Manuel d'utilisation
et
Guide pas à pas

Licence informatique 2ème année (2018/2019)
Sorbonne Université Sciences (UFR922)
Module 2I013 - Projet Écosystème

Binôme : Thibault ROCHE & Célia RESTES
Professeurs : Nicolas BREDECHE (& Paul ECOFFET)

Composantes principales du projet de simulation d'écosystème

- Un écosystème composé de quatre biomes aux propriétés faune/flore différentes.
- Une représentation graphique en 2D JAVA avec de nombreuses images.
- Un système d'agents *Firefox* & *Iceweasel* interagissant avec la flore.
- Un système d'analyse des données par graphes dynamiques en python.

Fichiers

Dossier Projet-ROCHE_RESTES à récupérer par clone sur github ainsi :

~$ git clone https://github.com/Exo2357/Projet-ROCHE_RESTES.git

Contenu du dossier Projet-ROCHE_RESTES :

AIDE.TXT

Manuel-ROCHE_RESTES.pdf

Scrum-ROCHE_RESTES.pdf

Programme-ROCHE_RESTES

Tables des matières

Guide d’installation et d’utilisation rapide

1) Prérequis d’installations

L’utilisateur s’assure de posséder les prérequis suivants :
(Linux) + (JAVA) + (Python2 OR Python3) + (librairies matplotlib 2.2.2 minimum)
Une version détaillée des prérequis est en fin de manuel.
Important : les PC en salles machines ne possèdent pas la version 2.2.2 de matplotlib

2) Téléchargement depuis github

L’utilisateur récupère le dossier Projet-ROCHE_RESTES par commande terminal depuis le dossier de son choix :

~$ git clone https://github.com/Exo2357/Projet-ROCHE_RESTES.git

3) Ouverture d’un terminal

L’utilisateur ouvre un terminal dans le dossier Programme-ROCHE_RESTES

3.bis) Compilation, étape optionnelle :

La compilation est optionnelle et se lance automatiquement lors du lancement de la simulation.
Mais il est tout de même possible de compiler en entrant dans le terminal :
~/Programme-ROCHE_RESTES$ make all

4) Lancement rapide du programme

L’utilisateur peut opter pour une utilisation avec les paramètres par défaut.

Depuis le terminal ouvert sur *~/Projet-ROCHE_RESTES/Programme-ROCHE_RESTES*

$ make runx Lance le programme (une fenêtre)

Attendre quelques secondes le lancement de la simulation puis :

$ make show Lance le programme d'analyse des données (deux fenêtres)

Le programme est ainsi composé d’une fenêtre principale : la simulation. Et de deux fenêtres optionnelles d’analyse graphique des données de la simulation.

4.bis) Utilisation avancée, paramétrable

Voir page sur le paramétrage pour utilisation avancée, permettant de paramétrer le monde avant de le lancer.

Scénario d’une utilisation pas à pas

Important : si à tout moment le scénario ne fonctionne pas comme prévu, voir section *Dépannage d’installation et d’utilisation*.

1) Lancement de la simulation :

Ouvrir un terminal sur : *~/Projet-ROCHE_RESTES/Programme-ROCHE_RESTES*
Entrez : ~$ **make runx**
Une fenêtre s’ouvre et la simulation commence.

2) Lancer l’analyse des données :

La fenêtre de la simulation peut être déplacée en cours d’utilisation.

Entrez la commande dans le terminal telle que :

~/Projet-ROCHE_RESTES/Programme-ROCHE_RESTES$ **make show**

Cela fait apparaître deux fenêtres dans lesquelles apparaissent :

- les donnée statiques de la distribution (en pourcentage de surface) des biomes,
- les données dynamiques de la faune et de la flore à travers 8 graphes.

4) Fin de la simulation

La simulation s’arrête :

- lorsque le nombre d’itérations est atteint ; paramétrable, voir page sur utilisation avancée.
- à tout moment par fermeture de la fenêtre [x] ou via alt+F4.

Les fenêtres de données se ferment :

- à tout moment par fermeture de la fenêtre [x] ou via alt+F4.

