RAPPORT TP2

Rapport de TP

Tables des matières:

Objectifs/Enjeux

Le TP consiste à réaliser un mini jeu avec la **bibliothèque graphique MLV,** on crée des fonctions ayant un but précis comme afficher la tablette et la manger puis on les assemble pour créer le jeu. L'objectif était également de modulariser les fichiers du projet, c'est pourquoi chaque fonctions et structures sont regroupées dans des fichiers séparés en fonction de leur(s) utilité(s).

Moyens mis en œuvre

Pour le bon déroulement de ce TP, nous avons eu recours à différentes documentations :

1. La documentation de la **bibliothèque graphique MLV;**
   http://www-igm.univ-mlv.fr/~boussica/mlv/index.html

2. L'utilisation d'un fichier **Makefile**;

Une fois le programme lancé sur le terminal, on a à présent devant nous le menu principal. Nous pouvons cliquer sur « **Nouvelle partie »**, « **Partie sauvegardée »** ou encore « **Quitter ».** Il est possible de sauvegarder sa partie en cours en cliquant sur le bouton **« Sauvegarder Partie »** puis de la retrouver en choisissant « **Partie sauvegardée »** dans le menu principal. Cliquez sur la **croix de la fenêtre** ou encore sur le bouton « **Quitter** » pour fermer le jeu. Pour rejouer la ou les parties il suffit de cliquer sur le bouton « **Rejouer** » qui se présente à la fin du jeu.

Questions du TP

Exercice 4 (modularisation) :

Question 1

On suppose que notre programme sera réparti en plusieurs fichier avec un nom explicite qui permettra de regrouper les fonctions du même genre.

On a découpé notre programme en **6 fichiers .c suivi de leur fichiers .h respectifs** et pour finir le **fichier .h** regroupant les **structures** :

- **Chomp.c** correspond au fichier principal ;
- **graphique.c** regroupe la partie graphique du jeu ;
- **jeu.c** qui contient les fonctions principale pour jouer à Chomp ;
- **tableau.c** correspond à l'affichage et la création de la tablette ;
- **sauvegarde.c** à les fonctions qui servent à sauvegarder le fichier ou reprendre un fichier existant pour reprendre le jeu en cours ;
- **menu.c** concerne toute les fonctions qui permettent l'affichage et l'utilisation du menu.

Question 3

Notre ligne de commande se trouve dans le fichier **Compilation.sh** :

#!/bin/bash

gcc -ansi -pedantic -Wall -o Chomp Chomp.c -lMLV

Notre fichier **Chomp.c inclut** tous les **fichiers .c** énumérés ci-dessus.

Question 4

La deuxième version modularisée est mieux adapté pour des programme avec plusieurs lignes de codes. Cela permet une meilleur lisibilité pour mieux coder et retrouver une fonction.

Fonctions principales

- void afficher_position(Position *pos, int partie_actuel, int partie_total, Dimension dim, Score gagnant);

  Cette fonction dessine case par case l'état actuel de la Tablette contenu dans pos.Les cases sont dessiné seulement si elles n'ont pas été mangé (égal à 1). On affiche aussi le numéro de la partie en cours. Aucune valeur n'est renvoyée.

- **Coup lire_coup(Partie *game);**

  La fonction attend un clic de souris. Si le clique est sur une case déja mangé, ou à l'extérieur de la grille redemande un nouveau clic de souris. Sinon donne à coup (de Type Coup) les coordonnées de la case cliqué. On prend en paramètre la partie ouverte pour récupérer le nombre de partie total et le nombre de partie joué pour les placer dans notre fichier sauvegarde. Renvoie un coup valide de type Coup en fonction de la position donné en paramètre.

- **void affiche_gagnant(Joueur joueur_g, Score *gagnant);**

  La fonction affiche le gagnant de la partie à l'écran, en prenant le joueur en paramètre. Aucune valeur n'est renvoyée.

- **void nombre_partie(int *partie_total);**

  La fonction prend en paramètre un pointeur de type entier qui représente le nombre de partie. Demande à l'utilisateur grâce aux boxes le nombre de parties que désire chaque joueur, les valeurs sont ensuite converti en entier. Ensuite, on fait une moyenne des deux entiers et on indique aux joueurs le nombre final de manche. Aucune valeur n'est renvoyée.

- **void choix_taille(int *n, int *m);**

  Fonction qui prend en paramètre deux pointeurs de type entier.

  Demande à l'utilisateur grâce aux boxes la taille de la hauteur et largeur désirée de sa tablette de chocolat qui seront assignées aux deux pointeurs respectifs. Les valeurs entrées sont converti en entier pour par la suite les utiliser lors de la création de la tablette. Aucune valeur n'est renvoyée.

- **void choix(Partie *game);**

  Fonction qui prend en paramètre la partie actuel. Elle permet d'assigner à la largeur et hauteur de la tablette :

  - les valeurs définies 8 et 16;

  ou

  - les valeurs que le joueur a décidés.

