OpenGL - Projet final

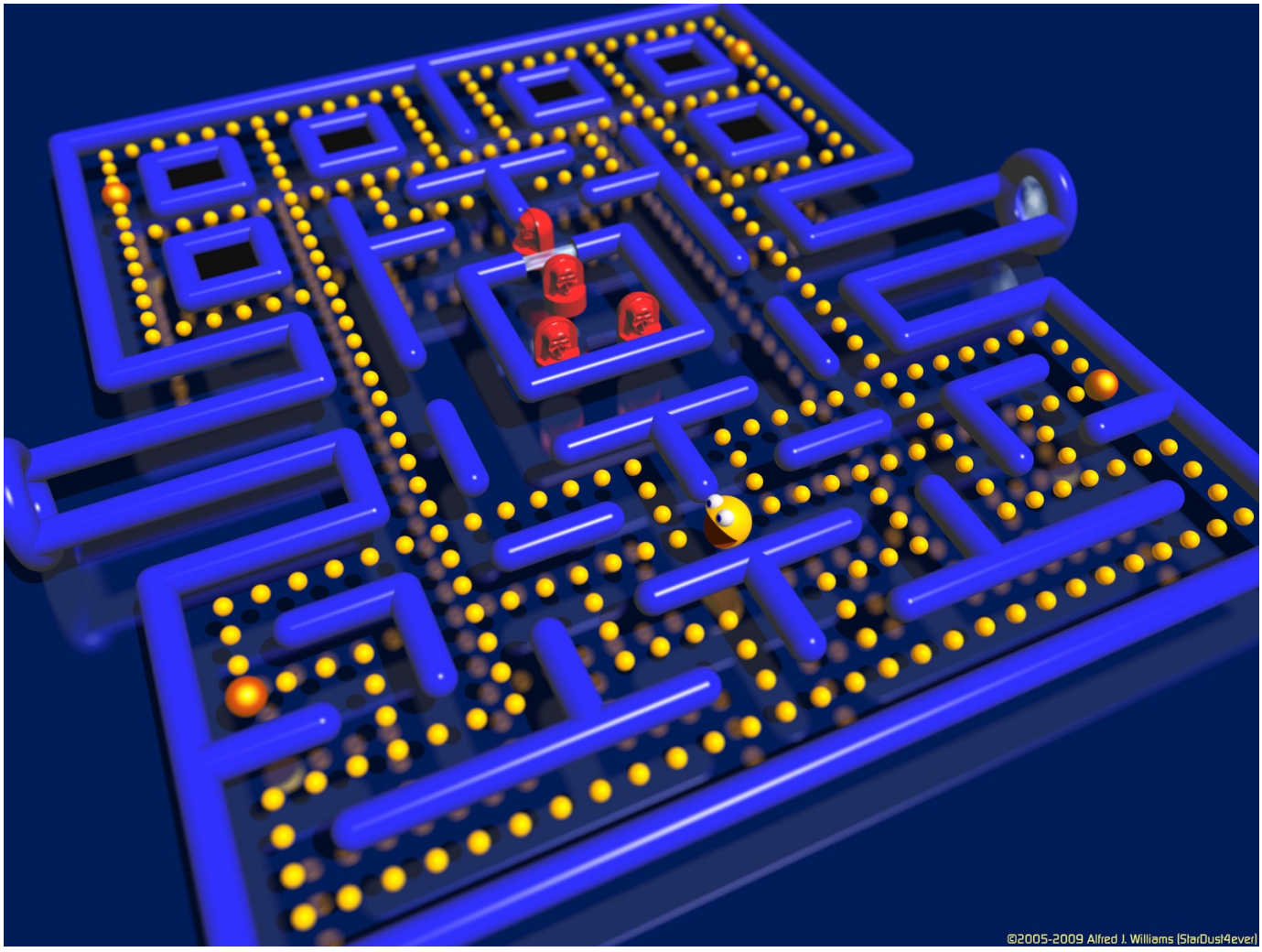
IMACMAN 3D

À rendre avant le : 11/01/2017 (veille de la soutenance)

Contacts : maxime.maria@esiee.fr/ biri@u-pem.fr

0 Introduction

L'objectif de ce projet est de développer un jeu de type « PACMAN » en 3D en C++/OpenGL 3+. Voici un exemple de jeu flash 2D : `http://www.playpacmanonline.net/`.



Exemple de Pac-Man en 3D.

