un entier

- Empaquetage: '01111010(r) 00010110(v) 11110100(b)' en binaire (24 bits) = 8001268 en décimal.

Étape 3: Dépaquetage de l'entier

- Dépaquetage de 8001268 en entiers: Rouge: 122, Vert: 22, Bleu: 244.

Exemple d'éxécution

```
r, v, b = (122, 22, 244)
print(empaquetage(r, v, b))
print(depaquetage(empaquetage(r, v, b)))
```

Listing 3: Exemple d'éxécution

```
>>> 8001268
>>> (122, 22, 244)
```

Listing 4: Output