Rapport sur le projet personnel en langage C :

NIBBLER
(ou jeu du Snake)

DUT1 informatique
Année 2012-2013

IUT de Marne-la-Vallée

Sommaire :

Introduction :

Le Nibbler (ou jeu du Snake) est un jeu vidéo où le joueur contrôle au clavier un serpent se déplaçant dans une aire de jeu. Si le serpent touche le bord ou bien s'il se mord lui-même, le jeu se termine. Si le serpent mange une pomme, son corps grandit d'une section.

Le but de notre projet en langage C était de programmer ce jeu du Snake sans librairie graphique (soit affiché dans la console), avec un plateau de taille fixe (donc sans allocation dynamique), et de rendre tout cela dans un délai de deux mois. Nous pouvions éventuellement ajouter des bonus (comme mettre des pommes bonus, des obstacles, un score, etc.), ce que nous avons essayé de faire. Le projet devant être fait en binôme, Steeve et moi avons décidé de nous mettre ensemble pour travailler, sous l'environnement d'UNIX. Nous avons ainsi divisé les différentes tâches, comme l'affichage, les bonus et autres, puis tout mettre en commun pour pouvoir finaliser le Nibbler et corriger d'éventuels problèmes.

I – Définitions des fonctions :

Nous nous sommes répartis les différentes fonctions afin de finir le Nibbler de façon la plus efficace possible, sans nous souciez dans un premier temps des fonctions Bonus. En effet, pour les bonus, étant facultatifs, nous avons décidé de s'en occuper une fois le jeu (juste le serpent mangeant de simples pommes) finit et sans problème.

Les différentes tâches se sont ainsi réparties de la façon suivante : dans une première partie, nous nous sommes occupés de tout ce qui concernant l'affichage, vérification du *Game Over* (fin du jeu) ainsi que la fonction principale (appelée aussi *main*) puis dans une autre partie, nous sommes occupés de tout ce qui touchait le *buffer*, à la fonction d'attente, ainsi que le passage en mode Brut/Normal. Chacune de ces fonctions ont été codées sous UNIX, puis mise en commun par mails.

II – Les fonctions d'affichage :

Dans une première partie, nous devions nous charger de l'affichage du jeu dans le terminal. Pour cela, nous avons commencé par coder la fonction d'affichage de l'aire de jeu (*aireJeu*), fonction assez simple car seulement composée de plusieurs boucles *for*. Cette fonction nous a donc permis d'afficher l'aire de jeu (un cadre de 62 x 20) dans le terminal. Une fois la fonction d'affichage de l'aire de jeu mise en place, nous avons créer une nouvelle structure, la structure *COORDONNEES* (comprenant les axes x et y), qui permettra de créer le serpent (comprenant la tête et ses parties du corps), les pommes et les bonus. Ensuite, nous avons cherché via un site de référence sur Internet, la fonction *goto_x_y*, permettant de « matérialiser » un repère dans le terminal, pour ainsi pouvoir se déplacer plus ou moins librement dans le terminal en utilisant les axes des abscisses et des ordonnées (d'où la création de la structure *COORDONNEES*), malgré ces deux axes sont inversés. Toutes ces fonctions ont été trouvées sans réel difficulté (sauf *goto_x_y*, fonction que nous n'aurions pu trouver seuls).

Après s'être occupé des fonctions les plus importantes de l'affichage (création de structure et d'axes de coordonnées), nous avons codé les fonctions d'affichage qui mettrons en place le jeu du

Nibbler : *afficheNourri*, *afficheTete*, *afficheQueue*, et *efface* (bien que celle-ci est été faite après). Ces quatre fonctions permettent au serpent et aux pommes de s'afficher dans le terminal. Les trois dernières concernent en particulier le serpent : *afficheTete* et *afficheQueue* pour afficher le serpent et la fonction *efface* pour effacer du terminal la queue du serpent lorsqu'il se déplace. La dernière fut cependant codée après car nous ne savions pas encore concrètement comment le serpent s'affichait dans le terminal ni quelle partie du serpent devait être placée en paramètre.

Pour finir les fonctions d'affichage du Nibbler, nous avons codé les fonctions *surSerpent* et *horsAire*, ces deux fonctions permettant de vérifier si le joueur a perdu (si le serpent se mord ou s'il sort de l'aire de jeu) ou bien quand le serpent mange la pomme. La fonction *horsAire* a posé un léger problème à cause des conditions : les axes des abscisses et des ordonnées, nous nous étions d'abord trompés en ne mettant pas les bonnes valeurs (qu'il fallait calculer aussi pour ne pas avoir de problème avec l'affichage du titre du jeu). Pour la fonction *surSerpent*, il n'y a pas eu de problème : la fonction ne comporte qu'une boucle *for* sans calcul de coordonnées.

III – La gestion du Buffer et du terminal :

Nous avons rencontré un plus grand nombre de problèmes pour les fonctions du mouvement du serpent en fonction des touches tapées. Il fallait gérer le temps d'attente entre chaque mouvement du serpent : en cherchant, nous avons trouvé la fonction *usleep* mais elle ne forçait pas l'écriture des caractères et les stockait dans le buffer de *stdout*. Nous avons donc cherché une fonction afin de pouvoir vider le buffer à chaque mouvement : *fflush*.

