**Projet C – Jeu vidéo**

# Objectif

Le but de ce projet C est de programmer un jeu vidéo 2D. Celui-ci est choisi à la fois assez simple pour être fait dans les délais, et assez complexe pour être intéressant à jouer. Il a pour but de vous familiariser avec l'utilisation de la librairie SDL, pouvant gérer le son, les entrées (clavier, manettes...), l'affichage de formes géométriques et images, et pas mal d'autres choses qui ne seront pas forcément utiles au projet (voir la page Wikipedia par exemple).

Je vous propose de partir d'un code très simple – mais assez limité – implémentant un Snake basique : `https://github.com/mahiuchun/Snake-SDL/tree/master/Snake-SDL`. Ensuite, pour améliorer cette ébauche de jeu vous devrez vous documenter. openclassrooms a quelques pages intéressantes, par exemple par ici pour la SDL et par là pour la gestion des collisions.

*Note :* ce ne sera pas précisé, mais au fil des questions vous êtes invités à séparer le code en plusieurs fichiers, à écrire un Makefile ...etc.

# Partie A : Améliorations

## Introduction

Faites en sorte que le jeu puisse passer en mode plein écran, et revenir en mode fenêtré. Aide. Au passage augmentez éventuellement la taille de la fenêtre dans le code (640x480 peut paraître petit pour les tests).

## Étape 1 : passage au modèle torique

Il est assez pénible de mourir dès qu'on touche un mur. Diverses versions alternatives de Snake utilisent une carte torique : le bord haut est connecté avec le bord bas, et le bord gauche avec le bord droit. Implémentez cette fonctionnalité.

## Étape 2 : passage à un modèle continu

L'expérience utilisateur serait plus sympa (liberté accrue) si le serpent se déplaçait sur tous les pixels et non sur une grille prédéfinie. Pour cela il faudra s'intéresser à la constante TILE_SIZE (et d'ailleurs la faire disparaître).

Ensuite, il serait agréable de pouvoir se déplacer dans toutes les directions. Il faudra pour cela redéfinir le comportement des touches directionnelles : un appui sur "up", "down", "left" ou "right" doit modifier la trajectoire "continûment", par exemple par paliers de 5 degrés (au lieu de 90 degrés dans la version initiale). Attention la mise à jour de la position du serpent (tête + corps) devient plus complexe.

*Note :* on peut éventuellement diminuer la vitesse du serpent dans les virages.

### Étape 3 : pommes et collisions

À ce stade vous devrez très probablement modifier le code calculant si le serpent mange une pomme, ou s'il entre en collision avec lui-même. On peut par exemple garder en mémoire les positions de chacune de ses "cellules", considérées comme des cercles (de rayon indiqué dans le code), puis calculer des intersections de cercles. Même principe pour les pommes.

## Partie B : extensions

L'univers de jeu est jusqu'ici très limité : un tore, un serpent, des pommes. On se propose d'y ajouter un peu de piment :

- des "mines" qui tuent le serpent si sa tête y passe, suppriment ou ajoutent une cellule si son corps les touche (suppression intéressante si le but du jeu est de maximiser la longueur ?) sinon l'ajout est plus logique car augmentant la difficulté.
- éventuellement d'autres types de fruits : optionnel mais ajoute de la diversité à pas cher (à vous de trouver des images).
- des murs tuant le serpent si sa tête les touche ; arrêtant son corps si celui-ci les touche (ajout d'une contrainte physique).
- des bonus : étoile d'invincibilité "à la Mario Kart", missiles permettant de détruire certains obstacles (touche espace par exemple).

Enfin, ajoutez du son :

- à chaque passage sur un fruit, sur un bonus ou sur une mine.
- lorsqu'on entre en collision avec un mur.
- éventuellement un fond sonore (pas nécessaire je pense, mais pourquoi pas).

## Partie C : améliorer le gameplay

On dispose à présent d'un joli jeu dans lequel un serpent progresse en croquant des fruits et en évitant des obstacles... mais ce n'est pas suffisant pour accrocher le joueur. Il faudrait y ajouter au minimum un score – que le joueur essayerait d'améliorer.

Finalement, un mode deux joueurs rendrait vos tests plus amusants. Ce mode utiliserait potentiellement deux threads : un par joueur ; voir par exemple https://computing.llnl.gov/tutorials/pthreads/. Les contrôles seront à définir pour le joueur 2, et les règles du jeu aussi. Quelques possibilités :

- les serpents s'allongent indéfiniment et le but est de piéger l'adversaire.
- l'objectif est de détruire le serpent adverse (missiles, bombes, ...).
- coopération, les deux joueurs affrontant des ennemis à définir...

Dans ce mode il peut être intéressant de permettre aux serpents de s'arrêter, changer éventuellement de direction puis repartir.

Quelques autres propositions pour finir et diffuser le jeu :

- Progressions à travers des niveaux, l'objectif étant de manger $n$ fruits par exemple.
- Un mode "labyrinthe" ? ...peut-être en niveau caché ?
- Scénario ? (motivation pour passer d'un niveau à l'autre).
- Boss final ? (par exemple un aigle dont on verrait l'ombre, qu'il faudrait éviter puis frapper lorsqu'il se pose − comme dans Zelda64 et sûrement d'autres jeux plus récents... :-) )

# Conclusion

SDL2 peut être compilé pour Android/iOS. Moyennant quelques efforts vous devriez pouvoir jouer à votre création sur votre téléphone. Par exemple si vous êtes sous Windows : http://lazyfoo.net/tutorials/SDL/52_hello_mobile/android_windows/index.php

# Informations

Le projet se fera par groupes de deux étudiants. L'évaluation du projet portera essentiellement sur le code rendu, mais aussi sur le compte-rendu et la soutenance.

Concernant le rapport, considérez le comme l'aide de votre jeu : il doit

- expliquer comment compiler le programme − idéalement, fournissez un Makefile ;
- illustrer son utilisation, ses options par divers exemples ;
- lister les bugs connus (6 semaines c'est court, il y en aura sans doute) ;
- parler éventuellement des choses que vous avez essayées mais qui n'ont pas abouti, ou encore des améliorations que vous n'avez pas apportées faute de temps, etc.

Pour la soutenance prévoyez de parler de votre code pendant quelques minutes avec une petite démo, montrant les points forts/faibles, ce qui reste à faire et comment, ...etc.