ACADÉMIE
DE NORMANDIE
Liberté
Égalité
Fraternité

Olympiades d'informatique 2023

- Classe de première –

Normandie

Durée de l'épreuve : 3 heures

L'épreuve se déroule par équipe et sur ordinateur.

Le sujet est constitué de deux parties indépendantes qui peuvent être traitées dans l'ordre souhaité.

Un dossier est mis à disposition des équipes avec les fichiers.

Les fichiers créés ou complétés sont à joindre dans un unique dossier compressé et qui seront nommés de la façon suivante (sans espace ni accents) :

NomEtablissement_VilleEtablissement_EquipeXX_pointillisme.py

NomEtablissement_VilleEtablissement_EquipeXX_representation.py
NomEtablissement_VilleEtablissement_EquipeXX_proiePred.py
NomEtablissement_VilleEtablissement_EquipeXX_param.py
NomEtablissement_VilleEtablissement_EquipeXX_initialisation.py
NomEtablissement_VilleEtablissement_EquipeXX_analyse_monde.py

où le numéro de l'équipe vous sera attribué par votre professeur.

À l'intérieur de chacun de ces fichiers, devront figurer en commentaire les noms de chaque membre de l'équipe.

Les calculatrices sont autorisées selon la règlementation en vigueur.

SIF Société Informatique de France

UFR SCIENCES & TECHNIQUES

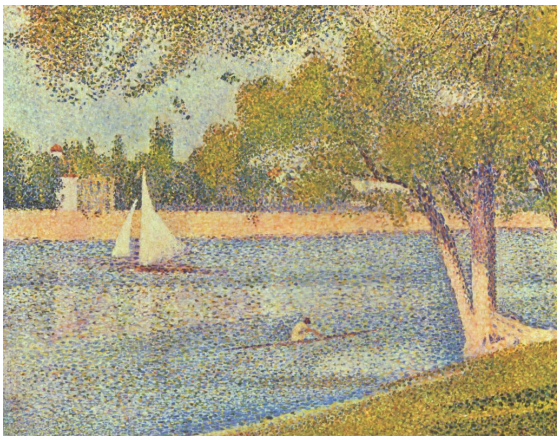
GREYC Laboratoire de recherche en sciences du numérique

Effet pointillisme

Présentation

D'après Wikipedia, "le pointillisme est un mouvement artistique de la peinture et une technique picturale qui utilise de petites zones de couleur juxtaposées plutôt que des mélanges de pâtes colorées."

Georges Seurat, Paul Signac, Pissaro font partie des artistes pointillistes les plus célèbres.



Georges Seurat, La Seine à la Grande Jatte. Printemps (huile sur toile, 1888)

Albert Dubois-Pillet, Les bords de Marne à l'aube (1886)

Consigne

Votre objectif est de concevoir un programme qui transforme une image en une version "pointilliste". Deux images sont fournies afin de tester votre programme [1]. La créativité et l'originalité seront appréciées dans l'évaluation (il est conseillé d'ajouter des commentaires dans votre code afin d'expliquer vos choix).

1. Source : Site Pexels, images libres de droit. Auteurs : Nextvoyage et Pixabay

Rappels sur les pixels et la bibliothèque PIL

On rappelle que la couleur d'un pixel est représentée par trois entiers variant entre 0 et 255, qui constituent la quantité de rouge, de vert et de bleu. Une image est en niveaux de gris lorsque n'importe lequel de ses pixels a la même quantité de rouge que de vert et de bleu. On rappelle également que la couleur noire correspond aux valeurs (0, 0, 0).

Pour la réalisation de votre code, vous allez utiliser la bibliothèque PIL, dont voici un usage possible (le fichier Python correspondant devant se trouver dans le même dossier que l'image `image1.jpg` :

```
from PIL import Image

img1 = Image.open("image1.jpg")
img2 = Image.new("RGB", img1.size)

largeur, hauteur = img1.size


img2.save("image_modifiee.jpg")
```

La variable `img2` représente une image, pour l'instant composée uniquement de pixels noirs, aux mêmes dimensions que l'image fournie.

