Manuel du Développeur

Alexandre Lerosier

9 novembre 2025

Sommaire

Travail effectué

- Création de labyrinthe
- Chargeemnt de labyrinthe
- Classement des joueurs par labyrinthe
- Jouer dans un labyrinthe

Travail à faire

- Suppression de labyrinthe par 'l'utilisateur
- Améliorer le déplacement des monstres : ils se déplacent après le joueur et tentent de le toucher
- Mettre en place une interface graphique (autre que la console)
- Permettre à un joueur de visualiser ses différents scores sur les différents labyrinthes
- Meilleure gestion du nombre maximal de labyrinthes
- Créer un fichier **README** en plus des manuels

Tests

- Test de l'interface utilisateur : `user_interface.c`
- Test de la faisabilité des labyrinthes : `labyrinth_generator.c`
- Test sur les labyrinthes existants : `labyrinth_repository.c`
- Test des classements : `labyrinth_repository.c`, `leaderboard.c`
- Test de l'exécution des fonctionnalités des menus : `menu.c`
- Certains tests ne sont pas exhaustifs

Points importants

Make

- Générer l'exécutable et l'exécutable de test : `make all`
- Générer la documentation : `make doc`
- Générer le manuel de l'utilisateur et le manuel du développeur : `make manuals`
- Nettoyer les fichiers temporaires `.o` : `make clean`
- Nettoyer les labyrinthes et leurs scores : `make clean_labyrinths`

Documentation

La documentation est générée par **Doxygen** à partir du fichier `Doxyfile`.

Manuels

Les manuels d'utilisateurs au format `.pdf` sont générés à partir de leurs fichiers `.md`.

Architecture

Stockage persistant

Les labyrinthes ainsi que leurs scores sont stockés dans les dossiers `/labyrinths` et `/scores`. La gestion du stockage persistant est assurée par le fichier `labyrinth_repository.h`.
Toutes les données sont enregistrées en binaire afin d'optimiser l'utilisation de l'espace de stockage. Les scores sont déjà triés lors de leur enregistrement afin d'éviter de devoir les trier plusieurs fois lors de chaque consultation du classement.

Interface utilisateur

L'intégralité de l'interface utilisateur est gérée par `user_interface.h`, ainsi que par `inputs.h` pour la gestion des entrées utilisateur.

Gestion d'une partie

La partie est gérée par `game.h`, qui coordonne le labyrinthe (`labyrinth.h`) et les monstres (`monsters.h`).

Orchestration des différents modules

Tous ces différents modules, ainsi que ceux qui ne sont pas cités, sont orchestrés par `menu.h`.

Limites et bugs

- Aucune vérification de la taille du paramètre `fileName` dans `repository.h`
- Le nombre de labyrinthes pouvant être lus est limité, mais il n'existe pas de limite pour en créer (certains labyrinthes peuvent donc ne plus s'afficher)
- Lors de la création d'un labyrinthe, si le nom saisi est trop long, il est simplement tronqué
- Les monstres se déplacent avant le joueur et ne peuvent pas se trouver sur la même case que lui, ce qui empêche le joueur d'être touché en allant vers un monstre
- Chaque nouvel ajout de monstre entraîne l'ajout de nombreux nouveaux types de `Square` dans l'énumération. Faut-il envisager une autre manière de modéliser le labyrinthe pour simplifier l'extension ?