  Aucune valeur de retour.

- **void manger(Tablette *t, int x, int y, Dimension dim);**

  La fonction modifie les valeurs de la case (x,y), deux entier donnés en paramètre, on les modifie en 0, ainsi que toutes ses cases en bas et à droite. C'est à dire les cases ayant des coordonnées égales ou supérieures à x et y. Ces modification ont lieu dans le tableau t donné grâce à la variable dim donné en paramètre. Aucune valeur n'est renvoyée.

- **int est_legal(Position *pos, Coup *coup);**

  Cette fonction prend en paramètre deux pointeurs : *pos de type Position et *coup de type Coup, elle vérifie si le coup joué est possible (si la position du coup dans la Tablette est égal à 1) ou non. Elle renvoi 1 si le coup est valide et 0 sinon.

- **int est_jeu_termine(Position *pos, Joueur *joueur_gagnant);**

  Cette fonction vérifie si le jeu est terminé en vérifiant si la case (0,0) de la Tablette est égal à 0, et évalue *joueur_gagnant par le joueur qui a gagné. La Tablette est donnée en paramètre en tant que pointeur avec le type Position et de même avec le joueur _gagnant qui donné avec le type Joueur. Elle renvoie 1 si la partie est terminé et 0 sinon.

- **int jouer_coup(Position *pos, Coup *coup, Dimension dim);**

  Le fonction jouer_coup vérifie si le coup joué est légal en faisant appel à la fonction est_legal(), si le coup est possible on utilise la fonction manger() pour modifier le contenue de la tablette à la localisation du coup. Ensuite on change le joueur qui vient de jouer le coup par l'autre joueur. La fonction prend deux pointeurs en paramètre *pos de type Position et *coup de type Coup ainsi que la taille dim de la tablette de type Dimension. Elle renvoi 1 si le coup joué est valide et 0 sinon.

- **void partie_jeu(Partie *game);**

  La fonction prend en paramètre en tant que pointeur la partie en cours. Tant que le nombre de partie donnée par les joueurs n'est pas atteint on continu le bon déroulement du jeu. A la fin du jeu on affiche le gagnant et on redemande aux joueurs s'ils veulent rejouer ou quitter la partie. Aucune valeur de retour n'est renvoyée.

- **void afficher_tab(Tablette t, Dimension dim);**

  Cette fonction sert à afficher le contenu de la Tablette tab donné en parmètre qui a pour taille les valeurs de la variable dim de type Dimension qui est en paramètre. Son utilité est seulement sur la console voir si le contenu à bien été modifié. Aucune valeur n'est renvoyée.

- **Tablette creer_tablette(Dimension dim);**

  La fonction prend en paramètre les dimension de la tablette du jeu qui va se lancer. Elle crée une nouvelle variable de type Tablette, et toutes les cases de ce Tableau seront initialisées à 1. Renvoi le tableau du type Tablette obtenue.

- **void recupereDimensionBouton(char nom[], int taille, int *largeur, int *hauteur);**

  La fonction récupère les dimensions nécessaire pour faire un rectangle en fonction du nom, de la taille, de la largeur et de la hauteur donné en paramètre. Aucune valeur n'est renvoyée.

- **void placeBouton(char nom[], int x, int y, float taille, MLV_Color couleur, Bouton *zone);**

  Cette fonction place les différents boutons sur la fenêtre en fonction de la dimension recupéré. Aucune valeur n'est renvoyée.

- **int menu();**

  Affiche le menu complet avec le nom principale et le début de partie qui peut se faire avec une partie sauvegardé ou une nouvelle partie. Aucune valeur n'est renvoyée.

Amélioration

En plus des consignes du TP, diverses fonctionnalités ont été implantées au programme. Il y a donc :

- Les joueurs choisissent un nombre de manches à jouer et cumulent des points à la fin de chacune d'entre elles. A la fin, le joueur gagnant est affiché à l'écran.

- Un fichier .txt est créé lorsqu'un joueur décide de sauvegarder, contenant les données relatives à la partie en cours. Par la suite il est possible de la charger via le menu principal.

- Via le menu des options, l'utilisateur a la possibilité de choisir le nombre de lignes et de colonnes de la tablette . La taille par fixe étant 8 * 16.

- Un seul style de couleur a été choisis, l'interface n'est pas trop chargée

Exécution du programme

L'exécutable **Chomp** se génère en exécutant le fichier le script **./Compilation.sh** mais aussi **./Chomp**. Si le fichier ne veut pas se lancer, il faut le rendre exécutable en faisant la commande **chmod +x Compilation.sh**.

Conclusion

En conclusion, ce TP nous a fait travailler sur deux points essentiels : l'utilisation de la bibliothèque graphique MLV, ainsi que sur la modularité. Le fait de découper son projet en plusieurs fichiers le rend plus organisé. De plus certains fichiers peuvent être indépendants et ainsi être utilisés dans de futurs projets.