UVHC
UNIVERSITE DE VALENCIENNES
ET DU HAINAUT - CAMBRESIS

INSTITUT
DES SCIENCES
ET TECHNIQUES
DE VALENCIENNES

Projet:

Jeu de Snake.

Compte-rendu.

Stéphan DONIN
Enseignant tuteur: S. ESTIVIE

Sommaire

Stéphan DONIN
Enseignant tuteur: S. ESTIVIE

Présentation du sujet

Règles du jeu originales:

« Snake, de l'anglais signifiant « serpent », est un jeu-vidéo populaire créé au milieu des années 1970 (…) . Il s'est de nouveau fait connaître dans les années 1990 avec l'émergence du nouveau support de jeu qu'est le téléphone portable. »
« Le joueur contrôle une longue et fine créature semblable à un serpent, qui doit slalomer entre les bords de l'écran et les obstacles qui parsèment le niveau. Pour gagner chacun des niveaux, le joueur doit faire manger à son serpent un certain nombre de pastilles, allongeant à chaque fois la taille de la bestiole. Alors que le serpent avance inexorablement, le joueur ne peut que lui indiquer une direction à suivre (en haut, en bas, à gauche, à droite) afin d'éviter que la tête du serpent ne touche les murs ou son propre corps.
Le niveau de difficulté est contrôlé par l'aspect du niveau (simple ou labyrinthique), le nombre de pastilles à manger, l'allongement du serpent et sa vitesse. »

D'après le site http;://fr.wikipedia.org/

Adaptation des règles:

Pour mon projet, j'ai choisi de simplifier les règles citées ci-dessus dans un premier temps, pour les complexifier par la suite.
Voici ce qui a été pensé dans un premier temps:

- Présence de quatre murs non pénétrables, formant la zone dans laquelle le joueur fera évoluer le serpent.
- Le joueur ne pourra qu'indiquer la direction dans laquelle le serpent doit aller (haut, bas, gauche, droite) et le serpent ne pourra donc pas s'arrêter: il avancera en continu jusqu'à la fin du jeu.
- Le serpent ne pourra pas faire demi-tour.
- Le serpent ne pourra pas « passer au-dessus » de sa queue.
- Apparition de petites boules de couleur jaune symbolisant la nourriture standard pour le serpent. Cette dernière rapportera donc des points, et apparaîtra avec des coordonnées aléatoires.
- Apparition d'une petite boule de couleur rouge toutes les 15 boules jaunes mangées. Cette boule rouge représente de la nourriture dite spéciale pour le serpent, puisqu'elle apparaîtra beaucoup moins souvent que la boule jaune, ne restera pas affichée à l'écran de manière permanente (dans le cas où elle n'est pas mangée par le serpent bien sûr), mais rapportera beaucoup plus de points.
- Les cas de défaite du joueur seront alors les suivants:

_ le serpent heurte un mur ou son propre corps.
_ le serpent heurte un rebord du jeu.

Voici maintenant ce à quoi j'ai pensé comme complexification des règles décrites ci-dessus:

- Sauvegarde des meilleurs scores faits par les différents joueurs.
- Mise en place d'obstacles sur le niveau.
- Intégration de différents niveaux de jeu.
- Intégration de bonus de jeu liés aux meilleurs scores sauvegardés par le système (exemples: nouvelles couleurs du serpent, nouveau design, nouveaux fonds d'écran...)
- Personnalisation du fond d'écran de jeu.
- Choix de la difficulté de jeu.
- Intégration d'une fonction de pause.
- Choisir si la collision avec les rebords est pénalisante ou si le système permet au serpent d'être téléporter sur le rebord opposé au moment de la collision (partie de son corps, par partie de son corps, et non son corps dans la totalité).

Choix du langage

JAVA

J'ai donc choisi JAVA comme langage pour la réalisation de ce projet, et ce, pour plusieurs raisons me paraissant évidentes.
Tout d'abord, JAVA est le langage que je maîtrise le plus jusqu'à maintenant, et pour me lancer dans un projet tel qu'un jeu-vidéo – terrain inconnu du développement pour moi auparavant – il me semblait plus prudent d'utiliser un langage avec lequel je suis le plus à l'aise, pour des soucis de productivité et de capacité à résoudre des problèmes liés au langage également.
De plus, JAVA est un langage qui, selon moi, possède une documentation de qualité et la possibilité de compléter la documentation officielle de Sun par d'autres documentations officieuses et autres forums d'entraide s'offrait alors à moi, de par le fait que c'est un langage utilisé par beaucoup de développeurs, à tout niveau.
Le développement de jeu-vidéo fait forcement appel à des bibliothèques graphiques et autres interactions visibles du côté utilisateur. JAVA est jusqu'à présent le seul langage de développement (hors développement Web) dont j'ai étudié un début de comportement graphique, et cela fait pour moi une raison de plus d'utiliser ce langage pour mon projet.
JAVA étant également un langage présentant des caractéristiques de portabilité intéressantes, si je souhaitais par la suite exporter mon projet sur des architectures différentes que celles d'un ordinateur personnel, JAVA serait, parmi les langages que je connais, le langage le plus convenable.

