

Abeilles versus frelons

L2 - Informatique, semestre 1

Année universitaire : 2023-2024

Table des matières




1. Guide de l'utilisateur.....	3
2. Présentation technique du projet.....	6
3. Implantation et la partie graphique.....	7
4. Pistes d'amélioration.....	8
5. Répartition des tâches.....	8

Guide de l'utilisateur

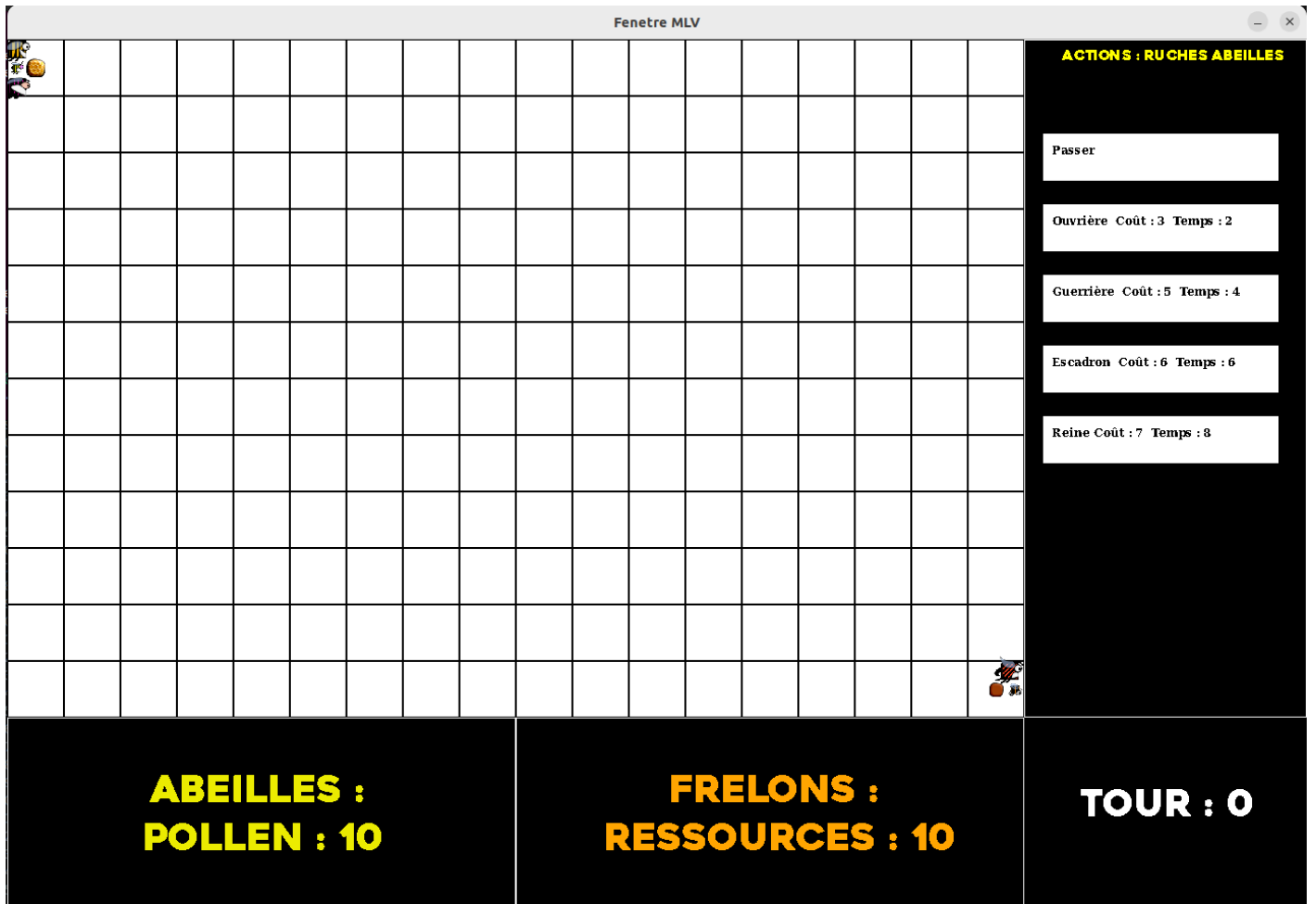
Unitées Abeilles:

	Reine Abeille
	Ouvrière
	Guerrier
	Escadron
	Ruche

Unitées Frelons:

	Reine Frelon
	Frelon
	Nid

Affichage au début de la partie



Comment jouer ?

