Projet C++ : Yellow

Pacman

KO Roqyun

CHEN Pascal

Table des matières

Introduction

Dans le cadre du projet C++ du thème « Yellow » en EISE4 (Electronique-Informtique Systèmes Embaqués en 4ème année), nous avons réalisé le jeu Pacman.

L'objetif de ce projet est de se familiariser avec le codage en C++ et d'apprendre la programmation orientée objet.

Description de l'application

Pacman

Le jeu vidéo Pacman est crée par Toru Iwatani pour l'entreprise Namco et il est sorti en 1980. Le but du jeu est de déplacer une personnage jaune « pacman » dans un labyrithe et de lui faire manger des pac-gommes. Il y a 3 types d'items : une pac-gomme petite, une super pac-gomme et fruit.

Une pac-gomme donne 10 points. Une super pac-gomme donne 50 points et une super pouvoir temporaire. Des fruits donnent beaucoup de points. Un fruit s'apparaît dans le jeu à un moment aléatoire.

Il y a aussi 4 fantômes qui essaie d'attraper Pacman. Chacun suit Pacman avec son propre algorithme. Par exemple,

- le fantôme rouge est un suiveur de Pacman.
- Le fantôme rose tend une embuscade à Pacman au lieu de le suivre comme le fantôme rouge.
- Le mouvement du fantôme cyan dépend du fantôme rouge.
- Le mouvement du fantôme orange est quasi-aléatoire.

Leurs algorithmes sont plus compliqué que les explications fournies mais elles décrivent leurs comportements principaux. Il y a aussi un mode « scatter ». Pacman mange une super pac-gomme, les fantômes devient vulnérable. Alors, ils s'enfuient. Les fantômes sauf le rouge doivent attendre dans la maison des fantôme au centre pendant une durée aléatoire. Ils peuvent pas suivre Pacman tout de suite dès une partie commence.

Quand ils touche Pacman, il meurt. Pacman possède 3 vies au début mais on peut incrémenter le nombre de vies en mangeant assez de gommes.

Le jeu a 255 niveaux en totale.

Les sprites ci-dessous sont utilisé dans nos projets :

Illustration 1: Pacman;

Illustration 2: Fantôme rouge : Blinky

Illustration 3: Fantôme orange : Clyde

Illustration 4: Fantôme cyan : Inkey

Illustration 5: Fantôme rose : Pinky

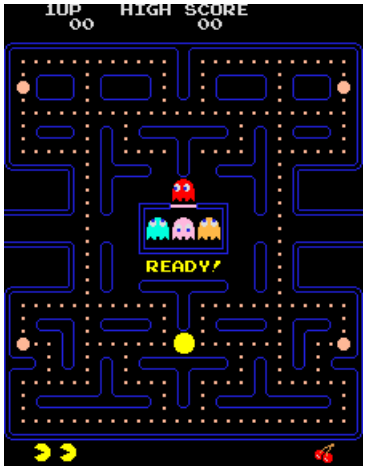


Illustration 6: Le jeu Pacman

Projet

Installation

Il faut la bibliothèque SFML d'une version supérieure à ou égale à 2.4.2 pour pouvoir compiler le jeu. Une commande « make » dans le répertoire « Pacman » où il y a le « Makefile » génère un exécutable « main ».

Jeu

Notre pacman est une variation de la version originale.

- Alors que le jeu original repart le même nombre de gommes à tous les niveaux, le nombre de gommes varie selon les niveaux et leurs emplacements sont purement aléatoires.
- Il n'y a pas de super pac-gommes. Donc les fantômes sont invincibles.
- Les algorithmes des fantômes ont été simplifiés.
  - « Blinky » ne suit que Pacman depuis le début d'une partie du jeu.
  - « Pinky » suit Pacman et essaie de faire un embuscade quand il est possible.
  - « Inkey » ne suit que « Blinky » depuis le début d'une partie du jeu.

- « Clyde » prend un chemin purement aléatoire.

