IFT 1015 – Programmation I

TP 1

- À faire en groupe de **deux** étudiants.
- Remise : Le 10.07.2024 à **23:59** au plus tard

1. Objectifs du TP1

Dans ce projet, vous aurez l'occasion de pratiquer les concepts suivants:

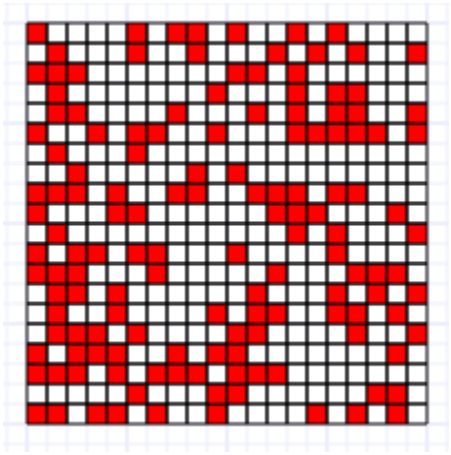
- Fonctions et la décomposition fonctionnelle
- Tableaux et enregistrements
- Opérations de dessin (tortue)

2. Introduction

Vous devrez écrire une implantation d'un jeu exécuté dans l'environnement codeBoot.

Le jeu de la vie, que vous devez coder, représente un automate cellulaire imaginé par John Horton Conway en 1970 (https://fr.wikipedia.org/wiki/Jeu_de_la_vie). Le jeu de la vie est un jeu dit à "0 joueur", au sens où le déroulement du jeu dépend seulement de sa configuration initiale.

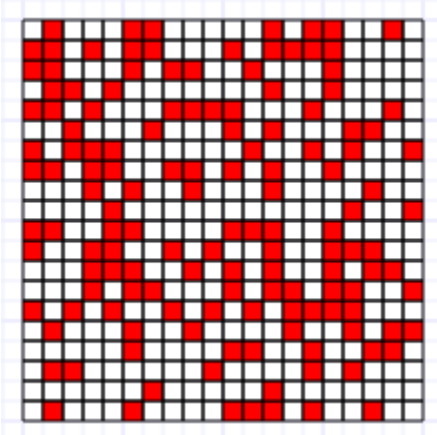
On dispose d'une grille à deux dimensions dont les cases représentent des cellules qui peuvent prendre deux états distincts : « vivante » ou « mortes ». Une cellule possède huit voisins, qui sont les cellules adjacentes horizontalement, verticalement et diagonalement et dont l'état change au fil du temps suivant des règles prédéfinies.

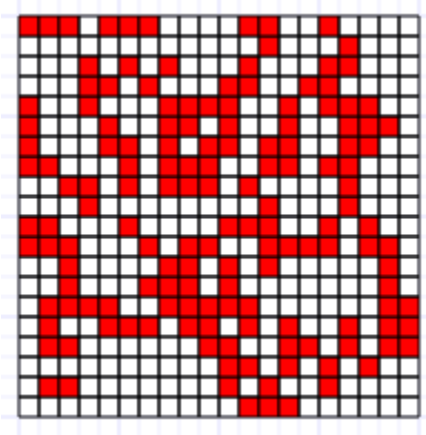


La partie se déroule de la manière suivante:

- Initialiser le jeu : laisser au hasard quelles cellules commencent vivantes et quelles commencent mortes
- Démarrer le jeu et laisser la grille changer d'état toute seule

- Apprécier les jolis motifs de l'automate qui évoluent sur l'écran.





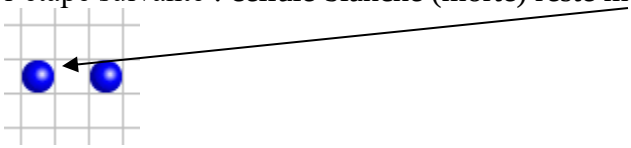
3. Déroulement du jeu

Une cellule est dite vivante lorsqu'elle est colorée (sur le dessin en rouge).

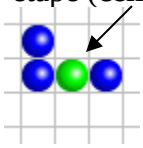
À chaque "tour", chaque cellule change d'état en fonction de ses voisins:

- Une cellule qui a moins de deux voisins vivants à une étape sera morte à la prochaine étape (sous-population);

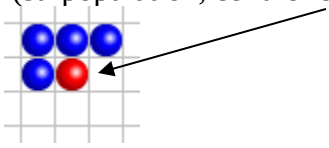
- Si une cellule a exactement deux voisines vivantes, elle reste dans son état actuel à l'étape suivante : cellule blanche (morte) reste morte.



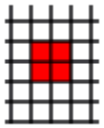
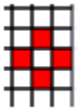
- Une cellule qui a deux ou trois voisins vivants continuera de vivre à la prochaine étape (cellule verte);



- Une cellule qui a plus de trois voisins vivants sera morte à la prochaine étape (surpopulation, cellule rouge);



- Une cellule morte qui a exactement trois voisins vivants "naîtra" à l'étape suivante et sera donc vivante (reproduction);

Quelques structures stables (plus de structures stables → https://fr.wikipedia.org/wiki/Jeu_de_la_vie)

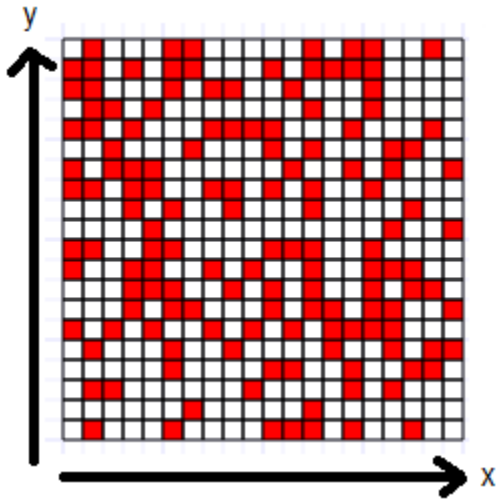
Structures qui oscillent

4. Spécification

Vous devez concevoir et coder les procédures spécifiées ci-dessous en respectant le nom spécifié exactement pour qu'on puisse les tester plus facilement. **Vous aurez sûrement à définir des fonctions et procédures auxiliaires (utilisez des noms appropriés de votre choix en camelCase).** Votre code doit être dans un fichier nommé **`jeuDeVie.py`**.

