

**Introduction**

Le projet Pac-Man est un projet réalisé en C++, il utilise OpenGL pour la gestion de l’affichage Graphique. Le choix de l’univers étant libre, celui de ce projet est Ms. Pac-Man.

Lorsque le jeu se lance, l’utilisateur se retrouve sur un menu où il doit appuyer sur la touche entrée, ensuite il peut consulter les commandes du jeux (touche C) ou lancer le jeu en choisissant entre deux niveaux.

**I) Fonctionnement de l’application**

Le jeu consiste à déplacer Ms. Pac-Man à l’intérieur d’un labyrinthe, afin de lui faire manger toutes les Pac-gommes qui s’y trouvent en évitant d’être touché par des [fantômes](https://fr.wikipedia.org/wiki/Fant%C3%B4me). Au lancement du jeu, le joueur possède trois vies, qu’il pourra perdre lorsqu’il rencontre un fantôme. Le jeu se termine alors quand le joueur à perdu ses trois vies ou bien lorsque le joueur à manger toutes les Pac-gommes et rejoint la case de sortie. Cette dernière apparaît de manière aléatoire une fois que toutes les Pac-gommes ont été mangés.

1. **Déplacements**

Ms. Pac-Man est contrôlé par le joueur, il peut se déplacer dans quatre directions : haut, bas, gauche, droite avec les touches directionnelles du clavier. Ainsi on se déplace de case en case et l’avatar s’affiche en fonction de la direction sélectionnée.

Les ennemis se déplacent de façon aléatoire dans le labyrinthe. Cependant, ils n’ont pas la possibilité de revenir sur leurs pas (faire demi-tour). A chaque itération, l’algorithme vérifie les possibilités de déplacement en fonction de la position de l’ennemi puis en choisit une au hasard parmi celle qui sont possible.

1. **Gestion du score et des vies**

Le score s’incrémente lorsque l’on mange une Pac-gomme (+ 10 points), une super Pac-gomme (+ 50 points) ou un fantôme (+200 points multipliés par le nombre de fantôme mangé). Il existe deux modes dans le jeu, un mode « normal » ou Ms. Pac-Man doit fuir les fantômes et un mode « spécial » ou les fantômes peuvent être mangés.

Le joueur possède 3 vies et n’a pas la possibilité d’en récupérer. Il perd une vie à chaque fois qu’il rencontre un fantôme sur une case. Lorsqu’il perd une vie, les personnages (ennemi et joueur) retourne à leurs positions initiales.

1. **Gestion des textures et Graphisme**

Pour les textures la bibliothèque SOIL a été utilisé. Toutes les textures du jeu sont générées à partir d’image .png et stockés dans un tableau dynamique. Ce dernier nous permet de récupérer les textures afin de les afficher. Pour afficher le Pac-Man dans les quatre directions, il y a qu’une seule image que l’on tourne lors de l’affichage. Les fantômes possèdent une texture en fonction de leurs noms passés au constructeur et change de texture lorsqu’on entre dans le mode spécial pour que l’utilisateur identifie dans quel mode il se trouve.

