



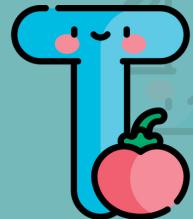
MINI PROJET

EN LANGAGE C

ENCADRÉ PAR:MR SMAILI

WORD HUNT

RÉALISÉ PAR :
BOULAHIA YASMINE
KHADRAOUI YOUNÉS
LEGHRIB RAMI
ALLAG WASSIM



1. INTRODUCTION



Le projet en question nous a permis d'acquérir des connaissances théoriques et pratiques, tout en nous permettant de découvrir le milieu professionnel et d'appliquer les compétences acquises au cours des trois derniers semestres. Il s'agit d'une réalisation d'un jeu vidéo en 2 dimensions, utilisant le langage C et la bibliothèque libre et multimédia SDL (Simple Direct Media Layer), qui a permis un accès à l'audio, la souris, le clavier, etc. Dans le rapport qui suit, nous vous présenterons le fonctionnement du jeu, ainsi que les techniques de programmation utilisées et les différentes idées, fonctions, procédures et bibliothèques utilisées.

(1) PRÉSENTATION DU JEU:

L'idée derrière ce jeu est très simple, c'est le jeu "jeu de mot" qui met en compétition une machine et un humain. Il se déroule dans une fenêtre contenant N lettres. L'objectif est de retrouver une combinaison de lettres obtenue par rotation et changement de position de n lettres adjacentes. La partie se termine lorsque la combinaison est trouvée, que ce soit par la machine ou par l'humain.



(2) COMPILATEUR ET INSTALLATION :

Nous avons opté pour l'utilisation de Code Blocks en tant que compilateur, en raison de sa capacité à prendre en charge plusieurs bibliothèques, et parce qu'il est compatible avec SDL. Nous avons téléchargé la version 20.03mingw-setup.exe de Code Blocks sur le site <http://www.codeblocks.org/downloads/binaries/> et nous avons téléchargé SDL depuis <https://sourceforge.net/projects/libSDL/>. En suivant des tutoriels sur YouTube, nous avons réussi à installer SDL sous Code Blocks. Nous avons également installé SDL_mixer pour la gestion du son, SDL_image pour afficher des images et SDL_ttf pour utiliser différentes polices.





2. GUIDE D'UTILISATION:



(A) ANALYSE DU JEU :

Ce jeu est un défi pour l'esprit car il demande à l'utilisateur ou à la machine de réfléchir à des combinaisons de lettres à partir d'une série de lettres initiales. Il peut être joué à différents niveaux de difficulté en modifiant la longueur de la chaîne de lettres (N) et le nombre de lettres adjacentes (n) à utiliser pour les rotations et les changements de position. C'est un jeu amusant qui peut être joué seul ou en compétition contre la machine.

(B) RÈGLES DU JEUX:

- Les règles de ce jeu de lettres sont les suivantes :
- La longueur de la combinaison de lettres doit être supérieure à 2 et inférieure à 8.
- La longueur de la combinaison initiale = La longueur de la combinaison finale.
- Le nombre de lettres adjacentes à pivoter doit être supérieur ou égal à la moitié de la longueur de la combinaison.
- L'état final de la combinaison est représenté par des 0 pour les lettres inversées et des 1 pour les lettres normales.
- La longueur de l'état de la combinaison = la longueur des lettres.
- Le BEST SCORE c'est la personne qui trouve la combinaison en peu de temps.

(c) DÉROULEMENT DU JEU

Lorsque le jeu débute nous sommes sur la page principale du jeu qui est un menu qui permet à l'utilisateur de choisir entre les trois modes, le téléchargement et la liste de meilleur score. Comme vous voyez dans la figure 1.



[FIGURE 1: LE MENU DU JEU]

Lorsque vous jouez et vous classez parmi les 5 premiers (Figure 2) vous devez entrer votre nom.



[FIGURE 2: VOUS ÊTES PARMI LES 5 PREMIERS]

Lorsque on clique sur le bouton "COMPUTER" vous retrouvez vous devant cette Figure(4) ou vous devez taper la combinaison initiale .



[FIGURE 4: MODE " COMPUTER " TAPEMENT DE LA PREMIÈRE COMBINAISON]

Après cette étape on se retrouve devant deux combinaisons de lettres (Figure 3), la première combinaison c'est celle qu'il faut permuter entre ces lettres adjacents pour trouver la deuxième .

