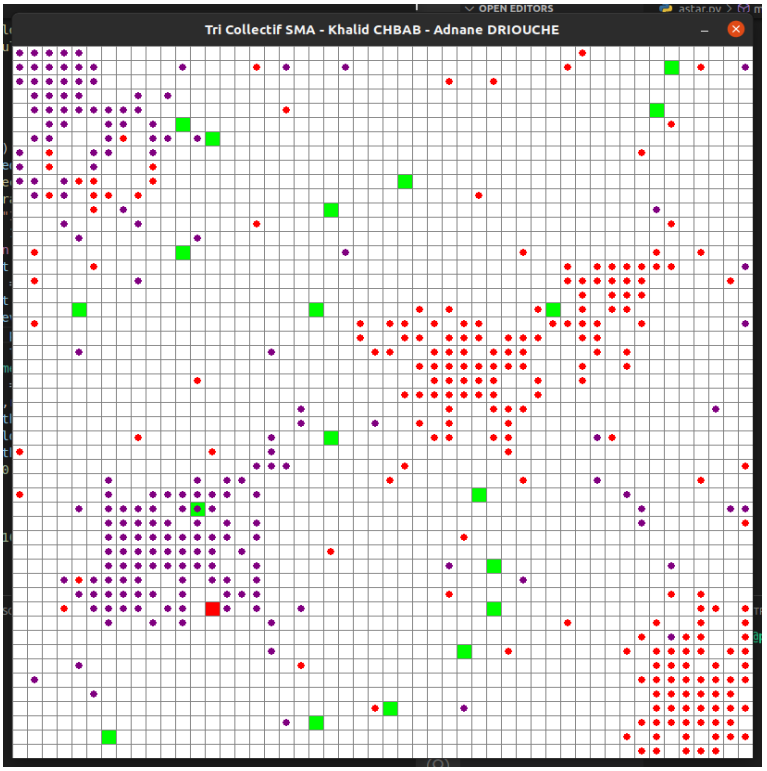
TP2 - V1 - TRI COLLECTIF

SMA



A l'intention de : Mme Salima HASSAS
Réalisé par : CHBAB Khalid & DRIOUCHE Adnane

I. Introduction

Nos agents(réactive) vivent dans un monde 2D constitué d'une grille(50x50), contenant des objets de type A et B en quantités spécifier lors d'exécution nA = nB = 200. Leur objectif principal est de regrouper les objets par type A ou B.

L'objectif de ce projet est de simuler le comportement émergent du tri collectif suivant des règles de perception/action décrits ci dessous :

- l'agent collecte des informations sur l'environnement dans les 8 directions .
- l'agent se déplace aléatoirement à chaque itération dans les 8 directions.
- l'agent peuvent prendre ou déposer selon des probabilités

$$P_{\text{prise}} = \left(k^{+}/\left(k^{+}+f\right)\right)^{2} \text{ et } P_{\text{dépôt}} = \left(f/(k+f)\right)^{2}$$

k + et k - des constantes et f représentant la proportion d'objet de même type A ou B dans l'environnement immédiat (voisinage de l'agent). Le voisinage de l'agent est défini par les cases atteignables en 1 pas de temps par l'agent dans les 8 directions.

l'agent est doté d'une mémoire de taille variable représentant les objets déjà rencontrés sur les N derniers pas.

II. Méthodologie

i. Définition des Agents et leurs comportements

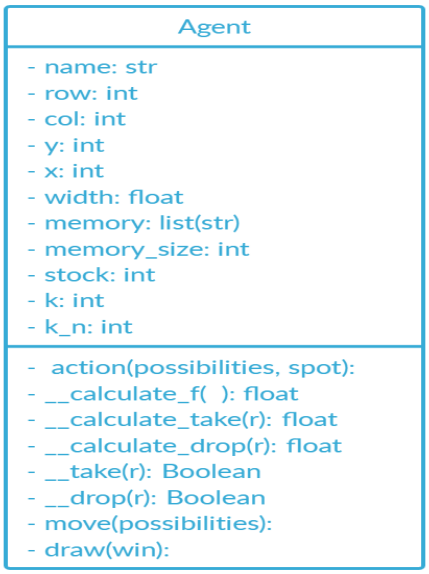


fig 1 : classe d'agent

Dans notre simulation, l'agent est représenté par un carré vert qui se déplace dans les 8 directions sur la grille. Les agents qui entrent en jeu sont tous des objets de la classe Agent :

Le programme d'un agent consiste en une boucle perception (), action().

