Adrien Patrizio
Thibault Guichard
Pierre Franc

Pacman 3D

Synthèse d'Image 2

A. Mode d'emploi

Il s'agit d'un jeu de Pacman classique.

Le joueur contrôle pacman par les quatre touches 'z', 'q', 's' et 'd', qui correspondent respectivement aux directions "haut", "gauche", "bas" et "droite". Dès que le joueur appuie sur une de ces touches, le jeu démarre et Pacman avance tout seul jusqu'à ce que le joueur appuie sur une autre touche ou qu'il rencontre un mur. Dans ce dernier cas, Pacman s'arrête jusqu'à ce que l'utilisateur utilise une touche pour le faire changer de direction, et s'il n'y a pas un mur dans cette direction, il va la prendre, et ainsi de suite.

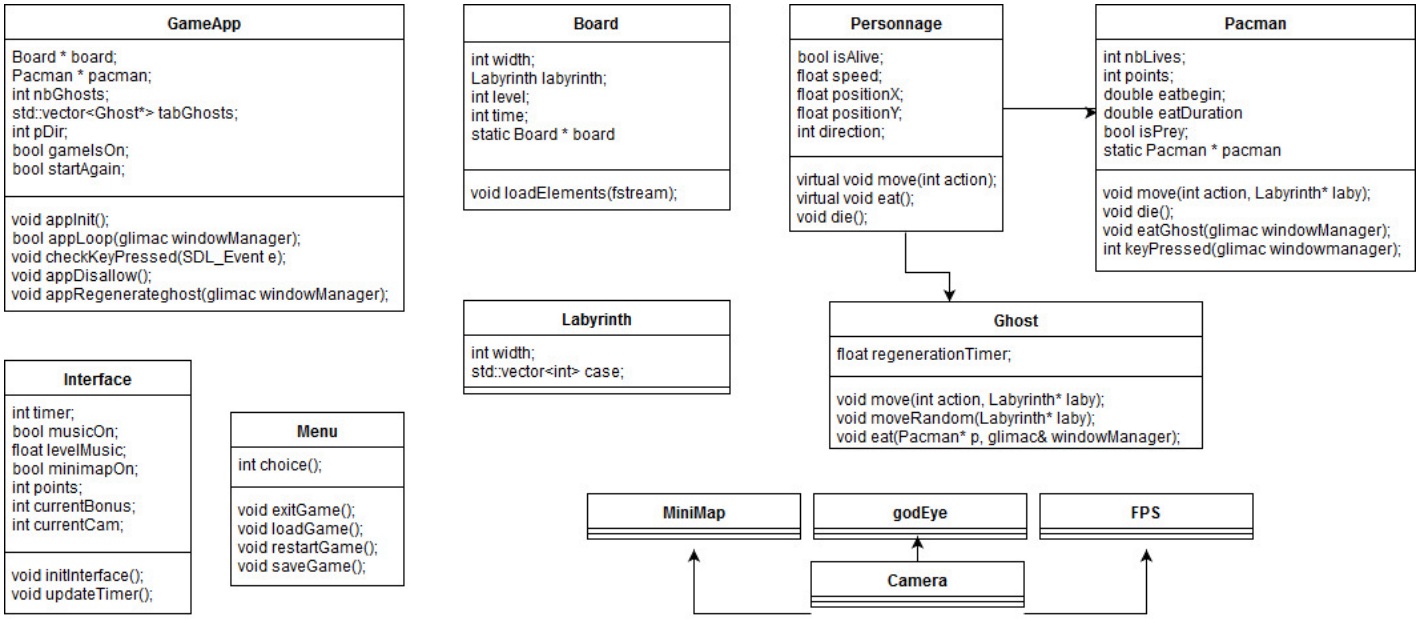
Les fantômes bougent aléatoirement, et tuent Pacman lorsqu'ils le croisent. Quand cela arrive, s'il reste des vies à Pacman, celui-ci reprend à son point de départ, sans perdre les points qu'il avait amassés jusqu'alors. S'il n'a plus de vies, la partie est perdue. Une nouvelle partie se lance alors automatiquement.

Sur tout le chemin du labyrinthe sont disposés des pacgommes, qui font gagner 100 points à chaque fois que Pacman passe sur l'un d'entre eux, et le fait disparaître. Il existe aussi quelques super pacgommes qui rapportent 500 points et inversent le jeu : pendant 20s, c'est Pacman qui chasse les fantômes. Plus il en tue, plus cela lui fait gagner des points.

