Les Tuiles de Wang

Atelier de programmation pour enfants

Nicolas Seriot, avril 2023

But du projet

- Générer des images de pavages
  - aléatoires
  - sur des grilles de taille variables
  - avec des dessins différents (terrain, rails, tuyaux, etc)
- De l’idée au résultat: comment organiser un projet

Outils et techniques

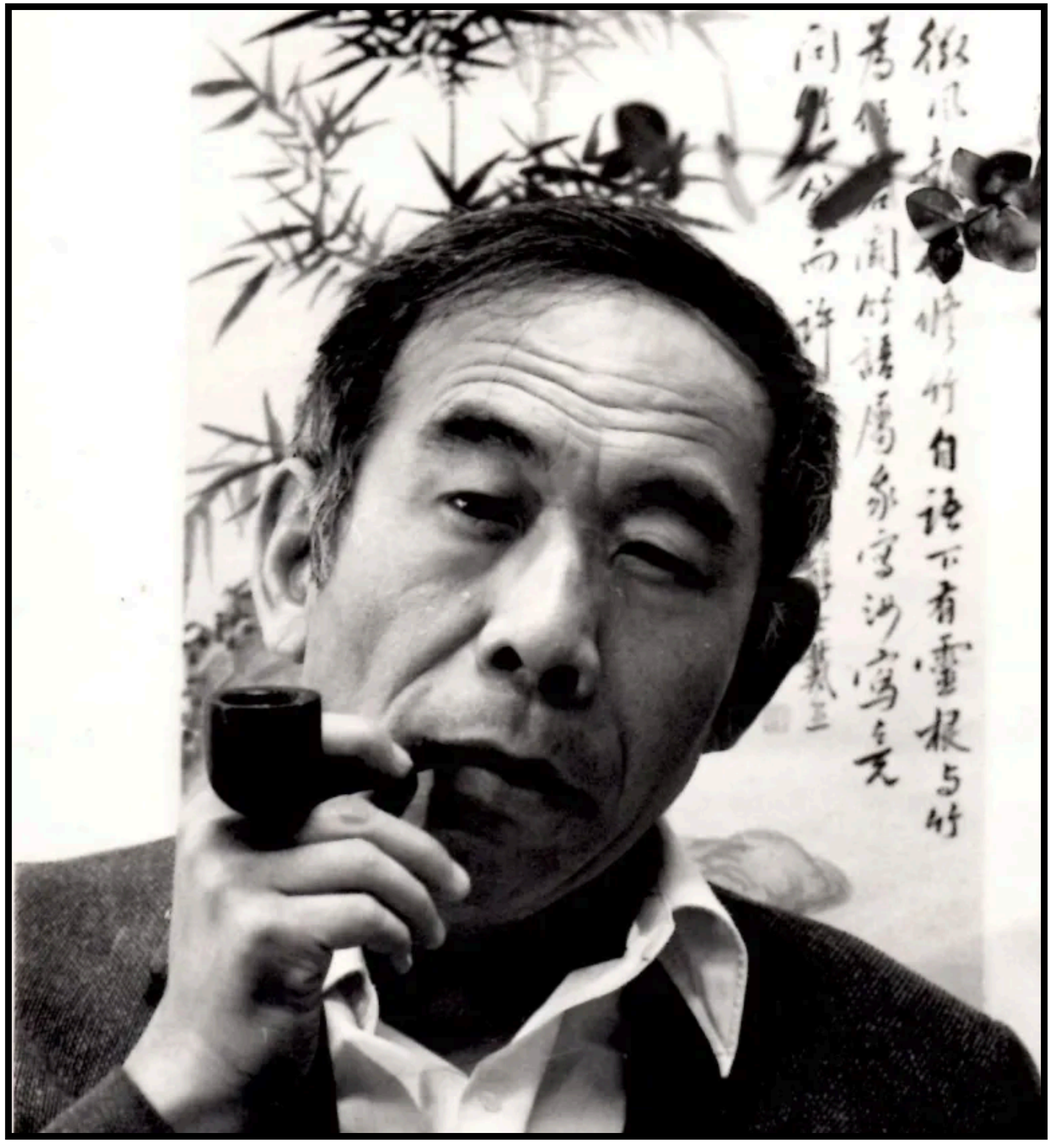
- Python
  - Listes et tableaux
  - Fonctions
  - Opérateurs binaires
- Pycairo
  - Translations
  - Crop