Compilation du programme : **make**

Lancement du programme : **./bin/AF**

Nettoyage du programme: **make uninstall**

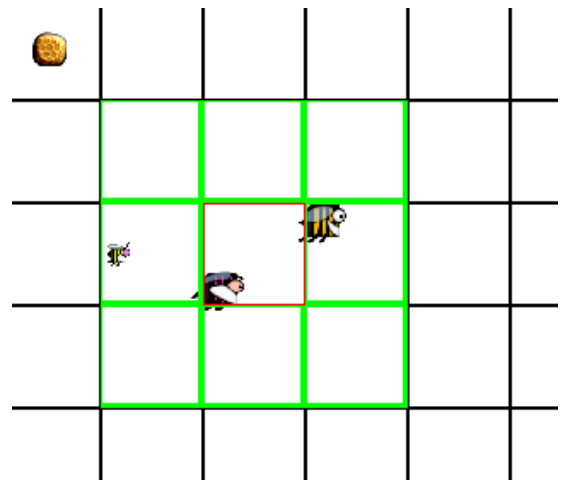
Le début du jeu détermine aléatoirement le camp qui prend l'initiative, ainsi soit les abeilles commencent, soit ce sont les frelons qui prennent le premier tour. À chaque tour, l'utilisateur doit interagir avec les boutons situés du côté droit de la fenêtre. Les actions pour chaque ruche ou nid apparaissent successivement.

Une fois que toutes les ruches ont reçu leurs ordres, les actions pour chaque unité sont présentées à leur tour. Pour faciliter l'identification de l'unité en attente d'actions, un carré rouge s'affiche sur la case correspondante du plateau.

Après avoir donné les ordres à toutes les unités, le même processus se répète pour le camp adverse. Lorsque l'autre camp a également donné tous les ordres, le tour suivant commence. Cette séquence se répète tout au long du jeu.

Si une unité est en attente, telle qu'une ouvrière en train de récolter ou une reine en train de créer un nid, l'affichage des boutons sera modifié en conséquence, et un message d'informations sera affiché. De même, lors de la création d'unités pour les ruches et nids, une adaptation de l'affichage des boutons sera effectuée, accompagnée d'un message d'informations.

Lorsque l'utilisateur souhaite déplacer une unité, les cases accessibles apparaissent en vert. Il est impossible pour le joueur de déplacer son unité vers une case qui n'est pas adjacente ou en dehors du plateau. Cette restriction garantit que les mouvements restent cohérents et limités aux positions directement connectées.



Présentation technique du projet

Au cours de ce projet, nous avons exploité toutes les technologies autorisées. Il est important de souligner que des commentaires ont été intégrés pour chaque fonction (dans tous les fichiers avec l'extension *.h), puisque nous avons opté pour l'utilisation d'un **Makefile** lors du processus de compilation.

Tout d'abord, grâce à la bibliothèque MLV, nous avons réussi à mettre en œuvre avec succès la partie graphique, comme expliqué dans la section suivante. En ce qui concerne les fonctions du "moteur" du jeu, il est essentiel de souligner que les tâches les plus complexes à gérer sont les fonctions d'ajout d'unités (à une case et à sa colonie) ainsi que les fonctions de suppression. Pour faciliter la lisibilité du code, nous avons divisé le code en plusieurs sections et décomposé chaque fonctionnalité en plusieurs étapes distinctes.

Les fonctions de création des nouvelles unités, réalisées grâce à l'allocation dynamique de mémoire (utilisant malloc), sont élaborées en fonction des structures fournies par les responsables du projet. Ensuite, les fonctions responsables de la création de la grille sont mises en place, de même que celles permettant de libérer la mémoire allouée par les unités, les colonies et la grille.