Enfin, dans le labyrinthe, il existe une paire de portails de téléportation. Lorsque pacman passe par l'un d'eux, il se téléporte à la sortie de l'autre. L'utilisateur peut récupérer 3 bonus : une vie en plus, une augmentation de la durée *glouton* et une augmentation de la vitesse

À tout moment, il est possible de sortir du jeu en appuyant sur la touche [espace] ou de mettre pause en appuyant sur la touche 'p'..

B. Architecture



C'est la classe `GameApp` qui contient la boucle du jeu. Au tout début du code, un objet `GameApp app` est créé, et 2 de ses méthodes sont appelées : `appInit();` et `appLoop();` qui contiennent l'essentiel du moteur de jeu.

La première chose que fait `appInit();` est d'appeler l'instance statique de la classe `Board`, qui est un singleton. Elle va commencer par initialiser son attribut `Labyrinth` grâce à un vecteur d'entiers représentants les différents états possibles de ses cases (0 pour un chemin avec pacgomme, 1 pour un mur, 2 pour un chemin vide...).

Une fois que le board est créé, `appInit()` crée un Pacman, qui est un enfant de la classe Personnage au même titre que les fantômes. Ainsi, les deux classes Pacman et Ghost dérivent de Personnage, en redéfinissant notamment sa méthode `move()`. Ils héritent donc de tous ses attributs parmi lesquels la position, la direction, la vitesse et divers booléens d'état.

Par ailleurs, nous avions prévu de créer une classe caméra dont descendrait trois classes correspondant à chacune des trois caméras disponibles : la caméra godEye qui voit tout du dessus et pivote de 90° lorsque Pacman passe à travers un portail, la caméra FPS, et optionnellement une caméra de minimap à afficher dans la vue FPS qui aurait pour particularité d'être toujours orientée vers le bas, mais relative à Pacman et à ses déplacements, rotations comprises)

C. Choix

Nous avons choisi de modifier un peu le comportement des portails par rapport au jeu de base. En effet, dans notre version, un portail ne mène pas au portail du côté opposé, mais à celui de l'un des côtés adjacents. Dans l'idée, cela devait déboucher sur un effet 3D de pivot de la caméra à 90° afin de correspondre à la nouvelle trajectoire. Nous avons fait ce choix afin de rajouter du sens à la contrainte de développer ce Pacman en 3D et non en 2D classique (en plus de la caméra FPS). Cela permet également d'ajouter une touche d'originalité (même s'il est certain que cette variation a déjà été faite dans des quantités de versions dérivées de Pacman).

D'un point de vue plus structurel, nous avons choisi d'appliquer le design pattern Singleton à la classe Board, car il nous semblait logique que cet objet n'ait toujours qu'une et une seule instance (comme l'aurait un plateau physique si c'était un jeu de table). Nous avons également hésité à faire de Pacman un singleton aussi, mais nous avons choisi de ne pas le faire pour deux raisons : tout d'abord cela ne faisait pas sens par rapport à la fonctionnalité de relance du jeu une fois la partie terminée. En effet, comme un singleton ne peut par définition pas être détruit, il ne semble pas logique de garder toujours une et une seule instance de Pacman alors que nous supprimons et recréons des fantômes ou d'autres éléments du plateau, à chaque début de partie. Ensuite, dans le cas ou nous aurions souhaité implémenter une fonctionnalité multijoueur, le singleton aurait tout simplement bloqué le concept, car il aurait été impossible d'avoir plusieurs Pacman sur le

plateau. Nous avons hésité à définir la classe Renderer à base de Singleton, mais ce n'était pas forcément une bonne idée, car il peut y avoir deux Rendu différents (en VR par exemple)

Par ailleurs, nous avons créé une classe abstraite Personnage, car il nous semblait clair que Pacman et les fantômes allaient avoir certains comportements et surtout certains attributs similaires.

D. Gestion de projet

Nous avons rencontré de nombreuses difficultés au cours de ce projet.

La première a été, il faut l'admettre, notre gestion du planning. Bien que nous nous soyons réunis dès l'annonce du sujet pour discuter du schéma UML et de la gestion de projet, il nous a fallu du temps pour réellement commencer à coder, et cela nous a desservi, car plus le temps passe au cours d'un semestre, et plus les projets s'accumulent, laissant de moins en moins de temps pour chacun. Sans surprise, le gros du code a donc été réalisé dans les deux semaines précédant le rendu, ce qui n'est pas vraiment la meilleure façon de procéder.