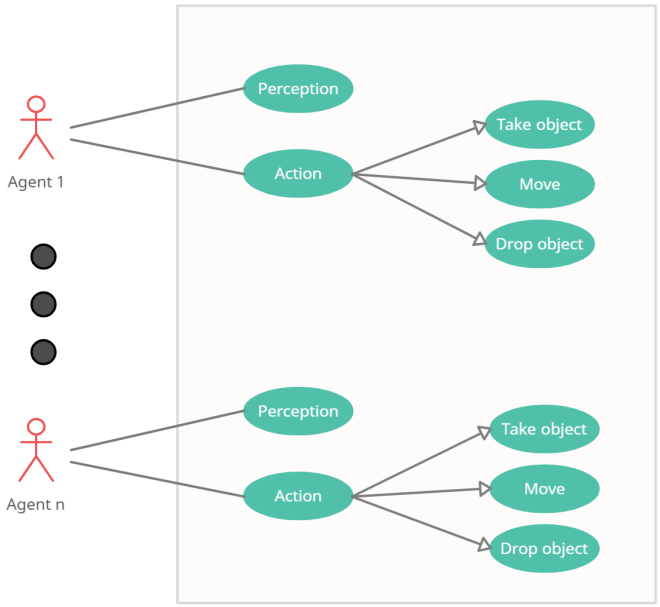


fig 2 : Diagramme de cas d'utilisation

La perception, permet à l'agent de récupérer les informations de l'environnement et l'action, dans le cas du sujet est soit : le déplacement (move) d'un pas, la prise (take object) d'un objet ou le dépôt (drop object) d'un objet.

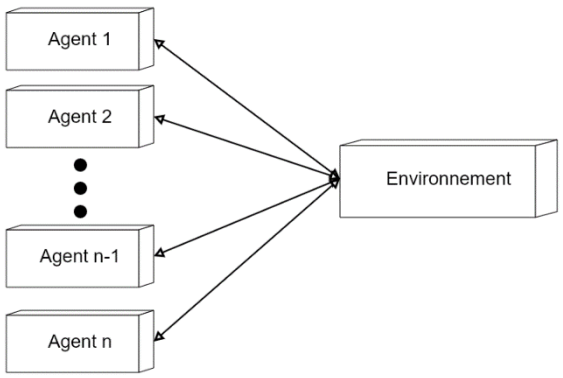


fig 3 : Interaction environnement agents

III. Implémentation de la solution

Cette simulation est construite au-dessus de la bibliothèque Pygame, elle utilise le concept POO pour représenter le monde et est développée en utilisant python. aucun multithreading n'est implémenté et les agents se relaient pour déplacer chaque itération.

