Rapport de projet

Mr.Classeau

Projet Nibbler

CHAMPAGNE Etienne

GUIDIS Antoine

DUT Info 1ère année 2012 / 2013

Afin de mettre en pratique les connaissances acquises pendant les cours de C du premier semestre, nous avons dû réaliser un projet. Il consistait à réaliser un jeu de Nibbler (ou Snake) sur une console de système UNIX, une façon ludique d'appliquer notre savoir. Nous avions donc dors et déjà défini le langage puisqu'il s'agit du projet final du cours de C, et fallait-il aussi faire attention à la compatibilité avec le système UNIX (certaines fonction ne fonctionnent que sur Windows et pas sur UNIX, et inversement). Le fait d'avoir eu un projet par binôme a aussi été un enjeu puisque nous avons dû apprendre à nous séparer le travail et à faire attention à rendre nos bouts de code compatibles ensembles.

I°/ Décomposition du projet en binôme

Dès que le projet nous a été communiqué, Etienne a commencé le Snake sur Windows. Parallèlement, Antoine s'est intéressé à l'aspect graphique du Snake, en trouvant des moyens d'améliorer l'affichage limité de la console. Lorsque le Snake basique sur Windows fût terminé, Antoine se chargea d'effectuer le portage sur UNIX. Seulement avec le portage des problèmes firent surface, c'est pourquoi Antoine recommenca un Snake compatible avec les systèmes UNIX, en fusionnant ce qu'il avait déjà fait avec le travail d'Etienne, pour finalement créer un nouveau Snake classique, qui, par la suite, a été optimisé au possible. Nous avons ensuite implémenté plusieurs bonus et améliorations, et ré-écrit des fonctions pour en améliorer la lisibilité/complexité.

II°/ Interface du Snake

Le snake doit être démarré sur un terminal d'au moins 30 lignes et 50 colonnes et est compatible avec les systèmes UNIX.

Une fois le programme lancé, un intro puis un menu s'affichent dans le terminal. Ce dernier dirige vers le jeu du snake classique, les modes alternatifs, le tableau des scores, les instructions de jeu ou l'option quitter.

L'option « Mode » propose une expérience de jeu différente de celle du Snake classique, mais repose sur les même mécaniques de jeu. Seul l'objectif change.
L'option « Tableau des scores » affiche le tableau des meilleurs scores.
L'option « Instructions » affiche les instructions à l'écran.
L'option « Jouer » lance une partie classique lancée, la partie commence, le but étant de faire le plus grand score en faisant manger des fruits au snake. Chaque fruit mangé fait grandir le corps du serpent de une case et fais monter le score. Si le snake tente de se manger ou fonce dans un mur, la partie est finie. Si le joueur a fait un nouveau highscore, on lui demande son nom et on affiche le tableau des scores. Sinon, on affiche simplement le tableau. Une fois une touche appuyée, on retourne sur le menu.

III°/ Explication essentielle du code

Dans le souci de rester simple, et étant donné que nos sources sont déjà très commentées, nous considèrerons que la partie essentielle repose sur les parties où il nécessite un peu plus d'attention à savoir : la fonction de déplacement, la probabilité d'apparition des différentes Food et la fonction smartSnake.

3.1°/ Déplacement

La fonction ChangeDirSnake permet à la première case du tableau theSnake, soit la tête du serpent (theSnake[THEHEAD]) de se déplacer d'UNE case vers "l'avant" en fonction de la touche appuyé, puisque les règles du jeu ne permettent pas d'aller dans la direction opposé du snake.

Dès lors il se crée un "espace" entre la tête et le reste du corps, c'est là que headPosPrec (soit la position précédente de la tête) intervient. C'est lui qui contient les coordonnées où se trouve cet espace censé représenter la deuxième section du snake.

Vient ensuite la fonction trietheSnake qui effectuera :

- la sauvegarde de la 2ème case du tableau représentant la 3ème section du snake
- l'insertion des coordonnées de la 2ème section du snake à la 2ème case du tableau avec headPosPrec
- la sauvegarde de la 3ème case du tableau représentant la 4ème section du snake
- l'insertion des coordonnées de la 3ème section du snake (qui a été sauvegardé auparavant) à la 3ème case.
- et ainsi de suite pour chaque section du snake.

En résumé, la fonction trietheSnake relie la tête et le corps du snake en triant le tableau de coordonnées puis en insérant les coordonnées de headPosPrec à la seconde case comme étant le « cou » du snake.

3.2°/ SmartSnake

La fonction smartSnake est un essai de bonus, afin de créer une « intelligence artificielle » pour le Snake. Mis à part son côté divertissant, cette fonction a permis de pouvoir faire des tests sans pour autant devoir « jouer » ce qui était plutôt agréable. De plus cela a permis de constater les nombreux cas auquel est confronté le snake durant le jeu.

Par contre, cette fonction prend en compte la tête du snake et la position où se trouve l'aliment, donc le snake est condamné à se manger lui-même tôt ou tard. Le fait de gérer la règle « le snake ne doit pas se manger lui-même » compliquait énormément le problème.

La fonction est construite de la manière suivante :

- Si le snake est dans la direction x, alors on test si l'aliment est : devant lui OU derrière lui. En fonction du résultat du test, on testera alors si l'aliment est au-dessus de lui ou en-dessous de lui.
- Parallèlement on testera si le snake se trouve sur la même ligne ou colonne ou s'il est dans cas particulier qui amène à le faire bouger d'une manière spécifique.

3.3°/ Probabilité d'apparition des nourritures

Nous avons intégré une fonctionnalité bonus : il existe différentes nourritures, qui rapportent chacune un nombre de point différents. Il faut donc pour rester logique qu'une nourriture rapportant plus de point qu'une autre apparaisse moins de fois. On définit donc une loi de probabilité appropriée. Afin d'illustrer cette dernière, on a le schéma suivant :

Schéma des probabilités (non proportionnel)

Nourriture	Couleur	Proba	Points
STAR		0.033554	+1024
COMET		0.041943,	+512
SUNSHINE		0.052429,	+256
CLOUD		0.065536,	+128
SHAMROCK		0.081920,	+64
CIRCLE		0.102400,	+32
DIAMOND		0.128000,	+16
TRIANGLE		0.134218,	+8
RECTANGLE		0.160000,	+4
SQUARE		0.200000,	+2

En résumé, la fonction rand_0_1(); nous renvoi un réel aléatoire entre 0 et 1 et selon la loi définie, la fonction alea_perso() détermine l'aliment.

Pour conclure, nous avons réussi à concevoir un snake basique et à y implémenter plusieurs bonus (graphiques comme mécaniques). Ce projet nous a aussi permis d'approfondir nos connaissances en C puisque toutes les notions nécessaires à sa réalisation n'ont pas été vu en cours.