Analyse du sujet, structures de données, fonctionnement

JAVA étant un langage orienté objet, mes structures de données allaient facilement être implémentées et réparties au sein de différentes classes. Dans le cadre de la première version du projet (c'est-à-dire, avec les règles basiques décrites plus haut), j'ai alors ainsi que j'ai obtenu les classes suivantes:

- Food: Cette classe représente la nourriture standard évoquée plus haut. Ainsi, cette classe contiendra la position courant de la boule jaune sur l'écran, en X et en Y, et contiendra également la hauteur et la largeur qu'occupe cette boule sur l'écran.
- SpecialFood:Cette classe est une classe héritant de la classe *Food*, décrite ci-dessus. Elle ne possédera en plus que des attributs et des méthodes de contrôle de son arrivée et sa sortie du jeu, étant donné qu'elle ne possèdera qu'un temps limité de « survie » sur l'écran de jeu avant d'être mangée par le serpent.
- Snake: Cette classe représente toutes les informations concernant le serpent tel qu'il apparaît à l'écran; ainsi, grâce à cette classe, on obtient la hauteur et la largeur que doit occuper une partie de corps du serpent, ainsi qu'une liste contenant des objets de classe *SnakeBody*, qui est une classe décrite ci-dessous.
- SnakeBody: Cette classe ne comprend pas toutes les informations concernant le serpent tel qu'il apparaît à l'écran, mais comprend toutes les informations essentielles concernant ce dernier, lorsque cette classe est contenue dans une liste, avec d'autres instances de la même classe. C'est à travers elle que l'on va représenter une « partie du corps du serpent », avec sa position sur l'axe X et l'axe Y, ainsi que la direction dans laquelle une partie du corps du serpent donnée se dirige (car, rappelons que le joueur ne contrôle pas le déplacement du serpent, mais seulement la direction dans lequel ce dernier se déplace).

NB: N'ont été décrites que les classes m'ayant sembler nécessaires de l'être; d'autres classes sont évidemment présentes, mais il n'y a nul besoin de les présenter au sein de ce rapport.

Avec ces classes, il sera alors possible de représenter graphiquement un serpent dans son environnement, à savoir: quatre murs, de la nourriture standard apparaissant de façon constante (c'est-à-dire dès que l'ancienne instance de la classe est mangée par le serpent, une nouvelle instance sera alors générée) *(voir annexe 1)* , ainsi que de la nourriture spéciale, avec un comportement comme décrit plus haut *(voir annexe 2)*.
Maintenant, il faut rendre tout ce système intelligent, et interactif avec le joueur. La façon le plus évidente et la plus instinctive pour un joueur de contrôler le serpent est bien évidemment les touches directionnelles du clavier de l'ordinateur mettant en place le système. J'ai alors étudier le comportement des classes *KeyListener* et *KeyEvent* (via l'API JAVA proposée par Sun) afin de donner au joueur la possibilité de contrôler le serpent avec le serpent, mais également un peu plus tard, de quitter le jeu ainsi que de faire un pause via les touches ESCAPE et SPACE du clavier. Ces

écouteurs de clavier seront alors actif en même temps que la méthode *move()* que j'ai développé.
Cette méthode, avec l'aide d'autres méthodes et d'exceptions développées à travers d'autres classes, a pour but de générer les déplacements du serpent, selon les directions des différentes parties du corps du serpent. Elle contrôlera également si le serpent ne se mord pas lui-même, ou bien s'il n'entre pas en collision avec un mur via des exceptions qui arrêteront le jeu et le quitteront en signalant au joueur la cause de la défaite *(voir annexe 3 et 4)*.

A ce stade d'avancement du projet, nous avons un jeu de Snake désormais intelligent (par là, j'entends qu'il gère les collisions, ainsi que le fait que le serpent ne doit pas manger son propre corps) et interactif par le fait que le joueur peut désormais déplacer son serpent à l'écran.
Il paraît alors intéressant de mettre à disposition du joueur une petite interface lui permettant de choisir un niveau de difficulté, ainsi qu'un indicateur du nombre de point qu'il a marqué à l'instant t.
Voici l'échelle des points que j'ai mis au point, selon la difficulté sélectionnée et selon le type d'aliment avalé:

ECHELLE DES POINTS	Nourriture standard	Nourriture spéciale
Facile	2	10
Moyen	5	25
Difficile	10	50

Comme la logique le voudrait, plus la difficulté est élevée, plus le joueur marque de points en mangeant un aliment (n'importe lequel des deux), mais plus le serpent sera rapide en contre-partie. A contrario, plus la difficulté est basse, moins le joueur marquera de points en nourrissant le serpent, mais plus ce dernier sera lent et il sera alors beaucoup plus simple d'éviter les différentes collisions.
Pour choisir la difficulté, le joueur passera par une interface accessible depuis une barre de menu standard (*voir Annexe 5 et 6)*, afin que n'importe quel utilisateur ne soit pas perdu. La difficulté choisie par défaut par le système sera la difficulté « Moyen ».
Concernant les compteurs de points, le score actuel est actualisé à chaque fois que le serpent se nourrit (quelque soit le type de nourriture), et le compteur des meilleurs scores, quant à lui, est mis à jour à chaque fin de partie, et vérifie ainsi si le score marqué par le joueur venant de terminer une partie est supérieur à celui déjà enregistré, ou non.