La suite de ce document donne les spécifications nécessaires au développement. Comme vous le verrez, certains points sont précis et devront donc être respectés scrupuleusement, d'autres vous laissent totalement libres, et d'autres sont un peu ambigus... Et oui ! C'est souvent le problème quand on développe un programme pour un client. « Welcome to the real world ! »

1 Règles du jeu

Vous incarnez un personnage (Pac-Man) dont le but est de manger toutes les billes (Pac-Gommes) à l'intérieur d'un labyrinthe, sans se faire toucher par l'un des quatre fantômes déambulant dans le labyrinthe. L'objectif est d'obtenir le plus de points possible.

Plateau de jeu

Le plateau de jeu est constitué d'éléments statiques :
— les murs du labyrinthe ;
— les Pac-Gommes classiques ;
— 4 Super-Pac-Gommes ;
— des bonus (représentés par des fruits dans le jeu initial),
et d'éléments dynamiques :
— Pac-Man, guidé par le joueur ;
— 4 fantômes se déplaçant dans le labyrinthe.

Début de partie

Au début d'une partie, le joueur a 3 vies et 0 point. La position initiale de Pac-Man et des différents éléments est précisée dans le fichier de description du niveau. Les quatre fantômes doivent être dans un espace réservé, duquel ils sortiront au fur et à mesure.

Déroulement d'une partie

Les déplacements de Pac-Man s'effectuent le long des murs du labyrinthe sans pouvoir les traverser. Le joueur contrôle sa direction. Aux bords du labyrinthe se trouvent des passages permettant à Pac-Man de passer d'un bord à l'autre (haut $\leftrightarrow$ bas et/ou gauche $\leftrightarrow$ droite).

Les fantômes déambulent dans le labyrinthe en essayant de toucher Pac-Man.

Lorsque Pac-Man passe sur une Pac-Gomme, celle-ci disparaît et le score du joueur est augmenté de 10 points. S'il s'agit d'une Super-Pac-Gomme, les fantômes sont immobilisés temporairement. Durant ce laps de temps, Pac-Man est capable de manger les fantômes : l'ambiance générale du jeu (les couleurs, *etc.*) doit changer de manière à ce que le joueur puisse se rendre compte facilement du changement. Lorsqu'un fantôme est mangé, il retourne immédiatement à sa position initiale et le score du joueur est augmenté de 200 points. Si plusieurs fantômes sont mangés pendant le même laps de temps, les points gagnés sont doublés (200, 400, 800, 1 600).

Lorsque Pac-Man passe sur un bonus, celui-ci disparaît et le score est augmenté de 500 points.

Si le score du joueur dépasse 10 000 points, il gagne une vie.

Si Pac-Man est touché par un fantôme, le joueur perd une vie. Pac-Man et les fantômes retournent immédiatement à leur position initiale.

N.B. : Les scores présentés ici sont à titre indicatif, il faudra sûrement les adapter en fonction de vos choix.

Fin de partie

La partie se termine lorsque toutes les Pac-Gommes ont été mangées ou lorsque le nombre de vies atteint 0.

2 Spécifications

Nom du jeu

Si vous n'aimez pas « IMACMAN 3D », n'hésitez pas à nommer votre jeu autrement !

Plateau de jeu

Le plateau de jeu est représenté par une grille dont chaque cellule est identifiée par un système de coordonnées classique. Une cellule contient un unique élément du plateau (Pac-Man, fantôme, mur, Pac-Gommes, vide, *etc.*)

Il doit être décrit à l'aide d'un fichier (chargement et sauvegarde). Le format de fichier n'est pas imposé. Par exemple, vous pouvez utiliser XML, JSON, ou encore un format binaire créé par vos soins.

Fenêtre de jeu

La fenêtre de jeu est constituée d'au moins deux éléments principaux :
— le champs de jeu (le labyrinthe, *etc.*) ;
— les informations relatives à la partie en cours (nombre de vies restant, score actuel, *etc.*).

Menu

Lorsque le joueur appuie sur Echap, le jeu est mis en pause et un menu pop-up s'affiche. Ce menu permet de recommencer le niveau, de sauvegarder la partie en cours, ou de charger une partie sauvegardée.

Déplacement des personnages

Les déplacements de Pac-Man s'effectuent le long des murs du labyrinthe sans pouvoir les traverser. Le joueur contrôle sa direction à l'aide des touches Z-Q-S-D (Haut-Gauche-Bas-Droite). Vous devrez donc déterminer à tout moment si le déplacement demandé par l'utilisateur est possible (*i.e.* s'il n'y a pas de mur sur le passage). Ces changements de direction ne sont possibles qu'aux croisements du labyrinthe.

