DIALLO Eric
DUFOUR Romain

ediall02@etudiant.univ-mlv.fr
rdufour@etudiant.univ-mlv.fr

L2-S3
TP Groupe 6
31/12/2013

TP 4
Rapport

Année 2013/2014

Introduction

Objectif, but du TP.

L objectif de ce projet est d'implémenter le jeu du serpent. C'est un jeu qui consiste à diriger un ver sur un damier et qu'il mange toutes les proies du damier sans que sa tête touche une partie de son corps.

Développement

Exercice 1: But

Les deux premières questions du TP consistent à montrer comment on peut afficher un damier en utilisant la bibliothèque graphique, puis comment déterminer la position aléatoire des proies sur le damier. Il est préparatoire à la programmation du jeu.

Explication des structures et types utilisées.

Pour implémenter le jeu, on utilise différentes structures de données et différents types:

- Un type `enum` booléen pour faciliter la lisibilité du code (pour différencier les fonctions booléennes des fonctions renvoyant un entier).

- SNAKE -
Pour coder le serpent, on utilise une structure `Snake` qui contient les éléments suivants:

-Un tableau de structure `CaseSnake` qui contient :

- Un type `enum StateSnake` qui code l'état de la case du serpent : (*`VIDE, TETE, CORPS, QUEUE`*).

- Une structure `Position` qui contient les coordonnées (x,y) de la case du serpent sur le damier.

- DAMIER -
Pour coder le damier (la map), on utilise une structure `Damier` qui contient :

- Un tableau à 2 dimensions de type `enum StateMap` qui contient l'état de chaque case du damier (*`VIDE, PROIE, VER`*).

- INPUT -
Les événements sont codés avec la bibliothèque SDL.
Pour gérer les événements, on utilise une structure `Input`.
Elle contient :
- Un `SDL_Event` (la structure d'événements SDL)
- Deux entiers récupérant les coordonnées des clics.
- un tableau de booléens qui indique si une touche de souris est enfoncée ou non.
- De même, un tableau de booléens pour les touches du clavier.
- Un booléen qui indique si l'utilisateur à cliqué sur la croix.

Les événements sont mis à jour grâce à la procédure `updateEvents()`.

- AFFICHAGE -
L'affichage est codé avec la bibliothèque SDL.
- La structure d'affichage contient :
- Un pointeur sur une `SDL_Window` (une fenêtre)
- Un pointeur sur un `SDL_Renderer` (un rendu)
- Un pointeur sur une `SDL_Texture` (le tileset du jeu)
- Un tableau de `SDL_Rect` (un rectangle) contenant les positions de chaque tile sur le tileset.
- Un tableau de `SDL_Rect` contenant les positions de chaque tile durant l'exécution du programme.

- SON -
Le son est codé avec la bibliothèque SDL_Mixer.
La structure de son contient :
- Un tableau de pointeurs sur un `Mix_Chunk` (une structure qui contient les sons WAV).

Un son est joué grâce à la fonction `playSound()`.

Algorithme principal du jeu

La fonction `playGame()` est la fonction essentielle au déroulement du jeu. Son algorithme est le suivant :

- on dispose des proies aléatoirement sur le damier.
- on affiche le titre du jeu.
- on dessine le damier avec le serpent initialisé à sa position de départ.
- Ensuite, dans une boucle infinie
- On met à jour les événements ainsi que les positions du serpent.
- Si l'utilisateur appuie sur la croix, on met fin au jeu.
- Si la position indiquée par l'utilisateur est valide, on ajoute cette position au tableau de positions du serpent.
- Si le serpent se mord lui-même, on indique que l'utilisateur à perdu, puis on met fin au jeu (on arrête la boucle).
- On met les positions sur le damier.
- On dessine le damier. Suivant la nouvelle position, on effectue une rotation sur l'image de la tête afin qu'elle soit orientée du bon côté.
- Ensuite, on teste si il reste des proies sur le damier; si il en reste, on continue la boucle, sinon on indique à l'utilisateur qu'il à gagné et on arrête la boucle.

Conclusion

- Ce TP permet de se perfectionner dans la programmation événementielle et à la conception d'interfaces graphiques.

- Il est possible d améliorer ce projet en implantant une fonctionnalité permettant d'avoir un score et de les sauvegarder.