UNIVERSITE DES SCIENCES ET DE LA TECHNOLOGIE HOUARI BOUMEDIENE
FACULTE D'ELECTRONIQUE ET D'INFORMATIQUE

MINI PROJET

JEU MODELE DE DROITE

Magrah Yacine

Tobok Mouaadh

Badada Hadjer

Merzouagui Amina

MODULE ALGORITHMIQUE 3

Mr Smaili Cherif

ANNEE UNIVERSITAIRE 2021/2022

INTRODUCTION

C’est grâce à ce Mini Projet que nous avons eu l’opportunité de cumuler les connaissances théoriques avec celles de la pratique. En codant un jeu 2D en langage C, pour la première fois, ceci a permet d’avoir une nouvelle expérience dans la vie active et de découvrir plus précisément le professionnel.

Ce Projet nous a permis à découvrir l’utilité de la bibliothèque SDL « Simple Direct-Media Layer », qui permet de réaliser des projets de contexte rendue 2D, un API qui nous donne la possibilité de toucher et gérer les interactions avec tout périphériques E/S, (souris, clavier … etc.).

Pour compléter notre Mini Projet nous avons rédigé ce rapport qui traite trois grandes parties, la première consiste à donner une vue globale sur le fonctionnement du jeu « modèle de droite », la deuxième explique comment créer les matrices du jeu et la troisième inclue la méthode pour réagir au saisit du joueur (cliques).

DESCRIPTION ET REGLE DE JEU

Conditions de jeu :

Le Jeu modèle de droite est un jeu 2D qui exige de simples conditions pour le joueur.

Au début, le jeu consiste à remplir une matrice 6x6 de diagonales, qui relient des points d'une matrice 7x7 de chiffres entre 0-4 soit de droite vers gauche «diag \ » ou de gauche vers droite « / » ,(**figure 1)**.

Il faut savoir que les fonctionnalités du jeu n'autorise pas d'indiquer l'existence d'une faute dans le saisit du joueur (**figure 2**), mais plutôt le jeu s'arrête des-que le saisit du joueur correspond à la réponse optimale du problème randomisé de la partie actuelle.

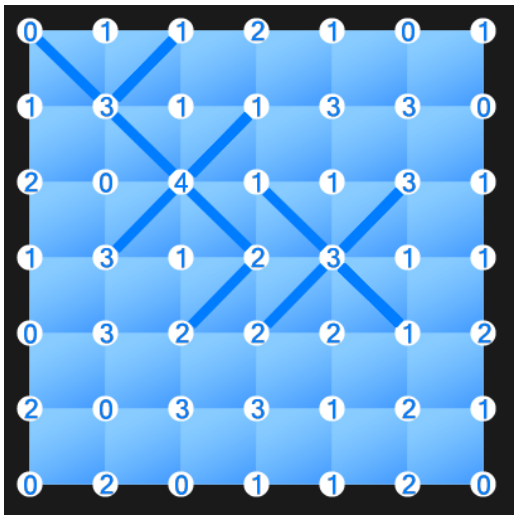


Figure 1

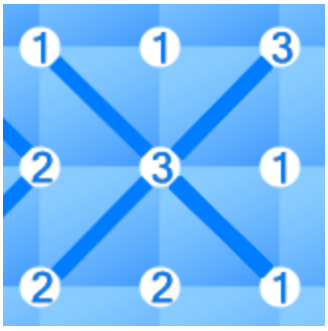


Figure 2

D’autre part, notre jeu est équipé par un chronomètre (**Figure 3)** qui indique le temps écoulé par le joueur avant victoire éventuelle ou arrêt, il démarre dès que le joueur clique sur START, ce chronomètre peut aussi se mettre en PAUSE.

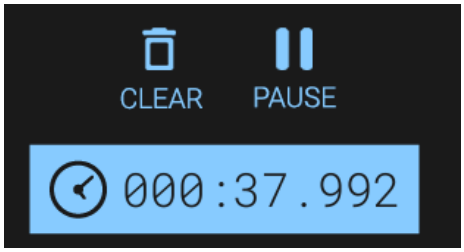


Figure 3

Fonctionnalités des buttons de fenêtre de jeu :

La fenêtre de jeu contient seulement deux buttons :