- 1. Introduction
  - Wang
  - Les tuiles
- 2. Créer le modèle
  - Listes et matrices
  - fill_board()
- 3. Créer la vue
  - draw_board()
  - draw_tile()
    - Sprites
- 4. Écrire l’algorithme
  - Opérateurs binaires
  - clean edges

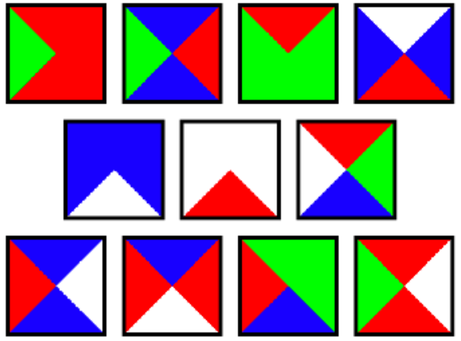
1. Introduction

Hao Wang

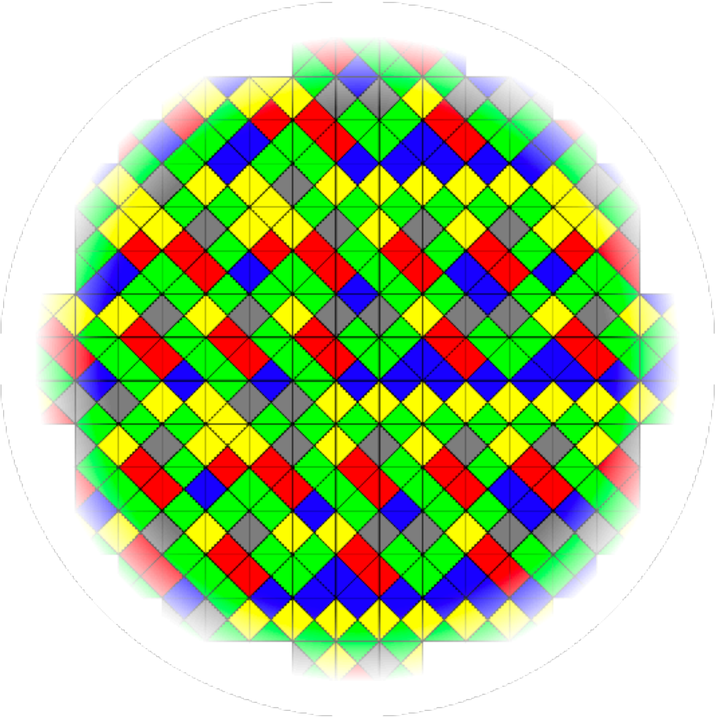
- Mathématicien et philosophe
- Chinois et américain
- Dans les années 1960
- Intéressé par le pavage de surfaces



Les pavés (“tiles”)



- les tiles sont carrés
- chaque côté a une couleur
- l’orientation est fixe, pas de rotation
- les pavés côte à côte se correspondent
- on peut ainsi former des motifs infinis



Les listes et les tableaux

```
# liste

l1 = [1,2,3]

l[0] = 1

# liste de listes

l2 = [ [1,2,3], [4,5,6] ]

l3 = [[1,2,3],
      [4,5,6],
      [7,8,9]]

l3[2][2]
```

Création de listes et tableaux

```
list(range(4)) # [0,1,2,3]

l4 = [0 for x in range(3)] # [0,0,0]

# initialiser un tableau avec des zéros

l5 = [[0 for x in range(3)] for x in range(3)] # [[0,0,0], [0,0,0], [0,0,0]]
```

Les fonctions

```
def mon_calcul(a, b, c):
    x = a/b + c
    return x

r1 = mon_calcul(4,2,1) # 3

r2 = mon_calcul(1,1,1) # 2
```

