GILLIARD Tallulah

Licence 3 MIASHS

Décembre 2015

Programmation

Projet Oscar, langage python

1- Structure globale du code

a) ordre de procédure (en se basant sur le « main »)

Création du World : chargement du fichier, initialisation graphique, affichage

Création d'Oscar : définition des méthodes d'évolution du monde

Boucle de tour de jeu et d'affichage

b) ordre linéaire

- Initialisation
  - Définition de la classe world
    `def__init__(self)`
  - Chargement de la fenêtre pygame
    `def init_fenetre(self)`
    `def affiche_fenetre(self)`
  - Chargement du fichier et Initialisation des classes agents
    `def charge_world(self,fichier)`
    boucle "tant que" fichier non terminé, charge `world, mineral, vegetal, animal, agent`
  - Référencement des agents dans des dictionnaires selon leurs types
    `class Mineral, class Vegetal, class Animal, class Agent`
- Noyau
  `class Oscar`
  - Définition de la classe
    `def__init__(self,world)`
  - Mise à jour de la variable principale
    `def__modifieVar__(self,agent,type_agent)`
  - Gestion de la mort d'un agent
    `def__mortAgent__(self, agent, type_agent)`
  - Gestion du changement d'état d'un agent
    `def__nouvelEtat__(self, agent, type_agent)`
  - Vérification du changement de status en fonction de la variable principale
    `def__testeStatus__(self, agent, type_agent)`
  - Gestion de la naissance
    `def__naissanceAgent__(self, agent, type_agent)`
  - Vérification de la naissance
    `def__testeNaissance__(self, agent, type_agent)`
  - Gestion de la trace
    `def__gereTrace__(self, agent)`
  - Gestion du champs
    `def__initField__(self)`
    `def__clearField__(self)`
    `def__testeField__(self, agent)`
    `def__meilleurSpot__(self, agent)` utile pour la naissance et le déplacement
  - Gestion des déplacements
    `def__deplaceAgent__(self, agent)`
  - Gestion d'un tour de jeu
    `def__tourSuivant__(self, agent, type_agent)`
- Main
  Création du monde et initialisations
  Création des méthodes du jeu
  Boucle de gestion des tours de jeu et d'affichage

2- Bilan chronologique

Voici les grandes étapes de création de ce programme :

Lecture d'un fichier (isoler les mots, classer les informations, répertorier selon des catégories)

Affichage d'un terrain (d'abord vide, sans agents puis avec des agents positionnés sur la grille)

Création de l'évolution temporelle

Gestion des modifications sur les variables :

- Statut
- Déplacement
- Naissance

Importante modification de structure des variables (* voir partie 3b)

Gestion des modifications sur les variables :

- Trace
- Champs

3- Problèmes rencontrés et résolus, choix d'organisation structurels et algorithmiques

a) Difficultés rencontrées (et résolues)