A ce stade, nous avons donc un jeu de Snake opérationnel, comptant les scores et avec un choix de difficulté possible. J'ai également veillé à ce qu'un petit côté soit personnalisable: le fond d'écran. En effet, le design n'étant malheureusement pas très poussé, j'ai souhaitais que le joueur puisse au moins chargé n'importe quelle photo en fond d'écran de sa partie. Ainsi, en indiquant en paramètre, au lancement du programme, le chemin d'accès de la photo que le joueur veut charger, le système va alors tenter de charger sa photo et charger le fond bleu par défaut si jamais il ne parvient pas à charger la photo proposé par le joueur. Cependant, si le chargement de la photo proposée par le joueur rencontre un succès, cette dernière sera chargée, et NON-redimensionnée; conséquence: les bords seront alors recalculés par le programme et le joueur pourra indirectement décider de la taille de l'environnement dans lequel il souhaite faire évoluer sa partie. *(voir Annexe 7)*.

Travail effectué / Fonctionnalités prévues initialement

Voici un tableau récapitulatif de ce qui avait été prévu en analyse préalable du projet, et de ce qui a été effectué au niveau du produit final:

Fondation d'un environnement basique (4 murs non pénétrables)	**OK**
Contrôle du joueur exclusivement sur la direction du serpent	**OK**
Impossibilité pour le serpent de faire demi-tour	**OK**
Le serpent ne peut pas se mordre lui-même	**OK**
Génération de la nourriture standard de façon aléatoire	**OK**
Génération et gestion de la nourriture spéciale	**OK**
Sauvegarde des meilleurs scores	**OK**
Mise en place d'obstacles sur le niveau	**-**
Intégration de différents niveaux de jeu	**-**
Intégration de bonus liés à la sauvegarde des meilleurs scores	**-**
Personnalisation des fonds d'écran de jeu	**OK**
Choix de la difficulté	**OK**
Fonction de pause	**OK**
Choix de la collision avec les parois du niveau est pénalisante ou non	**-**

NB: Tout le code source est également à la JAVADoc et le code source est donc totalement commenté.

Stéphan DONIN
Enseignant tuteur: S. ESTIVIE

Difficultés rencontrées

Je n'ai pas rencontré de difficultés majeures au cours du développement de mon projet. Les seules difficultés que j'ai rencontré ne sont se sont posées que sur des légers détails; ces détails sont allées à l'encontre de la réussite et du résultat final de mon projet, mais ont été surmontées assez rapidement, grâce à une certaine prise de recul et une capacité à réfléchir sur le langage et ses conventions. Par exemple, la toute première difficulté que j'ai rencontré fut la surcharge, dans une classe que j'ai développé, d'une méthode héritée d'une classe de JAVA. Initialement, ne savant pas que je surchargeais une méthode déjà existante (étant transmise à travers plusieurs classes, elle n'apparaissait pas dans l'API JAVA de Sun pour la classe héritée directement), la méthode surchargée était censée ne bouger que mon serpent, mais elle bougeait en fait tout le panneau graphique et décalait ainsi tout ce qui était contenu dans ce dernier. Cette erreur fut en fait commise par le fait que je nomme mes méthodes et attributs en corrélation avec les conventions d'écrire de JAVA et ce fut donc pour cela que j'ai surchargé une méthode dans m'en rendre compte et sans vraiment le vouloir.
Une autre difficulté rencontrée, celle-ci un peu plus complexe cette fois-ci, fut le fait que des décalages avaient parfois lieu au sein du corps de mon serpent, et notamment de la tête de la tête de ce dernier. Après des recherches plus ou moins longues, j'ai découvert que c'était en fait un conflit entre le processus faisant tourner ma méthode principale (celle faisant bouger le serpent) et les écouteurs de clavier de JAVA. J'ai pu régler ce problème grâce à des contrôleurs booléens dans chacune des deux méthodes, faisant le même travail que des sémaphores, si l'on se réfère à des fondements un peu plus profond de l'informatique. Ce problème fut, je pense, le plus délicat à résoudre, mais surtout à détecter, puisqu'il se manifestait de manière totalement aléatoire. Il fut cependant résolu et désormais, le programme tourne sans aucun problème de ce genre.
Une autre chose m'a également pris un peu de temps: les *GridBagLayouts*. Ils sont en fait les *LayoutManagers* les plus flexibles de JAVA, certes, mais également les plus complexes, d'après Sun (http://java.sun.com/docs/books/tutorial/uiswing/layout/gridbag.html). Cependant, leur usage m'étant indispensable pour le rendu final que je souhaitais obtenir, je pris alors la peine d'étudier ces derniers et ne fut pas déçu du résultat de mes études.

Voici donc plus ou moins les seules difficultés que j'ai rencontré au cours du développement de mon projet.

Améliorations

Je pense vraiment qu'un jeu aussi léger que le jeu de Snake doive être exportable sur des technologies mobiles et embarquées, telles que des téléphones portables, comme il fut souvent le cas dans l'histoire de ce jeu. Ainsi, devant la montée en flèche de la popularité du téléphone portable d'*Apple*, le *iPhone*, je pense que d'exporter ce projet sur le *iPhone* serait une tâche intéressante. J'ai ainsi commencé à me renseigner à ce propos, et il semble, selon mes études peu approfondies sur le sujet, que le langage acceptée par l'*iPhone* pour ses applications soit l'*Objective C*. Cependant, il existe des traducteurs JAVA ↔ *Objective C*, pouvant ainsi permettre l'exportation de mon projet. Reste à vérifier ensuite l'efficacité de ce genre de traducteurs, bien évidemment.
J'ai également pensé à exporter ce jeu de Snake sur les téléphones portables de la marque *BlackBerry*, qui semblent tout de même beaucoup plus ouverts qu'*Apple* sur la politique concernant leurs applications. J'ai également pensé téléphones portables équipés des micro-logiciels *Symbian OS* (tels que les *Nokia*), *Windows Mobile*, ou bien encore, plus récemment, *Androïde*, proposé par le géant *Google*, et qui semble être compatible avec le langage JAVA.

