Projet Informatique – Sections Electricité et Microtechnique

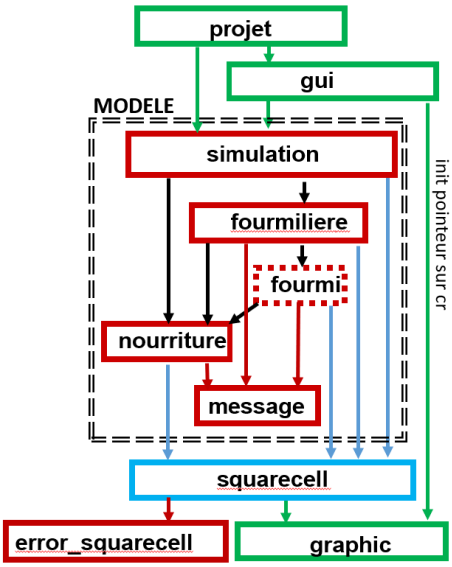
Printemps 2022 :Tchanz © R. Boulic

Rendu3 (dimanche 22 mai)

Objectif de ce document : *comme pour le document des rendus précédents, ce document identifie des ACTIONS à considérer pour réaliser le rendu de manière rigoureuse. Ces ACTIONS ne sont pas notées, elles servent à vous organiser. Vous pouvez adopter une approche différente du moment que vous respectez l'architecture minimale du projet (donnée Fig 6c). A nouveau on s'appuiera sur la série sur les méthodes de développement de projet.*

1. Buts du rendu3 : dialogue avec l'interface graphique et simulation

Votre approche peut évoluer entre ce que vous avez décrit pour le rendu2 en matière de structuration des données et ce que vous mettez en œuvre finalement du moment que vous respectez les responsabilités des différents modules (Fig ci-contre).



Donnée Fig 9b

Lancement et comportement attendu:

Le programme sera lancé soit en indiquant un nom de fichier à ouvrir (rendu2) :

```
./projet test1.txt
```

Soit sans aucun argument sur la ligne de commande :

```
./projet
```

Dans ce second cas, l'interface est crée et le programme attend qu'on lui demande d'ouvrir un fichier avec le bouton Open.

1.1 Rapport final (MAX 2 pages):

Le Rapport ne contient PAS de page de titre, ni de table des matières. Il est écrit avec une police 11 au minimum et 14 au maximum, interligne simple. Le rapport est écrit en français ou en anglais ; une orthographe ou une grammaire défaillante peut induire les correcteurs en erreur. Le Rapport ne duplique pas le précédent. Il contient :

- **la description de votre approche** pour les déplacements non -précisés de la donnée générale.
  - Generator : conserve-t-elle sa position initiale ?
  - Lieu de naissance des fourmis : comment est-il choisi ?
  - Collector :
    - quel chemin est choisi en cas de 2 chemins équivalents ?
    - que fait une fourmi Collector sans nourriture « cible » ?
  - Defensor : comment son but est-il mis à jour ?
  - Predator : que fait une fourmi Predator sans fourmi Collector « cible » ?

- **Captures d'écran (zoom ok) de plusieurs étapes de votre simulation** pour un fichier de test fourni (à préciser) et pour un autre fichier de votre choix (avec au moins une entité de chaque type). La légende doit expliquer les mouvements d'une capture d'écran à la suivante. Indiquer de nombre de mises à jour entre chaque capture.

- **Méthodologie et conclusion** : comment avez-vous organisé votre travail à plusieurs, indiquer la repartition des contributions par module et comment vous avez organisé le travail au sein du groupe (par quels modules avez-vous commencé, comment les avez-vous testés). Indiquer la proportion de travail simultané en groupe (c'est à dire côte à côte ou en-ligne sur le même code) par rapport au travail indépendant (chacun de son coté). Avec le recul, est-ce que vous modifieriez cette proportion?

    - Quel était le bug le plus fréquent, pourquoi ? Quel est celui qui vous a posé le plus de problème et comment a-t-il été résolu, ...).

    - Pour conclure fournissez une brève auto-évaluation de votre travail et de l'environnement mis à votre disposition (points forts, points faibles, améliorations possibles).

Le rapport final doit être inclus dans le fichier archive du rendu final (en format pdf).

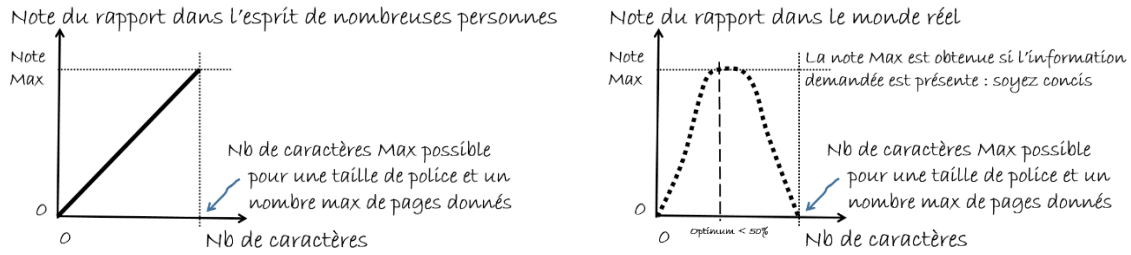


Figure 1 : un mauvais rapport dilue l'information utile jusqu'à atteindre le nombre maximum de caractères possibles sur la page (fig gauche) : en fait ce type de rapport sera pénalisé parce qu'il est peu lisible ; un bon rapport est celui qui fournit les informations demandées avec concision avec une mise en page aérée et lisible (fig droite)