- Compréhension et interprétation de l'énoncé
- Moyens de lecture d'un fichier en python
- Stockage des variables dans des dictionnaires : dictionnaires de dictionnaires, listes de dictionnaires (* plus voir partie 3b)
  Ci-dessous, extraits d'organisation du dictionnaire et infos importantes (le type, le field et la trace) pour le fichier « world2.txt »

```
world.liste_type_agents[].variables=
{'mineral': {'statusName': 'pierre', 'statusIcon': 'pierre.png'},
 'var': {'variable': 0, 'stepValue': 20, 'initValue': 0},
 'status': {'signe': '>', 'variable': 0, 'newStatus': 'caillou', 'thresholdValue': 200},
 'field': {'variable': '1', 'decreaseValue': -2},
 'sensor': {'variable': 'variable', 'sensitivityValue': 10}}

{'mineral': {'statusName': 'caillou', 'statusIcon': 'caillou.png'},
 'var': {'variable': 0, 'stepValue': 30, 'initValue': 0},
 'status': {'signe': '>', 'variable': 0, 'newStatus': 'DEATH', 'thresholdValue': 300},
 'field': {'variable': '1', 'decreaseValue': -1},
 'sensor': {'variable': 'variable', 'sensitivityValue': 10}}

{'vegetal': {'statusName': 'arbrisseau', 'statusIcon': 'arbrisseau.png'},
 'var': {'variable': 0, 'stepValue': 30, 'initValue': 0},
 'status': {'signe': '>', 'variable': 0, 'newStatus': 'arbre', 'thresholdValue': 200},
 'field': {'variable': '1', 'decreaseValue': -1},
 'sensor': {'variable': 'variable', 'sensitivityValue': 20},
 'birth': {'signe': '>', 'variable': 0, 'newbornStatus': 'IMPOSSIBLE', 'thresholdValue': 100}}

{'vegetal': {'statusName': 'arbre', 'statusIcon': 'arbre.png'},
 'var': {'variable': 0, 'stepValue': 20, 'initValue': 0},
 'status': {'signe': '>', 'variable': 0, 'newStatus': 'tronc', 'thresholdValue': 200},
 'field': {'variable': '2', 'decreaseValue': -1},
 'sensor': {'variable': 'variable', 'sensitivityValue': 20},
 'birth': {'signe': '>', 'variable': 0, 'newbornStatus': 'arbrisseau', 'thresholdValue': 100}}

{'vegetal': {'statusName': 'tronc', 'statusIcon': 'tronc.png'},
 'var': {'variable': 0, 'stepValue': 60, 'initValue': 0},
 'status': {'signe': '>', 'variable': 0, 'newStatus': 'DEATH', 'thresholdValue': 500},
 'field': {'variable': '1', 'decreaseValue': -1},
 'sensor': {'variable': 'variable', 'sensitivityValue': 20},
 'birth': {'signe': '>', 'variable': 0, 'newbornStatus': 'IMPOSSIBLE', 'thresholdValue': 100}}

{'animal': {'statusName': 'poussin', 'statusIcon': 'poussin.png'},
 'var': {'variable': 0, 'stepValue': 25, 'initValue': 0},
 'status': {'signe': '>', 'variable': 0, 'newStatus': 'poule', 'thresholdValue': 200},
 'field': {'variable': '1', 'decreaseValue': -1},
 'sensor': {'variable': 'variable', 'sensitivityValue': 20},
 'birth': {'signe': '<', 'variable': 0, 'newbornStatus': 'IMPOSSIBLE', 'thresholdValue': -1},
 'trace': {'signe': '>', 'variable': 'variable', 'newtraceStatus': 'newtraceStatus', 'thresholdValue': 3}}
```

```
world.liste_agents[].variables=
{'agent': {'yPosition': '1', 'xPosition': '0', 'initStatus': 'pierre'}}
{'agent': {'yPosition': '4', 'xPosition': '3', 'initStatus': 'pierre'}}
{'agent': {'yPosition': '2', 'xPosition': '7', 'initStatus': 'pierre'}}
{'agent': {'yPosition': '7', 'xPosition': '4', 'initStatus': 'pierre'}}
{'agent': {'yPosition': '8', 'xPosition': '2', 'initStatus': 'pierre'}}
{'agent': {'yPosition': '4', 'xPosition': '8', 'initStatus': 'arbrisseau'}}
{'agent': {'yPosition': '5', 'xPosition': '1', 'initStatus': 'arbrisseau'}}
{'agent': {'yPosition': '6', 'xPosition': '6', 'initStatus': 'arbrisseau'}}
{'agent': {'yPosition': '8', 'xPosition': '8', 'initStatus': 'arbrisseau'}}
{'agent': {'yPosition': '0', 'xPosition': '5', 'initStatus': 'poussin'}}
```

```
0 pierre.png
1 caillou.png
2 arbrisseau.png
3 arbre.png
4 tronc.png
5 poussin.png
6 poule.png
```

```
Oscar.field = [
    [-2, -2,  0,  0,  0,  0,  0,  0,  0,  0],
    [-2, -2,  0,  0,  0,  0, -2, -2, -2,  0],
    [-2, -2,  0,  0,  0,  0, -2, -2, -2,  0],
    [ 0,  0, -2, -2, -2,  0, -2, -3, -3, -1],
    [-1, -1, -3, -2, -2,  0,  0, -1, -1, -1],
    [-1, -1, -3, -2, -2, -1, -1, -2, -1, -1],
    [-1, -1, -1, -2, -2, -3, -1, -1,  0,  0],
    [ 0, -2, -2, -4, -2, -3, -1, -2, -1, -1],
    [ 0, -2, -2, -4, -2, -2,  0, -1, -1, -1],
    [ 0, -2, -2, -2,  0,  0,  0, -1, -1, -1]]

Animal.traces = [(5, 1), (4, 0), (5, 0)]
```