Les fantômes sortent de leur emplacement initial à tour de rôle avec un pas de temps fixé. Ils déambulent ensuite de manière aléatoire dans le labyrinthe, à vitesse constante. Ces changements aléatoires de direction ne sont possibles qu'aux croisements du labyrinthe. Une gestion particulière devra être mise en place pour sortir les fantômes de leur emplacement initial.

Contrôle de la caméra

Le jeu doit contenir deux types de caméra :

— une caméra « vue du dessus », avec le possibilité de zoomer sur Pac-Man (avec la souris, mais n'utilisez pas la molette) ;
— une caméra permettant de voir à travers les yeux de Pac-Man.

À tout moment, le joueur peut changer de vue en appuyant sur la touche C.

Apparence du jeu

Vous êtes libres de choisir le style général du jeu. Par exemple, les murs peuvent être des haies, Pac-man un chat, les fantômes des chiens, et les Pac-gommes des souris. Pour charger un modèle 3D (par exemple un .obj), ne codez pas la lecture du fichier, utilisez une bibliothèque telle que Assimp (`http://assimp.sourceforge.net/`).

Vos seules obligations sont :
— utiliser au moins une texture ;
— utiliser plusieurs lumières (par exemple, les Super-Pac-Gommes peuvent produire de la lumière) ;
— la scène doit être englobée dans une boîte texturée (de votre choix), par exemple un ciel nuageux ;
— Le jeu doit être jouable sans faire perdre 5 points à chaque œil du joueur.

3 Extensions possibles

Une fois l'ensemble des fonctionnalités implémentées, testées et validées, vous pouvez améliorer votre programme en proposant par exemple (attention, certaines extensions sont beaucoup plus complexes que d'autres) :
— prise en compte du temps pour finir le niveau dans le calcul du score final ;
— jeu à deux joueurs sur le même clavier ;
— comportement intelligent des fantômes (vous pouvez vous inspirer du jeu initial, *cf.* Annexe A) ;
— génération aléatoire de labyrinthe ;
— niveaux sur plusieurs étages, accessibles par des rampes ;
— animation des personnages (par exemple, Pac-Man ouvre et ferme la bouche quand il avance) ;
— ajout d'effets visuels (brouillard, « trainée » de particules suivant les déplacements...)
— rendu des ombres en utilisant des shadow maps ;
— système pour sauvegarder les meilleurs scores...

Si vous avez d'autres idées, n'hésitez pas ! Par contre, pensez à nous soumettre votre idée avant de vous y mettre. Ce serait dommage de travailler sur quelque chose que nous trouvons sans intérêt...

4 Contraintes techniques

— Groupe de 3 étudiants.
— Programmation en C++.
— Rendu avec OpenGL 3+ (utilisez les shaders, pas de pipeline fixe).
— Fonctionne au moins sous Linux : programme qui compile avec `g++` et exécutable sur les machines de la fac (typiquement celles de la salle 1B077).
— Gestion de la fenêtre et des évènements avec la SDL.
— Pas de fuite mémoire ! (utilisez Valgrind pour vérifier).
— Bonne architecture logicielle (par exemple, séparez bien le côté « Moteur de jeu » du côté « Moteur de rendu »).
— Code propre, commenté de façon pertinente.
— Utiliser un outil de gestion de versions (type Git) pour partager votre code.

5 Aide à la gestion de projet

1. Avant de vous lancer tête baissée dans le code, vous devez réfléchir ensemble à l'architecture de votre programme. Cette étape est essentielle à la réussite du projet : une architecture mal réfléchie au départ entraînera des modifications par la suite et donc un retard possiblement conséquent dans l'avancement du projet. Néanmoins, ne passez pas trop de temps sur cette réflexion, l'objectif est d'avoir une architecture simple, répondant aux attentes des spécifications. Il n'est pas nécessaire que votre programme soit ultra-modulable.
2. Optez pour de petits cycles de développement (type analyse, spécification, développement, test, validation).
3. Découpez le projet en tâches courtes, intégrables et testables rapidement, pour pouvoir mieux jauger l'avancement du projet.
4. Définissez, en amont, le rôle de chacun en fonction de ses compétences et ses affinités.
5. Utilisez un outil de gestion de versions type GIT, par exemple GitHub (`https://github.com/`) ou Bitbucket (`https://bitbucket.org/`).
6. Utilisez un outil de répartition et suivi de tâches, par exemple Trello (`https://trello.com/`) ou Azendoo (`https://app.azendoo.com`)
7. Codez une surcouche à OpenGL pour faciliter le développement. Elle pourrait contenir des classes simplifiant la gestion des ressources (VBO, textures, *etc.*) ou encapsulant des techniques utilisées régulièrement.
8. Ne passez pas les trois quarts du projet à modéliser de beaux objets pour remplacer Pac-Man ou les fantômes. Vous aurez une meilleure note si le programme fonctionne bien mais que les personnages sont de simples sphères, qu'avec de beaux assets et un programme tout buggué...

