EISTI

Mini-Projet : Application des tris

Algorithmique et programmation procédurale

Rapport de Projet

David RIGAUX

Mandry MBUNDU

18 avril 2016

Table des matières

1 Introduction

La consigne de ce mini-projet était de créer un jeu du serpent. En fait, le jeu du serpent est un jeu consistant à piloter un serpent virtuel qui grandit, à lui faire avaler des bonus, et à éviter que sa tête ne rentre en collision avec l'environnement ou avec lui-même. Pour faire ceci, nous devions utiliser les listes chaînées qui sont idéales pour modéliser un tel serpent. La méthode qu'on utilise avec ce type de structure de données est de stocker dans une liste les cases occupées par le serpent, et à chaque itération la nouvelle position de la tête est ajoutée en tête de liste alors que la position de la queue est supprimée en fin de liste (sauf si le serpent est en train de grandir). Nous devions tout d'abord faire une partie obligatoire qui consistait à écrire un programme qui permet à l'utilisateur de piloter un serpent à l'aide de quatre flèches du clavier. Et, tous les n pas de temps, le serpent grandira. Et une consigne facultative, qui consister à ajouter une collision quand le serpent touche les bords du terrain, ainsi que d'ajouter une collision quand le serpent rentre en contact avec lui-même. Et finalement, la possibilité d'ajouter des bonus et malus, des vies, un score, etc.

2 Étude du projet

La première étape pour réaliser un mini-projet comme celui qui est demandé il fallait tout d'abord décider comment procéder à la mise en forme du code et ce qu'il fallait coder. Pour nous orienter, il y avait de petites aides sur l'énoncé, par exemple quel *type* il fallait utiliser, des fonctions qu'on devait utiliser pour pouvoir faire marcher le programme, ou encore un programme qui nous permettait de reconnaître la touche saisie sur le clavier. Avant toute chose, nous avons décidé de commencer par faire le menu du programme, ce qui nous permettait de bien organiser le code dès le début pour le modifier au fil des fonctions ou procédures codées. Nous avions également fait le cadre dans lequel le serpent se trouverait durant le jeu. Ensuite, nous nous sommes concentrés sur le mécanisme du serpent (initialisation,mouvement, changement de direction, collisions, grandissement, affichage) et finalement le jeu lui-même, c'est-à-dire, calculer les limites du cadre, créer les boucles pour le faire s'animer, etc. Nous avions aussi été demandé de faire le programme réparti en *Units* ce qui nous permettra de mieux nous retrouver parmi les lignes de codes. Il était demandé de faire un unit que pour le serpent et

le programme principal qui sera le fonctionnement du jeu. Pour notre part, nous avons décidé de faire plus de Units, avec en plus, un unit avec toutes les procédures et fonctions d'affichage (l'affichage des multiples menus, l'affichage des cadres, et gros titres, etc.) et un autre unit qui nous permet de créer un classement des top 10 ayant les meilleurs scores.

3 Le serpent

Le but du jeu est de pouvoir manipuler un serpent virtuel. Pour faire ceci, il faut tout d'abord commencer ce serpent en question. Comme dis précédemment nous avions eu consigne d'utiliser un *type* bien précis, qui était un record ainsi que la création d'un type nœud contenant les coordonnées d'une case occupée par le serpent, ainsi que d'un record de type *serpent* qui avait plusieurs variables (la taille du serpent, la direction et une liste chaînées pour le corps). Pour faire ceci, il fallait tout d'abord que nous ayons les fonctions qui nous permettent de manipuler une liste chaînée. Nous nous sommes donc servis dans certains Cours et TD fais précédemment sur les listes chaînées. Nous avons donc pris les fonctions permettant de créer une liste, ajout en début (créer une nouvelle tête) et supprimer la fin de la liste (supprimer la queue). Ces fonctions sont indispensables pour créer le grandissement du serpent et de générer un mouvement sur l'écran. Maintenant que nous avions ceci, il nous fallait trouver la longueur initiale du serpent. C'est en s'aidant de l'énoncé où il y avait une image d'un jeu du serpent ayant tout juste commencé que nous avions trouvé la longueur initiale du serpent qui à une taille de 5 caractères incluant la tête. Pour afficher le serpent nous avons utilisé une suite du caractère * selon la taille du serpent pour le corps ainsi que les caractères $\wedge, >$,v, $<$ pour représenter la tête, selon la direction du serpent. Après avoir initialisé le serpent, nous avions ensuite commencé à nousoccuper des coordonnées de collisions avec le cadre et lui-même. Nous avons donc fais quelques calculs pour pouvoir trouver celles du cadre ainsi que de créer une fonction parcourant toute la liste chaînée du serpent faisant des tests si la tête du serpent a les mêmes coordonnées que d'une partie de son corps. Après ceci, nous avions décidé de créer la fonction permettant le grandissement, c'est-à-dire d'ajouter un noeud en plus à la fin de la liste ayant les mêmes coordonnées que l'ancienne queue. Maintenant, il ne nous restait plus qu'à faire le mouvement pour avoir plus ou moins fini le jeu. Pour créer le mouvement, nous avons créé une procédure de mouvement qui