2. Remplir le tableau

Créer et remplir le tableau aléatoirement

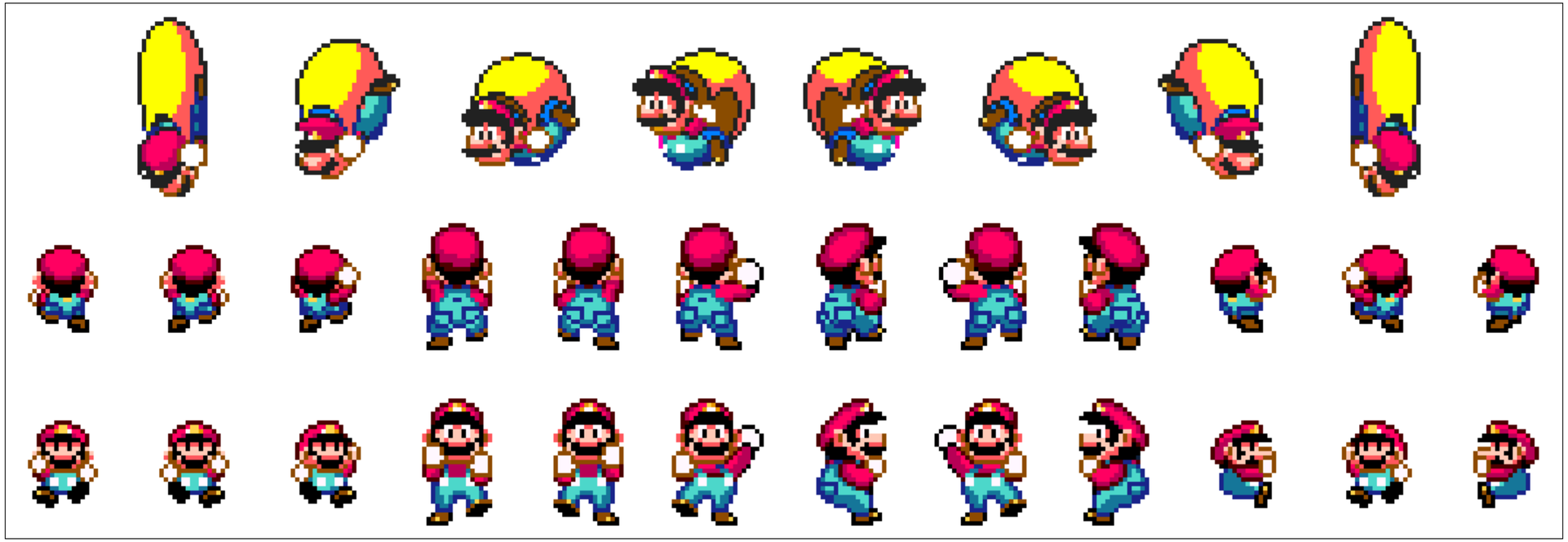
```
def fill_board():
    pass
```

3. Dessiner le tableau

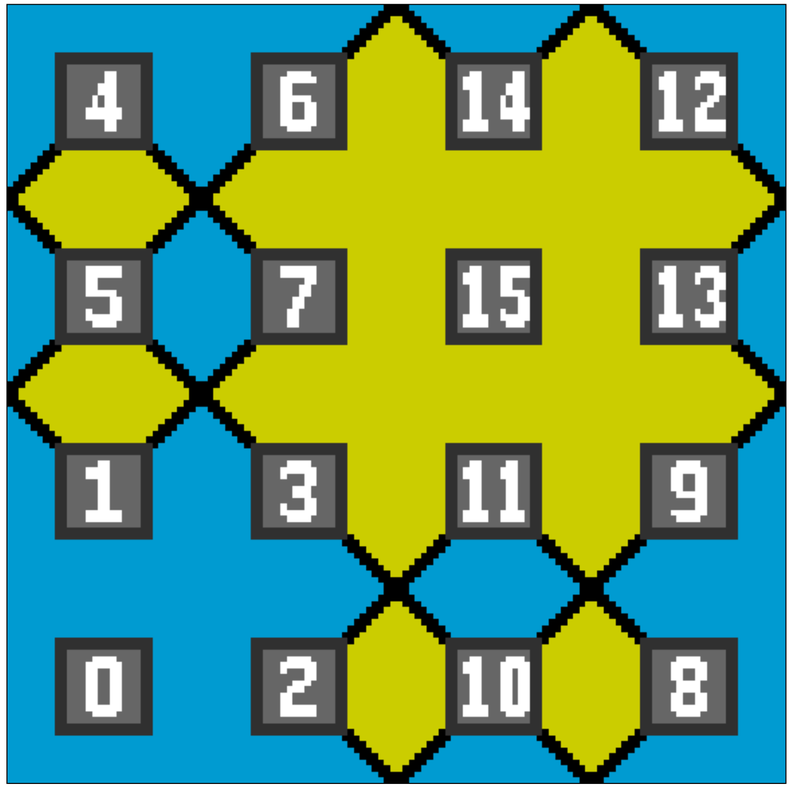
Dessiner le tableau

```
def draw_board():
    pass
```