- Les vitesses des fantômes s'incrémentent à chaque niveau et la vitesse de Pacman s'incrémente aussi mais plus lentement.
- Le nombre de vies donné est 2.
- Il ne faut que 500 points pour gagner une vie.
- En théorie, le niveau du jeu peut être incrémenté sans limite. En pratique, il y aura un niveau où les fantômes sont trop rapides pour Pacman.
- Lorsque les fantômes sont proches à Pacman, un son d'alerte est déclanché.

Capture d'écran

Illustration 7: L'écran du démarrage

Illustration 8: Le jeu en cours. Les couloirs qui sont ouverts sont liés (portails)

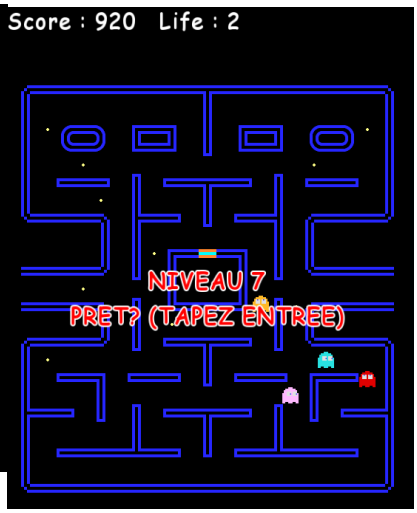


Illustration 9: l'augmentation du niveau de jeu

Illustration 10: Le jeu en pause

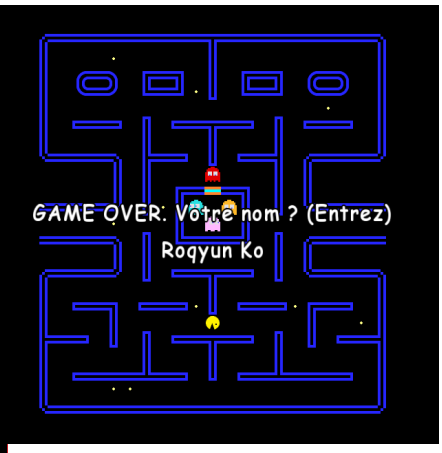


Illustration 11: Enregistrement du score et le pseudo du joueur à la fin du jeu

Illustration 12: La fin du jeu

Illustration 13: tableau des scores

Manipulation

On peut appuyer sur la touche d'échappement pour sortir du jeu ou du programme.

Pour bouger Pacman, on utilise les touches directionnelles. Pacman se déplace en autonome dans une seule direction à l'appui sur une touche directionnelle.

Pour mettre le jeu en pause, on appuie sur la touche 'P'.

Programmation

Hiérarchie

La hiérarchie du jeu est découpable en 3 parties :

- Jeu / GUI (Graphic User Interface)
  - Les son les images sont gérés ici et la fenêtre graphique est lancé depuis ici. Les événements de clavier/souris sont également gérés ici.
  - Le déroulement du jeu est géré ici. On détecte la collision des fantômes et Pacman, etc.
- Carte / routage
  - Une structure des données pour une carte du jeu est générée ici. Cette structure est utilisée principalement pour faire le routage et reconnaître les positions des entités (Pacman, les fantômes et les items). *Voir illustration 15*
- « Entité »
  - Les classes dans cette hiérarchie sont les créatures et les items.