- Mise à jour des variables (avec la copie « deep copy ») : organisation cohérente des dictionnaires et des classes afin qu'il n'y ait pas de confusion entre les mises à jours de variables (éviter une synchronisation générale), structuration de la classe « agent » et des 3 types « minéraux », « végétaux », « animaux »

b) Quelques précisions sur les choix algorithmiques et structurels

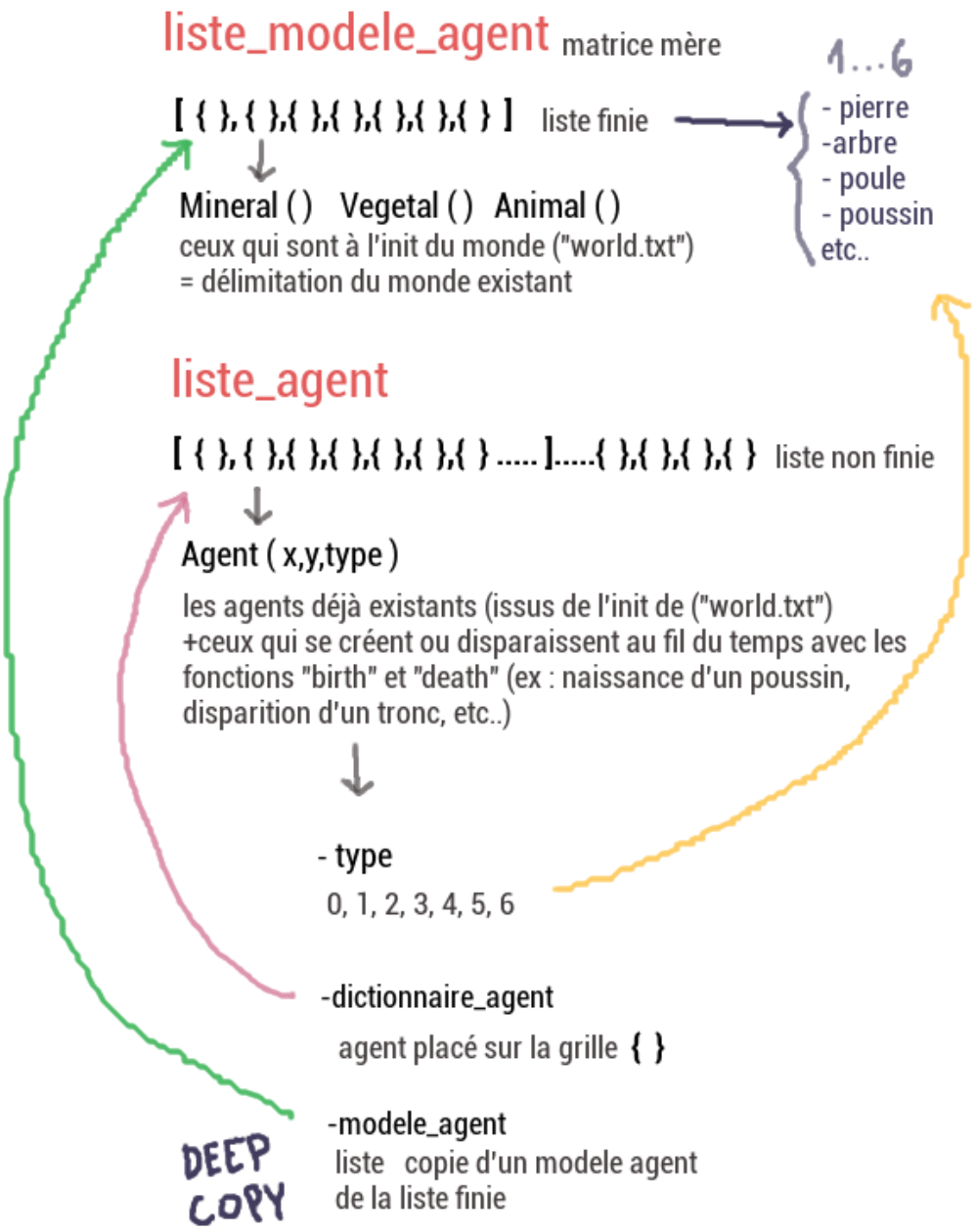
Choix de ce sujet « Oscar » plutôt qu'un autre car il semblait plus intéressant que les autres

Choix de programmer l'interface graphique en Pygame plutôt qu'en Tkinter

Deux agents peuvent se trouver sur la même case

* Organisation en dictionnaires

(Ci-dessous, modèle général d'organisation du programme pour le stockage et l'utilisation des variables)



4- Continuité du projet

- Gérer le « sensor »
- Approfondir la gestion de la trace (changement d'état des traces)
- Approfondir la gestion du champs
- Création de propriétés de rencontre entre les agents : orienté système de proies-prédateurs (ex : des renards qui tuent les poules, les poules tuent des vipères, les vipères tuent les renards) pour la création d'un système plus équilibré basé sur le principe de chaine alimentaire et d'écosystème.
- Ajout d'agents (plus de types d'agent dans chaque classe)