Votre code utilisera `getpixel` et `putpixel` de la bibliothèque PIL. Par exemple :

```
(rouge, vert, bleu) = img1.getpixel((0,0))
img2.putpixel((0,0), (rouge, vert, bleu))
```

Ce code permet de récupérer les quantités de rouge, vert, bleu du pixel de coordonnées (0,0) de la première image et de colorer le pixel correspondant de la variable `img2`.

Enfin, l'instruction `img2.show()` vous permet à tout moment d'afficher l'image correspondante à la variable `img2`.

Des loups et des moutons

Introduction

Le programme qui vous est fourni permet de simuler ce que l'on appelle un système Proies-Prédateurs : dans un monde, représenté par une grille carrée, des moutons et des loups se déplacent.

À chaque tour, la population qui se déplace (loups ou moutons) est tirée au hasard. Mais pour représenter le fait que les loups sont plus rapides que les moutons, ils sont en moyenne activés 3 fois plus souvent que les moutons.

Dès qu'un mouton n'a que des loups sur les cases autour de lui, il est attrapé. Par ailleurs, si, lors de 4 activations successives, un mouton n'a pas pu se déplacer car toutes les cases autour de lui sont restées occupées, il meurt de stress.

Le programme est exécuté en appelant la fonction `animer`. Cette fonction prend 3 paramètres :

— un entier précisant le nombre de moutons à mettre dans le monde ;

— un entier précisant le nombre de loups à mettre dans le monde ;

— un booléen (optionnel, à *vrai* par défaut) pour préciser si l'on veut que le monde soit affiché régulièrement (toutes les 10 étapes) ou non.

Dans la version qui vous est donnée, les loups se déplacent au hasard. Vous devez proposer une nouvelle version de la fonction `deplacer_un_loup` pour que ceux-ci, quelle que soit la taille de la grille, et la répartition des loups et des moutons, fassent disparaître le plus efficacement possible l'ensemble des moutons.

Présentation générale des différents fichiers

a. Fichier `initialisation.py`

Ce fichier contient l'ensemble des fonctions permettant de créer le monde initial, avec ses populations de moutons et de loups. vous n'avez pas à modifier ce fichier.

Le **monde** (un carré de côté `TAILLE` est représenté par une liste de listes. Chaque case peut ne rien contenir (`NONE`), contenir un loup ou contenir un mouton.

Un **mouton** est représenté par une liste de 6 cases :

— case `INDICE_TYPE` : contient `TYPE_MOUTON` pour préciser qu'il s'agit d'un mouton ;

— case `INDICE_NUM` : contient un entier correspondant au numéro du mouton (cette information n'est pas utilisée dans le code fourni) ;

— case `INDICE_LIGNE` : contient un entier représentant le numéro de la ligne occupée par le mouton ;

— case `INDICE_COLONNE` : contient un entier représentant le numéro de la colonne occupée par le mouton ;

— case `INDICE_INFOS` : peut contenir d'éventuelles informations propres au mouton. Non utilisée ici, donc initialisée à `NONE` ;

— case `INDICE_STRESS` : contient le niveau de stress du mouton. Initialisée à 0. S'il dépasse `STRESS_MAX`, le mouton meurt de stress.