Si le temps m'avait permis de terminer les quelques aspects manquant de mon projet (montrés plus haut dans la rubrique intitulé *Travail effectué / Fonctionnalités prévues initialement)*, je me serais certainement pencher sur l'exportation de mon projet vers des technologies mobiles telles que celles citées ci-dessus.

Apports personnels

Ce projet est ma première expérience en terme de développement de jeu-vidéo. Je ne retire strictement que du bien de cette expérience. Tout d'abord, personnellement parlant, je suis un passionné de jeu-vidéo, et ce fut alors plutôt enrichissant de voir comment un jeu-vidéo, même à bas niveau, est fait du côté développeur plutôt que du côté utilisateur. De plus, ayant connu le jeu de Snake à travers plusieurs des téléphones portables que j'ai eu la chance d'acquérir par le passé, j'affectionne particulièrement ce jeu; ceci a rendu l'avancement de mon projet d'autant plus agréable.

Ensuite, d'un point de vue technique et scolaire, j'ai trouvé que le développement de jeu-vidéo fait appel à beaucoup de connaissances diverses. Un tel développement fera appel à des connaissances en matière de programmation et d'algorithmique, cela va de soi, mais va également nécessitait une très bonne connaissance du langage utilisé (notamment pour l'optimisation des ressources prises au système hôte), et fera appel à des connaissances mathématiques également. Ainsi, dans ce projet, j'ai pu m'occuper du rendu graphique, comme de tout le travail d'algorithmique qui est fait derrière l'interface utilisateur.

Ce fut pour moi une expérience très agréable, et surtout très complète, et je suis heureux de partager ce point de vue avec plusieurs de mes camarades.

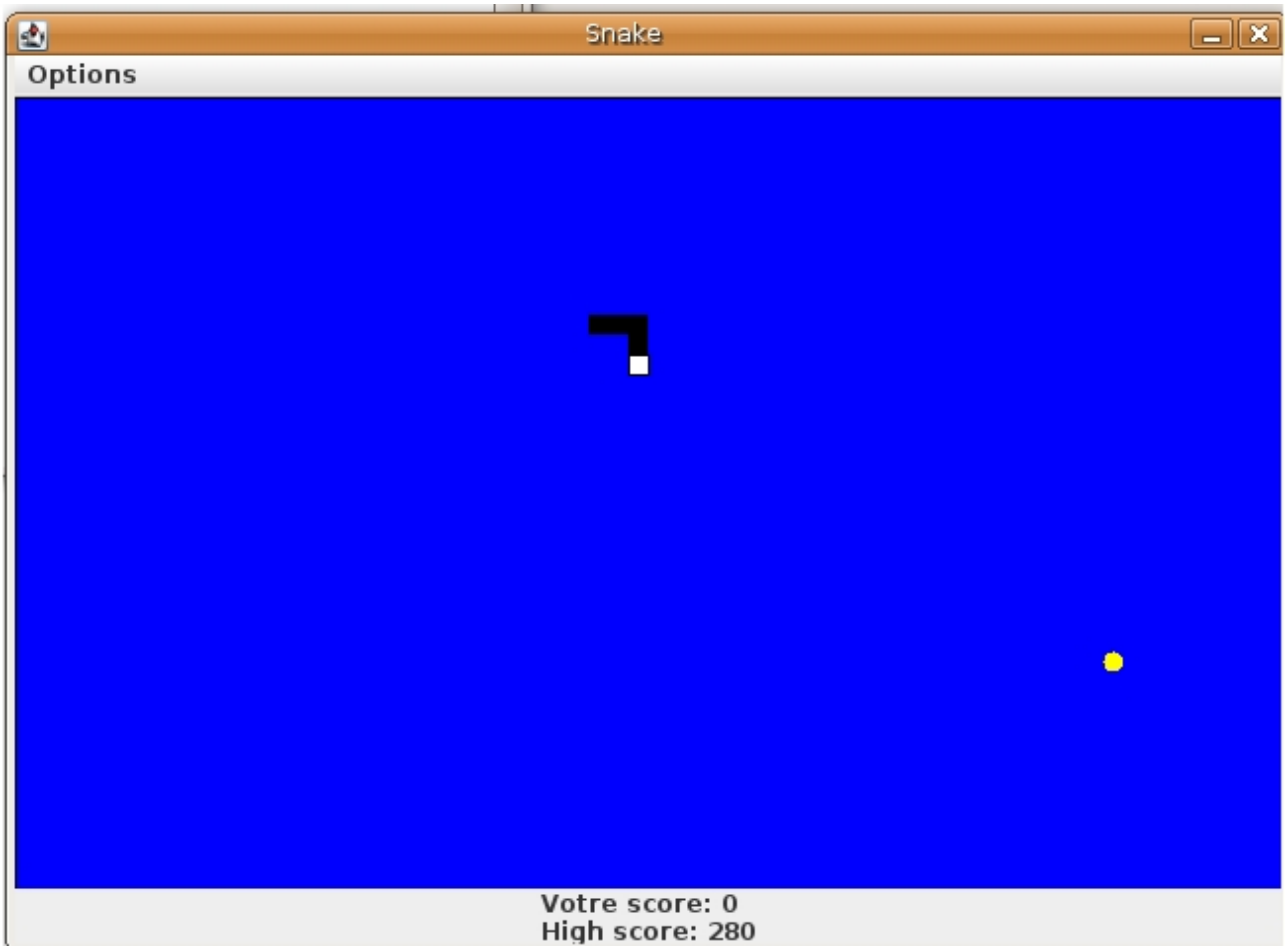
Stéphan DONIN
Enseignant tuteur: S. ESTIVIE

