Ecole Publique d'Ingénieurs en 3 ans

Rapport de projet

SIMULATION D'UNE FOURMILIERE

le 10 décembre 2021

Antonin HUAUT

Quentin FONTAINE

ENSI CAEN

ÉCOLE PUBLIQUE D'INGÉNIEURS
CENTRE DE RECHERCHE

www.ensicaen.fr

Choix de conception

Notre architecture utilise différentes fonctionnalités objets du C++, nous avons un héritage sur les fourmis : Une fourmi abstraite avec des classes enfantes pour chaque type de fourmis. Nous avons aussi utilisé le multi-héritage pour les fourmis ayant un âge : pouvant être enfant/adulte.

Nos classes de rendu graphiques sont séparées des classes de jeu, permettant d'avoir un respect des règles SOLID : une classe a un objectif et non plusieurs. Nous avons implémenté certains paternes comme le Singleton pour la configuration. En effet, la configuration est unique pour toute l'application, elle est chargée au démarrage de l'application et toutes parties de l'application peut y accéder à travers le paterne Singleton.

Voici un diagramme UML simplifié, nous avons retiré certaines fonctions *(getters, setters, fonctions héritées)* :

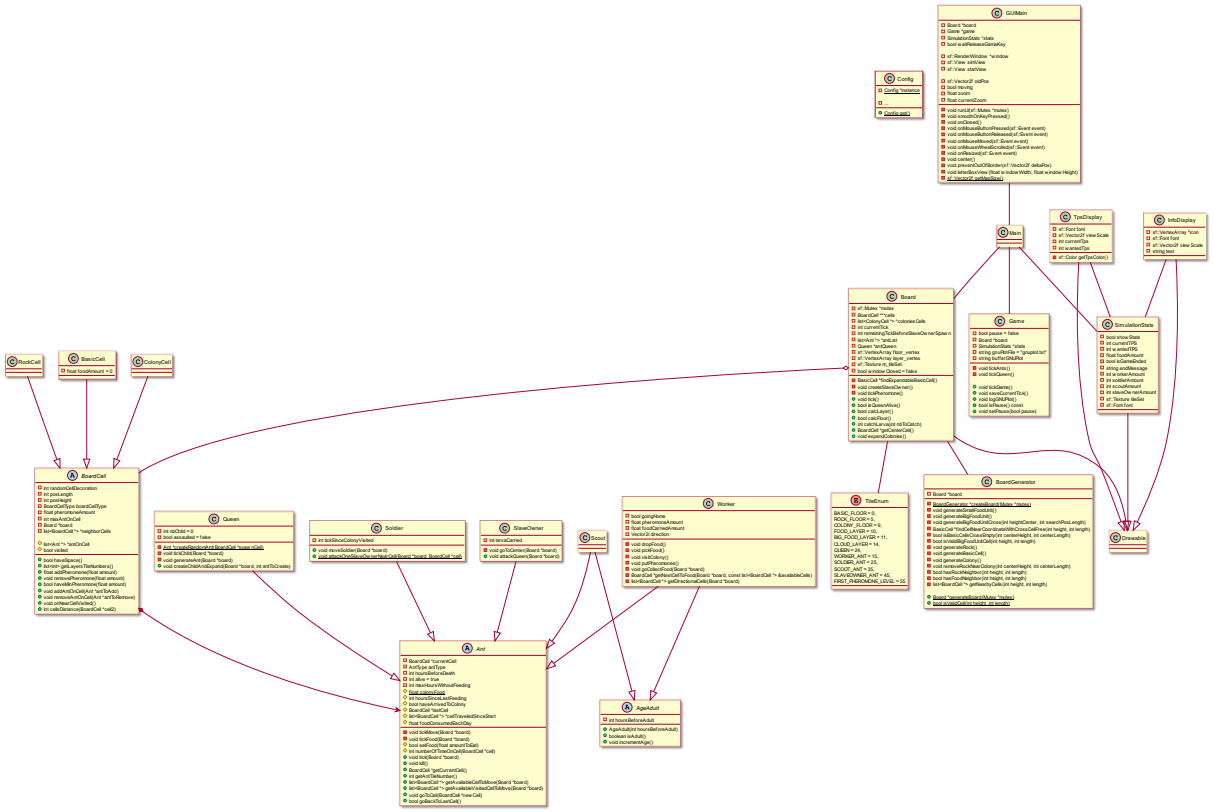

Image 1 : Diagramme de classe réalisé avec PlantUML

L'image du diagramme et le fichier PlantUML associé peuvent être trouvés dans le dossier Doc (pour voir en meilleur qualité l'image).

