Université Gustave Eiffel

PROGRAMMATION C AVANCEE

COMPTE RENDU DE TP

TRAVAUX PRATIQUE NUMERO : 4

UYAR ORHAN

SOMMAIRE

- INTRODUCTION

- MODEL DU PROJET

- MANUEL D'UTILISATION

- DETAIL DES DIFFERENTS MODULES

INTRODCUTION

Le but de ce TP était de programmer un jeu. L'objectif du jeu étant de résoudre un puzzle.
Ce puzzle étant crée a partir d'un image.

Le joueur possède un délai assez cours avant le début de la partir afin de mémoriser l'emplacement exacte des différent pièce. Le but étant de remettre tout les éléments a leur place.

Après ce cours instant le puzzle se retrouver mélanger. Une petite animation, permet de visualiser le bon fonctionnement du mélange, afin de ne pas brusquer les yeux de l'utilisateur.

La complexité du jeu tire principalement sa force de nombre de mélange executer.
Comme nous allons le voir par la suite, plus le puzzle est mélanger plus la complexité pour le résoudre est élève.

Quand la partie est commencer l'utilisateur dispose de son clavier et de la souris effectuer des mouvement afin d'essayer de résoudre le puzzle.
Nous verrons le fonctionne du clavier et de la souris par la suite.

A la fin du puzzle, lorsque toutes les pièces sont a leur emplacement initiale. Une animation tiré du jeu Motus commence et l'image original apparait par la suite et laisse un délai de 5 seconde a l'utilisateur pour contempler sa réussite avant la fermeture de la fenêtre automatique.

MODEL DU PROJET

Voici l'arborescence de notre projet :

```
|___ TP4
|  |___ images
|  |  |__ default.jpg
|  |  |__ flowers-1.jpg
|  |  |__ flowers-2.jpg
|  |  |__ flowers-3.jpg
|  |  |__ flowers-4.jpg
|  |___ out
|  |___ sources
|  |  |__ graphics
|  |     |__ plateau_drawer.h
|  |     |__ plateau_drawer.c
|  |  |__ utils
|  |     |__events
|  |        |__events.h
|  |        |__events.c
|  |     |__game_logic
|  |        |__game_logic.h
|  |        |__game_logic.c
|  |     |__plateau
|  |        |__plateau.h
|  |        |__plateau.c
|  |  |__ launcher.c
|  |___ Makefile
|  |___ README-TP4.pdf
```

Pour ce projet le choix a été fait de mettre en place différent module.
Tout d'abord le dossier image permet de stocker et d'être facilement accessible par l'utilisateur, a fin d'y importe plus facilement les images dans le jeu.

Le dossier out permet de stocker tout les object nécessaire a la compilation de l'executable.

Le dossier sources contient les différents module nécessaire au jeu.
Ces modules sont tout d'abord le module graphics et utils.

Le dossier graphics contient notre bibliothèque graphique nécessaire pour ce jeu.

Le dossier utils contient tout les utilitaire du jeu.
Les utilitaire du jeu se decline en plusieurs sous module qui sont events, game_logic et plateau.

Le sous module events, contient tout les fonctions qui permet de gère les différents évènement, c'est a dire rendre comprehensible tout les interactions de l'utilisateur par notre programme.

Le sous module game_logic contient tout la logic du jeu.

Le sous module plateau contient tout notre model conceptuel du plateau de jeu et des différents elements qui le constitue. Il contient aussi tout les méthode assurent leur bonne création et bonne instanciation.

Vous trouverai également un Makefile qui permet de compiler l'ensemble du projet et des modules qu'il compose.

MANUEL D'UTILISATION

Avant de tout, la compilation du projet est nécessaire. C'est pourquoi a la racine du Makefile, il vous suffit d'executer la commande « make » afin de compiler le projet.

Le « make » vas se charger de généré tout les objects nécessaire pour ce projet dans le dossier « out ». Et par la suite génère l'executable a la racine du projet.