Liste des fonctionnalités

On distingue les fonctionnalités dans plusieurs catégories :
- Système : Makefile
- Programme Java : écosystème
- Programme Java : agents
- Programme Python : analyse des données

I. Système : Makefile

Les fichiers Makefile gèrent la compilation et l'exécution des programmes.
Le Makefile est situé dans le dossier *~/Projet-ROCHE_RESTES/Programme-ROCHE_RESTES*

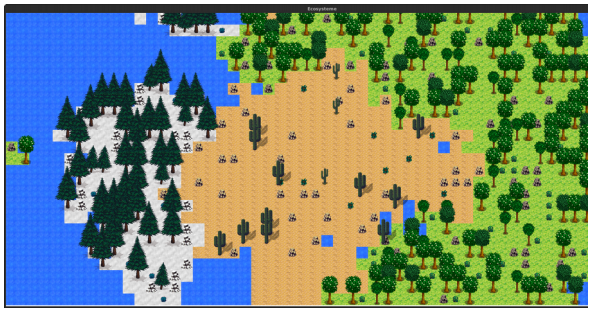
Liste des commandes disponibles est :

$ make all	compilation des fichiers objets
$ make run	compilation puis lancement de la simulation
$ make runx	compilation puis lancement de la simulation dans un autre terminal
$ make show	lancement de l'analyse de données
$ make clean	effacement des fichiers objets et des données d'analyse

II. Programme Java : écosystème

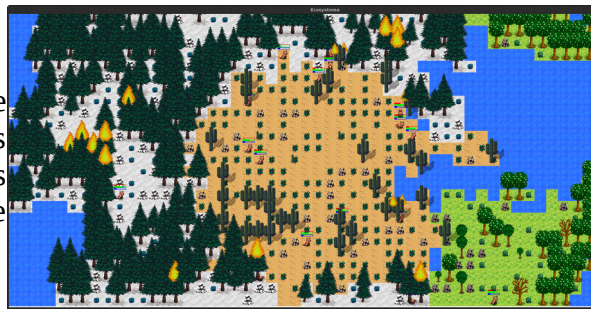
Génération de biomes

Le monde est torique et génère des biomes unifiés de formes aléatoires. Les biomes sont crées en partant d'une cellule qui se répand. Le code est conçu pour rendre accessible l'ajout d'autres biomes. La version proposée contient quatre biomes : océan, tempéré, désertique et montagneux.
Chaque biome (hormis l'océan) ont une taille minimale. La variabilité du coté aléatoire de la distribution des biomes dépend du paramètre : *BIOME_EQUALITY_RATE dans MainWorld.java*



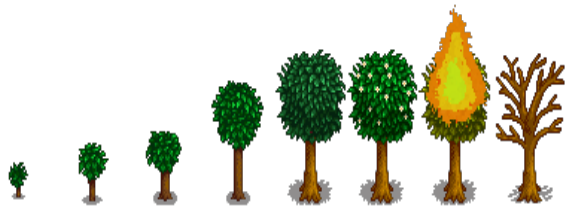
Feu de forêts

Le feu se propage d'arbre en arbre en voisinage de Von Neumann. Les arbres sont mis en feu par les Firefox et le feu peut être éteint par les Iceweasel. Les arbres brûlent sur plusieurs cycles et laissent un arbre mort en cendre.



Liste des fonctionnalités *(suite)*

Cycle de vie des arbres
Les arbres grandissent selon un cycle de vie de jeune pousse à arbre fleuri.
Lorsqu'ils sont en feu, ils brûlent sur plusieurs cycles puis sont des arbres morts, et disparaissent.
Lorsque le feu est éteint, l'arbre retourne au stade d'arbre non fleuri.



Cycle de vie des arbres

III. Programme Java : agents

Firefox
Les Firefox naissent dans le désert. Petits, ils restent dans leur territoire, où ils peuvent manger et dormir. A l'âge adulte, ils défendent leur territoire de leur Némésis les Iceweasel et peuvent s'aventurer dans les autres biomes et déclencher des feux de forêts.

