**Documentation du Projet Motus**

**1. Introduction**

Le projet **Motus** est un jeu de mots développé en langage C utilisant les bibliothèques **SDL2** et **SDL2\_ttf** pour l'affichage graphique et la gestion de la saisie. Le joueur doit deviner un mot secret en proposant des réponses. Pour chaque tentative, le jeu fournit un retour visuel :

* **Lettre verte** : la lettre est correcte et bien placée.
* **Lettre jaune** : la lettre est présente dans le mot secret mais à une mauvaise position.

Le joueur dispose de 6 tentatives maximum. Le jeu inclut également un système de sauvegarde/reprise et la gestion des scores.

**2. Structure du Projet**

Le projet est organisé en plusieurs modules pour faciliter sa maintenance et son évolutivité. L'arborescence du projet est la suivante :

pgsql

Copier

motus/

├── common.h // Déclarations globales : constantes, types et énumérations

├── common.c // (Peut être vide, réservé aux définitions globales éventuelles)

├── game.h // Interface de la logique du jeu

├── game.c // Implémentation de la logique du jeu

├── utils.h // Fonctions utilitaires (gestion du dictionnaire, conversion de texte, etc.)

├── utils.c // Implémentation des fonctions utilitaires

├── render.h // Interface de rendu graphique (affichage des écrans, boutons, pop-up d’aide)

├── render.c // Implémentation du rendu graphique

└── main.c // Point d’entrée du programme, boucle d’événements, gestion des états

**3. Description des Modules**

**3.1. Module Common**

**common.h** définit les constantes, les types et les énumérations utilisés dans tout le projet :

* **Dimensions de la fenêtre** (WINDOW\_WIDTH, WINDOW\_HEIGHT)
* **Nombre maximum de tentatives** (MAX\_ATTEMPTS)
* **L'énumération Difficulty** qui définit les niveaux de difficulté (DIFF\_DEBUTANT, DIFF\_AMATEUR, DIFF\_CONFIRME, DIFF\_EXPERT) et donc le nombre de lettres du mot secret.
* **L'énumération AppState** qui gère les différents états de l'application : MENU, HELP (pop-up d’aide), DIFFICULTY, GAME, GAME\_OVER et EXIT.
* **Les structures Player et MotusGame** qui représentent respectivement le joueur (pseudo et score) et l’état d’une partie (mot secret, tentatives, feedback, etc.).

**3.2. Module Game**

Ce module gère la logique du jeu :

* **getFeedback** compare le mot secret et la tentative du joueur pour déterminer le nombre de lettres bien placées et mal placées.
* **initGame** initialise une nouvelle partie en sélectionnant aléatoirement un mot du dictionnaire en fonction du niveau de difficulté.
* **processAttempt** enregistre la tentative du joueur et met à jour l'état de la partie (vérifie si le mot a été trouvé ou si le nombre maximum de tentatives est atteint).
* Les fonctions **saveGame** et **loadGame** permettent de sauvegarder et de reprendre une partie.
* Les fonctions **loadPlayerScore**, **savePlayerScore** et **updatePlayerScore** gèrent la lecture et l’enregistrement des scores dans un fichier externe.

**3.3. Module Utils**

Ce module propose des fonctions utilitaires :

* **loadDictionary** lit un fichier externe (dictionary.txt) contenant des mots (de 4 à 7 lettres) et les charge en mémoire.
* **pickRandomWord** sélectionne aléatoirement un mot du dictionnaire correspondant au niveau de difficulté choisi.
* **toUpperCase** convertit une chaîne de caractères en majuscules pour garantir une comparaison cohérente.

**3.4. Module Render**

Ce module s'occupe de l'affichage graphique :

* **drawText** affiche du texte à l'écran.
* **drawButton** dessine des boutons avec un texte centré.
* **renderMenu** affiche le menu principal avec les options "Nouvelle Partie", "Charger Partie", "Quitter" et un bouton "?" pour l'aide.
* **renderDifficulty** affiche l'écran où le joueur entre son pseudo et choisit la difficulté (le nombre de lettres du mot secret est déterminé par la difficulté sélectionnée).
* **renderGame** affiche l'écran de jeu qui indique le pseudo du joueur, son score, le nombre de lettres du mot à deviner et le nombre de tentatives effectuées sur 6. Les tentatives sont affichées sous forme de carrés colorés (vert ou jaune) pour indiquer le feedback.
* **renderGameOver** affiche l'écran de fin de partie avec un message indiquant si le joueur a gagné ou perdu.
* **renderHelpPopup** affiche une pop-up d’aide agrandie (700×450) avec les consignes du jeu, réparties sur plusieurs lignes (environ 10 mots par ligne). Un bouton "Retour" est positionné en bas à droite pour fermer la pop-up.

**3.5. Module Main**

Le fichier **main.c** est le point d’entrée du programme et gère :

* L'initialisation de SDL2 et SDL2\_ttf.
* La création de la fenêtre et du renderer.
* La boucle d’événements qui gère la saisie du clavier (pour le pseudo et les tentatives) et la souris (pour cliquer sur les boutons).
* La gestion des états de l'application (menu, aide, choix de difficulté, jeu, fin de partie).
* Le changement de curseur pour indiquer les zones cliquables.

**4. Exécution**

Assurez-vous que les fichiers externes **font.ttf** et **dictionary.txt** se trouvent dans le répertoire de travail (généralement le même dossier que l'exécutable ou le dossier de sortie configuré par votre IDE). Lancez ensuite l'exécutable **Motus.exe** (ou simplement **Motus** sous Linux/Mac).

**5. Utilisation du Jeu**

* **Menu Principal** :  
  Le joueur voit les options "Nouvelle Partie", "Charger Partie", "Quitter" et un bouton "?" qui ouvre une pop-up d’aide.
* **Choix de la Difficulté** :  
  En cliquant sur "Nouvelle Partie", le joueur accède à l'écran de difficulté où il doit saisir son pseudo et choisir un niveau (Débutant, Amateur, Confirmé, Expert). Le nombre de lettres du mot secret est défini par ce niveau.
* **Écran de Jeu** :  
  L'écran de jeu affiche le pseudo du joueur, son score, le nombre de lettres du mot à deviner et le nombre de tentatives réalisées sur 6. Chaque tentative est affichée sous forme de carrés colorés indiquant le feedback (lettres vertes ou jaunes).
* **Sauvegarde et Reprise** :  
  Le joueur peut sauvegarder sa partie et la reprendre ultérieurement via l'option "Charger Partie".
* **Fin de Partie** :  
  L'écran de fin de partie affiche un message indiquant si le joueur a gagné ou perdu et propose de retourner au menu.
* **Pop-up d’Aide** :  
  En cliquant sur le bouton "?" du menu principal, une pop-up d’aide apparaît, affichant les règles du jeu de manière lisible (environ 10 mots par ligne). Un bouton "Retour" en bas à droite permet de fermer la pop-up.

**6. Conclusion**

Ce projet démontre la mise en œuvre d'un jeu graphique en C à l'aide de SDL2 et SDL2\_ttf. Le code est organisé en modules clairs et structurés, permettant de gérer efficacement la logique du jeu, le rendu graphique et la gestion des états. La documentation fournie ici détaille l'architecture, le fonctionnement de chaque module et les instructions pour compiler et exécuter le projet.

Par Yassine Badaoui & Nathan Sion