Dimanche 7 Mai 2023

Université de Strasbourg

Projet Programmation Avancée
Réalisation du jeu PAC-MAN (Namco 1980)

Contributeurs :
Jardot Charles
Derousseaux Nathanaël

Introduction :

Pour mener à bien le projet PAC-MAN, nous avons décidé de nous organiser de la manière suivante : Nathanaël Derousseaux organise la structure du projet, l'affichage et les collisions au niveau des différentes instances. Charles Jardot s'est occupé de la gestion des fantômes et a également participé à la structure du projet.

En ce qui concerne les attendus du projet, la jouabilité, la gestion des fantômes ainsi que les différentes animations (fin de niveau, déplacements, ...) ont bien été implémentées. La gestion du score n'a pas été quant à elle implémentée.

Ce rapport s'articule en deux temps. D'abord, nous présenterons les différents choix de développement du jeu puis, nous pourrons conclure avec les parties fantômes et structures sur lesquelles j'ai personnellement travaillé durant ce projet.

I / Choix d'implémentation

Pour gérer les différents objets sur affichés sur la surface SDL, nous avons décidé de créer un objet Window, qui a comme attribut un container. Ce container est une liste d'objets éléments (ou classes hérités). On parcourt la liste à chaque itération du jeu, et on appelle pour chaque instance d'élément la fonction update. Lors de l'appel de cette fonction, l'élément réagira comme il doit : PAC-MAN bougera selon les inputs claviers, changera son sprite, etc…, les fantômes exécuteront leur algorithme de chasse et le terrain gérera les collisions avec les PAC-GOMMES. Un tour de jeu dure 16ms. Nous avons décidé de donner à chaque objet movable (classe héritée d'élément), la possibilité de se déplacer de X pixels par tour. PAC-MAN peut bouger de 4 pixels par tour, et les fantômes de 3 pixels en mode CHASE.

II / Tâches personnelles

2.1) Réflexion autour de la structure de déplacement

En ce qui concerne la structure de déplacement, il fallait trouver une méthode à la fois optimale et permettant à Pacman de ne pas traverser les différents obstacles qu'il rencontrait. Nous avons donc opté pour un système de "rail" où PAC-MAN peut se déplacer uniquement d'intersections en intersections. Nous avons ainsi défini 64 intersections de "déplacements" et 5 intersections de "position initiale".

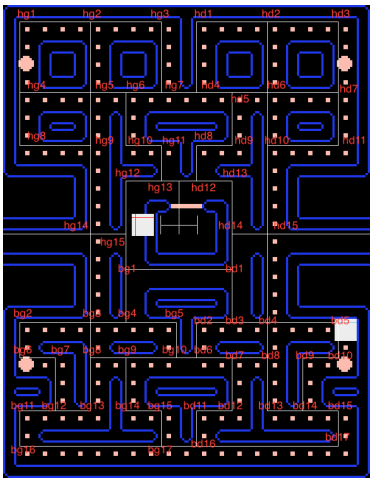

Figure 1 : Représentation des différentes intersections

A partir d'ici, j'ai défini un système de "directions" afin d'attribuer à chaque intersection des voisines (top, bottom, right et left). Si nous nous trouvons dans le cas de bg16 (bas gauche 16), nous remarquons qu'elle n'aura pas de voisines de gauche ou de bas, il y a aura donc une attribution d'un *nullptr*, qui constitue un critère pour les instances pour pouvoir se déplacer par la suite.

2.2) Fantômes

Durant ce projet, ma mission majeure fut de régler l'intelligence et la gestion des fantômes. Pour ce faire, notre choix concernant l'ia à implémenter se reposa sur le site web de Chad Birch [1] décrivant avec précision le système que nous avons reproduit. Ainsi, nous avons divisé les tâches en quatre classes distinctes : **Blinky** (fantôme rouge), **Clyde** (fantôme orange), **Pinky** (fantôme rose) et **Inky** (fantôme bleu). Ces différentes classes sont héritées de la classe Fantom, elle-même héritant de Moveable puis Element. Intéressons nous dès à présent aux différentes programmations des modes CHASE (chasse de PAC-MAN) des fantômes.

2.21) Le fantôme Blinky

Blinky a un objectif précis : il vise PAC-MAN. Ainsi, j'ai décidé de mettre en place un système de calcul de distances euclidiennes afin d'obtenir le chemin vers sa cible.
Ainsi comme expliqué avec notre système d'intersection, Blinky regarde les directions accessibles au moment présent, puis avec le calcul de distance, on "simule" à la prochaine intersection quelle serait la direction la plus courte pour lui. Il décide alors de choisir ce mouvement afin de se rapprocher de sa target. C'est l'algorithme "de base" au niveau des fantômes.

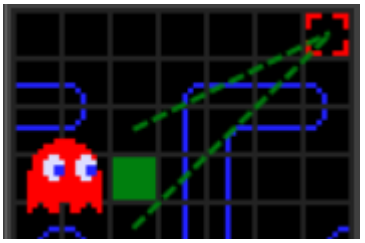

Figure 2 : Illustration d'un choix pour Blinky

2.22) Le fantôme Clyde

Clyde est un cousin de Blinky. En effet, ce dernier a un comportement complètement similaire à Blinky. A l'exception près que ce dernier est un couard. Lorsqu'il se situe à moins de 8 tiles (ici calculé avec les dimensions de notre jeu (zoom et taille) à environ 188 pixels), ce dernier file en direction de son objectif en mode "Scatter" (que nous verrons par la suite dans le séquençage).