Iceweasel
Les Iceweasel naissent dans le froid du biome montagneux. Petits, ils restent dans leur territoire, où ils peuvent manger et dormir. A l'âge adulte, ils défendent leur territoire de leur Némésis les Firefox et peuvent s'aventurer dans les autres biomes et éteindre des arbres en feu.

Recherche de chemin
Les agents utilisent la méthode de recherche ***A**** pour leurs objectifs de recherche : nourriture, proie pour la poursuite, arbre à brûler ou à éteindre et enfin endroit par défaut.

Agent qui dort

Agent qui mange

Agents qui s'accouplent

Une barre de vie
Une barre rouge au-dessus de l'agent indique ses points de vie. Celle-ci diminue si :
- l'agent est affamé et ne trouve toujours pas de buissons pour se nourrir.
- l'agent est affamé et ne peut manger (par exemple : il dort)
- l'agent est fatigué et ne peut dormir (par exemple : il mange)
- l'agent est coincé.

L'agent meurt lorsqu'il n'a plus de points de vie ou lorsque son prédateur a réussi à l'atteindre.

Comportements réalistes : Manger, dormir, se reproduire : le cycle de la vie

Faim

La barre de faim figure en vert au-dessus de leur tête. Arrivés sous un seuil de faim défini, les agents font une recherche de chemin A* vers le buisson le plus proche d'eux dans leur champ de vision. Une fois le buisson atteint, ils s'arrêtent et entrent dans l'action de manger pendant un nombre d'itérations paramétrable.

Énergie

La barre d'énergie figure en bleu au-dessus de leur tête. Arrivés sous un seuil d'énergie défini, les agents s'arrêtent et entrent dans l'action de dormir pendant un nombre d'itérations paramétrable.

Reproduction

Les agents sont sexués. Les femelles sont aptes à la reproduction lorsque la barre rose, figurant au-dessus d'elles, est à 0. Ainsi, si dans son voisinage (Von Neumann) il y a exactement un agent mâle de la même espèce, il y a une probabilité X paramétrable qu'ils se reproduisent.
Si elle entre en période de gestation, elle ne peut sortir de son biome d'origine qu'après avoir donné naissance au nouvel agent.
A noter que la reproduction n'est possible que dans le biome où les bébés peuvent vivre. De plus, en période de gestation, les femelles ont le même comportement que leurs semblables bébés : elles ne peuvent mettre feu aux arbres ou les éteindre et ne peuvent pas chasser les ennemis de leur territoire.

Système de proie et prédateur

Le système de proie et prédateur dépend du biome. Si un Iceweasel se retrouve dans le territoire désertique, il peut être pourchassé par un ou des Firefox si ceux-ci les repère dans leur champ de vision. Et vice-versa.
Pour atteindre sa cible, le prédateur fait une recherche A* vers elle et la poursuit si il trouve un chemin vers elle. Celle-ci peut s'échapper des mains de son assaillant en se rendant dans un autre biome ou lorsque son agresseur est fatigué et s'endort ou est affamé et mange, lui donnant ainsi le temps de s'éloigner.

Arbre de décisions

Les agents agissent en fonction de leurs priorités.

Est-ce que l'agent est fatigué ?
Oui => il s'endort
Non => Est-ce qu'il a faim ?
    Oui => il mange
    Non=> Est-ce qu'il a trouvé une proie ?
        Oui => la poursuit pour le tuer
        Non => Est-ce qu'il a trouvé un arbre (arbre en feu) ?
            Oui => le brûle si Firefox (l'éteint si Iceweasel)
            Non => déplacement aléatoire dans son champ de vision

Fluidité des mouvements et animations

Les agents ont plusieurs animations. Une pour les déplacements Nord, Sud , Ouest et Est, une pour son action de manger et une dernière pour son action de dormir.

La réalisation d'un spritesheet sur GIMP est préférée pour plus de clarté. Ainsi à partir du spritesheet, on peut extraire les images souhaitées à l'aide de fonctions spécifiques au vue de la réalisation des animations.