Bibliographie, annexe

Bibliographie

Pour ce qui est de la bibliographie, je me suis essentiellement inspiré de deux sources, à savoir Le Site Du Zero (http://siteduzero.com) pour les comportements et animations graphiques de JAVA, et surtout de l'API JAVA officielle de Sun, disponible à l'adresse suivante: http://java.sun.com/javase/6/docs/api/ .
Je ne me suis vraiment servi que de ces deux sources-ci.

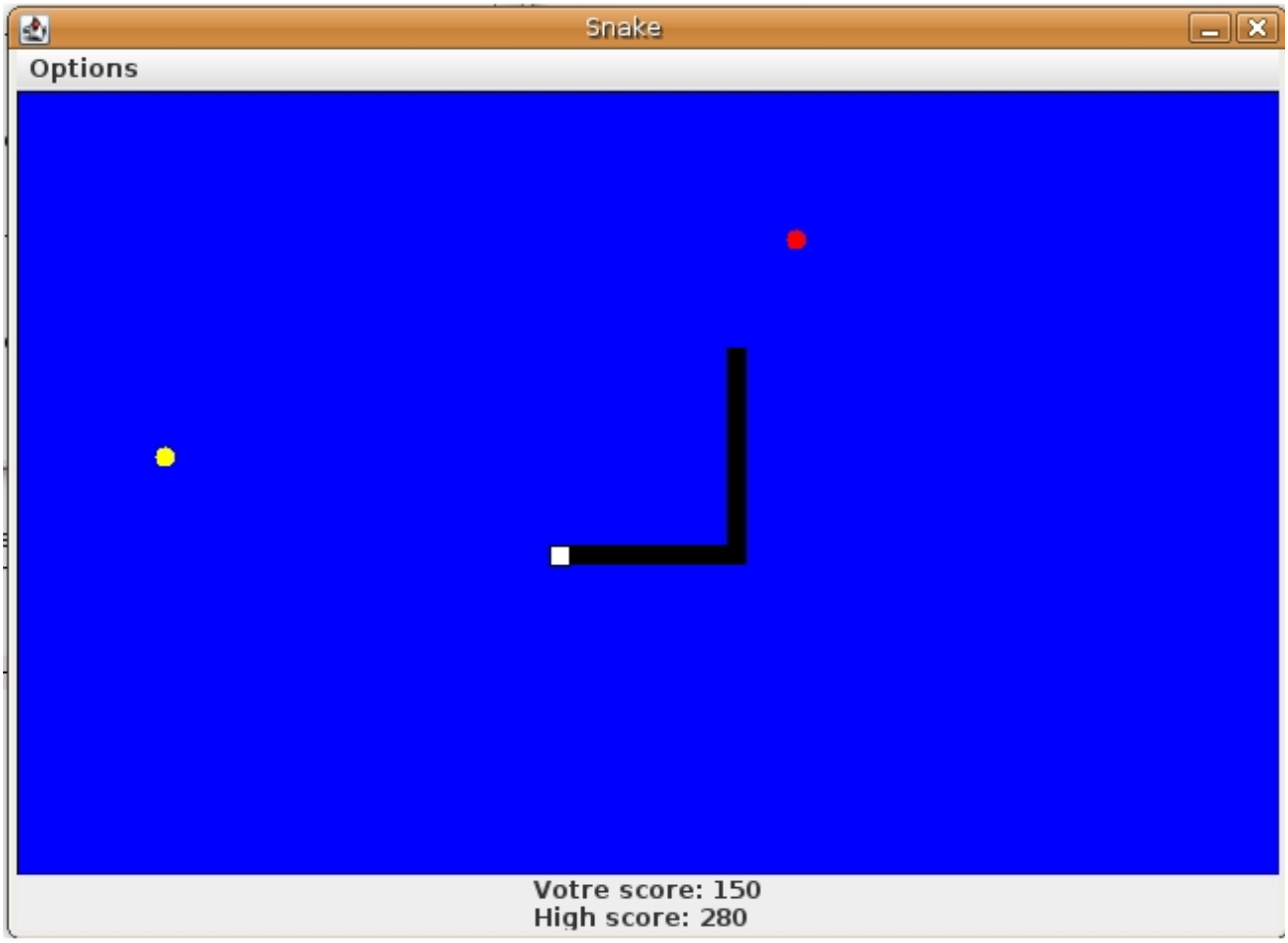
Annexe



Annexe 1

Stéphan DONIN
Enseignant tuteur: S. ESTIVIE

Le serpent à peine naissant, avec la boule jaune comme nourriture standard à engloutir.

