
Langage C

Croque-carotte




Florentin LEPELTIER
Théo BENARD

Cahier des charges

Cahier des charges de base	oui
Cahier des charges (extensions)	non (recherches sans réel aboutissement)
Cahier des charges pour les plus forts	un peu (Quelques suppléments)

Menu

- 1- Jouer au Croque Carotte
- 2- Charger la sauvegarde d'une partie
- 3- Voir les règles ainsi que le but du jeu
- 4- Quitter le programme



```
Croque_Carotte — main — bash — Basic — ttys002 — 100x25

BIENVENUE DANS LE JEU DU CROQUE-CAROTTE

Jouer au Jeu : '1'

Reprendre votre partie : '2'

Règles du Jeu : '3'

Quitter le Jeu : '4'

Que souhaitez-vous faire ? :
```

Le plateau : tableau d'entiers

Valeur	0	0	1	0	-1	2	0	0	0	0	...	5
Affichage	—	—	L1.	—		L2.	—	—	—	—	...	_c_

```
Typedef enum
{
    TROU      = -1,
    VIDE       = 0,
    JOUEUR_1  = 1,
    JOUEUR_2  = 2,
    CAROTTE   = 5
} plateau_e;
```

cases_rotatives	plateau[?]	plateau[?]	plateau[?]
-----------------	------------	------------	------------



tourner_carotte(...)

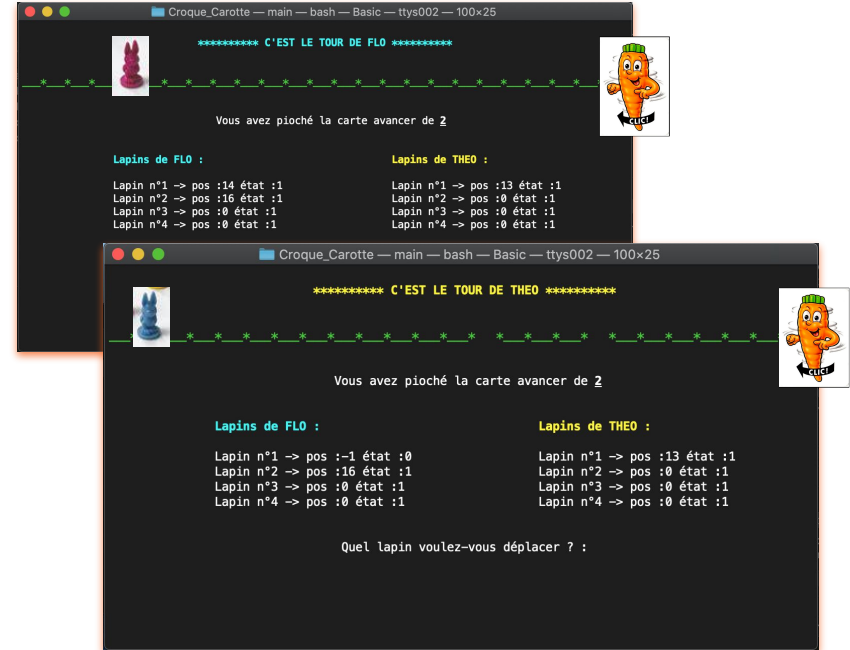
Case ouverte → plateau[cases_rotatives[...]] = -1

Case fermée → plateau[cases_rotatives[...]] = 0

L'affichage

L'affichage permet de nous renseigner sur le tour du joueur.

Il nous permet également d'afficher aux joueurs la position ainsi que l'état de leurs lapins respectifs.



```
Croque_Carotte — main — bash — Basic — ttys002 — 100x25

***** C'EST LE TOUR DE FLO *****

Vous avez pioché la carte avancer de 2

Lapins de FLO :
Lapin n°1 -> pos :14 état :1
Lapin n°2 -> pos :16 état :1
Lapin n°3 -> pos :0 état :1
Lapin n°4 -> pos :0 état :1

Lapins de THEO :
Lapin n°1 -> pos :13 état :1
Lapin n°2 -> pos :0 état :1
Lapin n°3 -> pos :0 état :1
Lapin n°4 -> pos :0 état :1
```

```
Croque_Carotte — main — bash — Basic — ttys002 — 100x25

***** C'EST LE TOUR DE THEO *****

Vous avez pioché la carte avancer de 2

Lapins de FLO :
Lapin n°1 -> pos :-1 état :0
Lapin n°2 -> pos :16 état :1
Lapin n°3 -> pos :0 état :1
Lapin n°4 -> pos :0 état :1

Lapins de THEO :
Lapin n°1 -> pos :13 état :1
Lapin n°2 -> pos :0 état :1
Lapin n°3 -> pos :0 état :1
Lapin n°4 -> pos :0 état :1

Quel lapin voulez-vous déplacer ? :
```

Le joueur : tableau de structures

Chaque lapin de joueur possède un tableau, avec à l'intérieur sa couleur (celle du joueur), son emplacement sur le plateau et son état (mort ou vivant).

Par exemple le tableau suivant représente les lapins du joueur 1.