Nous devions ensuite trouver un moyen de ne pas afficher ce que la touche tapée par le joueur lors du jeu (la fonction *getch* n'existant pas sous UNIX)pour pouvoir faire bouger le serpent et changer sa direction. Nous avons donc décidé à mettre le terminal en mode brut. Pour cela nous avons dû étudier le manuel de la librairie *termios.h* ainsi que celui de la fonction *raw*(*brut* en anglais) : ainsi nous avons stocké la structure du terminal de base, créé une nouvelle structure que nous avons mis en *brut*(sauf le paramètre de *stdout* que nous avons laissé comme celui initial sinon il saccageait l'affichage). Nous avons ensuite paramétré le terminal avec cette structure et la fin du jeu pour pouvoir remettre le terminal en mode normal.

Nous nous sommes ensuite occupé de la gestion du *buffer* et de stdin. Le problème du retour chariot a été résolu lorsque nous avions mis le terminal en mode *brut*, mais lorsque nous faisions un *getchar* ou un *scanf*, le jeu se mettait en pause : même s'il ne demandait plus de retour chariot il y avait quand même une interruption du programme.

Pour résoudre ce problème nous avons opté à utiliser la fonction *select* : cette dernière permet de voir si un élément se trouve dans le *buffer* d'un flux standard (*stdin*, *stdout* ou *stderr*) à travers des descripteurs de ficher lorsque le terminal est en *brut*(car ce dernier récupère les touches tapées même sans demande du programme). Il nous a donc suffi de créer un ensemble descripteur, le mettre a zéro, de créer un descripteur pour *stdin* puis de vérifier si ce dernier contenait quelque chose en lui laissant un intervalle nul, pour que la fonction renvoie 1 s'il contenait quelque chose sinon 0 . Après avoir fait cela, il nous a juste suffi de faire un *getchar* lorsque le *buffer* contenait quelque chose pour récupérer la touche entrée.

Une fois ces différents problèmes réglées, nous avons pu coder sans trop de difficultés les fonctions permettant au serpent de se déplacer dans l'aire de jeu en fonction des touches tapées par

le joueur (celles-ci reposant essentiellement sur cette gestion du *buffer* et du *stdin*).

IV – Mise en commun des codes et bonus :

Une fois toutes les fonctions du Nibbler finies, nous avons tout mis en commun afin de bien vérifier s'il n'y avait pas un problème dans une d'entre elles. Nous avons ensuite codé la fonction main, où tout le jeu du Nibbler se passe. Au début, le serpent avait un problème : il s'affichait et se déplaçait correctement dans le terminal, cependant, au bout de deux pommes mangées, la queue du serpent ne s'effaçait pas, laissant ainsi la traîner du serpent à chaque déplacement. Heureusement, nous avons trouvé l'erreur assez vite : il fallait initialiser toutes les parties du corps du serpent.

Nous avons ensuite essayé de moduler le projet (toutes les fonctions se trouvant dans le fichier principale). Cependant, nous avons rencontré un problème au niveau des différentes fonctions, en particulier celles du gestion du *buffer*. Nous avons ensuite moduler notre projet, afin de le rendre plus lisible. Nous avons donc décidé de diviser le projet en plusieurs parties : l'affichage, les déplacements, les tests (dans l'aire, etc.), les fonctions générales (pour passer en mode brut, etc.) ainsi que la fonction principale. Nous avons eu tout de même quelques problèmes au niveau de la modulation du projet à cause des fonctions générales et des variables globales : certaines fonctions utilisaient ces variables, créant des problèmes au niveau des variables (« variables déjà définies » ou bien « variable non définies »).

Une fois le projet fini (du moins les fonctions obligatoires pour le fonctionnement du Nibbler), nous avons essayé de coder les fonctions bonus. Nous avons tout d'abord codé la fonction permettant d'afficher le score en bas du terminal ainsi que les instructions pour que le joueur puisse abandonner ou mettre pause (une petite fonction composée de *printf* et d'un *goto_x_y*). Pour les pommes bonus, ce fut simple : nous avons repris la fonction affichant les pommes puis ajouter à la fonction main une boucle pour afficher les pommes bonus au bout d'un certain temps et durant un certain temps, et augmenter le score de plus de points que les pommes normales. Pour la fonction créant des obstacles, nous avons rencontré quelques bogues (erreurs de segmentation, serpent qui mange l'obstacle sans mourir et problèmes pour afficher plusieurs obstacles). Pour pouvoir afficher plusieurs obstacles, nous avons décidé de stocker les obstacles dans un tableau de structures (COORDONNEES obs). Pour les erreurs de segmentation et le serpent mangeant l'obstacle, nous avions juste oublier de mettre les obstacles dans les tests (pour voir si un point était occupé par l'obstacle).

Nous nous sommes ensuite occupés de la fonction pour le score (enregistrer le score dans un fichier, l'afficher et pouvoir y mettre les meilleurs scores). Nous avons rencontré des problèmes lors du codage de cette fonction au niveau de l'enregistrement du meilleur score : nous avions en effet mis en option du *fopen* « w+ » au lieu de « r+ » (le score était toujours enregistré comme étant le meilleur).

Conclusion

Notre projet consistant à créer le jeu du Nibbler en langage C et dans le terminal, ne s'est pas déroulé sans encombre. Nous avons divisé notre travail en plusieurs parties, afin d'être plus efficaces : une première partie pour les fonctions d'affichage du Nibbler et des vérification du *Game Over*, une autre pour créer de la gestion du *buffer* et de *stdin*, tout en gardant contact par mail l'un et l'autre, pour pouvoir nous aider ou bien pour mettre, une fois tout fini, le tout en commun et

coder les bonus. Bien que nous n'avons pas pu créer de modules pour notre projet, le jeu du Nibbler fonctionne tout de même bien, tout comme les bonus implémentés au programme.