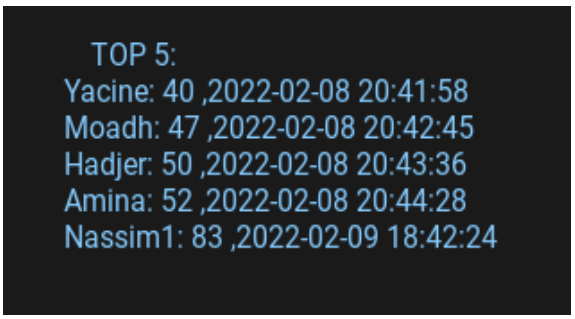
- CLEAR : pour réinitialiser la matrice de saisit du joueur.
- PAUSE : pour mettre en pause le jeu et afficher la fenêtre de pause.
- LA MATRICE : Sur la fenêtre de jeu il s’affiche une matrice 7x7 qui a sur les coins des chiffres entre 0-4, et pour les lier il faut faire des cliques sur les cellules , Le 1er clique place une diagonale de gauche vers droite , le 2eme de droite vers gauche et le 3eme fait disparaitre la diagonale.

1er click: 

2eme click: 

3eme click: 

Le chronomètre prend le rôle d’un élément arbitre dans ce jeu, il représente le score acquit par le jouer. À l’arrêt de jeu (pause ou victoire), les scores seront disposés en ordre croissant (TOP 5) où le plus petit est meilleur .



Fonctionnalités des buttons de fenêtre Arrêt:

Le jeu possède 2 fenêtres d'arrêt :

- Fenêtre WON (Victoire) :
  - SAVE : sauvegarder la partie.
  - MAIN : retour à l'accueil.
  - RPLAY : Nouvel partie.
- Fenêtre PAUSED (pause) :
  - SAVE : sauvegarder la partie.
  - MAIN : retour à l'accueil.
  - RESUME : Nouvel partie.

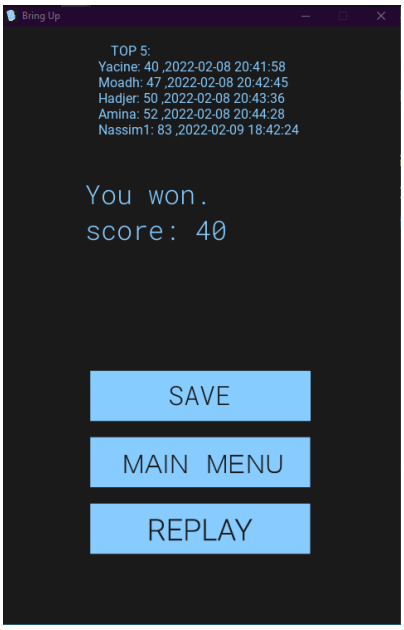
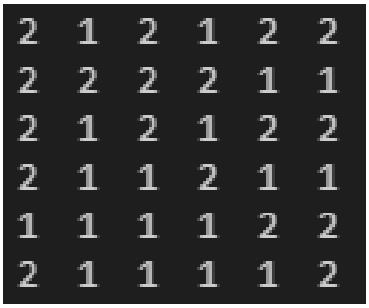


Figure 5

CRÉATION DES MATRICES

On utilisera dans ce jeu des matrices dynamiques crées avec *malloc (size) ;*

Matrice modèle (RANDM):



La matrice modèle (RANDM) est la matrice de problème de la partie, grâce à la fonction RANDM, on remplit chaque case soit avec 1-2, tel que ;

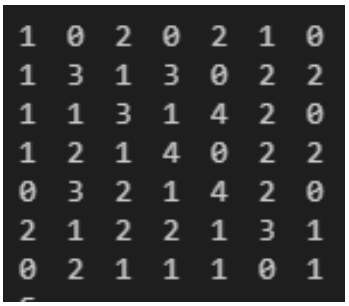
1 = / diagonale de gauche vers droite.

2 = \ diagonale de droite vers gauche.

Matrice des chiffres :

Pour que le jeu affiche les chiffres dans les coins de chaque cellule, sachant qu'on a 6 cellules par colonnes et 6 par lignes, on doit avoir une première matrice de type 7 × 7 veut dire une matrice de 7 colonnes et 7 lignes.

Cette matrice sera remplit par des (0, 1, 2, 3, 4) et qui représentent le nombre de lignes connectant à ce point. Notons un point important, le positionnement de ces chiffres ne doit pas être fixe mais il se change à chaque nouveau départ du jeu cela nous garantit l'innovation dans le jeu et réduit la possibilité de tricher, donc jouer avec plus de crédibilité.



Pour que cela soit possible on a utilisé dans notre programme la fonction INITGRID qui fait un appel à la fonction RANDM qui randomise la matrice modèle 6x6, remplit la matrice 7x7 en fonction de la matrice modèle et initialise la matrice du joueur à zéro.

