Aziz Nacef
L1 B.I
A.U 20/21

Documentation du projet de fin semestre

a. Les librairies utilisés:

- <stdlib.h> : Allocation dynamique de la mémoire avec malloc()
- <conio.h>: Traiter les touches de clavier avec kbhit(), getch()
- <unistd.h>: Simuler un délai de temps avec sleep()
- <windows.h>: Plusieurs fonctions spécifiques au fonctionnement de la console, permettant de rafraîchir l'écran
- <ctype.h>: librairie de fonctions sur les caractères, contient la fonction toupper() (Gestion des clés entrés par le joueur)

b. Notes générales

- Le jeu permet à l'utilisateur de personnaliser quelques paramètres, notamment le mouvement automatique ainsi que le pouvoir de traverser les bordures (et sortir de l'autre côté).

- L'espace ou le jeu se déroule est un tableau 2-dimensionnel de caractères, initialisé par des espaces à l'aide de deux boucles For imbriqués. On y écrit '#' sur les bordures pour créer le sandbox.

- Les insectes sont générées chaque 8 itérations, si leur nombre dépasse un seuil de 5, ils sont générées après (2*Sum) itérations, avec Sum un entier global qui garde nombre d'insectes. Cela maintient un nombre

raisonnable d'insectes et garantit que le tableau ne se remplira pas au bout.

C. Mécanisme de mouvement et de collision avec les bordures/les insectes

Comme le montre le diagramme attaché, après chaque touche significative (ZQSD), on réécrit les coordonnées de la tête puis on vérifie plusieurs contraintes avec les fonctions:

- BodyCheck (vérifier si le snake est en contact avec son propre corps)

- BorderCheck (vérifier si la tête est en contact avec un mur, il doit sortir de l'autre côté si les paramètres le permettent sinon le jeu se termine)

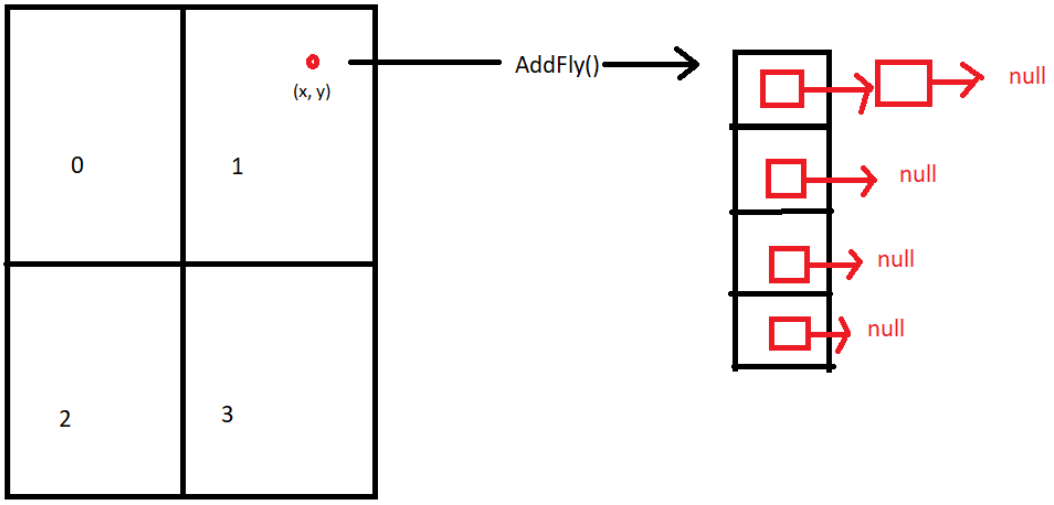
- CheckCoordinates (vérifier si le Snake a mangé un insecte)

R.Q: La fonction CheckCoordinates utilise une "hashtable" (Tableau associatif).

En effet, on déclare, en globale, un type liste chaînée qui garde les coordonnées de chaque insecte. Le tableau "hashtable" est un tableau de ce type de liste.

Le tableau (espace du jeu) est divisé en 4 quadrants, de 0 à 3, chaque fois qu'un insecte est généré, ses coordonnées sont passées à une fonction AddFly

Aziz Nacef
L1 B.I
A.U 20/21



qui l'ajoute au tableau.

Le tableau est dit associatif car le programme dispose d'une fonction CheckCoordinates qui, ayant reçu deux coordonnées (x,y), retourne le quadrant approprié.

En utilisant cette structure de donnée, la vérification de la collision tête-insecte est plus optimisée (On ne compare pas les coordonnées de la tête avec celles de tous les insectes).

Certes, dans le contexte de ce jeu ce n'est qu'une micro-optimisation, mais ce genre de structure est extrêmement utile dans le contexte des recherches répétées.

- La logique du jeu respectée, on fait appel à la fonction MoveBody pour finaliser le mouvement. Il s'agit d'une fonction récursive qui prend en paramètres les anciennes coordonnées de la tête ainsi qu'un pointeur a la cellule qui lui succède.

La condition d'arrêt est le cas d'une cellule ou le pointeur "prev" est nulle, dans ce cas ses anciens coordonnées dans le tableau reçoivent " " (le caractère espace).

D. Les techniques typiques (Listes - piles - files) utilisés

- AddToSnake(): On réalise un empilement (Ajout fin) d'une cellule au Snake s'il consomme un insecte.

- AddFly(): On réalise un ajout fin a liste chaînée de coordonnées chaque fois qu'un insecte est généré.

- RemoveFly(): Lorsqu'un insecte est consommé, on doit supprimer ses coordonnées de la liste chaînée concernée. La fonction RemoveFly reçoit en paramètres une liste et une paire de coordonnées et supprime la cellule en question, gérant tous les cas particuliers.

- Destroysnake() et DestroyTable() : ces procédures désallouent l'espace réservé au liste chainées des insectes ainsi que celle du snake si l'utilisateur choisit de continuer à jouer après avoir perdu.
  Pour la structure Snake, le pointeur est réinitialisé avec la fonction Snakeinit(). Ce n'est pas le cas pour le tableau, donc le pointeur sur liste contenu dans chaque case est remis à null.

Aziz Nacef
L1 B.I
A.U 20/21