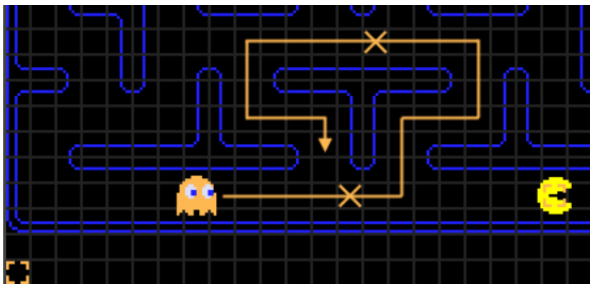

Figure 3 : Les deux cibles de Clyde

Ici dans un premier temps Clyde pourchasse PAC-MAN, arrivé à la première croix, il dévie pour se diriger vers le carré orange avant de s'éloigner loin de PAC-MAN l'obligeant à prendre son courage pour repartir vers PAC-MAN.

2.23) Le fantôme Pinky

Pinky est différent des autres fantômes précédents. Sa cible n'est jamais PAC-MAN. En effet, il tente d'embusquer ce dernier en choisissant comme cible le pixel se situant 4 tiles devant PAC-MAN (soit 96 pixels dans notre projet). J'ai décidé de suivre les recommandations de notre source et donc d'implémenter le "bug" lié à Pinky. Si le joueur choisit d'emmener PAC-MAN vers le haut alors Pinky aura pour cible 4 tiles en diagonale à gauche de PAC-MAN (au lieu de 4 tiles vers le haut directement).

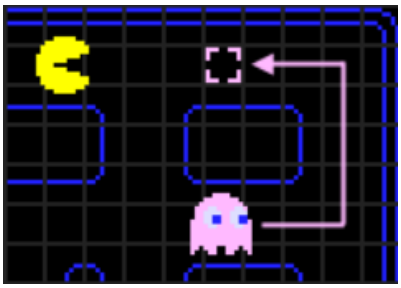

Figure 4 : Cas classique

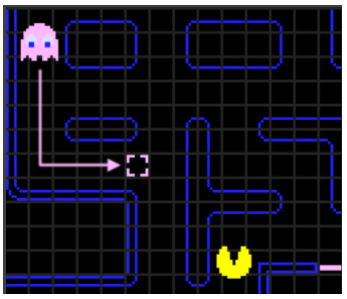

Figure 5 : Bug illustré

2.24) Le fantôme Inky

Inky reprend la mécanique de Pinky. Nous configurons 2 tiles (soit 48 pixels) devant PAC-MAN (dans la direction que ce dernier regarde). Puis un vecteur est tracé jusqu'à la position de Blinky. Ce dernier est alors doublé avec comme base Blinky donnant un point symétrique à la position de ce dernier. Ainsi, Inky est comme Pinky un fantôme cherchant l'embuscade.

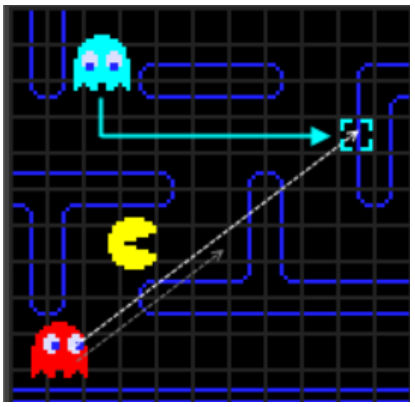

Figure 6 : Illustration de la stratégie d'Inky

2.25) Mode FRIGHTENED et mode EATEN

En plus du mode CHASE vu ici, nous avons 2 modes supplémentaires et essentiels : le mode FRIGHTENED et le EATEN.

Le premier correspond au mode où PAC-MAN dévore une PAC-GOMME. Les fantômes s'enfuient alors dans une direction aléatoire avec une vitesse diminuée de 1.

Le deuxième mode est le mode "mangé", les fantômes partent au point de spawn afin de redevenir menaçant. Leur vitesse est ici doublée. Cependant, nous n'avons pas réussi à faire revenir les fantômes à leur point exact de spawn (leur propre INITIAL_X et INITIAL_Y), c'est pourquoi nous avons décidé de considérer le spawn de Blinky, seul fantôme en dehors de la cage, comme point de reboot.

2.3) Séquençage des phases de Gameplay

Enfin, j'ai eu en charge la programmation du modèle de séquences du jeu. En effet, dans PAC-MAN, les fantômes ne font pas que suivre leur objectif. Il existe alors un quatrième mode appelé le mode SCATTER. Ce dernier a pour but de "soulager" PAC-MAN car les fantômes ont chacun leur cible située dans chaque angle du terrain. Ainsi, on assiste à 8 phases différentes pour le niveau 1 reproduit ici. Le jeu alterne successivement les phases suivantes : 7s SCATTER, 20s CHASE, deux fois d'affilée, puis 5s SCATTER, 20s CHASE, 5s SCATTER avant de finir définitivement en CHASE jusqu'au bout du jeu.

Les phases ont bien été implémentées cependant, la feature de stopper le temps d'écoulement des phases durant une pause ou durant le mode FRIGHTENED n'est pas mise en place.

III / Bibliographie

[1] Chad Birch, Understanding Pac-Man Ghost Behavior 2010, https://gameinternals.com/understanding-pac-man-ghost-behavior

[2] Wiki de la librairie SDL2 , https://wiki.libsdl.org/SDL2/FrontPage