Problèmes rencontrés

Nous avons rencontré divers problèmes durant la réalisation de ce projet. Outre notre connaissance du langage, car nous le découvrons, nous avons rencontré des problèmes liés à l'optimisation ou encore à l'interface graphique.

Pour résoudre ces problèmes, nous nous sommes aidés de StackOverflow, OpenClassRoom, la documentation SFML ou encore la documentation C++.

Optimisation

Un problème majeur que l'on a rencontré est l'optimisation. En effet, nous avons développé notre interface pour pouvoir jouer le jeu et l'affichage en séparé, permettant de faire beaucoup de tick de jeu. Cependant, nous nous sommes rendu compte que quand la colonie devenait grande, le jeu se ralentissait : on ne pouvait plus atteindre le tick souhaité. Pour régler ce problème, nous avons repris tous nos algorithmes de déplacement de fourmis (car nous avons détecté en profilant que c'était ces fonctions le problème) et, nous avons pour certains optimiser les algorithmes, et pour d'autres changer l'algorithme.

Cela a réglé une majorité des problèmes, cependant, on peut ressentir une baisse de vitesse du jeu si l'on est sur un PC portable qui n'est pas branché sur secteur : le CPU est alors « bridé » et le jeu tourne beaucoup moins vite.

Interface graphique

Un des premiers problèmes concernant l'UI est la gestion du tileset : une image contenant toutes les images de notre jeu et la gestion des vertices : la matrice de points pour placer nos images dans l'interface. C'était une notion nouvelle pour nous que nous ne connaissions pas, nous avons donc utilisé la documentation et l'aide de certains forums pour résoudre ce soucis.

Le second problème de l'UI est la synchronisation du thread de rendu et du thread du jeu. En effet, si pendant le rendu graphique, on accède par exemple à une fourmi qui vient d'être supprimée dans le jeu : ça plante. Le fixe a été de créer un mutex, placé aux endroits « dangereux », qui permet aux threads de s'attendre entre eux. Le point négatif est que cela ralenti le programme car les threads doivent par moment s'attendre.

Le dernier problème a été lors du test de compatibilité avec Ubuntu. Ubuntu utilise le système de fenêtrage X.org (X11) et ce système ne supporte pas le multithreading rendu / gestion des évènements d'interfaces. Cela faisait planter notre programme. Nous avons trouvé une solution mais nous ne l'avons pas implémenté. En effet, la solution aurait ralentie très fortement le programme car le thread de rendu et le thread des évènements devraient s'attendre en permanence. Notre programme n'est donc compatible que pour Windows. Cependant, si une distribution linux n'utilise pas X.org mais un autre système de fenêtrage, elle a de forte probabilité d'être compatible.

Fonctionnalités

Toutes les fonctionnalités du sujet ont été implémentées. De plus, nous avons ajouté quelques fonctionnalités bonus très pratique (ergonomiques ou visuelles). Voici un résumé des fonctionnalités :

Système global :

- Système de configuration (fichier JSON) : toutes les variables du jeu sont configurables (dont les ticks par seconde de jeu, les frames par seconde)
- Fichier GNUPlot contenant l'évolution des données à chaque tick de jeu
- Multi-thread : un thread de jeu, un thread de rendu, un thread de gestion des évènements. (Synchronisation jeu ⇔ rendu via un mutex)

Interface graphique :

