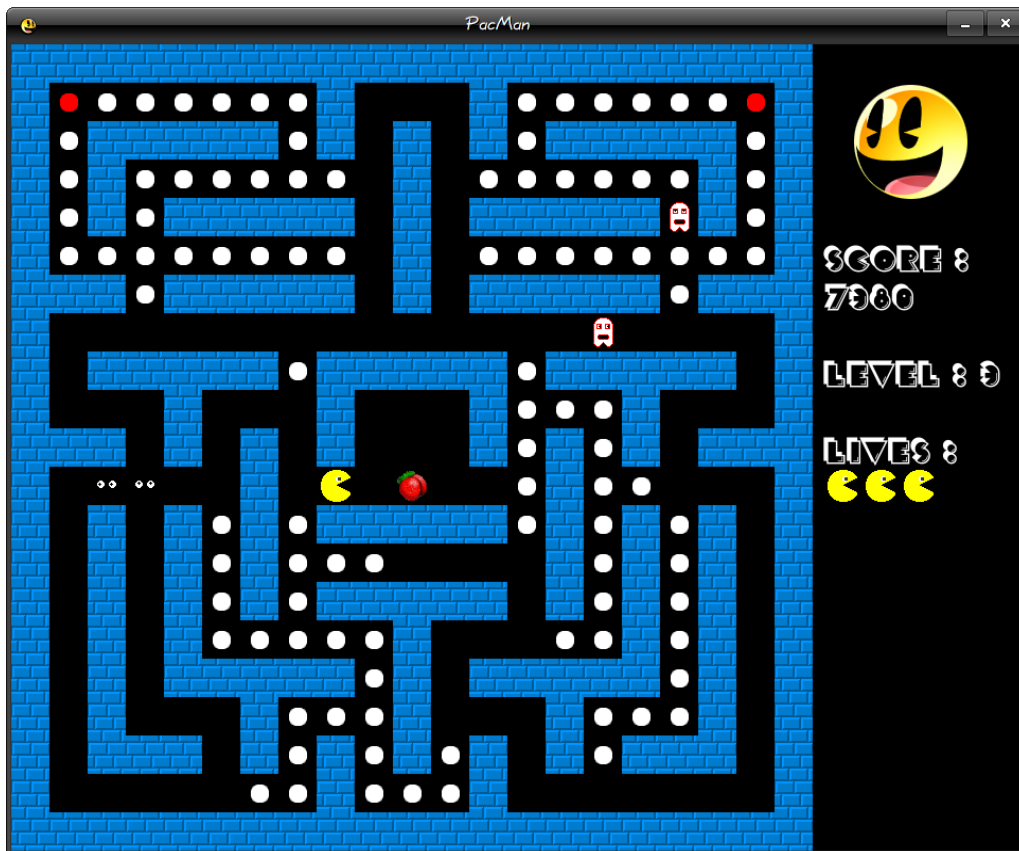


Méthodologie de la programmation

Rapport du « projet Pacman »



Pierrick Vandembroucke

Bachelor professionnel en informatique de gestion

2006-2007

Université du Luxembourg

Table des matières

Le jeu du Pacman.....	3
Description.....	3
Avant de commencer.....	3
C versus Java.....	3
Librairie graphique.....	3
Le projet.....	4
Programmation modulaire.....	4
Step by step.....	5
Fonctionnement de SDL.....	5
Gestion des évènements.....	6
Gestion du temps.....	7
Chasse aux bugs.....	7
Memory leak spotted.....	7
Le jeu.....	9
Comment ça marche?.....	9
Comment jouer?.....	10
Les caractéristiques du jeu.....	10
Améliorations possibles.....	11
Sources	12

Le jeu du Pacman

Pacman consiste en un jeu vidéo créé par le concepteur *Toru Iwatani* pour l'entreprise japonaise Namco. Le but du jeu consiste à déplacer Pacman, un personnage en forme de camembert, à l'intérieur d'un labyrinthe, et de lui faire récolter toutes les pac-gommes qui s'y trouvent en évitant d'être touché par des fantômes.

Description

Pacman, personnage emblématique du jeu, est un personnage en forme de rond jaune doté d'une bouche: il doit manger des pac-gommes et des bonus (sous forme de fruits, et d'une clé à 5000 points) dans un labyrinthe hanté par quatre fantômes. Quatre pac-gommes spéciales rendent les fantômes vulnérables pendant une courte période au cours de laquelle Pacman peut les manger. Le jeu original comprend 255 labyrinthes différents (le jeu était considéré comme allant à l'infini, mais l'écran se brouillait dès le 256e niveau).

