Rapport d'IN104 : Pacman

Thomas BOYER et Quentin KREMPP

22 mai 2019

Table des matières

1 Présentation du projet

L'objectif du projet était de reproduire le plus fidèlement possible le classique jeu de Pacman, avec toute liberté sur le langage et les structures de code à utiliser.

Le projet est disponible à l'adresse suivante : `https://github.com/Flying-Cucumber/PacMan`

2 Choix du langage et des structures

Le choix du langage étant libre, nous avons décidé de nous tourner vers le langage C, appris lors des cours d'IN102 et IN103, son caractère bas niveau permettant d'une part d'être plus proche des contraintes d'origine du jeu, et d'autre part une meilleure optimisation du code, notamment du point de vue de la mémoire occupée. De plus, les structures vues en IN103 permettent d'avoir une approche semblable à la programmation orientée objet vue en cours d'IN104, et semblait ici particulièrement adaptée à la situation.

2.1 Le terrain

2.1.1 Description

Pour représenter le terrain, nous avons choisi de le diviser en cases (que l'on a appelé "slabs", c'est à dire des "dalles"). Le terrain est donc une structure qui mémorise l'adresse d'une case initiale à partir de laquelle on va construire toutes les autres. Lors de l'initialisation, il va aussi retenir en mémoire un pointeur vers la case d'apparition de Pacman et vers la case d'apparition des fantômes. De plus, deux entiers représentent le nombre de cases en hauteur et en largeur du terrain, pour des commodités d'affichage. Ils sont initialisés après la création du terrain.

L'inconvénient

2.1.2 Le code

```
struct terrain{
    int size_x;
    int size_y;
    struct slab* initial_slab;
    struct slab* spawn_slab;
    struct slab* ghost_house;
};
```

2.2 Les dalles (slabs)

2.2.1 Description

Les entités (Pacman, les fantômes et les fruits) devant se déplacer à l'écran, et interagir avec ce qu'ils rencontrent, il était commode de fonctionner par cases, chacune ayant un type déterminant ses interactions avec les entités (par exemple les pac-gum ou super pac-gum). De plus, chaque case a en mémoire l'adresse des quatres cases adjascentes, afin de naviguer facilement dans les quatres directions. De plus, chaque case a en mémoire ses coordonnées et

deux pointeurs vers des objets de type SDL-Surface (structures SDL servant à l'affichage)

2.2.2 Le code

```
struct slab{
    struct slab* up;
    struct slab* down;
    struct slab* left;
    struct slab* right;
    int type;
    int x;
    int y;
    SDL_Surface* affichage;
    SDL_Surface* objet;
};
```

2.3 Les entités

2.3.1 Description

L'entité est une sous structure commune à toutes les entités, c'est à dire, d'un point de vue orienté objet, une super classe. Elle a en mémoire la case sur laquelle elle est, et la direction dans laquelle elle va, qui sera actualisée par le programme. Elle a de plus en mémoire, comme les cases, un pointeur vers un objet SDL-Surface pour l'affichage. Les différentes entités (Pacman, les fantômes et les fruits) héritant de cette structure, ils ne seront pas détaillés ici.

2.3.2 Le code

```
struct entity{
    struct slab* current_slab;
    int dir;
    SDL_Surface* affichage;
};
```

3 Le choix de la librairie graphique

3.1 Justification du choix

Nous avons choisi d'utiliser pour l'affichage graphique la librairie SDL, très populaire et donc bien documentée afin de trouver facilement des réponses

aux éventuelles questions que l'on pourrait se poser. De plus, elle dispose de tutoriels d'utilisation détaillés, notamment celui d'OpenClassrooms.

3.2 Problèmes rencontrés lié au choix de la SDL

L'inconvénient d'une telle librairie est qu'elle est de très bas niveau, ne permettant d'afficher que des rectangles et des images. C'est pourquoi nous n'avons pas eu le temps d'implémenter l'affichage d'images et encore moins des animations. La SDL ne pouvant pas afficher de chaînes de caractère, nous n'avons pas non plus eu le temps d'afficher le score du joueur.

4 Répartition du code au cours du projet

Echéance	Thomas BOYER	Quentin KREMPP
Semaine 1	Créations des structures des entités	Création des structures du terrain
Semaine 2	Fonctions d'initialisation des entités et intégration au terrain	Fonctions d'initialisation du terrain
Semaine 3	Fonctions de gestion des collisions entre entités	Fonctions de déplacement des entités et interctions avec le terrain
Semaine 4	Ré-agencement du code, commentaires et corrections. Intelligence artificielle des fantômes	Documentation sur le fonctionnement de la librairie SDL
Semaine 5	Implémentation des déplacements d'un point de vue graphique, gestion des événements clavier	Implémentation de l'initialisation graphique de l'ensemble du jeu
Semaine 6	Gestion de la fin de niveau	Affichage du score

5 Difficultés rencontrées

1. La SDL étant bas niveau et difficile à prendre en main, nous avons passé plus de temps que prévu à la maîtriser et n'avons, par conséquent, pas pu nous occuper des animations et de affichages de texte.
2. La programmation bas niveau, notamment dans l'utilisateur entraîne de nombreuses erreurs, notamment de segmentation qui sont difficiles à débugger car n'étant pas relevées à la compilation et entraînant un crash immédiat du programme. Cela nous a beaucoup ralenti lors des semaines 2 et 3

3. La gestion des événements clavier nécessitant une structure de code particulière, nous avons étés forcés de restructurer notre code en profondeur lors de la semaine 5