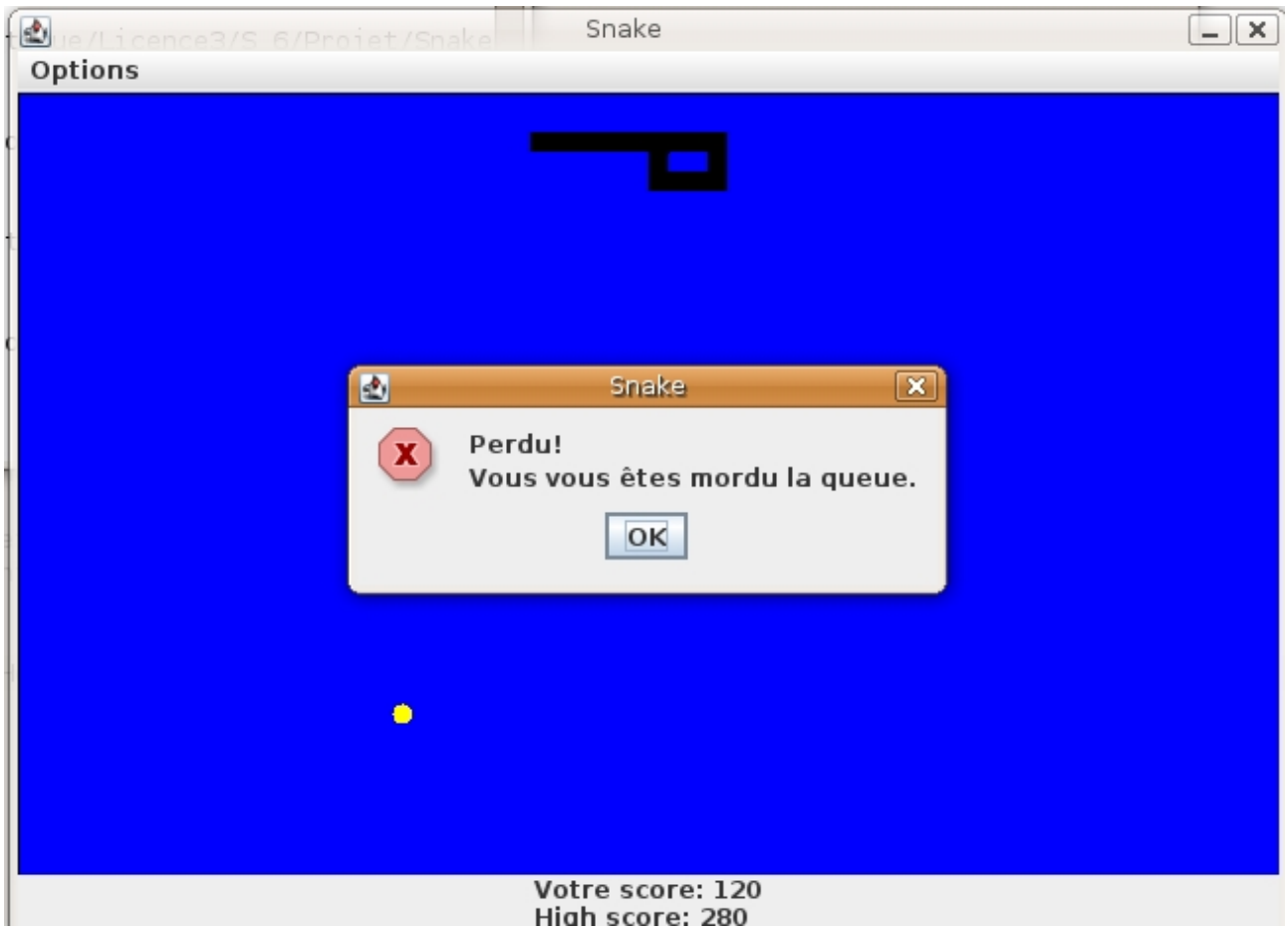


Annexe 2

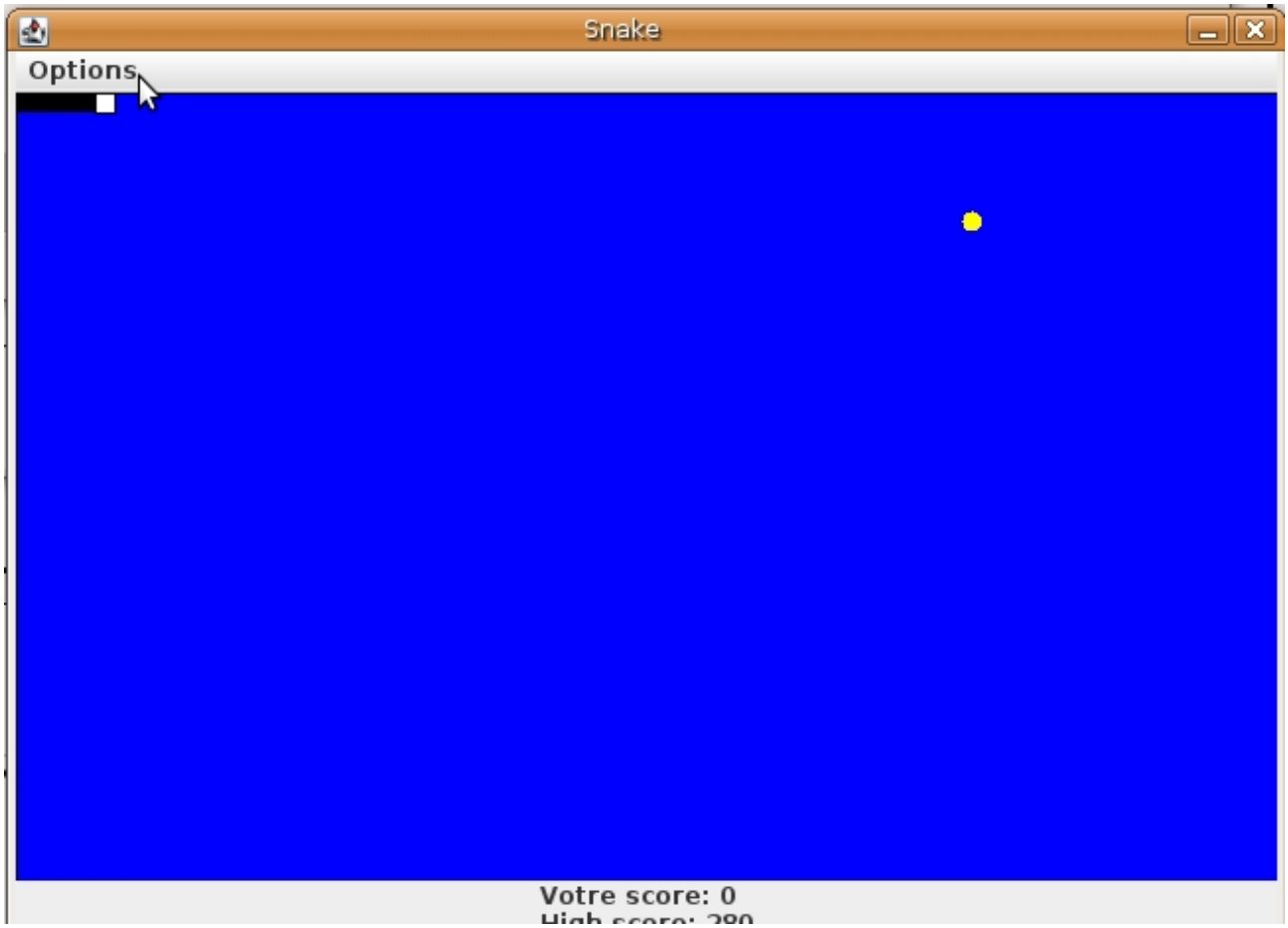
Le serpent, un peu plus grand, avec l'apparition de la nourriture spéciale, en rouge, et toujours la présence de la nourriture standard, en jaune.


Snake
Options
Snake
Perdu!
Vous avez heurté un rebord.
OK
Votre score: 0
High score: 280