Cependant, un second problème est venu s'ajouter à cela, et il a nettement compliqué les choses. En effet, nous avons perdu un temps considérable à tenter de gérer l'aspect crossplatform du projet. Car il se trouve qu'au sein du groupe, il y a des utilisateurs de Mac et d'autres de Windows. Nous souhaitions donc naturellement être capable de coder séparément sur le projet, et que le code fonctionne de la même manière sur n'importe laquelle de nos machines. Mais malgré des heures passées sur cette affaire, seuls ou aidés de gens connaissant bien mieux le sujet que nous, nous n'avons jamais réussi à régler ce problème. Nous avons donc dû nous résoudre à coder totalement séparément le moteur de jeu et le moteur de rendu, voire même à certains moments à coder à l'aveugle, sans pouvoir compiler. Sans cela, nous aurions pu nous concentrer davantage sur le code même, et ainsi livrer un projet bien plus abouti.

Par ailleurs, on peut noter que cette complication n'a pas seulement un mauvais effet sur le projet lui-même en le retardant, mais également sur sa composante humaine. En effet, il est particulièrement frustrant de chercher à résoudre des problèmes pour pouvoir seulement être capable de commencer le projet, et d'ainsi voir tomber de nombreuses heures perdues et autant de fonctionnalités écartées. Cela a de toute évidence un mauvais impact sur la motivation et sur le moral.

Nous avons néanmoins fini par réussir à dépasser ce souci, non pas en le réglant, mais en le contournant, car il n'y avait plus d'autres choix, nous ne pouvions plus nous permettre de perdre du temps - et vu le résultat, nous aurions même sans doute dû tenter de contourner le problème plus tôt.

Nous nous sommes également confrontés à une autre petite difficulté d'ordre organisationnelle. En effet, bien que ce soit possible grâce aux outils tels que Git, il est souvent plus compliqué de travailler à plusieurs en même temps sur la même portion de code. Pour éviter ce problème, nous nous sommes à certains moments retrouvés à devoir

attendre qu'un autre push ses modifications et fasse une pause pour nous y mettre de notre côté, ce qui n'est vraiment pas optimal. Nous avions pourtant bien séparé le code en tâches courtes, intégrables et testables rapidement, mais nous avons fait l'erreur de ne pas faire attention à leur indépendance. Heureusement, le logiciel CLion comporte un excellent outil de Merge qui nous a fortement aidé.

Au niveau de la gestion de projet, nous avons comme dit précédemment commencé par nous réunir afin d'établir le diagramme des classes du projet, d'établir un planning et de séparer les tâches.

Pour cela, nous avons utilisé deux outils en lignes : Draw.io pour le diagramme de classe, et Trello pour la gestion de projet à proprement parler.

Nous avons séparé les tâches en petites unités facilement vérifiables, et les avons rapidement réparties en fonction des affinités et des compétences de chacun, voyant que le délai était trop court pour que l'on puisse tous toucher à tout comme nous le souhaitions à l'origine.

Adrien a géré le moteur de rendu, tandis que Pierre et Thibault se sont occupés du moteur de jeu.

Pour finir sur une note positive, nous nous félicitons tout de même d'avoir passé un certain temps à réfléchir à la structure au début du projet, car une fois les classes créées et munies de leurs getters et setters, cela nous a permis de coder le moteur de jeu relativement rapidement et efficacement sans être confronté à de gros problèmes de structuration.

E. Résultat

Le résultat n'est certes pas à la hauteur de ce que nous avions en tête (l'est-il jamais ?), mais nous sommes tout de même satisfait de l'avancée du projet aux vues des soucis que nous avons rencontrés. La grosse déception est de ne pas avoir d'interface openGL. C'était pourtant proche, mais trop de complications sont venues s'ajouter.

Il y aurait bien sûr de nombreuses améliorations possibles, à commencer par finir le moteur de rendu donc. Une fois ceci fait, nous envisagerions en premier lieu la création d'un menu et d'une interface.

Parmi les améliorations que nous aurions souhaitées implémenter avec plus de temps, il y a notamment la possibilité de générer de manière procédurale le labyrinthe, et également la création d'un plateau à étage multiple, contenant des rampes d'accès d'un niveau à l'autre. Cela aurait constitué une évolution intéressante du jeu, et aurait renforcé et justifié l'aspect 3D du projet.

Enfin, dans les améliorations plus basiques car s'inspirant directement du jeu de base, il aurait pu être intéressant de donner des comportements plus recherchés aux fantômes. grâce à de l'Intelligence Artificielle forte