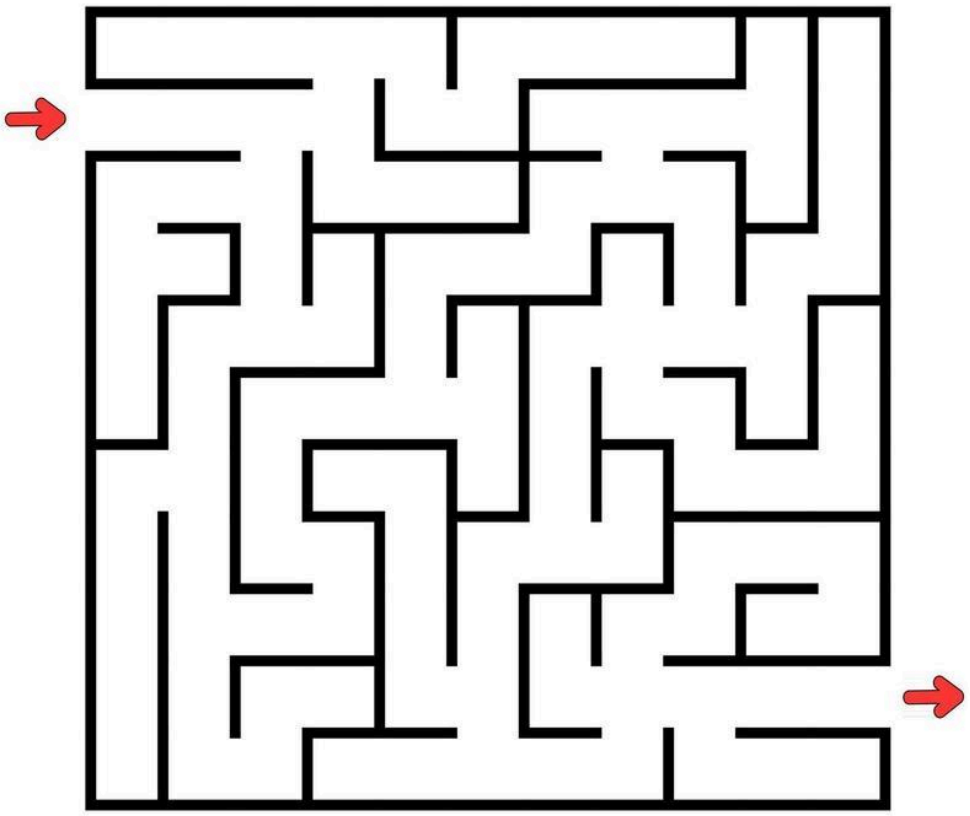
Université Abdelmalek Essaâdi
Faculté des Sciences et Techniques de Tanger
Département : Génie Informatique

Année universitaire 2024-2025
Filière : Licence IDAI
Prof. Badr-Eddine Boudriki Semlali



Rapport : Jeu de labyrinthe avec obstacles en C++ utilisant Raylib

réalisé par :
ABBOUTI YOUSRA, AKDIM RIHAB

introduction:

Dans le cadre de ce projet, nous avons réalisé un jeu de labyrinthe interactif en utilisant la bibliothèque **Raylib** pour le rendu graphique. Le principe du jeu est simple : le joueur contrôle un personnage, **Jerry**, qui doit traverser un labyrinthe pour atteindre une cible représentée par un morceau de fromage. Le labyrinthe est généré de façon procédurale sous forme de grille, avec des dimensions et une difficulté qui augmentent selon le niveau.

Nous avons ajouté plusieurs fonctionnalités essentielles pour rendre le jeu dynamique et agréable à jouer. Par exemple, le joueur peut déplacer Jerry grâce aux touches directionnelles, et le labyrinthe s'affiche de manière fluide avec quelques animations pour améliorer l'expérience utilisateur. Dans les niveaux avancés, nous avons intégré des **obstacles mobiles** pour ajouter du défi, obligeant le joueur à élaborer une stratégie pour progresser.. Une **gestion des états du jeu** a également été mise en place pour passer entre les différentes phases : l'écran d'accueil, le jeu principal et l'écran de fin.

Ce projet nous a permis de mettre en pratique nos compétences en **programmation C++** tout en utilisant une bibliothèque graphique moderne pour concevoir un jeu interactif complet. C'était une expérience enrichissante qui nous a permis de combiner logique, créativité et rigueur technique.