4.1 Fonction `creerGrille(tailleGrille)` crée un modèle de la grille de jeu et retourne la variable modèle de taille spécifiée (un tableau 2D, 1D, un enregistrement, …). Le paramètre **`tailleGrille`** est un enregistrement possédant 3 champs :

- **`nx`** – nombre des cases sur l'axe x
- **`ny`** – le nombre des cases sur l'axe y
- **`largeur`** – largeur d'une case de grille (en pixels)



Vous pouvez modéliser une grille de 2 dimensions en utilisant un tableau 1D et les fonctions auxiliaires permettant de transformer un index 1D en index 2D et vice versa. Par exemple, une grille 4*4 peut être modélisée comme un tableau 1D de 16 éléments. L'élément avec index 15 correspond à l'index (3, 3) sur une matrice de 2 dimensions.

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

4.2 Procédure `init(grille)` permet de créer l'état initial du modèle. Le paramètre **`grille`** est une variable modèle du jeu.

La procédure **`init(grille)`** doit d'abord déterminer le nombre des cellules vivantes et ensuite, générer les indexes des cellules vivantes sur la grille. On va utiliser le hasard pour effectuer ces deux tâches. Le nombre des cellules vivantes devra être généré aléatoirement en fonction de la taille d'une grille : de 10% à 50%. Donc, si le nombre des cases dans la grille de jeu est 25 et le pourcentage généré aléatoirement est 35%, donc vous devriez initialiser comme cellules vivantes, $0.35*25 = 8.75 \approx 9$ cellules ($8.25 \approx 8$). Ces 9 cellules, encore une fois, doivent être choisies sur la grille de manière aléatoire.

Le paramètre **`grille`** pourrait être une donnée structurée (un tableau, enregistrement, …).

4.3 Procédure `dessinerGrille(tailleGrille,grille)`

La procédure **`dessinerGrille(tailleGrille,grille)`** prend 2 paramètres, la taille de grille et le modèle grille, et affiche l'état du jeu selon le contenu de votre variable modèle. Le paramètre **`tailleGrille`** est décrit dans 4.1. Choisissez la couleur rouge pour colorier les cellules vivantes. Les cellules mortes laissez en blanc. Pour effectuer les dessins de l'évaluation du jeu, utilisez la tortue.

4.4 Procédure `jouer(tailleGrille)` : la procédure principale qui crée une grille de jeu (modèle), l'initialise et démarre le jeu en boucle infinie. Cette procédure doit implémenter l'évaluation du jeu avec l'animation :

- Étape 1. Création de la grille du jeu (modèle, une variable contenant un état de jeu) – appel à la fonction **`creerGrille(tailleGrille)`** ;
- Étape 2. Initialisation du modèle de jeu vide retourné par la fonction **`creerGrille(tailleGrille)`** en utilisant l'appel à la fonction **`init.`** Initialisation de l'état initial : générer le nombre des cellules vivantes et choisir les cellules vivantes.
- Étape 3 : Lancer le jeu en boucle infinie :

```
aff1 :    afficher l'état courant
          former l'état future
          affecter à la variable d'état courant la valeur de
          l'état future
```

`sauter à l'étape d'affichage aff1`

Redessinez les états du jeu chaque 0.1 seconde pour avoir une animation.

La procédure **`jouer(tailleGrille)`** prend un paramètre : l'enregistrement **`tailleGrille`** décrit dans 4.1

5. Remise

Vous devez dessiner une grille carrée 20*20 avec les cases de 10 pixels chacune. Pour ce devoir il faudra implémenter les tests unitaires seulement pour les fonctions auxiliaires utilisées et non pour les fonctions de dessins et les fonctions 4.1 – 4.4.

6. Évaluation

Indiquez vos noms clairement dans les commentaires au début de votre code.
Remettez le fichier `jeuVie.py` La remise doit se faire sur le site Studium du cours.
Voici les critères d'évaluation du travail :

- L'exactitude (respect de la spécification)
- L'élégance et la lisibilité du code
- La présence de commentaires explicatifs lorsque nécessaire
- Le choix des identificateurs
- La décomposition fonctionnelle et le choix de tests unitaires pertinents

Indications :

- La performance de votre code doit être raisonnable.
- Chaque fonction devrait avoir un bref commentaire pour indiquer ce qu'elle fait.
- Il devrait y avoir des lignes blanches pour que le code ne soit pas trop dense (utilisez votre bon sens pour arriver à un code facile à lire)
- Les identificateurs doivent être bien choisis pour être compréhensibles (évitez les noms à une lettre, à l'exception de i, j, . . . pour les variables d'itérations des boucles).
- Vous devez respecter le standard de code pour ce projet (soit, les noms de variables en «camelCase »).