Les Sprites



Les Sprites



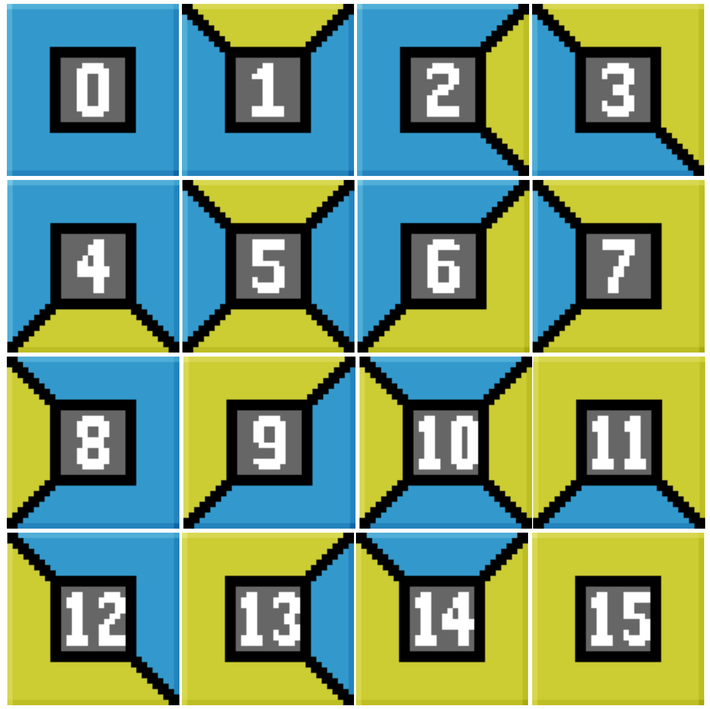

```
pos = [(0,3),(0,2), …]
```

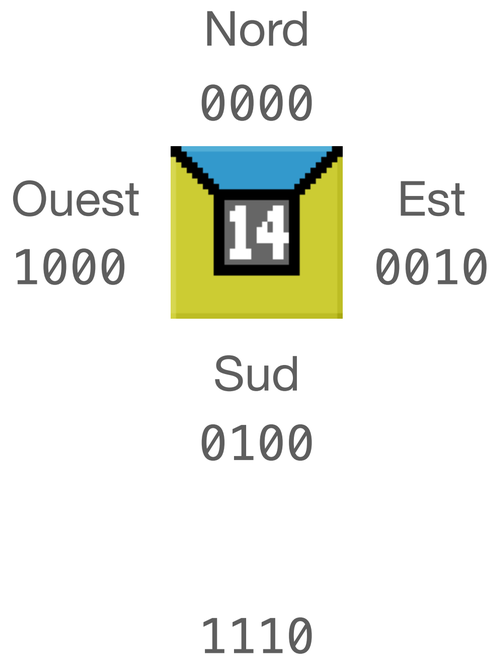
Dessiner une tile

```
def draw_tile():
    pass

def draw_board():
    # …
    # draw_tile()
    # …
```

4. Faire que les côté se correspondent

Les tiles à 2 couleurs, codage binaire





1x1 + 0x2 + 0x4 + 1x8 = 9

Opérations sur les bits

```
# ET logique
# (si les deux sont vrais)

1 & 1 = 1
1 & 0 = 0
0 & 1 = 0
0 & 0 = 0

# OU logique
# (si au moins un des deux est vrai)

1 | 1 = 1
1 | 0 = 1
0 | 1 = 1
0 | 0 = 0
```

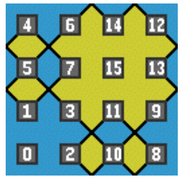
```
1100 & 1101 = 1100

0110 & 0001 = 0000

1110 & 1000 = 1000



1100 | 1101 = 1101

0110 | 0001 = 0111

1110 | 1000 = 1110

0010 | OSEN = 00E0
```

L’algorithme

```
def fill_board():
    # idée
    # on parcours toutes les cases de haut en bas et de gauche à droite
    # on propage les couleurs d’est en ouest
    # on propage les couleurs du sud au nord
```

Sources

- https://en.wikipedia.org/wiki/Hao_Wang_(academic)
- http://www.cr31.co.uk/stagecast/wang/intro.html
- https://en.wikipedia.org/wiki/Wang_tile


4 6 14 12
5 7 15 13
1 3 11 9
0 2 10 8