Par la suite, nous avons implémenté des fonctions visant à déterminer la force, le coût et le temps de construction d'une unité souhaitée. Nous avons également fait des fonctions permettant l'ajout et la suppression d'une unité sur une case, ainsi que dans sa colonie (pour les unités) et parmi d'autres colonies (pour les colonies).

En ce qui concerne les reines, il est à noter que nous conservons les adresses des reines ayant déjà construit une colonie.

Si vous avez encore des questions concernant la partie technique, nous vous encourageons à consulter chaque fichier portant l'extension *.h et à examiner la fonction qui suscite votre intérêt.

Implantation et la partie graphique

La partie graphique a commencé à être développée après la création des fonctions du moteur.

La partie graphique dispose de plusieurs fonctions qui ont pour unique but d'afficher les éléments graphique comme par exemple la fonction ***MLV_Image** affiche_ruches(Grille *grille)*** qui affiche les ruches des colonies d'abeilles et de frelon, ainsi que ***MLV_Image** affiche_unites(UListe colonie)*** qui affiche toutes les unités d'une colonie d'abeille ou de frelon. Le type de retour est ***MLV_Images***** car il s'agit d'un pointeur vers une liste d'images MLV. Cela permet ensuite de pouvoir libérer les images en mémoire lorsqu'elles ne sont plus utilisées pour éviter toute fuite de mémoire.

Pour faciliter l'implémentation des boutons, nous avons créé la struct Bouton suivante :

```
// structure pour les boutons
typedef struct {
    int x, y, largeur, hauteur;
    char texte[50];
    char action;
} Bouton;
```

Cette structure a pour objectif de faciliter la gestion des boutons et de leurs actions associées. Ainsi, pour déterminer sur quel bouton l'utilisateur a cliqué, il suffit de parcourir une liste de boutons de type ***Bouton**** et de vérifier si les coordonnées de la souris se trouvent à l'intérieur des dimensions du bouton en question. Ensuite, l'action à effectuer peut être déterminée en consultant la valeur de *char action*.

La structure Bouton est utilisée dans les fonctions ***Bouton* afficher_commandes_prod(int joueur, int* nb_boutons, Grille *grille)*** et ***Bouton* afficher_commandes_unite(int* nb_boutons, Unite* curr, int tour)*** qui permettent respectivement d'afficher la liste des boutons liés aux actions de la ruche ou du nid en fonction du tour et d'afficher la liste des boutons liés aux actions de l'unité. La fonction affiche également un carré rouge sur la case de l'unité en cours.

Pistes d'amélioration

Pour améliorer notre projet, il nous faut implémenter:

- Le combat entre deux unités (la fonction est faite mais ne peut pas être bien utilisée).
- La sauvegarde et le chargement.
- La fin de partie (la fonction est fait mais ne peut pas être bien utilisée)

Répartition des tâches

Nous avons décidé de diviser le projet en 2 parties: le “moteur” du jeu et la partie graphique.

MUNAITPASOV a fait le moteur du jeu et toutes les fonctions qu’elles sont liées à l’initialisation au début, la création des nouvelles unités et l’ajout et la suppression des unités. NAVARRO a fait la partie graphique en utilisant la bibliothèque MLV et la correction de quelques fonctions de moteur du jeu.

Pendant toute la création du jeu, nous avons maintenu une bonne cohésion au sein du binôme, avec des échanges réguliers avant de prendre des décisions concernant l'implémentation de chaque fonctionnalité. Nous avons réussi à mettre en place efficacement la structure requise du jeu, y compris son affichage graphique avec MLV.

Néanmoins nous avons rencontré des difficultés dans l'implémentation du système de combat. Par manque de temps, nous n'avons pas pu le mettre en œuvre de manière suffisamment correcte pour le laisser dans la version finale du jeu. Cependant, nous avons créé la fonction responsable de la destruction de l'unité perdante, ainsi que la fonction qui élimine la colonie et toutes les unités associées.

Ces destructions sont déclenchées en fonction de la victoire de l'unité du camp des frelons sur la colonie adverse. Dans ce cas, la Ruche devient un Nid, et l'unité frelon victorieuse est affiliée à ce nouveau Nid.