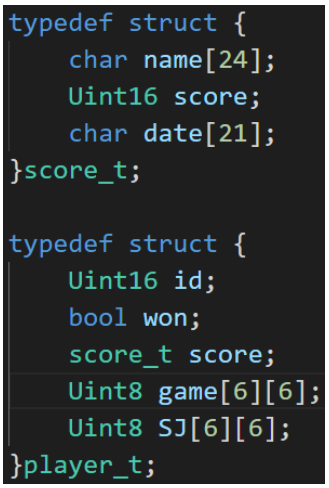
Pour que le joueur arrive à manipuler le jeu on doit lui afficher une matrice de type 6 × 6 veut dire 6 colonnes et 6 lignes, cette dernière sera remplit par des zéros et qui signifie pas de diagonales et après chaque clique elle va ajouter 1 (x=(x+1)mod3).

Contrôle de fin de partie :

La partie se fini des que la matrice de saisit du joueur correspond à la matrice modèle du problème, on utilisera la fonction MATCMP pour vérifier cela.

Sauvegarde de partie :

La sauvegarde de partie se fait sur un fichier locale, avec un type d'enregistrement *Player_t ;* qui est de taille 124 octets et se compose de:



```
typedef struct {
    char name[24];
    Uint16 score;
    char date[21];
}score_t;

typedef struct {
    Uint16 id;
    bool won;
    score_t score;
    Uint8 game[6][6];
    Uint8 SJ[6][6];
}player_t;
```

- Id du joueur (2 bytes).
- Booléen de victoire (2 bytes).
- ENG score (48 bytes):
  - Nom (24 bytes).
  - Score/temps (2 bytes).
  - Date départ (22 bytes).
- Matrice modèle (36 bytes).
- Matrice saisit (36 bytes).

IMPLÉMENTATION SDL

Utilisation du SDL:

Pour ce projet on utilisera SDL pour gérer le saisit du joueur (que ce soit cliques sur bouton ou cellules de matrice), Afficher l'information du jeu sur l'écran (chrono, matrice, boutons …).

De plus, la bibliothèque principale du SDL ne suffit pas dans ce type de projet, donc on s'appuiera sur d'autres bibliothèques SDL supplémentaires comme *SDL_TTF* et *SDL_IMG* pour faciliter le travail.

Initialisation du SDL:

Notre programme commence par un appel au fonctions *SDL_Init(), TTF_Init()* et *IMG_Init()* qui sont nécessaires pour utiliser les biblio SDL. Ensuite, on crée une fenêtre *SDL_Window** et un moteur de rendu *SDL_Renderer** qui va dessiner des textures *SDL_Texture** sur la fenêtre.

Fonctionnement du SDL:

L'utilisation usuelle du SDL consiste à avoir deux boucles principales concaténées l'une dans l'autre :

- Boucle de Rendu. *While(!quit){*
    - Boucle des Événements. *While(SDL_PollEvent){…}}*

Boucle Événements :

Cette boucle fait parcourir les événements ***SDL_Event*** qui se passent dans la durée avant Rendu et réagit en conséquence, grâce à des fonctions ***Type_porcess_event***.

Par Exemple :

Événement= clique souris droit, position (65,90) est équivalent à un clique sur matrice, case (1,1), donc on incrémente la valeur de la case 1,1 de la matrice de saisit (*SJ[1,1] = (SJ[1,1]+1)%3 ;).*

Boucle Rendu :

La boucle responsable au dessin, on considérant les changements déclenchés par la boucle Evt.

Remarque : pour maintenir un certain niveau de stabilité, on ajoute *SDL_Delay();* en but de ne pas surcharger le system.

Clôture du SDL:

Après que la boucle Evt avait déclenché le changement du booléen de sortie *bool quit,* la boucle Rendu ne serait plus active, en conséquence le code de sortie situé après sera exécuter, ce code fait libérer toute sorte de textures et allocations mémoire faites, et ferme les initialisations faites au début avec *IMG_Quit,TTF_Quit,SDL_Quit*.

CONCLUSION

En conclusion, nous devons avouer que rétrospectivement nous sommes satisfaits de notre travail dans ce Mini Projet puisque nous avons atteint des nouveaux objectifs.

En effet, ce Mini Projet nous a permis de comprendre et maitriser dans une certaine mesure, la programmation en C et de savoir exploiter les fonctionnalités du API SDL pour créer des jeux vidéo en 2D sans complexité.

Finalement, nous n'avons qu'a exprimer notre satisfaction envers cette expérience qu'il nous a appris à gérer notre temps afin de faire le maximum de travail prévu, pour fournir un jeu prêt dans le délai imposé.