Source: Wikipédia

Avant de commencer

C versus Java

Préalablement à la réalisation du Pacman, il convient d'arrêter le choix du langage de programmation: C ou Java? Le Java présente l'avantage d'un langage de haut niveau, doté de nombreuses fonctionnalités, bibliothèques graphiques et autres. Toutefois, il nécessite une familiarité dont je ne disposais pas à suffisance pour aisément mener mon projet à bien (notions de base, l'orienté objet pouvait me poser problème). Le C, à l'inverse, s'il se caractérise par son bas niveau, m'est nettement plus familier. Petit bémol, il ne dispose pas de bibliothèque graphique. Le recours à divers sites comparatifs m'a permis de peser les avantages et inconvénients de chaque langage.

Finalement, j'ai opté pour le C, un langage où je suis à l'aise, afin d'optimiser le rapport qualité/temps. L'apprentissage de Java s'avérait superflu pour un projet gourmand en temps, sans que cela se justifie vraiment.

Bibliothèque graphique

Le choix du langage C posé, il restait l'écueil du choix de la bibliothèque graphique. Cela supposait de me documenter sur les différentes bibliothèques graphiques (SDL, OpenGL, Allegro, GTK,...). Mes critères de choix reposaient sur la 2D, la portabilité et la disponibilité de documentation, tutoriels et forums spécialisés. La documentation s'avérait essentielle; les nouveautés dans ce projet

par rapport aux autres se rapportaient à la gestion de l'interface graphique ainsi qu'à celle des interactions avec l'utilisateur.

Finalement, mon choix s'est porté sur SDL, un bon tutoriel étant disponible, de même que le code source d'un jeu de Mario afin de bien intégrer son fonctionnement. Après quelques essais, il m'a fallu conclure que SDL se révèle nettement plus ardu qu'il n'y paraît au premier abord. Dans de précédents projets réalisés en VBA et Delphi, l'interface se créait par simples drag-and-drop et quelques lignes de code. Avec SDL, toute commande requiert du code, de l'initialisation à la libération des surfaces en passant par le chargement, le positionnement, l'affichage,...

Il n'empêche: SDL tout comme C reste un langage de bas niveau. Une fois son fonctionnement assimilé, il s'apparente à un vrai jeu d'enfant.

Le projet

Il faut souligner ici plusieurs détails d'importance. Certains se montrent assez techniques et d'autres plus axés sur la globalité du projet.

Programmation modulaire

Coder un Pacman nécessite un grand nombre de lignes de code. Plus on en rajoute, moins il est facile d'y évoluer. Pour contrer cet aspect, j'ai recouru à la simplification maximale des fonctions, tout en divisant et d'organisant l'ensemble en différents fichiers. Chaque fichier contient des fonctions spécifiques à une fonctionnalité du programme:

- main.c: menu d'accueil, lance la nouvelle partie, lance l'affichage des contrôles ou quitte le programme;
- jeu.c et jeu.h: fonctions du jeu, gestion des rencontres;
- chargement.c et chargement.h: charge les sprites et la carte;
- gestion_Pacman.c et gestion_Pacman.h: fonctions de déplacement du Pacman
- gestion_fantômes.c et gestion_fantômes.h: fonctions de déplacement et de transformation des fantômes
- pause.c et pause.h: gestion de la pause;
- game_over.c et game_over.h: affiche l'écran de fin de jeu avec le score obtenu;
- controles.c et controles.h: affiche la fenêtre avec les différents contrôles du jeu;
- constantes.h: contient les différentes constantes (tailles d'un bloc, déclaration des structures,...).

Step by step

Un programme tel qu'un Pacman ne se conçoit pas globalement. Coder le jeu étape par étape, comme pour la construction d'une maison, s'impose:

- Le menu;
- Le plateau de jeu;
- Ajout du Pacman et gestion de ses déplacements;
- Ajout des fantômes:
 - Intelligence artificielle pour les déplacements;
 - Gestion des rencontres avec le Pacman.
- Ajout des bonus;
- Ajout des animations pour rendre le jeu un peu plus vivant.

Fonctionnement de SDL

Il s'agit d'un point essentiel, nécessitant un développement. Le SDL, langage choisi notamment pour manipuler l'interface graphique, se caractérise, on l'a dit, par son bas niveau. Son fonctionnement diffère de Delphi, où on glisse les éléments sur l'interface,...