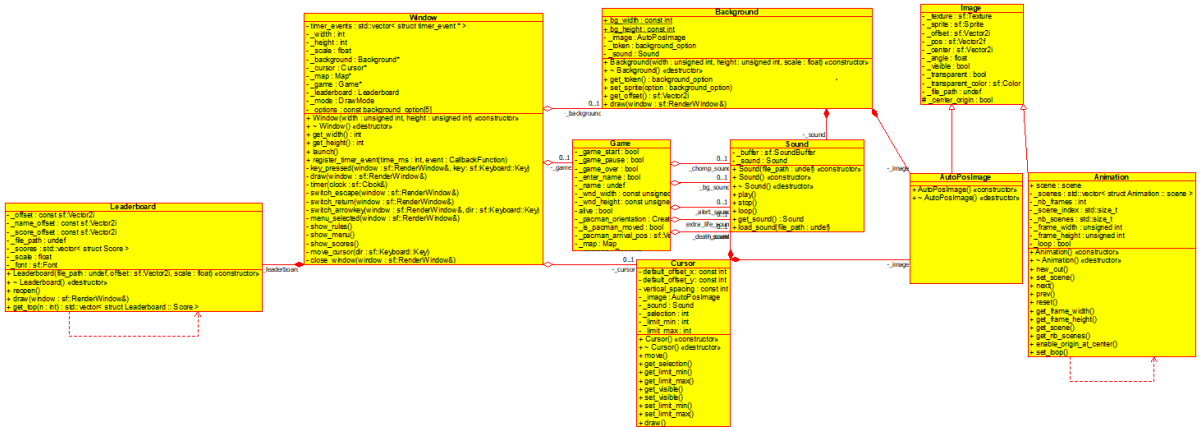


Illustration 14: UML de la partie jeu/GUI

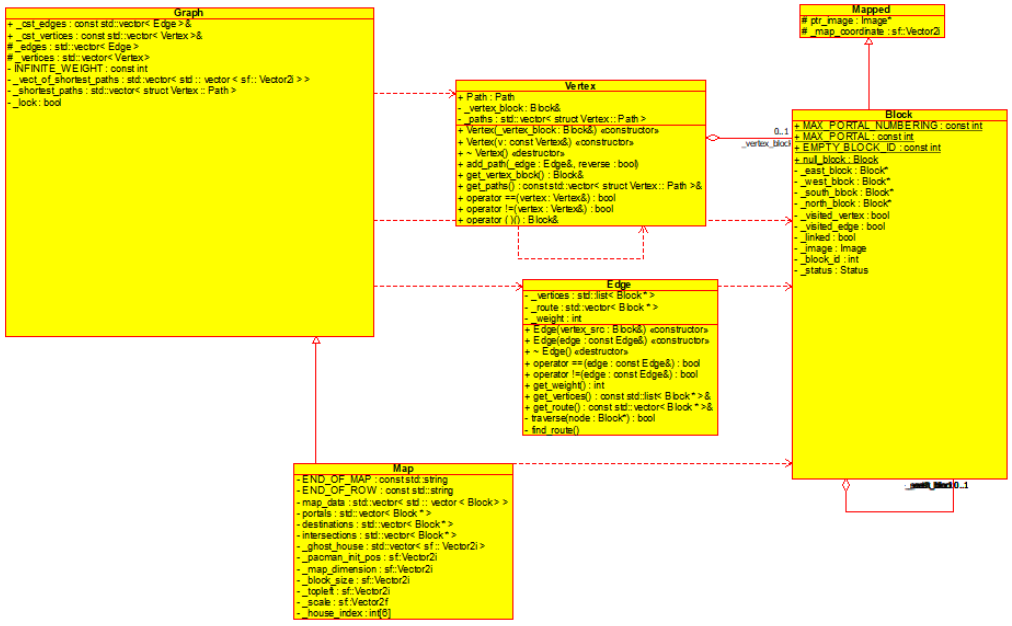


Illustration 15: La partie carte/routage

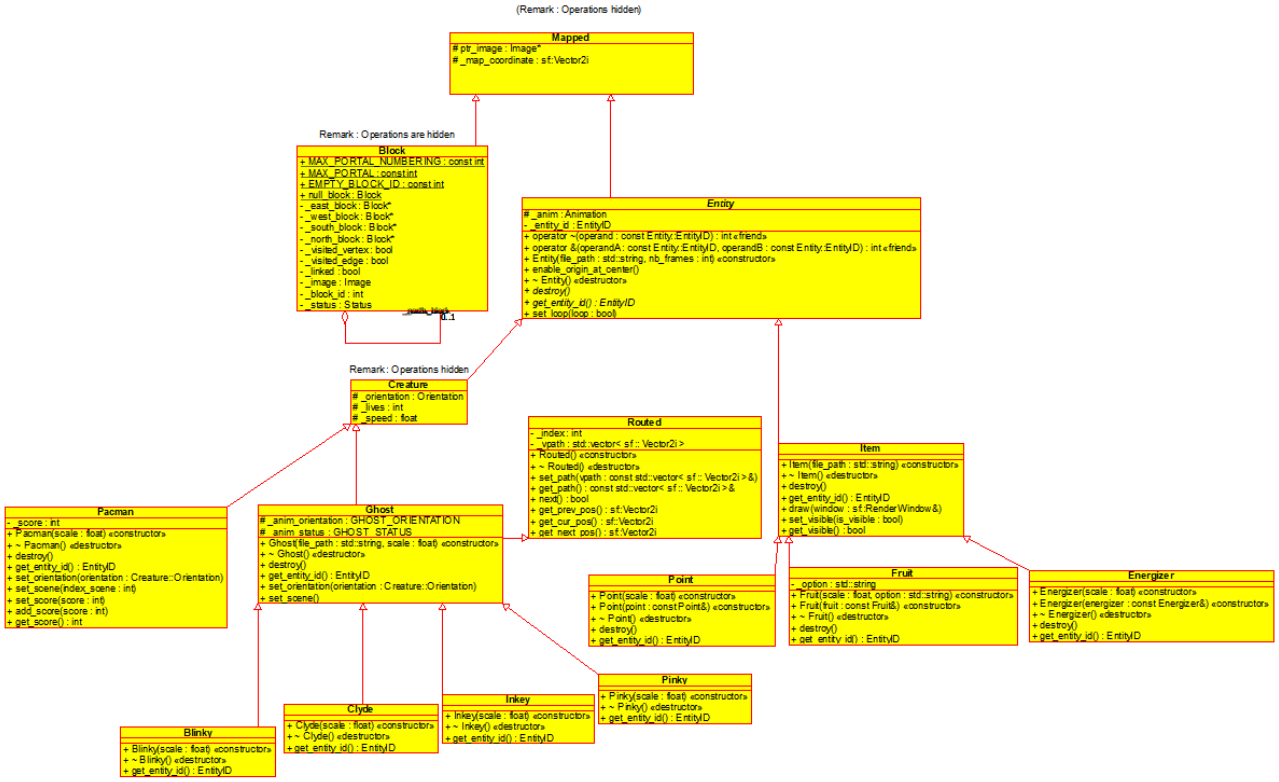


Illustration 16: La partie « Entité »

Classe

- **Window** : Les événements des périphériques de l'entrée et de la sortie sont gérés. Cette classe est découpé en 3 fichiers sources (*switch.cc* pour les fonctions déclenchées lors de l'événement de clavier, *winevents.cc* pour les événements à détecter / lancer, *window.cc* pour lancer la fenêtre)
- **AutoPosImage / Background / Curseur**: La classe AutoPosImage sert à centrer les images à l'écran. La classe Background sert à afficher les image du fond.
- **Game** : Le mouvement de Pacman et des fantômes sont graphiquement gérés. Une transition lisse entre une position à une autre position par exemple ou sauter d'un portail à un autre portail. La détection des collisions et les sons sont également gérés ici.
- **Carte/Block** : Une carte est découpé en plusieurs morceaux. Ainsi, on peut facilement positionner les « entité » et faire les murs et les portails. La classe Map fait une carte crée par utilisateur depuis un fichier texte. Ce fichier texte contient l'information sur les « blocks » comme sa nature (mur, case vide, portail, etc) et l'image qu'il faut utiliser pour afficher ce « block » ; l'information sur les positions initiales des fantômes et Pacman. Enfin, la carte crée un lien entre les « blocks » adjacents pour créer des chemins.
- **Graph / Vertex / Edge** : Ils servent à faire le routage entre les chemins et trouver le chemin le plus court. Nous avons utilisé un algorithme récursif.
  Un exemple :*On qu'il y a 4 chemins d'un sommet A à un sommet Z et les nombres en parenthèse sont les distance entre deux sommets adjacents :*

