
Gilet Jaune

Rapport de Projet C++

Par

KARL MONGOSSO
ANTOINE DAFLON



Mathématiques Appliquées et Informatique Numérique
SORBONNE UNIVERSITÉ

JANVIER 2018

L'APPLICATION

Description générale

Le jeu Gilet Jaune est un jeu d'arcade dans lequel le joueur incarne un manifestant du mouvement "gilet jaune". Le joueur se déplace donc dans Paris en manifestant. Son but est alors de récupérer un maximum de carburant et d'argent pour subvenir à ses besoins, et cela en évitant les projectiles, lancés dans sa direction par les forces de l'ordre, ainsi que les biens brûlés par ses camarades.

Lorsque le joueur récupère du carburant ou un sac d'argent, il voit respectivement son score augmenter d'un point d'une part et son score et ses points de vie augmenter de 20 d'autre part. Le sac d'argent est, du fait de ses gains, un item (il apparaît rarement) et aura une durée d'apparition limitée (environ 10 secondes). Le carburant quant à lui apparaît dès que le carburant précédent a été attrapé par le joueur. Au contraire, lorsque le joueur est touché par un bien brûlé ou par un projectile, il perd respectivement 5 et 10 points de vie. Les biens brûlés ont également une durée d'apparition limitée qui augmentera au fur et à mesure du jeu. Il doit également faire attention aux forces de l'ordre qui, dès lors d'un contact avec le joueur, lui font également subir des dégâts qui sont proportionnels au temps de la collision. Le jeu se termine lorsque le joueur ne possède plus de points de vie (initialisés à 100) à cause des dommages qu'il a pu subir.

Pour rendre le jeu plus intéressant, le jeu devient de plus en plus dur au cours du temps. Les forces de l'ordre ainsi que les biens brûlés se font de plus en plus présent au fur et à mesure de la manifestation faisant ainsi vivre un véritable calvaire au manifestant. Nous lui souhaitons donc bon courage pour survivre dans cet environnement hostile.

Description de l'implémentation

Dans notre implémentation, il faut tout d'abord savoir que la fenêtre graphique sera représentée par une grille aux mêmes dimensions que l'écran. La grille sera initialisée à 0 et sera mise à jour au fur et à mesure que l'écran se remplit avec les objets et les personnages.

D'autre part, le jeu Gilet Jaune a été implémenté à l'aide de 13 classes, voici le détail de chacune des classes

TextureManager

La classe TextureManager contenant les méthodes nécessaire à l'affichage des élément du jeu. Elle a donc une méthode permettant de créer une texture à partir du nom d'un fichier photo et une méthode permettant d'ajouter la texture sur le rendu du jeu

GameElements

La classe GameElements est probablement la deuxième classe la plus importante du jeu. C'est un classe abstraite permettant de gérer tous les éléments "matériels" du jeu autrement dit, les objets et les personnages. Elle va contenir toutes informations de l'élément (position, texture) et notamment 2 fonctions virtuelles pures permettant la mise à jour des informations de l'élément et l'ajout de la texture sur le rendu. Elle est la classe mère des classes Object et Character.

Character

La classe Character est une des classes fille de la classe GameElements et elle se charge de toutes les actions liées au personnage (joueur,ennemies). Tout comme la classe gameElements, c'est une classe abstraite qui sera la classe mère des classe Player et Enemy. Elle permet d'ajouter la texture des personnages sur le rendu et elle contient les accesseurs aux coordonnées des personnages.

Player

La classe Player est une classe fille de la classe Character. Elle permet de gérer le joueur. Elle va, essentiellement, s'occuper des déplacements du joueur, de la mise à jour des coordonnées et de la position sur l'écran du joueur et de la détection d'une collision avec un ennemie à travers l'attribut privé damage.

Enemy

La classe Enemy est une classe fille de la classe Character. Elle permet de gérer les ennemies. Elle va gérer son déplacement en fonction de sa position lors de son apparition¹ et ensuite mettre à jour ses coordonnées et sa position sur l'écran. Elle indique également si l'existence de l'ennemi (si l'ennemie est ressorti de l'écran, il n'existe plus et disparaît).