Au contraire, toute action s'opère via des lignes de code. Bien comprendre son fonctionnement est indispensable pour éviter les bugs ainsi que les fuites de mémoire. D'apparentes aberrations apparaissent logiques après réflexion. Exprimé en français et précisant les fonctions utilisées, le principe se montre identique à nombre des jeux ou programmes utilisant SDL.

- Avant l'initialisation, on doit signaler qu'en cas de bug, on doit quitter SDL (*atexit(SDL_Quit())*).
- Initialisation de SDL (*SDL_Init()*)
- Initialisation des différentes bibliothèques dérivées (*TTF_Init()* par exemple pour les polices)
- Déclaration des variables (plus acceptable pour les débogueurs)
- Chargement de l'icône de la fenêtre
- Création de la fenêtre (*SDL_SetVideoMode()*)
- Modification du titre de la fenêtre
- Chargement des sprites statiques (*IMG_load()*)
- Affectation des polices d'écriture (*TTF_OpenFont()*)

- Chargement des textes statiques (*TTF_RenderText_Blended()*)
- Entrée dans la boucle « infinie »
 - Gestion des événements
 - Effacement de l'écran pour éviter les superpositions et fuites de mémoire
 - Chargement des sprites dynamiques (images et textes)
 - Positionnement des différents sprites sur l'écran (*SDL_BlitSurface()*)
 - Rafraîchissement de l'écran à l'aide du buffer (*SDL_Flip()*)
 - Libération des surfaces (sprites) dynamiques (*SDL_FreeSurface()* ou *TTF_CloseFont()*)
- Fin boucle « infinie »
- Libération des sprites statiques
- Fermeture des librairies dérivées (*TTF_Quit()* pour les polices)
- Fermeture de SDL (*SDL_Quit()*)

Evidemment, comme pour une majorité des fonctions du C, une grande partie des fonctions de SDL comportent une valeur de retour. Et, celle-ci requiert d'être testée pour s'assurer du bon fonctionnement du programme.

Gestion des évènements

SDL permet deux types de gestion événementielle: bloquante (*SDL_WaitEvent()*) et non-bloquante (*SDL_PollEvent()*). Au cours du jeu, les deux types sont implémentées:

- Dans le menu, les pauses et l'écran de fin de jeu, le système bloquant est utilisé: il attend une action de l'utilisateur pour poursuivre ses opérations. Cela permet de mettre le processus en *wait* et d'utiliser moins de CPU.
- Au cours du jeu normal, le système non-bloquant intervient: les fantômes doivent continuer de bouger même si le Pacman reste immobile. Cela consomme beaucoup de CPU. Cet inconvénient sera est annihilé grâce au système de timer. Autrement dit, toutes les X milli-secondes le programme effectuera certaines opérations.

Gestion du temps

Le projet s'est révélé nettement plus gourmand en investissement temps qu'initialement prévu. Une schématisation préalable sur papier aurait permis de surmonter partiellement cet écueil. Toutefois, dresser un tel plan n'est pas simple, il est difficile de prévoir à l'avance les besoins et l'organisation du programme. Ainsi, des phases de test de SDL et les tests au départ de variables simples.

Prometteuses, ces démarches se complexifient par la multiplication des paramètres. Ce qui conduit à l'intrégration de structures. Ensuite, dans le développement, pour optimiser le programme, le recours à des pointeurs sur des structures s'est imposé.

Finalement, le projet a excédé mes prévisions. Cependant, s'il me fallait maintenant coder le Pacman en Java, je cerne clairement que les objets s'imposeraient d'emblée: les structures.

Chasse aux bugs

Entre chaque nouvelle « couche », déboguer apparaît indispensable. Car il est plus simple de retrouver des erreurs sur des petites parties de codes que dans 2000 lignes. Au cours du développement, j'ai rencontré tous les types de problèmes (débordements en mémoire, fuites de mémoire,...). Le tout a été débogué à coups de gdb et de valgrind.

Les phases de débogage, dans un tel projet, se révèlent finalement plus formatrices que la programmation au sens strict. Elles permettent aussi de réaliser que trop de tutoriels sont truffés sinon d'erreurs, du moins d'approximations et de madresses.

De l'élaboration de ce Pacman, je retiendrai certainement qu'il m'a familiarisé avec les outils de débogage, leur maîtrise mais également leurs faiblesses.