Sans oublier qu'on a l'option de mettre la partie en pause , le temps consommé dans cette partie , le meilleur temps et le nombres de lettres adjacentes qu'on peut permuter.



[FIGURE 3: MODE " ON PLAYER " JOUER]

Après avoir sélectionné "DONE", vous devrez choisir le nombre de lettres adjacentes à faire pivoter, comme indiqué dans l'image ci-dessous.(Figure 5)



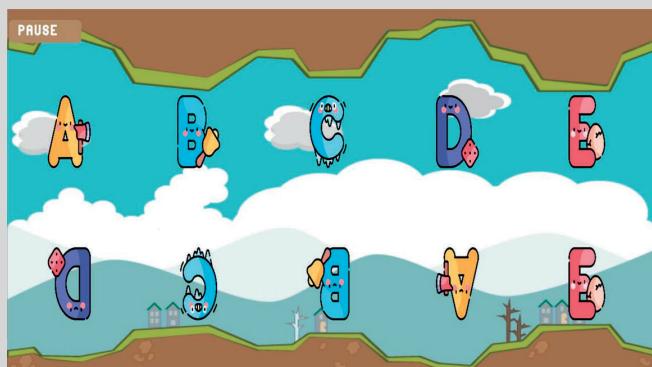
[FIGURE 5:MODE " COMPUTER " CHOIS DU NOMBRES DE LETTRES ADJACENTES A PERMUTER.]

Après cette étape on se retrouve devant la(figure 6)pour choisir la combinaison qu'on veut atteindre .



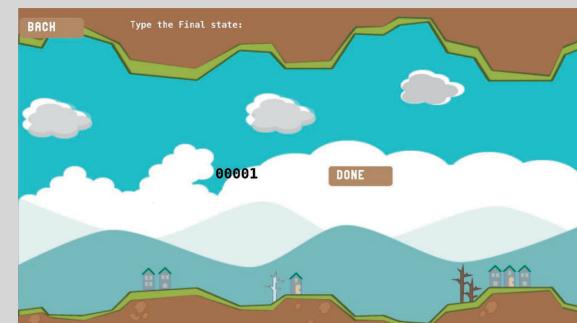
[FIGURE 6::MODE " COMPUTER " CHOIS DE LA COMBINAISON FINALE]

Quand vous cliquez sur «DONE» l'ordinateur commence à jouer avec tes combinaisons saisies .(Figure8)



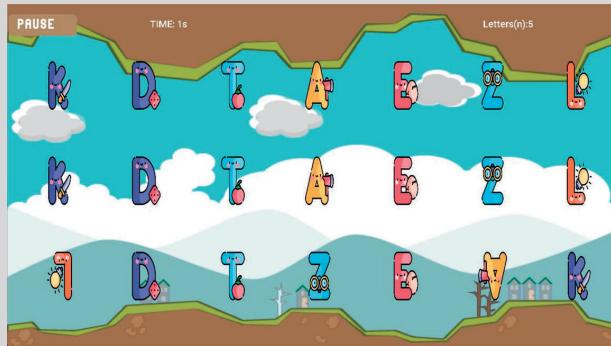
[FIGURE 8:MODE " COMPUTER " L'ORDINATEUR JOUE]

Après avoir sélectionné "DONE", vous devrez choisir la combinaison finale vous cliquez sur «DONE» vous passez a la fenêtre en bas (Figure 7) pour taper l'état des lettres de la combinaison précédente .



[FIGURE 7:MODE " COMPUTER " L'ÉTAT DE LA COMBINAISON FINALE.]

Lorsque vous appuyez sur «VS COMPUTER» vous êtes sur (figure9)



[FIGURE 9:MODE “ VS COMPUTER ”]

- Sinon si vous voulez quitter le jeu vous appuyez sur " EXIT " .

Quand vous cliquez sur «BEST SCORE» vous voyez une liste de 5 meilleurs joueurs(figure10).



[FIGURE 10:MEILLEUR SCORE]

3. GUIDE DU PROGRAMMEUR:

LES CAS POSSIBLES DES COMBINAISON:

L'idée principale pour réaliser ce jeu est: comment enregistrer tous les combinaison de rotation ?

Nous avons opté pour une approche simple pour trouver toutes les combinaisons possibles qui peuvent être obtenues après plusieurs rotations des lettres adjacentes de cette combinaison.