1.2 Evaluation du rendu3 : en plus d'évaluer les critères sur la qualité du code (ne pas oublier de corriger tout warning obtenu aux rendus précédents), nous effectuerons une évaluation manuelle de votre programme comme suit :

- lancement avec les 2 syntaxes : avec et sans nom de fichier sur la ligne de commande.

- évaluation de la mise à jour de la simulation dans différents scénarios pour faire apparaître les comportements demandés. La majorité des scénarios sera constitué des cas très simples discutés dans la section suivante. Seuls quelques scénarios auront une complexité plus grande pour l'évaluation des performances.

- **Affichage** :
- les couleurs ne doivent pas changer en cours de simulation quand une fourmilière disparait.

1.3 Barème (total 25 pts) : Exécution (14pts), Architecture (2pts), Encapsulation/externalisation (2 pts), Style (4pts), Rapport (3pts)

2. Organisation du travail

Nous recommandons de tester votre simulation en mode Step avec l'affichage graphique de chaque état successif de la simulation car c'est un puissant outil de mise au point.

N'hésitez pas à compléter avec du code de scaffolding qui affiche la valeurs des variables importantes dans le terminal pour compléter vos outils de mise au point. D'ailleurs, en cas de bug, l'affichage dans le terminal devrait être redirigé vers un fichier de texte (cf cours redirection de la sortie) pour pouvoir l'ouvrir avec un éditeur de texte après le crash du programme et analyser l'évolution des valeurs de vos variables.

fichier	contexte	But du test: vérifier…
C01.txt	Simulation vide	1) apparition de nourriture avec la probabilité de 0.1 (on devrait voir 4 à 6 éléments de nourriture après 50 clics sur Step) 2) repartition aléatoire dans tout l'espace 3) mise à jour simultanée du compteur de nourriture
f01.txt	End of life de generator	1) L'état initial est visualisé mais la fourmilière disparait dès la première mise à jour car la taille calculée à partir du nombre de fourmis est beaucoup plus petite que la taille indiquée dans le fichier. De ce fait la fourmi generator ne peut pas se déplacer suffisamment pendant la première mise à jour et elle disparait, ce qui fait disparaitre la fourmilière. Le programme doit continuer à s'exécuter (production de nourriture).
f02.txt	End of life de defensor	1) L'état initial est visualisé mais la fourmi defensor disparait dès la première mise à jour car la taille calculée à partir du nombre de fourmis est beaucoup plus petite que la taille indiquée dans le fichier. De ce fait la fourmi defensor ne peut pas se déplacer suffisamment pendant la première mise à jour et elle disparait. Il n'y a pas de problème pour la fourmi Collector.
f03.txt	Test sur Total_food	1) Demander à voir les information de la 1ière fourmiliere pour visualiser la diminution de la réserve de nourriture à chaque step: initiale=1 puis 0.6 puis 0.2 puis disparition de la fourmilière par manque de nourriture.
f04.txt	Pas de nourriture et mettre temporairement food_rate à 0	1) fin de vie des 3 fourmis initialement présente (après 1 step) 2) naissance d'une fourmi à chaque step (grâce à la valeur très importante de total_food), avec les proportions de l'état **FREE** (0.85, 0.1, 0.05), ce qui donne la sequence de nombres de fourmis (C, D, P): (1,1,1), (1,0,0), (1,1,0), (2,1,0), (3,1,0), (4,1,0), (5,1,0),(6,1,0),(6,1,1). 3) La fourmilière s'agrandit avec le nombre de fourmis, dans l'ordre: sizeF=13, 11, 13, 14, 15, 16,… 4) Les fourmis créés trouvent un emplacement valide dans la fourmilière
f05.txt	Pas de nourriture et mettre temporairement food_rate à 0	1) naissance d'une fourmi à chaque step (grâce à la valeur très importante de total_food), avec les proportions de l'état **CONSTRAINED** (0.6, 0.1, 0.3), ce qui donne la sequence de nombres de fourmis (C, D, P): (1,0,0), (1,1,0), (2,1,0), (2,1,1), (3,1,1), (3,1,2). Cependant, selon l'endroit où la fourmi Generator est placée, il n'est pas certain que la fourmi soit créée. Donc la sequence peut rester bloquée à une des étapes indiquées ci-dessus (ça n'est pas pénalisé). 2) Les fourmis créées trouvent un emplacement valide dans la fourmilière, si un tel emplacement n'existe pas, la fourmi n'est pas créée.
f06.txt	Comportement d'une fourmi Collector. Mettre temporairement food_rate à 0	1) L'élément de nourriture le plus proche est atteignable en ligne droite, le Collector doit se diriger vers lui et le capturer en 7 steps et revenir au contact de la fourmilière en 3 steps pour le stockage (la valeur de total_food doit augmenter) 2) Le second élément de nourriture n'est pas sur la même famille de diagonale ; le Collector ne peut pas le récupérer. La fourmi peut rester immobile.
f07.txt	Comportement d'une fourmi Collector. Mettre temporairement food_rate à 0	1) Le seul élément de nourriture (18,115) est atteignable avec un seul changement de direction, en 9 step vers le haut à droite puis 2 vers le haut à gauche. 2) au retour, le Collector fait d'abord 2 step vers le bas à droite puis 5 step vers le bas à gauche (ou l'inverse) suffisent pour qu'il y ait contact des coins.
f08.txt	Comportement d'une fourmi Collector. Mettre temporairement food_rate à 0	1) L'élément de nourriture (18,115) est le seul atteignable mais les autres bloquent les 2 chemins possibles car il ne sont pas de la même famille. Le chemin avec le moins de superposition est choisi (verse en haut à gauche d'abord). 2) Le collector restera bloqué devant le premier élément de nourriture qui fait "obstacle"
f09.txt	Comportement d'un Collector. Mettre temporairement food_rate à 0	1) L'élément de nourriture (23,122) est le seul atteignable mais le second (17,117) bloque le passage car il n'est pas de la même famille. Le Collector fait donc des reflexions contre la bordure pour atteindre (23,122). 2) Même problème au retour, l'élément (17,117) bloque le chemin avec un seul changement de direction ; il fait donc des reflexions contre la bordure.
f10.txt	Collector vs Defensor. Mettre temporairement food_rate à 0	1) Au second step, en chemin vers l'élément de nourriture (28,48) le Collector entre en contact avec un Defensor d'une autre fourmilière, ce qui le détruit.
f11.txt	Collector vs Predator. Mettre temporairement food_rate à 0	1) La fourmilière rouge est en mode constrained. Son predator doit se diriger vers le collector vert le plus proche (distance euclidienne). 2) Si elle reste immobile la fourmi collector est détruite (après 3 à 4 steps) par contact d'un coté ou d'un coin ou par superposition avec le Predator