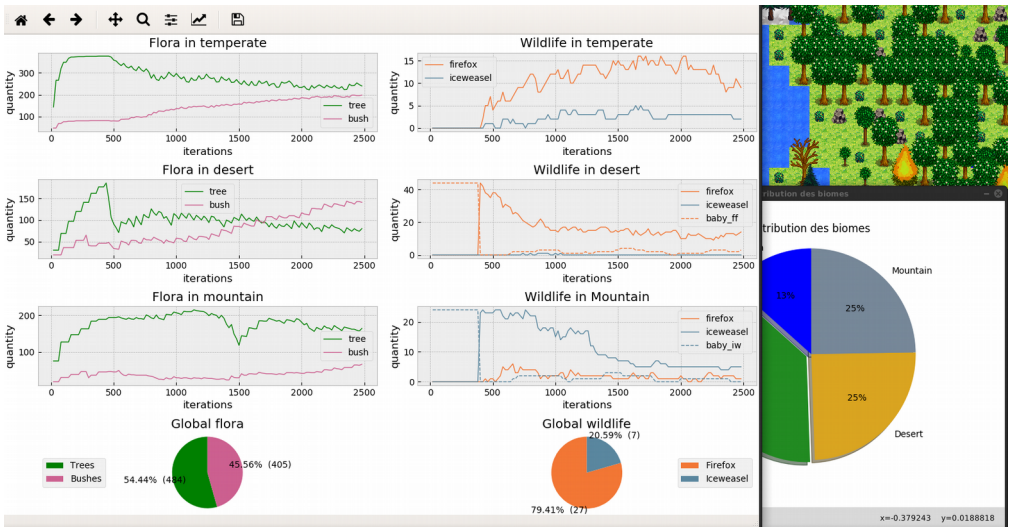
L'agent possède un état interne qui est l'orientation, mais également une animation. Pour le déplacement, son orientation définit son animation.

Chaque animation de déplacement se décompose en 4 images. A chaque itération, l'image courante passe à la suivante et à l'itération X, si l'agent change d'orientation en fonction de son futur déplacement, son animation se retrouve aussi changée.
Les deux autres animations, manger et dormir, sont composées de 2 images. Jusqu'à la fin de l'action de manger ou dormir, les images se succèdent et rebouclent.

IV. Programme Python : analyse des données

La commande $make show lance deux programmes Python qui affichent les données de la simulation sous forme de graphiques :

distribution des biomes présente par *pie chart* la distribution des biomes en pourcentage.
analyse faune & flore présente par 8 graphes la faune et la flore de manière dynamique.



Paramétrage pour utilisation avancée

La simulation peut être paramétrée en modifiant les paramètres en lettres capitales, situés en tête des fichiers suivants :

Les options affectant les performances de la simulation sont indiquées en rouge.
Les options utiles au debug sont indiquées en italique.

MainWorld.java

DX, DY	**Int**	**Taille du monde**
ITERATIONS	Int	Nombre de cycles avant l'arrêt de la simulation
DELAY	**Int**	**Délai entre chaque cycle**
FREQ_AGENT_ACTION	Int	Fréquence d'animation des actions
FREQ_AGENT_ANIM	Int	Fréquence de changement d'animation de déplacement
FREQ_DATA	Int	Fréquence par cycle de récupération des données
FREQ_WORLD	Int	Fréquence par cycle de la MAJ de la flore
SHOW_MATRICE	*Bool*	*Debug*

World.java

SPRITE_LENGTH	**16/32**	**Taille des sprites**
BIOME_SIZE_EQUALITY_RATE	Float	Variation de la taille des biomes
P_REPRODUCTION	Float	Probabilité de reproduction des agents
P_FIREFOX	Float	Probabilité de création initiale d'un Firefox
P_ICEWEASEL	Float	Probabilité de création initiale d'un Iceweasel