- Toutes les images (32x32) dans un tileset
- MinFPS / maxFPS configurable
- Système de couches visuelles
    - Première couche : cases du plateau statiques
    - Seconde couche : nourritures / colonies
    - Troisième couche : fourmis / nuage (les cases non visitées sont affichées avec un nuage)
- Gestion dynamique de la résolution d'affichage (affichage de bandes noirs pour garder un aspect/ratio)
- Impossibilité de sortir de la map (on aperçoit juste les bordures noirs lorsqu'on dézoom)
- Raccourcies :
    - Espace => Toggle la pause du jeu (+ dump gnuplot)
    - C => Recentre sur la colonie
    - Touches "flèches" : Permet de se déplacer dans la map *via le clavier*
    - Clic de souris + slide : Permet de se déplacement dans la map *via la souris*
    - Shift : Accélère le déplacement lors du déplacement dans la map via les touches "flèches"
    - Molette de la souris : Zoom/Dézoom la map
    - S : Affiche ou cache le HUD des statistiques
- HUD des statistiques : affiche le nombre de fourmis en temps réel, la nourriture, le TPS :

Image 2 : Capture d'écran de notre simulation de fourmilière

Types des fourmis par couleur :

- Marron : Eclaireur
- Noir : Ouvrière
- Bleu : Soldat
- Rouge : Esclavagiste

Terrain

- Le terrain mesure 211x201 avec la reine et la colonie en son centre
- Une case ne peut contenir que 12 fourmis (0 si obstacle, 100 pour la colonie)
- Les obstacles sont présents sur 30% du terrain
    - 50% sont d'une case
    - 30% de deux cases
    - 10% de trois cases
    - 5% de quatre cases
    - 4% de cinq cases
    - 1% de six cases
- Il existe 2 gros stock de nourritures sur deux coins de la map, chacun répartie en 5 piles de 20000 unités de nourriture
- Il existe 0.02% de petit stock de nourritures, chacun de 10 unités de nourriture
- La colonie s'étend si il n'y a plus de place pour accueillir toutes les fourmis (si total fourmis * nbCaseColonie)

Fourmis

- Elles mangent depuis la colonie à distance (sauf les esclavagistes)
- Elles peuvent mourir
    - De vieillesse : un an pour les fourmis classiques et 10 ans pour la reine
    - De faim, elles mangent 0.1% de nourriture pour une fourmi et 1% pour la reine
        - Les fourmis esclavagistes peuvent survivre 10j sans manger
- Ouvrières
    - Elles mangent d'abord leur stock
    - Elles sont mineurs au début puis deviennent majeurs après 14 jours
    - Elles partent chercher la nourriture (dans les cases explorées) pour la ramener à la colonie
    - Elles déposent des phéromones (%) lorsqu'elles ont de la nourriture
    - Elles suivent la phéromone au sol
    - Elles déposent leur nourriture si elles meurent
- Reine
    - Elle a 2 larves / jour suivant les proba de l'énoncé (et la 1ère = éclaireur)
    - Elle peut se faire attaquer et ne pas en produire
    - Elle vit au centre de la colonie, elle ne bouge pas
- Eclaireurs
    - Elles sont mineurs au début puis deviennent majeurs après 2 jours
    - Elles explorent les cases non explorées en priorité (et non au hasard)
- Soldats
    - Elles partent traquer les esclavagistes (dans les cases explorées) pour les tuer
    - Elles partent en direction de la fourmilière au bout de 100 jours

- Esclavagistes
    - Elles apparaissent dans les coins
    - Elles se dirigent vers la fourmilière puis repartent d'où elles viennent
    - Elles capturent les larves pour les ramener chez elles
    - Elles peuvent manger la nourriture de la fourmilière si elles sont sur une case de la colonie
    - Si elles meurent, les larves capturées seront réintégrées dans la colonie

Jeu

- La colonie commence avec 200 unités de nourriture
- Un tour correspond à une heure de simulation
- Phéromone
    - Limité à 1000 par case
    - S'évapore à 0.5%
    - Se disperse à 0.02%
    - Une ouvrière ne peut posséder que 500 phéromones, qui est restauré lors de la visite de la colonie
- Les esclavagistes apparaissent aléatoirement tous les 5 à 10 jours
- GNUPlot
    - Le fichier gnuplot.txt est généré lorsque la reine meurt ET / OU lorsque le jeu est mis en pause via le bouton ESPACE
- Le fichier gnuplot.txt est écrasé à chaque lancement de partie
- Chaque tour de jeu enregistre des paramètres pour le fichier GNUPlot :

<tickActuel> <nbFourmisTotal> <nbSoldat> <nbEsclavagiste> <nbOuvriere> <nbEclaireur> <nourritureColonie>