- C1 : A←(1)→ B←(100)→ C←(1)→ Z
  C2 : B←(1)→ U←(2)→ Z
- C3 : A←(1)→ D←(1)→ J←(1)→ K←(2)→ H←(1)→ Z
- C4 : A←(5)→ O←(6)→ P←(1)→ Z

On parcourt :

1. A→B→C→ Z**, arrivé** ; distance(C1) = 102, **la distance minimale (C1) = 102**
2. Revenir : Z→C→B ; distance (C2) = 1, la distance minimale(C1) = 102
3. B→U→Z, **arrivé** ; distance(C2)= 4, **la distance minimale(C2) = 4**
4. Revenir : Z→U→B→A ;
5. A→ D→ J→ K→ H ; distance(C3)= 5, la distance minimale(C2) = 4
6. Revenir :H→ K→ J→ D→A. (La distance est supérieur à la distance minimale)
7. A→O; distance(C4)= 5, la distance minimale(C2) = 4
8. Revenir :A→O. (La distance est supérieur à la distance minimale)
9. Le chemin le plus court est C2.

- **Routed** : Les fantômes prennent le chemin le plus court et enregistre le chemin à suivre dans cette classe.
- **Mapped** : Cette classe est pour enregistrer les coordonnées de la carte, les coordonnées de l'écran, etc.
- **Entity :** Une classe abstraite pour les fantômes, Pacman et les autres items.

Fantôme

Les fantômes prennent toujours des chemins les plus courts. Les fantômes ne peuvent pas revenir à l'arrière jusqu'à ce qu'il arrive à une intersection.

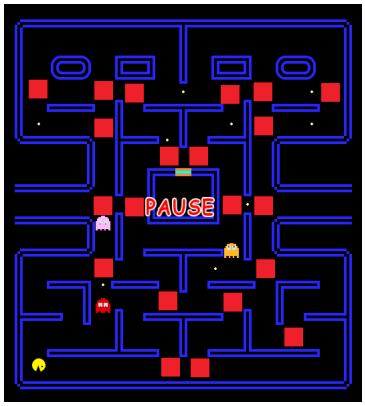


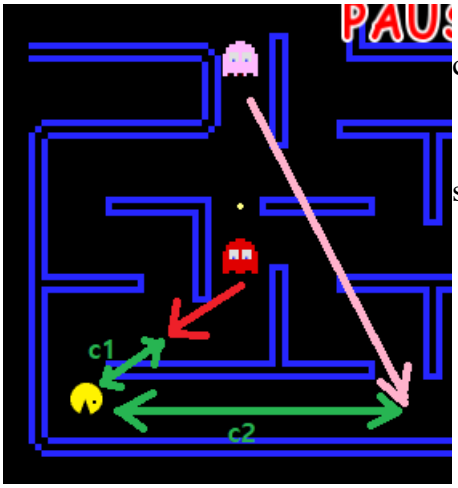
Illustration 17: Les intersections sont marqué en rouge

Comportement

Blinky suit Pacman en choisissant une intersection la plus proche de Pacman.

Pinky tend une embuscade à Pacman en choisissant une intersection la deuxième plus proche de Pacman.

Ces comportements sont justifiés par l'exemple suivant :



Si Pacman n'arrive pas avant Pinky à l'intersection au bout du chemin ***c2***, Pinky réussi à tendre une embuscade à Pacman.

Inkey suit Blinky lentement. Leurs parcours sont rarement superposés à cause de la différence en vitesse.

Timer

Dans le programme, on crée un chronomètre et on implémente un « timer » en comparant le dernier moment où les fonctions ont été exécutées. Pour facilement utiliser cette fonctionnalité, on utilise un « Callback ».