Object

La classe Object est une classe fille de GameElements. C'est une classe abstraite permettant de gérer tous les object du jeu (carburant,bien brûlé,argent,projectile). Elle contient :

¹les ennemies sont créés à l'extérieur de l'écran, ils entrent dans l'écran envoient un projectile puis ressortent

- une méthode ajoutant la texture de l'objet sur le rendu
- une méthode virtuelle pure déterminant si l'objet est attrapé
- une méthode indiquant si l'écran est déjà occupé pour ne pas que l'objet se superpose sur un autre élément
- une méthode redéfinissant aléatoirement les coordonnées de l'objet
- les accesseurs de l'objet

Flame

La classe Flame est une classe fille de la classe Object. Elle permet de gérer les événements liés aux biens brûlés. Elle va mettre le bien brûlé sur l'écran à condition qu'il n'est pas été attrapé et qu'il existe encore². Elle contient également le temps d'apparition et le nombre d'itération du jeu durant lequel le bien brûlé est apparu³.

Gas

La classe Gas est une classe fille de la classe Object. Elle permet de gérer les événements liés au carburant. Elle va mettre le carburant sur l'écran à condition qu'il n'est pas été attrapé.

Money

La classe Money est une classe fille de la classe Object. Elle permet de gérer les événements liés à l'argent. Elle va mettre l'argent sur l'écran à condition qu'il n'est pas été attrapé et qu'il existe encore⁴. Il contient également le temps d'apparition et le nombre d'itération du jeu durant lequel l'argent est apparu⁵.

Projectile

La classe Projectile est une classe fille de la classe Object. Elle permet de gérer les événements liés aux projectiles. Elle va gérer les déplacements du projectile qui va vers le joueur. Pour cela, elle calcule la pente de la droite liant le joueur et l'ennemie lançant le projectile à l'aide de l'équation :

$$(1) \quad \text{pente} = \frac{y_1 - y_0}{x_1 - x_0}$$

Elle incrémente la position de x de 1 et de y de pente en fonction de la direction du projectile. Elle fait disparaître le projectile si il touche le joueur ou si il sort de l'écran.

²Rappel:les biens brûlés disparaissent au bout d'un certains temps

³pour pouvoir le supprimer lorsqu'il atteint sa durée d'apparition

⁴Rappel:l'argent disparaît au bout d'un certains temps (attention il disparaît assez vite)

⁵pour pouvoir le supprimer lorsqu'il atteint sa durée d'apparition

Menu

La classe Menu permet de gérer l'écran du menu (écran apparaissant lors du lancement du jeu). Elle ajoute les éléments du menu sur le rendu et lance le jeu lorsque l'utilisateur clique sur l'icône de lancement.

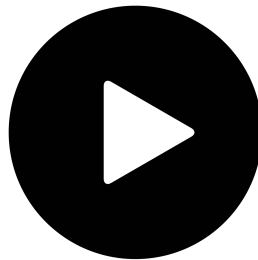


Figure 1: Icône de lancement

Map

La classe Map gère la création de la map. Il est possible de créer 2 types de map : une map à partir d'une image (map par défaut) et une map personnalisée. La map à partir d'une image est chargée et ajoutée sur le rendu par défaut mais vous pouvez également changer la map en remplaçant l'image ou en créant votre propre map⁶.

Game

La classe Game est la classe la plus importante. C'est la classe qui va être appelée dans la fonction principale et qui fait appel à toutes les autres classes. Elle va se charger de gérer tous les éléments du jeu excepté la musique qui est gérée par la fonction principale. Elle va donc gérer tout ce qui concerne l'affichage des éléments du jeu ainsi que la mise à jour des éléments du jeu. Elle va notamment permettre de créer tous les éléments (en faisant appel à leurs constructeurs), mettre à jour (en faisant appel à leurs méthodes) et afficher le joueur, les ennemis et les objets en gérant leur fréquence d'apparition et leurs textures. Elle va également gérer tout ce qui est lié à l'interface graphique c'est-à-dire l'initialisation, l'affichage du rendu et la gestion des événements.

Finalement, le dernier élément de l'implémentation est la fonction principale qui va premièrement se charger de lancer la musique de fond. Elle va majoritairement être chargée de la boucle de jeu, c'est-à-dire que c'est elle qui va arrêter le jeu quand il est fini et qui va faire appel aux méthodes de la classe Game pour mettre à jour et dessiner les éléments du jeu. Mais, elle va également être à l'origine de la musique de fond du jeu.