Pour certains scenarios de test, il est demandé de temporairement mettre à zero certaines constantes pour ne pas perturber l'évolution de l'action en cours: food_rate. NE PAS oublier ensuite de redonner la valeur initiale de food_rate. Ces tests simples seront complétés par quelques autres plus complets pour évaluer la robustesse et les performances du programme (ex: le fichier public avec 25 fourmilières).

3. Forme du rendu

Documentation : l'entête de vos fichiers source doit indiquer le nom du fichier et les noms des membres du groupe avec, **pour les fichiers .cc**, une estimation du pourcentage de contribution de chaque membre du groupe au code de ce fichier.

Rendu : pour chaque rendu **UN SEUL membre d'un groupe** (noté **SCIPER1** ci-dessous) doit téléverser un fichier **zip**[1] sur moodle (pas d'email). Le non-respect de cette consigne sera pénalisé de plusieurs points. Le nom de ce fichier **zip** a la forme :

SCIPER1_ SCIPER2.zip

Compléter le fichier fourni **mysciper.txt** en remplaçant 111111 par le numéro SCIPER de la personne qui télécharge le fichier archive et 222222 par le numéro SCIPER du second membre du groupe.

Le fichier archive du rendu2 doit contenir (**aucun répertoire**) :

- Fichier texte édité **mysciper.txt**
- Votre fichier **Makefile** produisant un executable **projet**
- Tout le code source (.cc et .h) nécessaire pour produire l'exécutable.
- Le fichier pdf du rapport

*On doit obtenir l'exécutable **projet** en lançant la commande* **make** *après décompression du fichier **zip**.*

Auto-vérification : Après avoir téléversé le fichier **zip** de votre rendu sur moodle (upload), récupérez-le (download), décompressez-le et assurez-vous que la commande **make** produit bien l'exécutable et que celui-ci fonctionne correctement.

Exécution sur la VM: votre projet sera évalué sur la VM à distance.

Backup : Il y a un backup automatique sur votre compte myNAS.

Gestion du code au sein d'un groupe :

- vous pouvez envisager d'utiliser **gdrive.epfl.ch** pour définir un répertoire partagé par les 2 membres du groupe et pas plus. Cependant il n'y a pas d'éditeur de code en mode partagé.
- Une approche qui demande un apprentissage supplémentaire serait d'utiliser **github** : cf ce tutorial sur moodle. Attention : il FAUT restreindre l'accès du code aux seuls 2 membres du groupes.

Rappel sur l'outil GDB de recherche de bug :

- Guide utilisateur de GDB avec son code de test ; GDB tutorial in english

[1] Nous exigeons le format zip pour le fichier archive