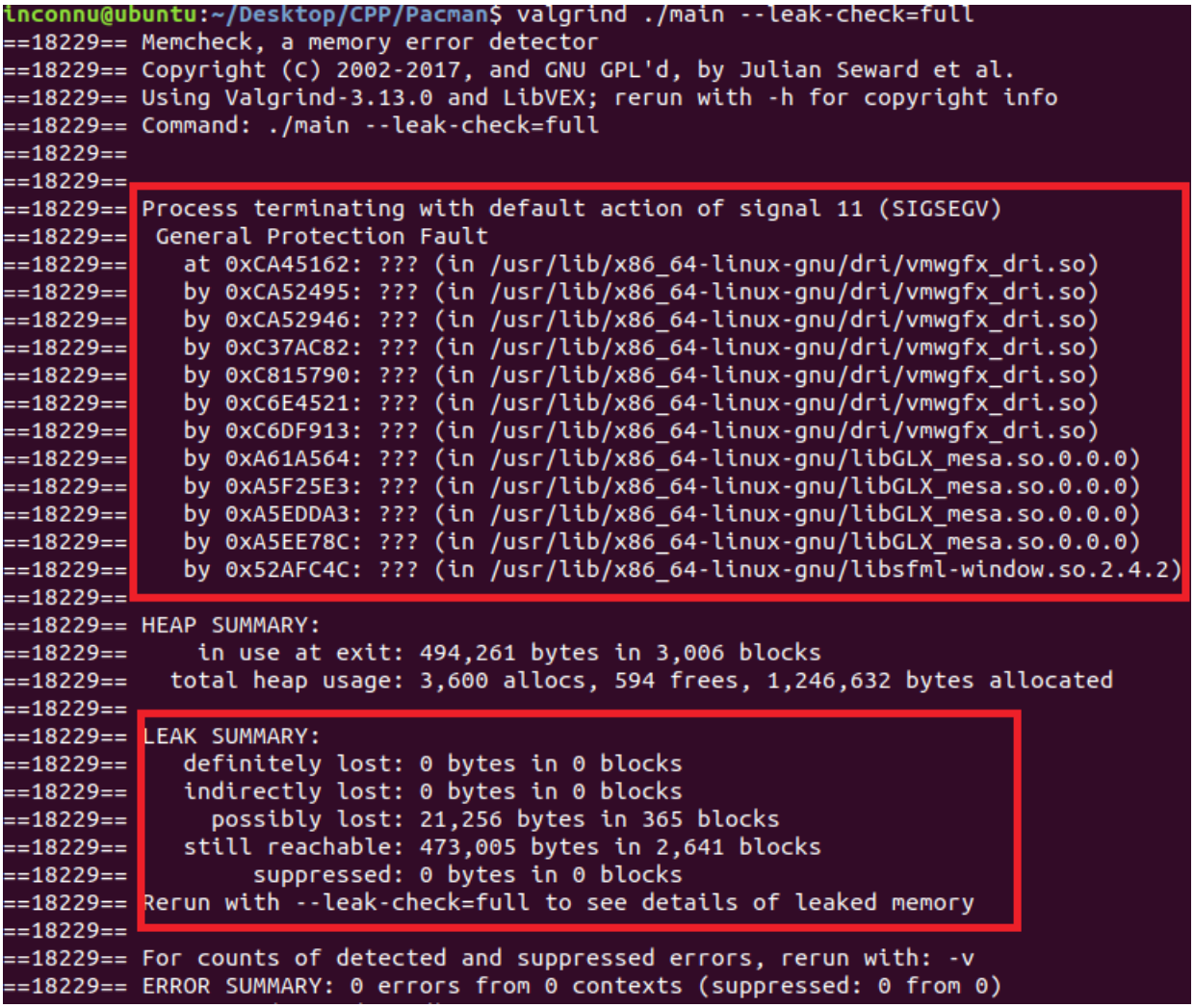
```
typedef std::function<void(void)> CallbackFunction;

struct timer_event {
        CallbackFunction event;
        int time_ms;
        sf::Time timer;
};

void Window::register_timer_event(int time_ms, CallbackFunction event) {
        struct timer_event *tmev = new struct timer_event;
        tmev->time_ms = time_ms;
        tmev->event = event;
        timer_events.push_back(tmev);
}

register_timer_event(100,[&]() {_game->animate();});
register_timer_event(1,[&]() {_game→loop();});
```

Résultat Valgrind



Les fuites de mémoire possible vient de la bibliothèques externes.