J1[0] = lapin n°1	J1[1] = lapin n°2	J1[2] = lapin n°3	J1[3] = lapin n°4
<u>Structure</u> générique d'un lapin : -couleur -position -etat	-couleur -position -etat	-couleur -position -etat	-couleur -position -etat



La pioche : tableau d'entiers

Pioche définie	0	0	0	0	0	...	0
----------------	---	---	---	---	---	-----	---

↓ init_pioche(...)

Pioche initialisée	-1	-1	-1	-1	-1	...	-1
--------------------	----	----	----	----	----	-----	----

↓ melanger_pioche(...)

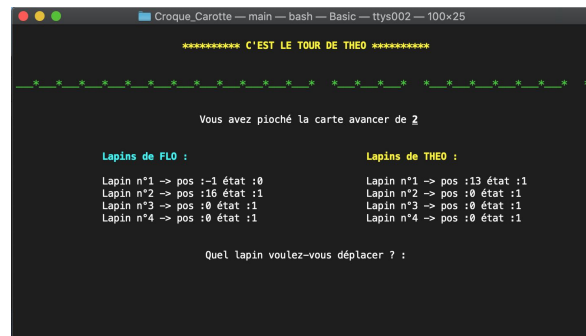
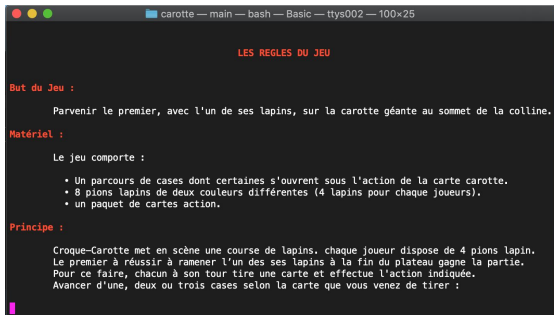
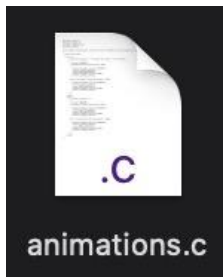
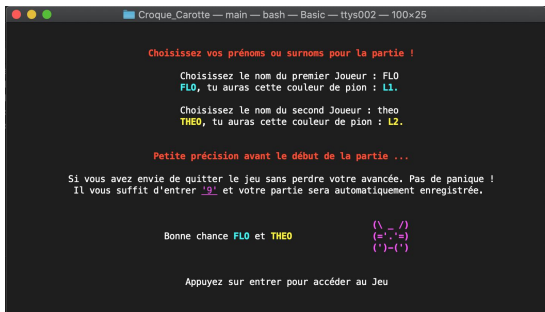
Cartes mélangées	0	1	2	3	1	...	3
------------------	---	---	---	---	---	-----	---

↓ piocher(...) → Carte avancer de 1

Nouvelle pioche	0	1	2	3	-1	...	3
-----------------	---	---	---	---	----	-----	---

Une variable nommée `carte_piochee` prend la valeur 1 : `carte_piochee = pioche[indice]` → vaut 1 ici
`pioche[indice] = -1`

Les suppléments ...



```

11 void init_plateau(int plateau[],int cases_rotatives[]){
12
13     plateau[NB_CASES-1] = CAROTTE; //On prépare l'affichage de la carte carotte
14     int case_alea1=0, case_alea2=0, case_alea3=0;
15 //ATTRIBUTION DE 3 NOMBRES ALÉATOIRES QUI DIFFÉRENT ENTRE EUX (Les cases rotatives doivent être différentes)
16 while ((case_alea1 == case_alea2) || (case_alea1==case_alea3) || (case_alea2==case_alea3))
17 {
18     case_alea1 = rand()%(NB_CASES-2)+1; //On exclut les valeurs 0 et carotte
19     case_alea2 = rand()%(NB_CASES-2)+1; //Entre 1 et 23 pour 25 cases/plateau[23] = case 24
20     case_alea3 = rand()%(NB_CASES-2)+1; //plateau[24] = case25 = carotte
21 }
22 //ON CONSERVE LES 3 NOMBRES DANS LE BUT DE LES RÉUTILISER LORSQUE L'ON PIOCHERA LA CARTE CAROTTE
23 cases_rotatives[0] = (case_alea1);
24 cases_rotatives[1] = (case_alea2);
25 cases_rotatives[2] = (case_alea3);
26 }
27

```

Les difficultés rencontrées

- Structures
 - Pointeurs
 - Modulos
 - Config
-

Bugs persistants

- Lorsque le joueur ne possède plus qu'un lapin vivant et que la carte le fait avancer plus loin que le plateau.
 - Il peut arriver qu'il y ait plus de cartes carotte de piochées que la normale, aussi, on peut piocher 25 cartes au lieu de 24...(la dernière case de la pioche se remet à 0 sans raison ?)
 - Il est possible que d'autres bugs soient présents, si c'était le cas, nous ne les avons alors pas rencontrés ou avons oublié de les noter...
-

Points forts/Points faibles

- Tableaux
 - Énumérations
 - Factorisation du code
 - Longueur des fonctions
-

Ce qu'on aurait pu améliorer ...

- Code plus configurable
 - La connexion en réseau
 - L'interface graphique à l'aide de SDL.
 - La pertinence des commentaires.
-

MERCI DE VOTRE ATTENTION !