Structure du code

Inclusion des bibliothèques

Nous avons commencé par inclure plusieurs bibliothèques importantes :

- **`iostream` : pour gérer l'affichage des messages dans la console et la lecture des entrées utilisateur.**
- **`raylib` : c'est la bibliothèque graphique que nous avons utilisée pour créer la fenêtre du jeu, dessiner les éléments graphiques comme le labyrinthe, et gérer les interactions du joueur.**
- **`class.h` : un fichier externe qui contenait des classes ou fonctions personnalisées, probablement pour gérer les objets du jeu comme le joueur ou le labyrinthe.**

Dimensions de la fenêtre et de la grille

Nous avons défini les dimensions de la fenêtre et du labyrinthe :

- La fenêtre avait une taille de **900 pixels en largeur** et **800 pixels en hauteur**.
- Chaque cellule du labyrinthe mesurait **40 pixels x 40 pixels**, et le labyrinthe était composé de **10 colonnes** et **7 lignes**, soit une grille de **10x7** cellules.

Centrage de la grille

Nous avons centré le labyrinthe dans la fenêtre en calculant des décalages.

- **Pour l'horizontalité** (`mazeOffsetX`), nous avons pris la largeur totale de la fenêtre, enlevé la largeur du labyrinthe, puis divisé ce reste par 2 pour obtenir le décalage.
- **Pour la verticalité** (`mazeOffsetY`), nous avons appliqué la même logique avec la hauteur de la fenêtre.
  Ainsi, peu importe les dimensions de la fenêtre ou du labyrinthe, il était toujours centré.

Variables pour le jeu

Nous avons créé plusieurs variables essentielles pour suivre l'état du jeu :

- **`gamewin`** : une variable booléenne qui indique si le joueur a gagné ou non.
- **`score`** : le score actuel du joueur.
- **`startTime` et `elapsedTime`** : utilisées pour suivre le temps écoulé depuis le début du niveau.
- Nous avons aussi créé un tableau **`levelScores`** pour enregistrer le score de chaque niveau et voir les progrès du joueur.

Gestion des niveaux

Nous avons structuré notre jeu pour qu'il ait plusieurs niveaux :

- Le niveau de départ était le **niveau 0** (indiqué par `currentLevelIndex`), et nous avons mis en place un mécanisme pour avancer d'un niveau à l'autre.
- Chaque niveau était associé à des obstacles spécifiques qui devenaient de plus en plus complexes à mesure que le joueur avançait.
  Nous avons utilisé un minuteur pour définir l'apparition ou le mouvement de ces obstacles.

États du jeu

Pour gérer les différentes phases du jeu, nous avons créé une **énumération** appelée `GameState` :

- **Menu principal** : c'est là où le joueur pouvait choisir de commencer le jeu ou de quitter.
- **Jeu actif** : c'était l'état où le joueur était effectivement en train de jouer, interagissant avec le labyrinthe et les obstacles.
  Cela nous a permis de structurer la logique du jeu et d'afficher les éléments appropriés en fonction de l'état actuel (par exemple, on n'affiche pas le labyrinthe pendant le menu principal).

Gestion du joueur et de l'objectif

Nous avons conçu deux éléments clés dans notre labyrinthe :

1. **Le joueur** : Nous avons suivi les coordonnées du joueur pour savoir où il se trouvait dans la grille.

2. **L’objectif** : C’était une case spéciale dans le labyrinthe, et le but du jeu était d’y parvenir.
   Quand le joueur atteignait l’objectif, il gagnait, et nous pouvions alors avancer au niveau suivant.

Utilisation de textures

Nous avons ajouté des textures pour rendre le jeu plus interactif et visuellement attrayant. Cela comprenait des images représentant le joueur et d’autres éléments du jeu, comme les murs ou les obstacles. Nous avons chargé ces textures dans des variables afin de pouvoir les afficher correctement dans la fenêtr

Points forts

1. **Multiniveaux** :
   - Le jeu propose trois niveaux de difficulté avec des labyrinthes de tailles croissantes.
   - Les niveaux introduisent des obstacles dynamiques pour augmenter la difficulté.
2. **Interface utilisateur conviviale** :
   - Écran de menu interactif.
   - Écrans pour indiquer la victoire et la fin d'un niveau.
   - Boutons pour rejouer ou passer au niveau suivant.
3. **Gestion des scores et du chronométrage** :
   - Les scores des niveaux sont calculés en fonction du temps pris pour les compléter.
   - Un score total est affiché à la fin.
4. **Obstacles mobiles** :
   - Les obstacles ajoutent un défi supplémentaire, surtout dans le niveau DIFFICILE.