i. Structuration du code

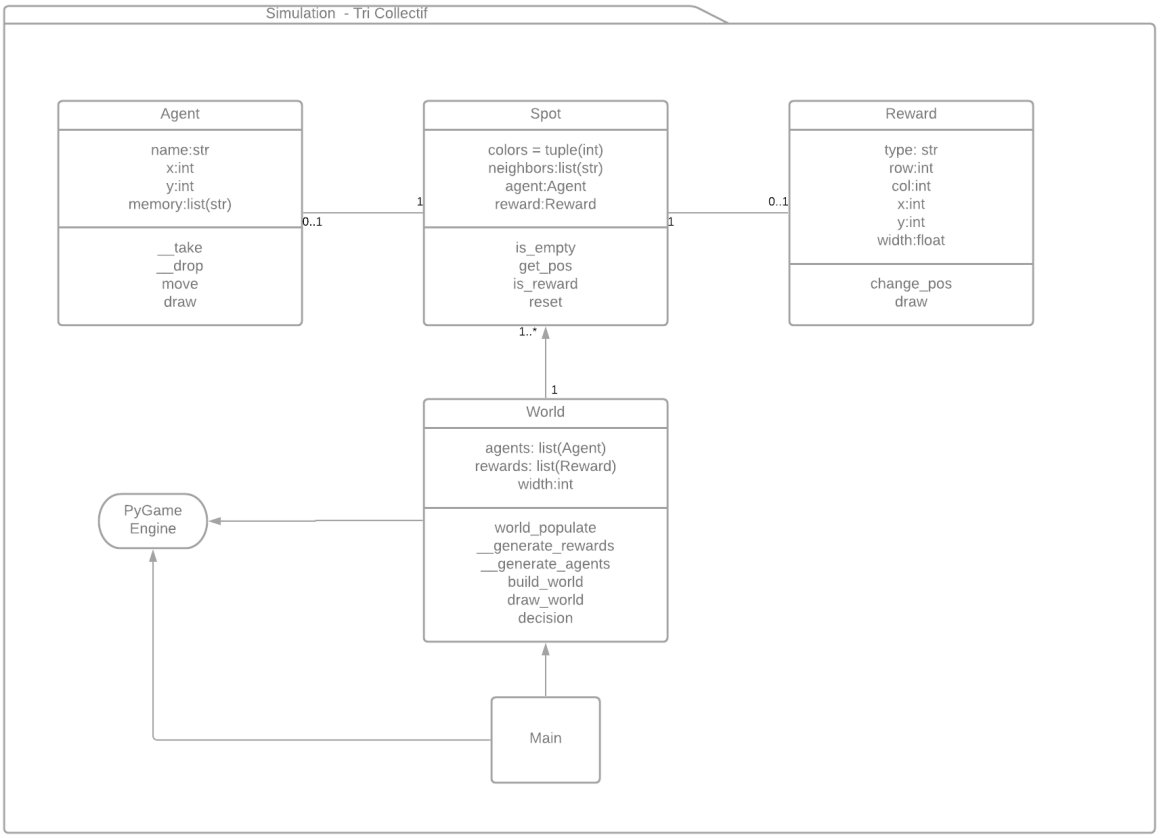


fig 4 : diagramme de classe

Les classes :

- **Agent** : la classe d'agent contient tous les opérations liées à l'agent notamment :
  - initialisation des propriétés d'agent
  - action on se basent sur les informations envoyées par l'environnement
  - déplacement aléatoire en suivant les possibilitées fourni par l'environnement
  - control d'état interne comme la mémoire
  - prise/dépôt des objets
- **World** : la classe World ou simulation englobe l'environnement et affichage(GUI) :
  - initialisation des propriétés d'agent
  - génération des agents, objets, grill et les cellules
  - réaliser la perception et envois aux agents
  - mise à jour de GUI par itération
- **Reward** : la classe Reward ou Objet contient l'abstraction de caractéristiques d'objet.
- **Spot** : le bloc minimal de construction de terrain de simulation/world.
- **Main** : point d'entrée de programme et exécution de simulation.