⁶voir l'annexe : Création de Map pour avoir plus d'info sur la création de map

Mise en valeur des contraintes

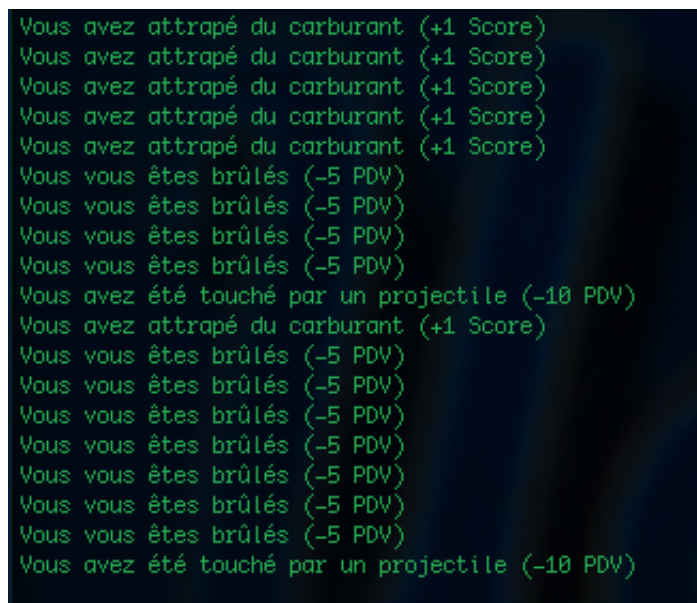
Globalement, les contraintes furent assez simples à remplir. L'utilisation d'au moins 8 classes avec 3 niveaux d'hérarchie s'est faite naturellement.

En effet, comme vous avez pu le remarquer, le jeu est composé d'un grand nombre d'éléments à gérer que ce soit au niveau des personnages, des objets ou encore de l'écran entre autre.

De même, les fonctions virtuelles coulaient de source du fait des différents niveaux de hiérarchie. Il était logique de créer des classes abstraites et ainsi des fonctions virtuelles pour des éléments du jeu qui fonctionne plus ou moins selon un même schéma.

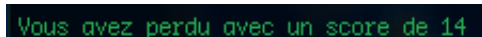
Les conteneurs quant à eux font office dans le programme de "bases de donnée". Ils contiennent l'ensemble des éléments concernant les objets et les ennemies (pour les list), et les données du joueur (score, point de vie pour la map).

La contrainte la plus dure à satisfaire a été la contrainte concernant les surcharges d'opérateur car nous n'avions pas réellement besoin. Finalement, nous avons décidé de surcharger les classes Game et Player pour effectuer dans le premier cas un récapitulatif de ce qui se passe dans le jeu au niveau du terminal (figure 2) et dans le deuxième cas indiquer le score du joueur en fin de partie toujours dans le terminal (figure 3).



```
Vous avez attrapé du carburant (+1 Score)
Vous avez attrapé du carburant (+1 Score)
Vous avez attrapé du carburant (+1 Score)
Vous avez attrapé du carburant (+1 Score)
Vous avez attrapé du carburant (+1 Score)
Vous vous êtes brûlés (-5 PDV)
Vous vous êtes brûlés (-5 PDV)
Vous vous êtes brûlés (-5 PDV)
Vous vous êtes brûlés (-5 PDV)
Vous avez été touché par un projectile (-10 PDV)
Vous avez attrapé du carburant (+1 Score)
Vous vous êtes brûlés (-5 PDV)
Vous vous êtes brûlés (-5 PDV)
Vous vous êtes brûlés (-5 PDV)
Vous vous êtes brûlés (-5 PDV)
Vous vous êtes brûlés (-5 PDV)
Vous vous êtes brûlés (-5 PDV)
Vous vous êtes brûlés (-5 PDV)
Vous avez été touché par un projectile (-10 PDV)
```

Figure 2: Surchage de Game



```
Vous avez perdu avec un score de 14
```

Figure 3: Surchage de Player

Fierté

La partie la plus gratifiante du jeu est l'implémentation de l'interface graphique. L'utilisation des bibliothèques SDL2⁷ (SDL2,SDL2_image,SDL2_mixer et SDL2_ttf) n'a pas été abordée en cours. Il a donc fallu que l'on "s'autoforme" et que l'on monte rapidement en compétence dans ce domaine pour mener à bien le projet. Après un travail rigoureux et acharné, nous avons finalement réussi à créer une interface graphique fonctionnelle pour un rendu très satisfaisant. Cela représente le plus grand accomplissement de ce projet.