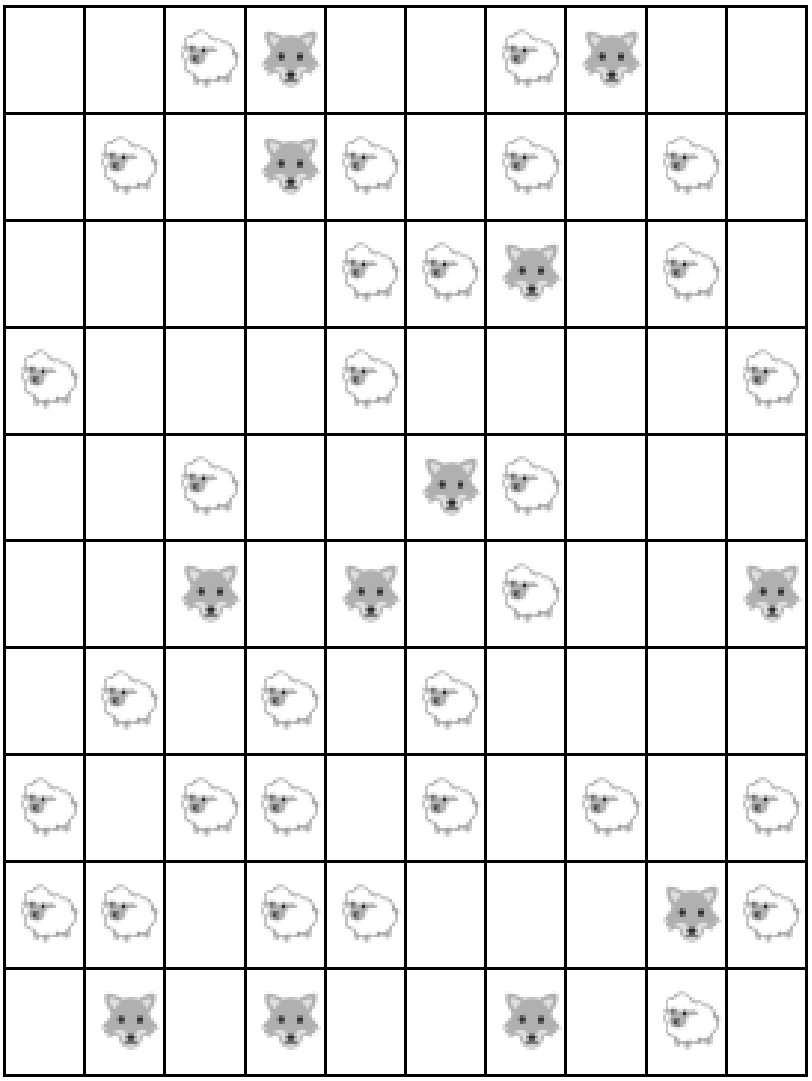


Figure 1 – Exemple de monde peuplé de moutons et de loups

Un **loup** est représenté par une liste de 5 cases :

— case `INDICE_TYPE` : contient `TYPE_LOUP` pour préciser qu'il s'agit d'un loup ;

— case `INDICE_NUM` : contient un entier correspondant au numéro du loup (cette information n'est pas utilisée dans le code fourni) ;

— case `INDICE_LIGNE` : contient un entier représentant le numéro de la ligne occupée par le loup ;

— case `INDICE_COLONNE` : contient un entier représentant le numéro de la colonne occupée par le loup ;

— case `INDICE_INFOS` : peut contenir d'éventuelles informations propres au loup. Non utilisée ici, donc initialisée à `NONE` mais vous pourrez éventuellement stocker ici une information voire plusieurs (avec une liste par exemple).

b. Fichier `representation.py`

Ce fichier contient l'ensemble des fonctions permettant d'afficher le monde à un instant donné. Vous n'avez pas à modifier ce fichier.

c. Fichier `param.py`

Ce fichier contient différentes constantes qui sont utilisées dans les différents fichiers du programme. Il est peu probable que vous ayez à modifier ce fichier.

d. Fichier `analyse_monde.py`

Ce fichier contient plusieurs fonctions permettant d'acquérir des informations à partir du monde. Certaines fonctions sont déjà utilisées dans le programme qui vous est donné, d'autres non, mais elles pourraient vous être utiles :