Nous avons utilisé deux structures de données de type liste. Une liste principale qui commence par la combinaison initiale, et une autre liste qui contient toutes les combinaisons que nous pouvons obtenir à partir de la combinaison initiale après une rotation de n paramètres. Après la construction de cette liste secondaire, nous passons à la vérification. Si une combinaison n'existe pas dans la liste principale, mais existe dans la liste secondaire, nous ajoutons cette combinaison à la liste principale. Et cette nouvelle combinaison pointera également vers une nouvelle liste qui contiendra toutes les combinaisons résultant de nouvelles rotations. Nous continuons ce processus jusqu'à ce que toutes les combinaisons de la liste secondaire existent déjà dans la liste principale.

3. GUIDE DU PROGRAMMEUR :

La liste des fonction et procédures utiliser en langage c de cette idée:

- Une fonction pour ajouter des combinaisons à la liste void Ajouter_Nouv(liste precedent,char combin[],char etat[]).
- Une procédure pour vérifier si une combinaison existe déjà dans la liste bool Existance(liste Tete1,liste cel).
- Une fonction pour détruire la liste void Detruire_Liste(liste Tete).
- Une fonction pour faire une rotation sur une combinaison donnée void Ajouter_Nouv(liste precedent,char combin[],char etat[]).
- Une fonction pour créer toutes les combinaisons possibles d'une combinaison donnée en effectuant des rotations void Creer_Combinaison_Cellule(liste combin,liste tete,int naj).
- Il y a aussi une fonction spécifique nommée "all_Combinaison_Cellule" qui prend en entrée une liste et un entier n. Cette fonction utilise une technique de parcours en largeur pour trouver toutes les combinaisons possibles de cette liste en effectuant des rotations de taille n. void all_Combinaison_Cellule(liste tete,int naj)

Bibliothèques utilisées

Pour accéder à des fonctionnalités supplémentaires, nous avons utilisé les bibliothèques suivantes : stdio.h, stdlib.h, string.h, stdbool.h, math.h, conio.h, time.h

MODE PLAYER :

Les action paramétrrer utiliser dans ce mode de jeux viennent de l'idée précédentes (LES CAS POSSIBLES DES COMBINAISON):

- void random_string(char * string, size_t length) est une fonction qui génère aléatoirement les combinaison.
- //Les suivantes son vue déjà dans combinaison possibles
- void all_Combinaison_Cellule(liste tete,int naj) .

- void Creer_Combinaison_Cellule(liste combin,liste tete,int naj).
- void Rotation(liste combin,int position,int naj,char newcombinaison[],char newetat[]).
- void Ajouter_Nouv(liste precedent,char combin[],char etat[]).
- void Detruire_Liste(liste Tete).

3. GUIDE DU PROGRAMMEUR :

LE CHEMIN POUR TROUVER LES COMBINAISON:

L'idée utilisé est de chercher la combinaison initiale dans les structures utiliser dans tous les cas de combinaison et l'empiler dans une pile et elle cherche sans parent et elle l'empile jus-qua que le parent = la combinaison initial. Les action paramétrer utiliser :

- void Empiler(liste *pile,char combin[],char etat[]): Cette fonction prend un pointeur vers une liste (liste chaînée) et deux chaînes de caractères (combin et etat) et ajoute une nouvelle cellule au début de la liste chaînée avec les valeurs combin et etat données.
- liste2 Recuperer_Parent(liste2 tete,liste2 child): Cette fonction prend deux listes chaînées de liste2 (tete et child) et trouve l'état parent de l'état enfant dans la liste chaînée tete.
- void Ajouter_Nouv2(liste2 precedent,char combin[],char etat[]): Cette fonction prend une liste2 (liste chaînée) et deux chaînes de caractères (combin et etat) et ajoute une nouvelle cellule à la liste chaînée avec les valeurs combin et etat données, en définissant le pointeur suiv de la cellule précédente pour pointer vers la nouvelle cellule.

MODE COMPUTER :

Ce mode de jeu est basé sur l'idée précédente (LE CHEMIN POUR TROUVER LES COMBINAISON) donc il utilise les même actions paramétrée.

LES AUTRES ACTIONS PARAMÉTRÉES UTILISÉES DANS LE JEU:

CONCERNANT LES IMAGES

- SDL_Texture* loadTexture(char* path): Chargement de textures.
- Struct image* createlimage(const char * path,int x,int y,int w,int h,int id);
: Création d'images.
- void loadMedia(int newTexture, char* path) : Chargement de médias.

CONCERNANT LE TEMPS

- void current_time() affiche l'heure et la date actuelles .