Memory leak spotted

Avec ces mêmes outils de débogage, m'est apparue une petite fuite de mémoire dans la fonction `SDL_Init` (sur Ubuntu 7.04 et `libsdl1.2 11-7`). Testée en dehors du Pacman afin de m'assurer que j'étais étranger à ce dysfonctionnement, cette fuite rapportée aux auteurs de SDL sera, m'ont-ils promis, corrigée pour la prochaine mise-à-jour. Le code concerné se trouve à la page suivante.

```
#include <SDL/SDL.h>

int main (){
    atexit(SDL_Quit);

    if ((SDL_Init(SDL_INIT_VIDEO)) == -1){        //ligne 5
        fprintf(stderr,"Error at SDL_Init  : %s\n",SDL_GetError());
        return -1;
    }

    SDL_Quit();
    return 0;
}

==9420== Address 0x47DD056 is 14 bytes inside a block of size 16,384 alloc'd
==9420==   at 0x402095F: calloc (vg_replace_malloc.c:279)
==9420==   by 0x45EE346: XOpenDisplay (in /usr/lib/libX11.so.6.2.0)
==9420==   by 0x4093BED: (within /usr/lib/libSDL-1.2.so.0.11.0)
==9420==   by 0x409E987: (within /usr/lib/libSDL-1.2.so.0.11.0)
==9420==   by 0x408B8B0: SDL_VideoInit (in /usr/lib/libSDL-1.2.so.0.11.0)
==9420==   by 0x405E1D0: SDL_InitSubSystem (in /usr/lib/libSDL-1.2.so.0.11.0)
==9420==   by 0x405E216: SDL_Init (in /usr/lib/libSDL-1.2.so.0.11.0)
==9420==   by 0x8048FF4: main (main.c:5)
....
==9420== LEAK SUMMARY:
==9420==   definitely lost: 10 bytes in 2 blocks. <- fuite de 10 bytes
==9420==   possibly lost: 0 bytes in 0 blocks.
==9420==   still reachable: 23,249 bytes in 393 blocks.
==9420==   suppressed: 0 bytes in 0 blocks.
```

Le jeu

Comment ça marche?

Téléchargez le jeu sur le site: <http://pacman.vanpie.be/>. La version présente sur le site tourne sur la majorité des distributions Linux qui ont SDL, SDL_image et SDL_ttf d'installés.

Une fois downloadé, décompressez le Pacman.zip et rendez-vous dans le dossier Pacman.

Exécutez le programme Pacman en double-cliquant sur l'icône:



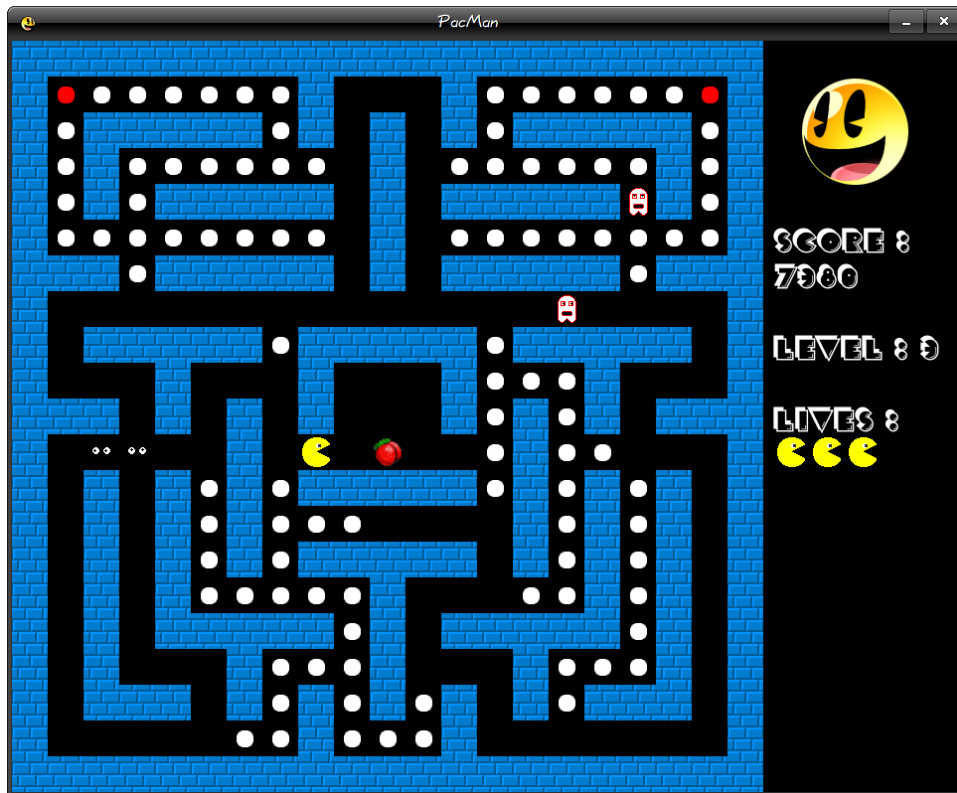
Une fenêtre apparaît à l'écran:



Faites votre choix via le menu avec les **flèches** de votre clavier et validez en appuyant sur la touche **Enter**.