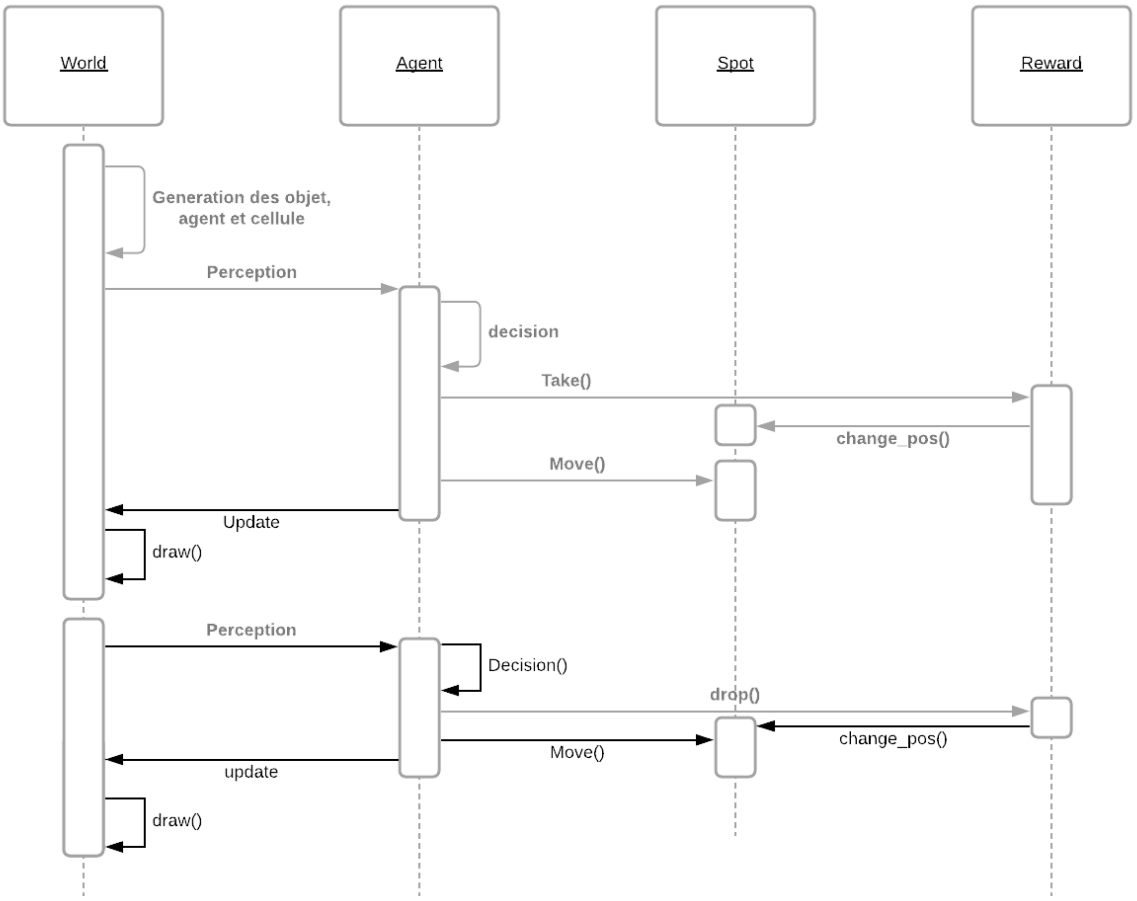


fig 5 : diagramme de conséquence "2 itérations de simulation pour un agent"

La simulation commence par initialisation du monde et GUI par instanciation de class World (la taille de la grille, le nombre d'agents et d'objets, les constantes k+, k-...), puis la génération des entités comme les agents, les objets et les cellules.

Une fois le monde initialisé, la simulation commence par récupérer des informations pour chaque agent dans les 8 directions qui les entourent. Le calcul des possibilités d'agent se fait par interrogation de 8 cellules, on leur demande si elles sont vides ou pas. l'agent se base sur ces informations pour alimenter sa mémoire et calcule des probabilités de dépôt et prise. On note que si un agent porte un objet sa couleur change. L'agent se déplace et sa position sur la grille change. Le choix de déplacement vers une cellule se fait d'une manière aléatoire. Le monde passe par ces étapes pour tous les agents a chaque itération afin de mettre à jour la GUI.

NB : la simulation peut être mise en pause en cliquant sur la touche ESPACE.

IV. Résultats d'exécution

Pour l'exécution, nous avons pris comme exemple :

Un environnement sous forme de grid de Taille = (50,50) qui contient :

20 agents : représenté par des carrés verre(vide) rouge(porte un objet)
400 objets : 200 de type A de couleur VIOLETTE et 200 type B de couleur ROUGE représenté par des petites cercles.

Chaque agent se déplace aléatoirement dans l'environnement et il prend ou dépose l'objet en se basant sur ces probabilités suivantes :

$$P_{prise} = (k^+ / (k^+ + f))^2 \qquad P_{dépôt} = (f/(k^- + f))^2$$

Avec : k+ et k- des constantes et f représentant la proportion d'objet de même type A ou B dans l'environnement immédiat.

Les captures d'écran ci-dessous sont prise respectivement après : 357, 11718, 100783 et 423897 itérations (10 s, 30 minutes, 4h25 minutes et 9h30).