Finalement, de manière plus général, la création d'un jeu en partant de rien est déjà une grande fierté en soi.

⁷Pour avoir plus d'information sur les éléments SDL utilisés dans le jeu veuillez vous référer à l'annexe SDL2

INSTALLATION ET COMPILATION

Installation

Le programme nécessite, en plus des librairies déjà présente sur la majorité des machine⁸, les librairies SDL2, SDL_image de SDL2, SDL2_mixer et SDL_ttf de SDL2.

Installation Mac

L'installation de ces 3 librairies sous mac nécessite simplement de taper les commandes suivantes dans le terminal :

1. brew install sdl2⁹
2. brew install sdl2_image
3. brew install sdl2_mixer
4. brew install sdl2_ttf

Installation Linux/Ubuntu

L'installation de ces 3 librairies sous mac nécessite simplement de taper les commandes suivantes dans le terminal :

```
hg clone https://hg.libsdl.org/SDL SDL
cd SDL
mkdir build
cd build
../configure
make
sudo apt-get install libsdl2-dev libsdl2-image-dev libsdl2-mixer-dev libsdl2-ttf-dev
```

⁸en plus des librairies iostream, string, list, map, vector, stdlib.h, time.h

⁹Si Homebrew n'est pas installer taper cette commande avant les commandes indiquées : `ruby -e "$(curl -fsSL https://raw.githubusercontent.com/Homebrew/install/master/install)" < /dev/null 2> /dev/null`

Compilation

Le dossier contenant les fichiers du jeu est équipé d'un Makefile. Il suffit juste d'exécuter la commande *make* sur le terminal en vous plaçant dans le dossier pour compiler le jeu.

```
[MacBook-Pro-de-Karlito-2:Arcade-Game-Gilet-Jaune- mac$ make ]
g++ -g -std=c++11 -Wall -o Projectile.o -c Projectile.cc
g++ -g -std=c++11 -Wall -o Money.o -c Money.cc
g++ -g -std=c++11 -Wall -o Gas.o -c Gas.cc
g++ -g -std=c++11 -Wall -o Flame.o -c Flame.cc
g++ -g -std=c++11 -Wall -o Enemy.o -c Enemy.cc
g++ -g -std=c++11 -Wall -o Object.o -c Object.cc
g++ -g -std=c++11 -Wall -o Player.o -c Player.cc
g++ -g -std=c++11 -Wall -o Character.o -c Character.cc
g++ -g -std=c++11 -Wall -o Menu.o -c Menu.cc
g++ -g -std=c++11 -Wall -o Map.o -c Map.cc
g++ -g -std=c++11 -Wall -o TextureManager.o -c TextureManager.cc
g++ -g -std=c++11 -Wall -o Game.o -c Game.cc
g++ -g -std=c++11 -Wall -o GameElements.o -c GameElements.cc
g++ -g -std=c++11 -Wall -o main.o -c main.cc
g++ -g -std=c++11 -Wall -o main Projectile.o Money.o Gas.o Flame.o Enemy.o Obje
ct.o Player.o Character.o Menu.o Map.o TextureManager.o Game.o GameElements.o ma
in.o `sdl2-config --cflags --libs` -lsdl2_image -lsdl2_ttf -lsdl2_mixer
```

Figure 4: Compilation avec make

```
[MacBook-Pro-de-Karlito-2:Arcade-Game-Gilet-Jaune- mac$ ./main
```

Figure 5: Exécution

```
[MacBook-Pro-de-Karlito-2:Arcade-Game-Gilet-Jaune- mac$ make clean
```

Figure 6: Nettoyage avec make clean

TABLE DES MATIÈRES

L'application	1
Description générale	1
Description de l'implémentation	1
TextureManager	2
GameElements	2
Character	2
Player	2
Enemy	2
Object	2
Flame	3
Gas	3
Money	3
Projectile	3
Menu	4
Map	4
Game	4
Mise en valeur des contraintes	5
Fierté	6
Installation et Compilation	7
Installation	7
Installation Mac	7
Installation Linux/Ubuntu	7
Compilation	8
	Page