tourne en boucle jusqu'à ce qu'une des deux fonctions de collisions renvoie le booléen de collision *true*. Dans la boucle *while*, il y a la fonction qui retire un noeud en fin de liste et qui ajoute une nouvelle tête. Pour pouvoir ralentir le défilement du serpent nous avons utilisé la procédure *delay*, qui fais une pause d'un certain nombre de millisecondes que nous choisissons. Cela créer donc un mouvement à l'aide de la procédure *ClrScr* qui permet d'effacer touts le contenu de la fenêtre de la console. En effet à chaque itération de la boucle while, on efface touts le contenu de la console, nous réaffichons le cadre et le serpent avant avancé ou changé de direction. Pour le faire change de direction nous avons mis une condition *if* qui reconnaît la touche venant d'être saisie et en fonction de celle-ci ajoute 1 ou soustrait 1 à la coordonnée en question. Pour changer à ce moment la la tête de direction nous affichons, comme préciser précédemment nous affichons l'orientation de la tête en fonction de ça direction avec des caractères différents. Maintenant, nous allons vous expliquer comment on fait le serpent grandir. Au lieu de faire grandir le serpent à n itérations, nous avions décidé de passer directement à le faire grandir quand la les coordonnées de la tête du serpent sont égale aux coordonnées d'un point choisies au hasard dans les limites du cadre, où sera affiché un petit "o" qui est en fait de la nourriture. Donc quand la tête et la nourriture ont les mêmes coordonnées nous ajoutons un noeud en fin de serpent ayant les mêmes coordonnées que l'ancienne queue. En suivant les étapes précédentes, nous avons réussi à créer un serpent qui bouge, et qui grandit d' un caractère quand sa tête rencontre une pomme. Il nous faut maintenant implémenter le serpent dans le programme principal et pouvoir naviguer dans le jeu.

4 Le jeu

Pour pouvoir créer un bon jeu, il faut d'abord créer un menu avec plusieurs options/choix. Quand nous lançons le programme, un écran d'accueil affichant le titre "Jeu du Serpent" qui est centré horizontalement et verticalement apparaît. Pour continuer, il nous suffit de saisir une touche quelconque. L'utilisateur fait ensuite face à l'accueil avec la possibilité de jouer, de lire les instructions de comment jouer, de consulter le classement des top 10 meilleurs scorent et la possibilité de quitter l'application. Si l'utilisateur choisit de jouer, il aura alors deux choix : jouer avec un cadre ayant des murs à l'intérieur ou jouer avec un cadre simple. S'il choisit de jouer avec

un cadre ayant des murs il ne pourra choisir par la suite que la rapidité de son serpent, alors qu'en choisissant le cadre simple, il pourra choisie l'option de pouvoir traverser les bords du cadre et fait réapparaître du bord opposé du cadre, ou ne pas avoir cette possibilité et perdre dès que le serpent entre en collision avec un des bords. À chaque fois que le serpent mange une pomme, son score augmente de 1 point. Après avoir perdu, c'est-à-dire être entré en collision avec le cadre ou avec lui-même, si vous avez fais un score qui à sa place dans les top 10 scores réalisé, alors le programme demandera à l'utilisateur d'entrer son surnom, et cela sera enregistré dans un fichier. Nous allons entrer dans de plus vastes explications dans la partie suivant. Finalement, si l'utilisateur choisit l'option de consulter les instructions de comment jouer, il lui sera affiché à comment jouer.

5 Classement

Pour faire le classement de ce jeu, nous avons dû utiliser des tableaux, les fonctions de lecture et d'écriture de fichier, et un algorithme de tri. En effet pour faire cela nous avons écris tout premièrement un fichier contenant des scores. Mais il fallait suivre une syntaxe bien exacte pour pouvoir lire correctement les données. Et nous avons mis un ordre non logique aux premiers abords, mais nous comprenons la logique très rapidement. En effet, à la fin de chaque partie, si le score de la partie est supérieur au dernier du top 10 alors cette partie en question va être enregistrée dans le fichier contenant le classement. Dans le fichier nous avons donc écrit le classement dans le sens inverse que d'habitude, le dernier du classement est celui écrit en premier. Nous avons fait cela, car pour savoir si le score à sa position dans le classement il nous suffit tout simplement de prendre le score du dernier. Si le score du dernier est inférieur au score de la partie venant d'être jouée, alors nous importons tout le fichier avec le classement et à ce moment nous le mettons dans l'ordre croissant. Après avoir fait cela, nous remplaçons les données du dernier avec les données de la nouvelle partie. À ce moment-là il ne nous reste plus qu'à trier le nouveau tableau pour avoir un nouveau classement. Le classement est ensuite récrit dans le fichier .txt pour le sauvegarder. Nous avons aussi écrit une procédure permettant d'afficher le classement, avec des fonctions calculant le nombre d'espaces à afficher pour que toute la table soit bien alignée et propre.

6 Bilan personnel

6.1 David Rigaux

Pour ma part, je me suis occupé de la majorité du mini-projet sauf pour les fonctions de collisions que Mandry a faites. Ce projet a été relativement simple à faire si l'on savait exactement comment procéder et connaissait un minimum le jeu du serpent. Ce projet m'a permis de faire un jeu, dont j'ai toujours rêvé de réaliser ainsi de comprendre comment on anime des mouvements sur une console. Après avoir fini le programme demandé, j'ai décidé de prendre en tant que défi d'ajouter de petits modes de jeu et des niveaux de difficulté (rapidités différentes) ainsi que de créer un système de classement. J'ai vraiment apprécié réaliser ce projet. Cela m'a également permis d'utiliser un certain nombre de notions vu auparavant (les ES avec des fichiers, les units, les tableaux, les listes chaînées ainsi que les algorithmes de tris).

7 Conclusion

La réalisation de ce projet était vraiment intéressante et motivante de faire un programme qui marche. La plus grande satisfaction réside dans le fait d'avoir réussi à coder un jeu que nous jouions quand nous étions petits. Ce projet nous a également permis de revoir pas mal de notions importantes vues au long de l'année.