RAPPORT LABYRINTHE

Introduction

Lors de ce projet, j'ai eu pour objectif d'implémenter un jeu « Labyrinthe ».

Ce logiciel, basé sur la programmation orientée objet C++, est stocké sous forme de projet Visual Studio 2022, dont le programme principal correspond au fichier « Programme.cpp ».

LES CLASSES

1) Objet graphique

Cette classe permet d'implémenter un objet graphique : il se trouve dans le plateau, et contient des coordonnées (ligne et colonne) ainsi qu'un type (explication plus tard de cet attribut).

Elle est utilisée pour implémenter les classes dérivées « Objet graphique fixe » et « Objet graphique mobile ».

2) Objet graphique mobile

Un objet graphique mobile a la capacité de se déplacer dans le labyrinthe.

Il servira à implémenter la classe dérivée « Personnage » correspondant au personnage du plateau.

3) Personnage

Le personnage est l'objet mobile contenu dans le labyrinthe, affiché sous la forme d'un « P » dans le jeu.

On lui attribue le type 4.

4) Objet graphique fixe

Un objet graphique fixe correspond à un élément stable du plateau : un mur, un couloir ou la sortie.

5) Mur & couloir & sortie

Ces classes sont toutes dérivées de la classe « Objet graphique fixe ».

Leur type est respectivement 1, 2 et 3.

Cet attribut « type » permet d'obtenir la classe de l'objet instancié, car il n'y a pas de méthode propre en C++ pour obtenir le type d'un objet.

6) Plateau

Un plateau possède une taille : son nombre de lignes et de colonnes.

Cette classe contient également une matrice de pointeurs vers des objets graphiques fixes.

En effet, cela permet de stocker la structure du plateau du labyrinthe, tout en gardant une approche permettant d'allouer dynamiquement de la mémoire.

A l'état initial (lors de la construction de l'objet), le plateau est composé uniquement de couloir.

Cette classe possède 2 versions pour la méthode « afficher » :

- La première, ne prenant aucun argument, permet d'afficher uniquement le plateau, sans le personnage
- A l'inverse, la seconde prend en argument une référence vers un objet graphique mobile, et affiche donc le plateau ainsi que le personnage.

De plus, étant donné la présence de pointeurs dans l'implémentation de cette classe, j'ai ajouté à cette dernière un constructeur de recopie, un destructeur ainsi qu'une surcharge pour l'opérateur « = ».

7) Labyrinthe

La classe labyrinthe est la pièce finale de l'implémentation de ce jeu : elle utilise l'ensemble des classes implémentées précédemment.

Lors de l'exécution du programme principal, l'objet « Labyrinthe » sera le seul à être manipulé.

Comme le plateau, le labyrinthe possède une taille.

De plus, cette classe contient un pointeur vers un plateau et un autre vers un personnage.

En effet, le jeu du labyrinthe nécessite un plateau et un personnage pour fonctionner.

Elle possède elle aussi un constructeur de recopie, un destructeur et une surcharge de l'opérateur « = ».

Elle permet de déplacer le personnage sur le plateau, et possède une méthode « fini » permettant de savoir si la partie est terminée.

A chaque déplacement de personnage, la méthode appelée pour déplacer ce dernier vérifie si la direction choisie est valide : si aucun mur ne bloque le déplacement.

EXCEPTIONS

Lors de l'implémentation de ce jeu, 3 exceptions nécessitant d'être traitées sont apparues.

1) Erreur de saisie de coordonnées.

Lors de l'instanciation d'un objet graphique possédant des coordonnées, le programme renvoie un message d'erreur si la saisie est incorrecte (ligne et/ou colonne négative).

2) Erreur de saisie de type

Comme vu précédemment, seuls 5 types d'objets existent : 0, 1, 2, 3 et 4. Dans le cas ou le type saisie ne correspond pas à un de ces derniers, le programme renvoie donc un message d'erreur.

3) Erreur de saisie de la taille

Lorsque le labyrinthe ou le plateau est instancié avec un nombre de lignes et/ou de colonnes négatif, le message renvoie un message d'erreur pour signaler cette incohérence.

METHODES ANNEXES

Ce projet contient également les fichiers « Fonctions.h » et « Fonctions.cpp », qui permettent d'implémenter les méthodes utilisées uniquement dans le programme principal.

1) Méthode « initPlateau »

Cette méthode renvoie un pointeur vers le plateau qui va être utilisé pour la partie.

Le plateau est créé de manière arbitraire : sa configuration est tout le temps la même.

2) Méthode « initPerso »

De même, cette méthode renvoie un pointeur vers le personnage qui sera utilisé lors de la partie.

Étant donné que le plateau est toujours le même, la position de départ du personnage est elle aussi identique entre chaque partie.

3) Méthode « initLabyrinthe »

De même que les deux méthodes précédentes, cette méthode renvoie un pointeur vers un labyrinthe, qui est identique entre chaque partie car prend en argument le plateau et le personnage créé avec les deux méthodes précédentes.

FONCTIONNEMENT DU JEU

C’est vers le programme principal que convergent tous les codes implémentés précédemment.

C’est ce programme qui permet le déroulement de la partie.

Dans un premier temps, on instancie les objets qui serviront lors du jeu : le plateau, le personnage et le labyrinthe.

Ensuite commence une boucle « while », qui s’arrête dès que le personnage a atteint la sortie (condition obtenue grâce à la méthode « fini » de la classe « Labyrinthe »).

Tant que la partie n’est pas finie, le programme :

- Affiche le labyrinthe.
- Demande au joueur de choisir une direction pour déplacer le personnage.
- Effectue le déplacement en appelant la méthode correspondant à la direction choisie. Dans le cas où la saisie est invalide, le programme renvoie simplement le message « Saisie invalide », et recommence la boucle « while ».

La partie se termine donc lorsque le personnage a atteint la sortie, et c’est alors que le programme renvoie un message de fin de la partie.

DIFFICULTES RENCONTREES

Pour construire le plateau via la méthode « initPlateau », je me suis inspiré du format proposé dans la partie de l'énoncé du TP.

Pour ce faire, j'ai donc dû écrire à la main la construction de chaque élément du plateau (murs, couloirs et sortie).

Pour construire ces objets graphiques fixes, j'ai à chaque fois défini la ligne et la colonne à attribuer à l'objet, instancié un pointeur vers l'objet en spécifiant ces coordonnées, puis placé cet objet dans le plateau en appelant la méthode « setCase » de la classe « Plateau ».