Annexe 3

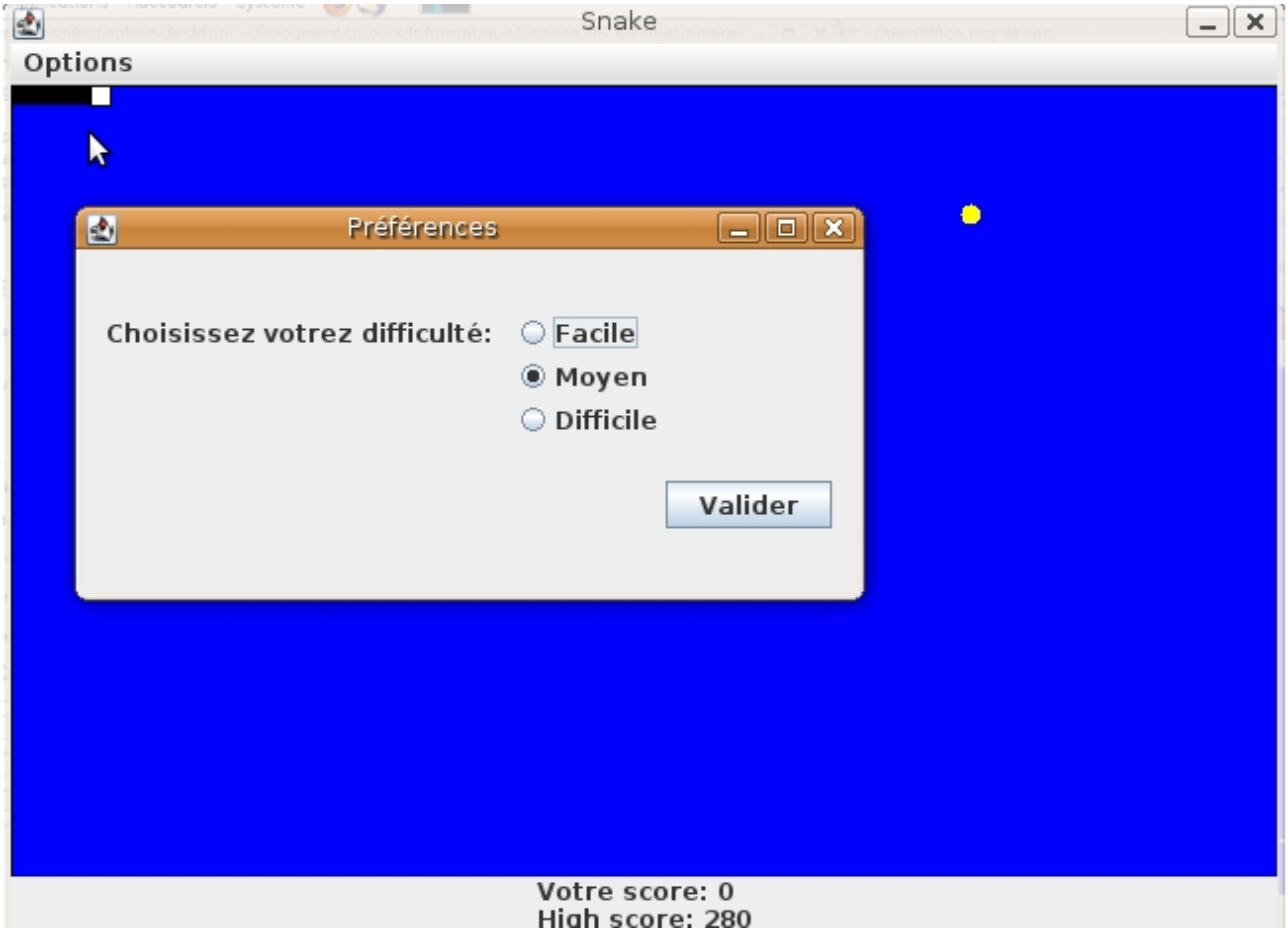

Snake
Options
Snake
Perdu!
Vous vous êtes mordu la queue.
OK
Votre score: 120
High score: 280

Annexe 4



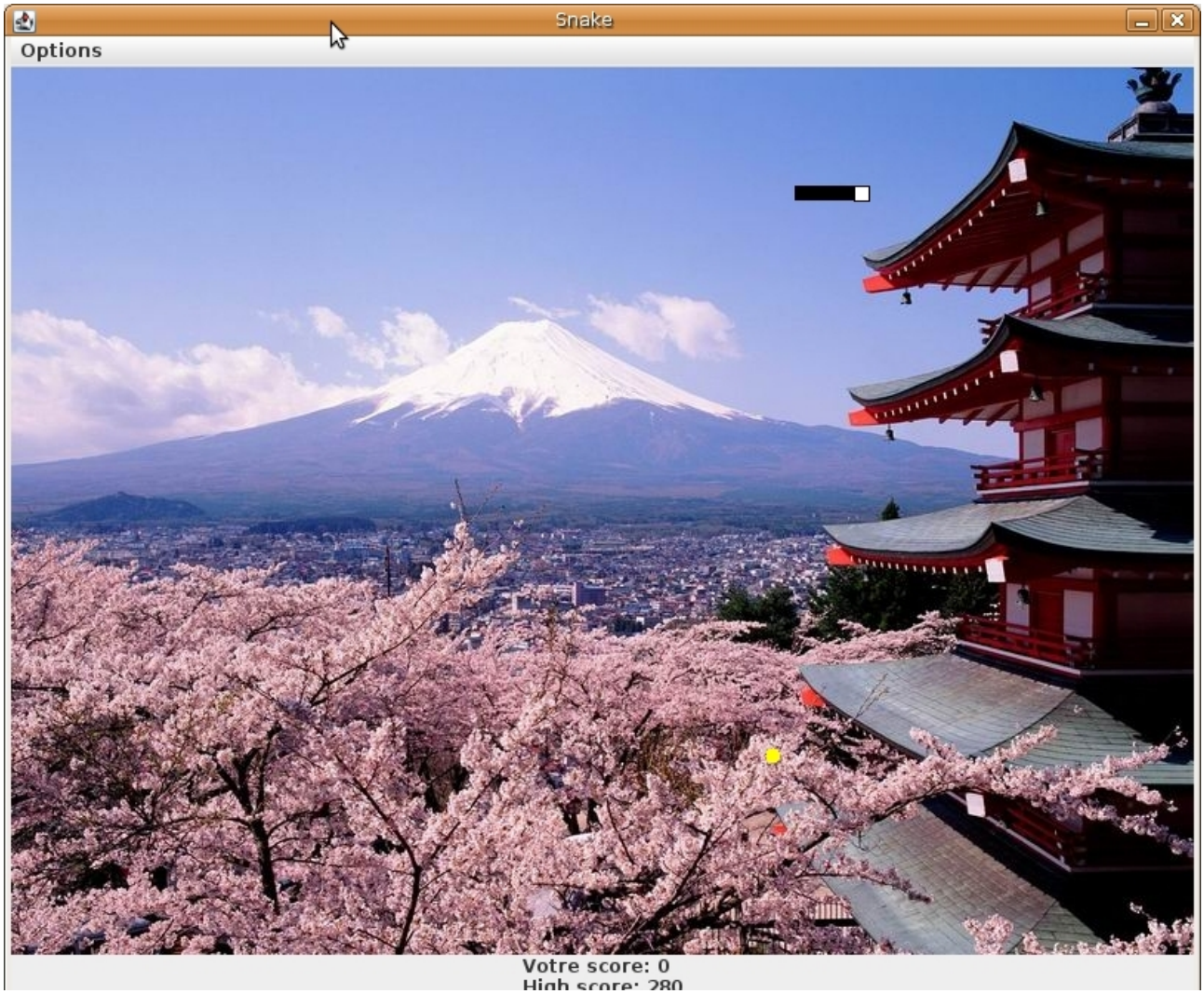
Annexe 5

Le curseur de la souris est sur la barre de menu



Annexe 6

L'interface de choix de la difficulté



Annexe 7

Jeu de Snake chargée avec une image personnelle