- fonctions `est_vide`, `est_mouton`, `est_loup` : 3 fonctions pouvant être utiles pour vérifier ce que contient une case du monde ;
- fonction `case_valide` : permet de savoir si un couple `(ligne, colonne)` est bien dans la grille ;
- fonctions `case_a_gauche`, `case_a_droite`, `case_au_dessus` et `case_au_dessous` : renvoient les coordonnées `ligne, colonne` de la case voisine de celle dont les coordonnées sont passées en paramètre dans la direction voulue. Si cela amène à sortir de la grille, alors ces fonctions renvoient `None, None` ;
- fonction `case_vers` : Cette fonction renvoie la liste des cases (une ou deux cases) voisines de la case courante et plus proche d'une case cible. La fonction ne vérifie pas si ces cases sont vides ou pas ;
- fonction `mouton_le_plus_proche` : Cette fonction renvoie le mouton le plus proche de la case dont les coordonnées sont passées en paramètres. S'il n'y a plus de mouton, la fonction renvoie `NONE` ;
- fonction `loups_les_plus_proches` : cette fonction renvoie les `n` loups les plus proches de la case passée en paramètre. S'il y a moins de loups que le nombre demandé, alors tous les loups sont renvoyés. Dans la liste renvoyée, les loups sont classés du plus près au plus éloigné ;
- fonction `voisins_libres` : cette fonction renvoie la liste des cases voisines de celle dont les coordonnées sont passées en paramètres et qui sont vides ;
- fonction `moutons_voisins` : cette fonction renvoie la liste des moutons situés sur les cases voisines de celle dont les coordonnées sont passées en paramètres.

e. Fichier `proiePred.py`

Ce fichier contient le programme principal. La plupart des fonctions se transmettent les informations suivantes :

- le *monde* : c'est la grille contenant les animaux ;
- les *infos* : ce sont des informations que les fonctions pourraient vouloir partager. Ce n'est pas utilisé dans la version proposée, mais si vous en avez besoin, n'hésitez pas à y stocker une information (ou plusieurs avec une liste) ;
- la *liste_moutons* : c'est la liste des moutons encore en vie ;
- la *liste_loups* : c'est la liste des loups dans le monde.

On trouve dans ce fichier les fonctions suivantes :

- fonction `animer` : c'est le programme principal. Elle exécute la simulation jusqu'à ce que tous les moutons aient disparu, puis affiche le nombre d'itérations qui ont été nécessaires pour éliminer tous les moutons ;
- fonctions `deplacer_les_loups` et `deplacer_les_moutons` : ces fonctions se contentent de déplacer respectivement tous les loups et tous les moutons. Vous n'avez pas à modifier cette fonction.

— fonction `deplacer_un_mouton` : cette fonction déplace aléatoirement le mouton passé en paramètre. Si c'est impossible, le niveau de stress du mouton augmente. Si ce niveau dépasse `STRESS_MAX`, le mouton meurt de stress. Vous n'avez pas à modifier cette fonction ;

— fonction `deplacer_un_loup` : c'est la fonction qui gère le déplacement d'un loup. Dans la version proposée ici, le déplacement est aléatoire. **C'est cette fonction que vous devez modifier**.

— fonction `deplacer_un_animal` : cette fonction est appelée par les 2 fontions `deplacer_un_loup` et `deplacer_un_mouton`. Il faut absolument passer par cette fonction quand on déplace un animal (mouton ou loup) pour être sûr que les informations sont bien modifiées correctement. Vous n'avez pas à modifier cette fonction ;

— fonction `gerer_prise_mouton` : cette fonction, appelée automatiquement lorqu'un loup est déplacé, supprime les éventuels moutons attrapés. Vous n'avez pas à modifier cette fonction.

Travail demandé

Vous devez modifier le programme fourni pour que les loups éliminent le plus efficacement possible les moutons. Parmi tous les fichiers qui vont sont donnés, vous pouvez (mais à chaque fois, ce n'est une obligation) :

— ajouter des fonctions au fichier `analyse_monde.py` ;

— ajouter des fonctions au fichier `proiePred.py` ;

— modifier la fonction `deplacer_un_loup` du fichier `proiePred.py` ;

— modifier la valeur initiale de la variable `infos_generales` dans la fonction `animer` ;

— modifier l'initialisation des informations spécifiques à chaque loup et chaque mouton dans les fonction `creer_mouton` et `creer_loup`.

Aucune autre modification que celles mentionnées ci-dessous n'est autorisée. Vous devrez commenter vos nouvelles fonctions, et expliquer en commentaire de la fonction `deplacer_un_loup` le principe de l'algorithme que vous avez choisi d'implanter. Outre la clarté de votre code, votre version sera testée sur des configurations différentes (taille de la grille, nombre de moutons, nombre de loups) et comparée aux versions des autres équipes participant à ces Olympiades.