**II) Structure du projet**

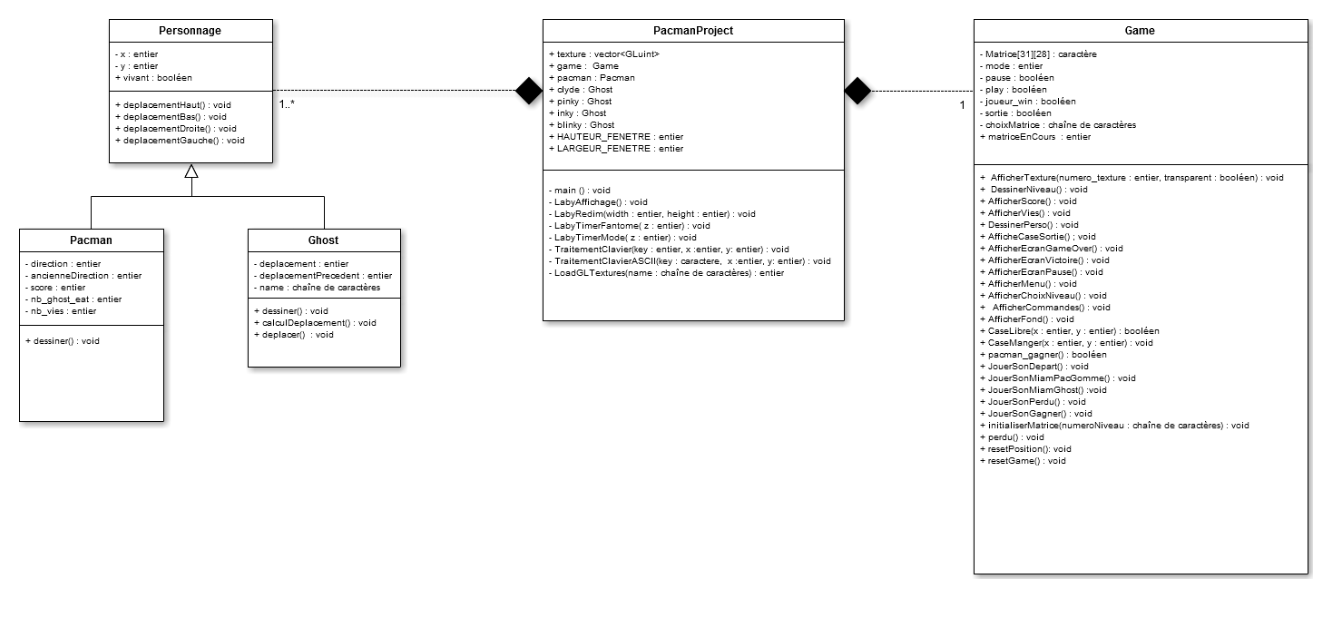


Figure 1 : Diagramme de classe simplifié de l'application

Le fichier PacmanProject.cpp est le point d’entrée de l’application, c’est ici que le programme ce lance grâce à la fonction main() qui lance la boucle glut, cette boucle tourne à l’infini tant que l’on a pas quitter le programme. C’est également dans ce fichier que l’on gère le chargement des textures.

On gère aussi Ce qui s’affiche dans la fenêtre (LabyAffichage) et le redimensionnement de la fenêtre (LabyRedim).

La classe Game contient la plupart des fonctionnalités du jeu. C’est donc la classe principale de l’application. Elle contient plusieurs booléens (play, pause, joueur\_Win et sortie) permettant de mettre à jour l’affichage en fonction de l’état du jeu. Les méthodes d’affichage sont appelées dans PacmanProject. Il y a aussi la fonction CaseLibre() qui renvoi si l’on peut aller sur une case ou non.

La classe Personnage représente un personnage dans le jeu avec ses coordonnées, s’en dérive donc la classe Pac-Man qui correspond au joueur et la classe Ghost correspondant aux ennemis du jeu.

Amélioration possible de l’architecture : Afin d’améliorer la clarté du code, il faudrait séparer les fonctions d’affichages du reste. Au lieu de les avoir dans la classe Game, on devrait créer une classe Affichage.

**III) Explication des solutions retenues pour le rendu graphique**

Pour l’interface, il y a plusieurs Viewport servant à :

* Afficher le fond du jeu : le fond est une texture mesurant la taille de la fenêtre et encadrant la zone de jeu.
* Afficher le score : pour afficher le score, il a fallu convertir l’entier en chaîne de caractère et utiliser une fonction pour séparer la chaine car OpenGl affiche seulement caractère par caractère.
* Afficher les vies restantes : on créer un Viewport affichant notre icone pour chaque vie non-utilisée.
* Afficher la zone de jeu. Celui-ci contient les menus, les niveaux et nos personnages. La texture de fond correspond à notre matrice « text » (Cette méthode n’est pas très adaptive, en effet, pour créer un bon éditeur de niveaux ou ajouter plusieurs niveaux, il faudrait générer le fond en fonction de ce qui se trouve dans notre matrice). Les personnages sont affichés au centre de la case où ils se trouvent.

L’affichage est fenêtré et se redimensionne bien grâce à la fonction LabyRedim, ce qui permet à l’utilisateur de choisir la taille de la fenêtre mais aussi de garantir la compatibilité lorsqu’on joue sur un autre écran.