CONCERNANT LE SCORE

- Player* get_scores(Player Tab[]) renvoie le tableau de joueurs rempli d'informations .
- int leaderboard_check(Player Tab2[], int score) vérifie si un score donné peut entrer dans le tableau des scores .
- Player* update_scores(Player Tab[]) la fonction renvoie le tableau de joueurs mis à jour.

3. GUIDE DU PROGRAMMEUR:

CONCERNANT LE SAUVEGARDE

- void update_save(char *initChaine,char *finChaine,unsigned int Time). • enregistre les informations de la partie en cours dans un fichier .
int read_save(char *initChaine,char *finChaine) lit une sauvegarde de jeu à partir d'un fichier .

CONCERNANT LE TEXTE

- SDL_Surface* TTF_RenderText_Solid(TTF_Font *font, const char *text, SDL_Color fg): Cette fonction retourne une surface de rendu de police TrueType avec le texte spécifié. Le texte est rendu avec la couleur "fg" dans une surface en noir et blanc.
- SDL_Texture* SDL_CreateTextureFromSurface(SDL_Renderer *renderer, SDL_Surface *surface): Cette fonction prend une surface de rendu et crée une texture qui peut être rendue en utilisant le renderer spécifié.
- void SDL_FreeSurface(SDL_Surface *surface): Cette fonction libère la mémoire allouée pour une surface.
- void TTF_SizeText(TTF_Font *font, const char *text, int *w, int *h): Cette fonction calcule les dimensions du texte rendu en utilisant la police spécifiée.
- void SDL_RenderCopy(SDL_Renderer *renderer, SDL_Texture *texture, const SDL_Rect *srcrect, const SDL_Rect *dstrect): Cette fonction copie la texture spécifiée dans le renderer en utilisant les rectangles source et de destination spécifiés.
- MText* createText(TTF_Font* font, SDL_Color fg,SDL_Renderer* renderer,const char* text,int x,int y) renvoie un pointeur vers un objet "MText" qui contient les informations nécessaires pour afficher le texte à l'écran (texture, position, etc.)
- void drawText(struct MText* mText,SDL_Renderer *renderer) fonction qui affiche le texte à l'écran .

CONCERNANT LES BOUTON

- struct Button* createButton(const char * text,int x,int y,int w,int h,int id) crée un bouton en utilisant la bibliothèque SDL2 TTF.
- struct Button* createButton2(const char * text,int x,int y,int w,int h,int id) crée un bouton en utilisant une police .

3. GUIDE DU PROGRAMMEUR:

CONCERNANT LA SOURIE

- int checkMouseClick(SDL_Event e,struct image** image,int N)
- vérifie si le clic de la souris se trouve sur l'une des images.
- void checkMouseClickHome(SDL_Event e,struct Button** buttons) vérifie si l'utilisateur a cliqué sur un des boutons de la page d'accueil
- void checkMouseClickMenu(SDL_Event e,struct Button** buttons)vérifie si un clic de souris a eu lieu sur l'un des boutons du menu. Si un bouton est cliqué, la fonction définit le mode de jeu actuel en conséquence .
- int checkMouseClickNumbers(SDL_Event e,struct Button** buttons)retourne un entier qui représente le numéro du bouton sur lequel l'utilisateur a cliqué. Si aucun bouton n'a été cliqué, elle retourne -1.

4. CONCLUSION

Ce projet a permis de mettre en pratique les algorithmes de base appris au cours du semestre et de tirer parti de la documentation disponible en ligne pour planifier et maîtriser la bibliothèque SDL, qui était une première pour le parcours universitaire .

Références:(SQL)<http://www.codeblocks.org/downloads/binaries/>

Références:(SQL)<http://sdz.tdct.org/sdz/les-animations-optimisees-avec-sdl.html>

Références:(SQL)<http://www.gnurou.org/writing/linuxmag/sdl/partie1>

Références:(SQL)<https://www.youtube.com/watch?v=SW-iY4h3Cbs>

Références:(SQL)<http://loka.developpez.com/tutoriel/sdl/installation/codeblocks/>

Références: (IMAGES):https://drive.google.com/drive/folders/1w6_u896ol0lBXQn2GigGdGINu4QduPE-

Références: (audio):<https://drive.google.com/drive/folders/1arG9VS4axfaitu1GtBSri6CLxRHQIPSx>

RÉFÉRENCES:

Références:(SQL)<http://alexandre-laurent.developpez.com/tutoriels/sdl-2/installation-et-configuration/>

Références:(SQL)<https://openclassrooms.com/courses/apprenez-a-programmer-en-c/installation-de-la-sdl>