Rapport de Projet de Fin de Module : Géneration d’une labyrinthe avec Interface

Encadré par :
Ikram Ben Abdel ouahab

Réalisé par :
Bambara Josias Junior Wilfrid
Akmam Aladine
Boulkassoum Ouologuem

Table of Contents

1

Introduction

Ce document détaille l'architecture, les fonctionnalités et la logique du programme écrit en C++ pour un jeu de labyrinthe interactif utilisant la bibliothèque graphique Raylib . Le jeu propose un menu, plusieurs niveaux de difficulté, un système de gestion des partitions, et des mécaniques de navigation dans un labyrinthe généré dynamiquement.

2

Spécifications Techniques

L'application a été conçue pour offrir une expérience utilisateur fluide et intuitive, grâce à une interface graphique épurée et réactive. Les utilisateurs peuvent interagir facilement avec les éléments à l'écran, grâce à l'intégration de la bibliothèque Raylib, qui simplifie grandement le processus de rendu.
Les fonctionnalités clés incluent la possibilité de manipuler des objets à l'aide de conteneurs STL, comme les vecteurs et les piles, et d'effectuer des opérations complexes grâce aux algorithmes intégrés. La résolution de 900x850 pixels offre une clarté visuelle optimale, garantissant que chaque détail est rendu avec précision.

Architecture du Code

Le programme est structuré en plusieurs classes pour assurer une clarté, une modularité extensibilité optimale

1. Enums :

- AppState : Gère les états du jeu (écran principal, choix du niveau, jeu, fin, affichage des scores).
- NiveauDifficulte : Définit les difficultés (Facile, Moyen, Difficile).

2. Structures :

- Score : Représente un score basé sur le temps de complétion du labyrinthe.
- Cell : Représente une cellule du labyrinthe avec ses murs et son état de visite.
- ScoreManager : Gère les scores pour chaque niveau de difficulté.

3

Architecture du Code

3. Classes :

- Niveau : Stocke les paramètres du niveau (difficulté et taille des cellules).
- Button : Représente les boutons interactifs de l'interface utilisateur.
- Labyrinthe : Gère la génération et l'affichage du labyrinthe.
- Joueur : Représente le joueur et gère ses mouvements.
- Jeu : Gère l'ensemble du jeu, les états, et les interactions.

4

Détails des Classes et Fonctionnalités

1. Enumérations

AppState

- Représente les différents états du jeu :
  - MENU : Écran principal.
  - SELECTION_NIVEAU : Choix de la difficulté.
  - JEU : Jeu actif.
  - FIN : Fin de partie.
  - SCORES : Affichage des scores.

4

Détails des Classes et Fonctionnalités

NiveauDifficulte

- FACILE : Grandes cellules (100x100 pixels).
- MOYEN : Cellules moyennes (50x50 pixels).
- DIFFICILE : Petites cellules (25x25 pixels)

5

Détails des Classes et Fonctionnalités

2. Structures

- Score
  - Représente un temps de complétion enregistré pour un joueur.
- Cell
  - Représente une cellule dans la grille du labyrinthe.
  - Attributs :
  - walls : Tableau de booléens indiquant la présence de murs (état haut, droite, bas, gauche).
  - visited : Booléen indiquant si la cellule a été visitée lors de la génération du labyrinthe.

Détails des Classes et Fonctionnalités

3. Classes

ScoreManager

- Gère les scores pour chaque difficulté :
  - ajouterScore : Ajoute un score pour le niveau choisi et trie la liste.
  - afficherScores : Affiche les 10 meilleurs scores pour un niveau.

Niveau

- Gère les paramètres d’un niveau :
  - getCellSize : Retourne la taille des cellules en fonction de la difficulté.
  - setDifficulte : Change la difficulté actuelle.

5

Détails des Classes et Fonctionnalités

Button

- Représente un bouton interactif :
  - isClicked : Détecte si le bouton a été cliqué.
  - draw : Affiche le bouton sur l'écran.

Jeu

- Classe principale pour gérer l'état et la logique du jeu :
  - Attributs principaux :
    - appState : État actuel de l'application.
    - labyrinthe : Pointeur vers l'instance du labyrinthe.
    - ratTexture, cheeseTexture : Textures pour le joueur et l'objectif.
  - Méthodes :
    - run : Boucle principale du jeu gérant les événements et le rendu graphique.
    - drawMenu, drawSelectionNiveau, drawGame : Gèrent les différents écrans du jeu.

5

Détails des Classes et Fonctionnalités

Labyrinthe

- Gère la création et l'affichage du labyrinthe :
  - Attributs principaux :
    - grid : Grille 2D de cellules.
    - cols, rows : Nombre de colonnes et de lignes.
    - cellSize : Taille des cellules.
  - Méthodes :
    - generate : Génère le labyrinthe avec l'algorithme de backtracking.
    - removeWalls : Supprime les murs entre deux cellules adjacentes.
    - draw : Affiche le labyrinthe et ses murs.

Joueur

- Représente le joueur :
  - move : Gère les déplacements du joueur dans la grille en fonction des entrées.
  - getPosition : Retourne la position actuelle du joueur.

5

Fonctionnalités Principales

1. Menu Principal
   - Affichage d’un bouton pour démarrer le jeu.
   - Navigation vers la sélection des niveaux.
2. Sélection de la Difficulté
   - Boutons interactifs pour choisir entre les niveaux : Facile, Moyen, ou Difficile.
   - Chaque niveau ajuste la taille des cellules du labyrinthe.
3. Générer un Menu Principale
   - Affichage d’un bouton pour démarrer le jeu.
   - Navigation vers la sélection des niveaux.

5

Fonctionnalités Principales

4. Navigation dans le Labyrinthe
    - Le joueur se déplace avec les touches directionnelles.
    - Les mouvements sont limités par les murs des cellules.
5. Gestion des Scores
    - Enregistrement du temps de complétion.
    - Affichage des 10 meilleurs scores par difficulté.
6. Affichage Graphique
    - Utilisation de Raylib pour dessiner les cellules, murs, et objets.
    - Textures pour représenter le joueur (rat) et la sortie (fromage).

7

Conclusion

Ce programme fournit une implémentation robuste et modulaire pour un jeu de labyrinthe avec gestion des scores. L’utilisation de Raylib simplifie le rendu graphique tout en assurant une exécution fluide. Avec des améliorations futures, ce projet pourrait être enrichi pour offrir une expérience encore plus engageante et personnalisable.

Merci !