Le nom de l'executable genre par ce makefile est « ./puzzle »

Avec le Makefile vous disposez de l'option « mrpropre » qui a pour consequence de supprimer tout les fichier généré par la compilation. Pour cela il vous suffit d'exécuter la commande « make mrproper »

Une fois le projet compiler. Vous pouvez commencer a jouer en executant l'exécutable avec la commande « ./puzzle »

Par défaut le jeu se lance avec l'image default.jpg avec un taille de 4 et 120 mélange.

Pour le lancement du puzzle vous disposer de plusieurs option de lancement afin de modifie les paramètre du jeu.
Les options disponible dans le projet sont les suivant :

Option -i : vous permet de choisir une image extérieur. Apres l'optons -i entre le chemin de l'image avec le quel vous voulez jouer. N'importe quel dimension d'image peut être prise en compte, étant donné que l'image et redimensionner dans le jeu afin de le rendre carré. Par consequent nous somme nullement responsable de la resemblance avec l'image d'origine.

Vous avez a votre disposition quelque image dans le dossier image afin de tester cet commande.

Option -s : vous permet de choisir la taille de jeu. C'est a dire le nombre de casses en verticale et en horizontale. -s devra être suivie d'un entier positif non nul.
Nous somme nullement responsable du comportement du programme en cas de non respect de ces règles.

Option -m : vous permet de choisir la complexité du jeu. C'est a dire que plus vous mélanger la carte plus il sera en théorie compliqué a résoudre, avec cet option vous choisissez le nombre de fois qu'une case est déplace aléatoirement. Par défaut cet valeur étant a 120.

Une fois la partie commencer le joueur dispose de son clavier et de sa souris pour interagir et déplacé les différents elements du jeu. Le fonctionnement du clavier est assez simple.
Par l'intermédiaire des flèches directionnel du clavier vous pouvez déplacer la case blanche. Pour la souris il faudra cliquer tout la case que l'on veut déplacer a l'emplacement de la case blanche. Donc si on click sur le carré a droite de la case blanche cela aura pour effet de déplacer celui-ci dans la case blanche.

DETAIL DES DIFFERENTS MODULE

Voici quelque détail conceptuel du projet.

Afin de réaliser ce projet nous avons mis en place deux structure. Qui nous permet de manipuler ce puzzle.
La première structure est la structure Box qui représente une boite dans le quel est stocker des coordonnée. Les coordonnée stocker dans ces Object Box représente une positon définie par un cordonnée x est y. C'est coordonné stocker dans chaque object représente l'emplacement initiale que devrai être l'objet box. En résumé chaque Box porte l'information de son emplacement d'origine.

Ces Object de type Box sont utiliser pour crée un liste dans Panel, qui est le plateau du jeu.
Le plateau de jeu est représenter par un liste de Box. Donc en résumé le puzzle est une liste de box. L'emplacement de chaque Box dans la liste nous donne l'information de sa position actuel dans le jeu.

Afin de manipuler la case « vide », il été mis en place un Object Box, a la différence des autre Box present dans le Panel celui-ci permet de stocker l'emplacement de la case noir afin de le manipuler plus facilement.
Donc a la difference des autre Box, l'object Empty_box représente les coordonnée de la boite vide et non sont emplacement d'origine. Nous utilisons donc la meme structure pour modéliser des object qui joue des rôle différents.
Cela nous permet de manipuler plus facilement un Box étant donnée de l'uni-type et d'économiser du code.

Deuxième point important dans le sujet c'est au niveau du mixer de Panel.
Méthode présente dans le fichier : « sources/utils/game_logic/game_logic.h »

Après quelque test de la premier version d'un mixer, je me suis rendu compte d'un problème liés au mouvement aléatoire, lors du mélange du puzzle, qui est le fait qu'on pouvez réaliser, le mouvement contraire au dernier mouvement effectué. Le problème ici est donc, le fait qu'on réalise des action inutile exemple le fait de faire HAUT puis BAS, ramené la case a sa position inutile, mais en meme temps nous avons réaliser deux mouvement.

Afin de corriger ce problème la méthode BKEY_mixer_logic permet de renvoyé un mouvement aléatoire qui n'est pas le mouvement opposé au dernier mouvement. Cela nous permet d'assurer le bon mélange du puzzle.