6 Livrables

La veille de la soutenance (*i.e.* le 11/01/2017), vous devrez nous fournir par mail (biri@u-pem.fr/ maxime.maria@esiee.fr) :

— Le code source : donnez nous accès à votre dépôt Git ;
— un rapport en PDF (environ 4 pages, hors page de garde, tables des matières, *etc.*), contenant au moins :
  — un mode d'emploi ;
  — une description simplifiée de l'architecture du programme (pas le détail de toutes les classes, concentrez vous sur l'essentiel) ;
  — la justification de vos choix ;
  — une présentation de la gestion de projet (qui a fait quoi, ce qui a fonctionné ou non, *etc.*) ;
  — les résultats.

7 Soutenance

Vous devrez présenter votre projet lors d'une soutenance d'une quinzaine de minutes, devant vos camarades et nous. Votre présentation devra comporter :

— une démonstration de votre jeu, en présentant les diverses fonctionnalités (spécialement les fonctionnalités supplémentaires). L'objectif ici est de « vendre » votre jeu. Imaginez que nous sommes des éditeurs, et que vous cherchez à éditer votre jeu : vous devez nous en mettre plein la vue !
— une présentation succincte de l'architecture de votre programme et des bibliothèques utilisées (pas OpenGL, ni SDL) ;

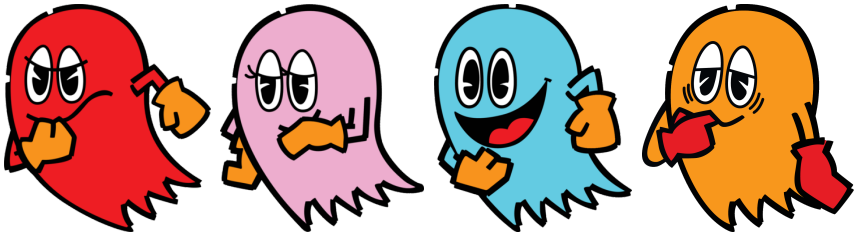
— un résumé de votre gestion de projet (qui a fait quoi, difficultés rencontrées, outils utilisés, si c'était à refaire vous changeriez quoi ?, *etc.*).

Nous attendons de vous une soutenance claire et fluide. Vous devez donc bien la travailler, et faire plusieurs répétitions avant le jour J (et pas avant la minute M). La présentation orale de ses travaux devant un public n'est pas une tâche triviale ! Voyez cette soutenance comme un entraînement car, dans vos futurs emplois, vous serez souvent amenés à effectuer ce type d'exercice.

A Comportement des fantômes

Dans le jeu initial, les 4 fantômes sont différents, ils ont un nom, une couleur et un comportement distincts (ça me rappelle un concept de C++ ça tiens !) :

1. Shadow (ombre) - Blinky (rouge) : suit Pac-Man en permanence ;
2. Speedy (rapide) - Pinky (rose) : vise l'endroit où se dirige Pac-Man ;
3. Bashful (timide) - Inky (bleu) : de temps en temps, part dans la direction opposées de Pac-Man ;
4. Pokey (bébête ?) - Clyde (orange) : de temps en temps, change de direction.



Les 4 fantômes de Pac-Man.

Lorsque Pac-Man a mangé une Super-Pac-Gomme, les fantômes fuient : ils sont plus lents et changent de directions aléatoirement.

Pour gérer le comportement des fantômes, il sera nécessaire de considérer le plateau comme un graphe ! Inutile de changer sa représentation mathématique mais il faudra sûrement adapter les algorithmes de parcours de graphes (Dijkstra par exemple) à cette structure de données particulière (ici une grille homogène).