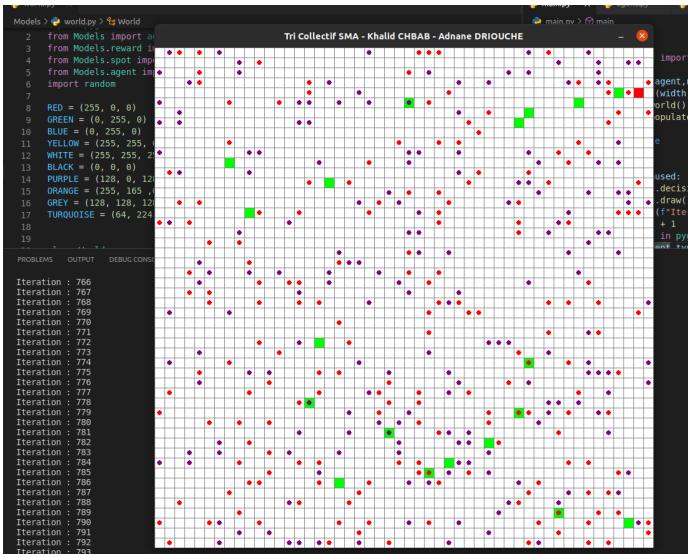


fig 6 : une minutes après l'exécution

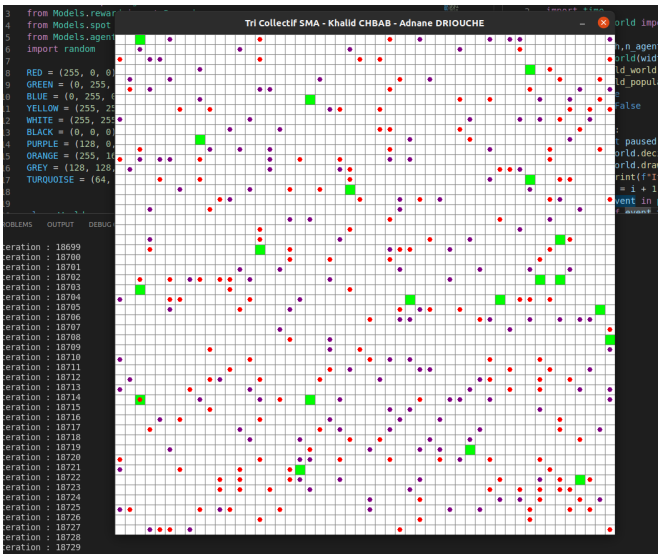


fig 7 : 15 minutes après l'exécution

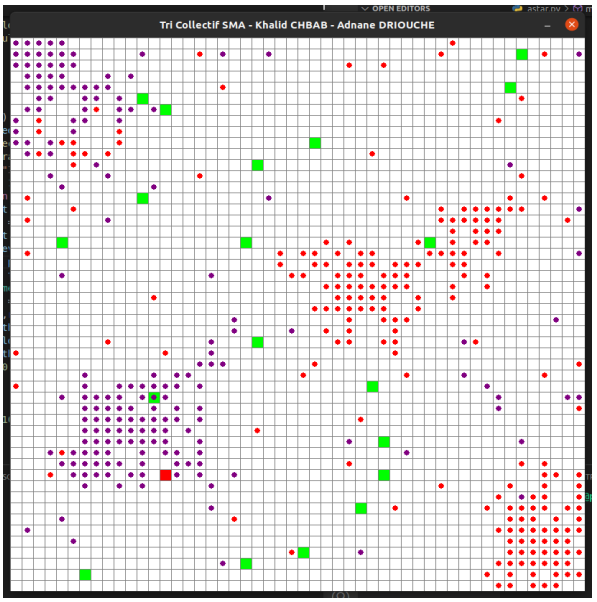


fig 8 : 255 minutes après l'exécution

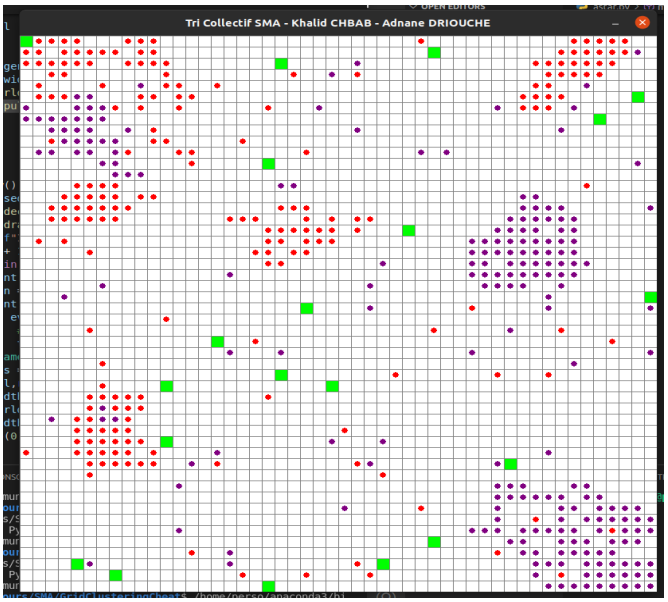


fig 9 : 570 minutes après deuxième exécution

Étant donné que les agents n'ont aucune intelligence sur l'endroit où mettre ou prendre des objets, la convergence vers un monde semi-cluster ou un monde clusterisé peut prendre beaucoup de temps. il sera intéressant de voir si le changement de variables comme la capacité de mémoire ou le mécanisme de l'environnement peut améliorer la simulation.