Comment jouer?

Sélectionnez New Game et la fenêtre du jeu s'affiche:



Pour vous déplacer dans le jeu, il suffit d'utiliser les **flèches** (la répétition des touches est activée).

Afin de mettre le jeu en pause, pressez la touche **P** et pour réactiver le jeu, réappuyez sur **P**.

Si vous désirez revenir au menu principal, fermez la fenêtre à l'aide de la souris ou appuyez sur échappe (**Esc**).

Les caractéristiques du jeu

En créant ce jeu, j'ai tâché de respecter le jeu original. Certes, quelques différences sont présentes mais dans la droite ligne de la philosophie globale.

Au début du jeu, le **Pacman** a 3 vies et commence au niveau 1 avec 0 point. La répétition des touches est activée pour qu'il se déplace toutes les 0,1 seconde. Cette constante n'évolue pas au fil des niveaux.

Les **fantômes**, au nombre de 4, se déplacent aléatoirement sur le plateau. S'ils sont en mode agressif (de couleur) et si le joueur se trouve sur leur chemin, ils le traqueront. Si l'un d'entre eux le touche, le joueur perd une vie. La vitesse de

déplacement des fantômes augmente de niveau en niveau afin d'accroître la difficulté des épreuves.

Les **pac-gommes** (pastilles blanches) avalées rapportent 10 points au compétiteur. Quand il n'y en a plus ni des blanches ni des rouges, celui-ci franchit le niveau et poursuit avec davantage de complexité.

Les **pac-gommes spéciales** (pastilles rouges) font gagner 15 points. En avaler une entraîne une vulnérabilité de 10 secondes des fantômes. Ingérer un fantôme durant ce temps, crédite le joueur de 200 points. Attention, une fois ce délai passé, les fantômes peuvent redevenir agressifs (clignotement préalable de 3 secondes).

Trois **bonus** enrichissent le jeu; la cerise, l'orange et les fraises rapportent respectivement 200, 400 et 600 points. Ces bonus apparaissent aléatoirement au centre du plateau durant 8 secondes.

A chaque palier de 5000 points, le joueur gagne une vie.

Le nombre de niveaux est illimité, le plateau ne change pas mais les fantômes vont de plus en plus vite.

Une fois la partie perdue, votre score s'affiche et le compétiteur est renvoyé au menu du jeu.

Améliorations possibles

Le projet, bien que fini et opérationnel, n'est pas parfait. Il subsiste deux problèmes mineurs et des améliorations envisageables.

Ainsi, quand le Pacman croise un fantôme et qu'il le mange, il n'enlève pas toujours la gomme qui se trouve en dessous.

Le second problème provient de la fenêtre du game-over qui disparaît trop vite/lentement. Je n'ai pu maîtriser cet inconvénient mineur. Or, Il faudrait s'assurer que le joueur dispose du temps de visualiser son score final avant de revenir au le menu. A l'inverse, le même joueur n'appréciera guère de se croire bloquer sur le tableau de résultat.

En outre, améliorer les algorithmes de déplacement des fantômes contribuerait à l'attractivité du jeu. Dans le Pacman original, chaque fantôme a son caractère et sa manière de se déplacer (utilisation de l'algorithme du plus court chemin de Dijkstra,...).

Enfn, l'ajout d'un fond sonore en utilisant SDL_mixer renforcerait le suspens inhérent au jeu.

Sources

- Site du zéro: tutoriels et sources diverses, pas assez précis sur certains détails mais les tutoriels sont captivants grâce à l'humour employé. (<http://www.siteduzero.com/>)
- Développez.com: communauté de développeurs, tutoriels et foires aux questions. Très riche en informations. (<http://www.developpez.com/>)
- Site officiel de SDL: documentation officielle, tutoriels, exemples de code,... (<http://www.libsdl.org/>)
- Pages man des différentes fonctions