Agents.java

MAX_HEALTH	Vie de l'agent
MAX_HUNGER	Satiété maximale
HUNGRY	Commence à avoir faim
HUNGER_DMG	Dégâts reçus si l'attribut hunger ≤ 0
FREQ_DMG_HUNGER	Toutes les FREQ_DMG_HUNGER l'agent s'affame
EATING_DURATION	Nombre de cycles requis pour manger
MAX_ENERGY	Énergie liée au sommeil
LOW_ENERGY	Commence à vouloir dormir
SLEEP_DMG	Dégâts reçus si l'attribut energy ≤ 0
FREQ_DMG_SLEEP	Toutes les FREQ_DMG_SLEEP l'agent se fatigue
SLEEPING_DURATION	Nombre de cycles requis pour un sommeil complet
ADULT_AGE	Age en cycles requis avant d'être adulte
PREGNANCY_DURATION	Durée de la gestation
P_REPRODUCTION	Probabilité de reproduction
VISION	Champ de vision (carré) de l'agent

BiomeTemperate.java
BiomeDesert.java
BiomeMountain.java

TREE_DENSITY	Densité des arbres
BUSH_DENSITY	Densité des buissons
ROCK_DENSITY	Densité des rochers
TREE_REGROWTH_RATE	Multiplicateur de vitesse de repousse des arbres
BUSH_REGROWTH_RATE	Multiplicateur de vitesse de repousse des buissons

Prérequis détaillé d’installation

– Un environnement LINUX

Un émulateur de terminal parmi la liste suivante :
https://packages.debian.org/fr/jessie/x-terminal-emulator

– JAVA

– Python ou Python3

Exemple avec conda : ~$ conda update python

– **Package de librairies matplotlib 2.2.2 minimum**

Instructions d’installation de matplotlib sur le site officiel :
https://matplotlib.org/users/installing.html
Exemple avec conda : ~$ conda update matplotlib
Vérification de version avec :

```
~$ python
>>> import matplotlib
>>> matplotlib.__version__
```

Dépannage d’installation et d’utilisation

1) Si les commandes *runx* et *show* ne fonctionnent pas :

Depuis un terminal : ~/Programme- ROCHE_RESTES
$ make run Lance le programme dans le terminal actuel (une fenêtre)

puis, depuis un terminal : ~/Programme-ROCHE_RESTES/**data**
Lancer l'analyse des données statiques :
$ python plot_distrib.py &
Lancer l'analyse des données dynamiques :
$ python plot_data.py &

2) Si une erreur est relative au graphes, vérifier la version de matplotlib :

```
$ python
>>> import matplotlib
>>> matplotlib.__version__
```

Dépendances mineures (utilisées dans ce programme):

matplotlib.pyplot
matplotlib.gridspec
matplotlib.animation
matplotlib.style
python-csv

Crédits et droits d’auteurs

Agent Firefox *est une allusion à*
Mozilla Firefox est un navigateur web libre et gratuit, développé et distribué par la Mozilla Foundation avec l'aide de milliers de bénévoles grâce aux méthodes de développement du logiciel libre/open source et à la liberté du code source.

Agent Iceweasel *est une allusion à*
Iceweasel est la version renommée par Debian de Mozilla Firefox.

Algorithme A*
https://github.com/tiagodopke/java-a-star

Sprites

Firefox et Iceweasel
Mods de Stardew Valley
https://community.playstarbound.com/threads/tokiris-pet-replacements-fox-wolf-red-panda-update-red-panda-finished.110263/

Emote cœur & Buissons & Arbres du biome tempéré
Stardew Valley https://www.spriters-resource.com/pc_computer/stardewvalley/

Sapins du biome montagne
"[LPC] Conifer Trees Pack" by bluecarrot16, b_o, Lanea Zimmerman (Sharm), Johann Charlot, Yar, Jetrel, Zabin, Hyptosis, Surt, and KnoblePersona

Cactus du biome désert & Sable & Herbe
"[LPC] Beach/Desert" by bluecarrot16, Guillaume Lecollinet, Johann Charlot, cynicmusic, Surt, vk, Yar, Buch, Jetrel, Zachariah Husiar aka (Zabin), Hyptosis, William Thompson, Frode Lindeijer aka (Modanung), Leonard Pabin, and Lanea Zimmerman aka (Sharm).

Neige
Downdate
https://opengameart.org/content/snow-texture

Eau
Jattenalle
https://opengameart.org/content/texture-water

Rochers
RayaneFLX
https://opengameart.org/content/rpg-terrains