Projet C++ PAC-MAN

L'objectif du projet est de programmer un clone aussi fidèle que possible du jeu Pac Man, en C++.

Il vous faudra développer votre propres classes pour modéliser le jeu, et vous adapter à une bibliothèque existante, responsable de l'affichage et des entrées/sorties : SFML.

L'évaluation du projet reposera sur 4 aspects :
- créer des classes correctement écrites, ce qui comprend des fichiers .h et .cpp, un makefile, l'encapsulation des attributs dangereux, les méthodes canoniques et d'encaspulation **quand elle sont nécessaires**
- utiliser des classes existantes provenant d'une grosse bibliothèque, qui dispose d'une documentation exhaustive.
- votre autonomie
- l'avancement global du projet

Le sujet vous propose une démarche étape par étape. En particulier, il est essentiel que le jeu commence par une modélisation claire et fonctionnelle, sur laquelle on ajoutera ensuite une couche graphique.

Pour chaque classe que vous définissez, vous devriez écrire un petit programme int main() de test, afin de valider le bon fonctionnement de la classe en dehors de toute considération graphique.

Classe Map

Une des difficultés de programmer un Pac-Man est de gérer le labyrinthe, pour autoriser/interdire les déplacements de Pac-Man et des fantômes.

Vous devriez commencer par définir des enum pour :
- les Tuile, avec pour valeurs admissible pour l'instant SOL et MUR
- les Directions, avec pour valeur admissible HAUT, BAS, GAUCHE, DROITE

La classe Map a pour attribut un tableau 2D de Tuiles, vous devriez utiliser un vector<vector<Tuile>> protégé.
Dans le jeu, tout les éléments sont calés sur cette grille : Pac-Man n'est jamais à cheval entre 2 Tuile, les Fantomes non plus, si il y a des Pac-Gums (on les implémentera plus tard), elle sont positionnées sur une case, etc.

Définissez des méthodes utiles pour la manipulation de la classe, comme par exemple :
- une méthode void charger(string filename) pour créer la Map à partir d'un fichier texte. Un exemple de fichier vous est fourni
- des accesseurs permettant de connaître les dimension de la Map
- une surcharge de l'opérateur < <

Vous serez sans doute amené à enrichir cette classe, par exemple, avec des méthodes :
- bool est_bloque(int ligne, int colonne, Direction d) qui renvoie vrai si, en partant de la case (ligne, colonne) on pourrait ou non se déplacer dans la Direction d
- int compter_sorties(int ligne, int colonne) qui compte le nombre de SOL autour d'une case de map
- etc.
Tout dépendra de vos besoins par la suite, mais il est probable que Pac-Man et les Fantome passent leur temps à faire ce genre de tests, alors regroupez les au même endroit.

Classe Personnage

La classe Personnage est destinée à servir de classe mère aux classes Pac-Man et Fantome que l'on développera par la suite.

Une Personnage est défini par les attributs protégés ligne et colonne, qui indiquent sa position sur la Map.
Il est également défini par sa Direction. Dans les règles du jeu originel, un Personnage ne change pas de direction sans raison. Si on lache le clavier, Pac-Man avance tout droit. Si un Fantome rencontre un croisement en forme de +, il ne tourne pas, etc.
Par ailleurs, une règle de déplacement pour les fantômes est de ne jamais faire un déplacement avant/arrière. Il est donc essentiel de connaître l'orientation.

Écrivez les méthodes qui vous semblent utiles pour le moment : constructeur, accesseurs, etc.

Idéalement cette méthode pourrait être abstraite, mais on va plutôt commencer par programmer un déplacement aléatoire dans le labyrinthe.

Écrire une méthode virtual void deplacer(Map); avec les règles suivantes :
- avant tout déplacement, le Personnage analyse les 4 cases autour de lui, et choisit aléatoirement une direction qui n'est pas bloquée ET qui ne le ramène pas en arrière. C'est sa nouvelle direction.
- puis il effectue un déplacement d'une case, en augmentant ou diminuant ses attributs ligne/colonne

Exemple 1 :

```
######
__P>__
######
```

Le personnage va actuellement vers la DROITE. Il analyse les directions :
- il ne peut aller ni en HAUT, ni en BAS parce qu'il y a des MUR
- il ne peut pas aller vers la GAUCHE parce que cela inverserait son mouvement
=> il continue donc obligatoirement à DROITE

Exemple 2 :

```
######
__P>__
##__##
```

Le personnage va actuellement vers la DROITE. Il analyse les directions :
- DROITE est possible, parce qu'il n'y a pas de MUR et que cela ne le ramène pas sur ses pas.
- BAS est possible, parce qu'il n'y a pas de MUR et que cela ne le ramène pas sur ses pas.
=> il tire aléatoirement une direction parmi les direction possibles : DROITE ou BAS.
Peut-être qu'il continue tout droit, peut-être qu'il tourne.

Couche graphique

Le système possède maintenant assez de richesse pour afficher quelque chose de relativement intéressant : un Personnage qui se déplace de manière autonome dans le labyrinthe.

Faites les tutoriels relatifs à Linux, l'affichage de Sprite et la gestion du clavier.

Utiliser le SFML pour afficher la Map et le Personnage, avec une boucle de jeu qui met constamment à jour la position du Personnage.

Une spritesheet libre de droit vous est fournie, elle a été créée par *drakzlin* pour le site www.OpenGameArt.org Elle contient le nécessaire pour l'affichage du jeu.
Quand ça marche pour un Personnage, essayez d'en afficher plusieurs.

Classe PacMan

Il est temps de créer Pac-Man ! C'est une classe fille de Personnage. Ce qui le différencie, c'est essentiellement les règles de déplacement : selon les touches du clavier et sans pouvoir traverser les murs.

Dans le programme principal, ajouter l'affichage de PacMan et vérifier qu'il ne rentre pas en contact avec un autre Personnages

Classe PacGum

Le but du jeu est de manger toutes les Pac-Gums dans le niveau. Modifier la classe Map pour ajouter des Pac-Gums sur toutes les cases de SOL à l'initialisation, compter le nombre de Pac-Gums en jeu, etc.
Puis gérer l'affichage des Pac-Gums. Et le ramassage des PacGums par PacMan.

Classe SuperPacGum

Dans le jeu originel, les SuperPacGums permettent à PacMan de devenir invulnérable pour un court laps de temps (vous pouvez le compter en secondes ou en nombre de frames, si vous ne voulez pas gérer un timer), et attaquer les fantômes.

Un fantome mangé est placé dans une « maison des fantômes » au centre de la carte (vous devriez rajouter un nouveau type de Tuile).

Quand ça marche, pensez à libérer les fantômes dans la maison quand Pac-Man redevient vulnérable

Classes Fantomes et règles de déplacements dans le jeu originel

Le site https://gameinternals.com/understanding-pac-man-ghost-behavior propose un article extrêmement détaillé sur les mouvements des Fantomes.

Classe Fantome

Fantome hérite de Personnage et définit les règles génériques de déplacement vers une case donnée (c.f. l'article de Games Internals), et les règles de déplacement en cas de fuite quand PacMan est invulnérable, et

Modifier l'affichage en conséquence.

Classes Blinky, Pinky, Inky et Clyde

Créez ces 4 classes, héritant de Fantome, afin de redéfinir pour chaque type de Fantome la selection de la case à atteindre. Ceci modifiera son comportement pour se déplacer.

Modifier l'affichage en conséquence.