Un combat singulier : deux pions sur un damier

En examinant de près une sorte de colin-maillard, découvrons certains principes utiles à la programmation des jeux sur les ordinateurs de poche et mettons-les en pratique sur une TI 57.

■ Dans le précédent numéro, en prenant pour exemple un jeu connu, nous avons abordé le problème du déplacement d'une pièce sur un plan quadrillé et nous avons examiné l'une des solutions programmables sur une calculatrice élémentaire.

Aujourd'hui, nous allons voir qu'il existe d'autres solutions mais nous ne changerons pas d'optique : il s'agira toujours de décrire par le menu certains procédés grâce auxquels on peut programmer les jeux. Cela dit, le jeu dont nous nous occuperons est plus original dans son principe. On remarquera par exemple qu'il serait assez difficile de s'y livrer sans l'aide d'un ordinateur de poche ; on ne tardera pas à comprendre pourquoi.

Le terrain sur lequel se dispute chaque partie est un damier de dix cases sur dix. Chacun des deux adversaires se trouve représenté sur ce damier par un pion : il connaît sa propre position, mais il ignore celle de l'autre (fig. 1). Le damier où se déroule le jeu doit donc rester invisible aux deux adversaires. C'est la calculatrice, arbitre impartial, qui seule connaîtra toutes les données de la partie.

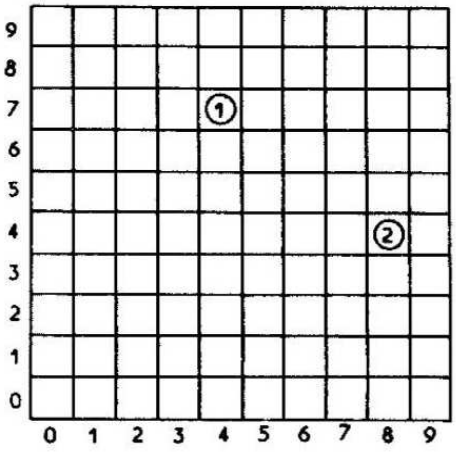
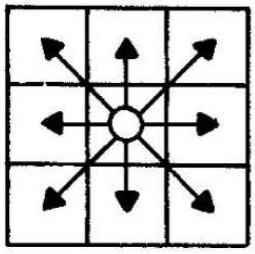


Fig. 1 : le terrain de jeu

A tour de rôle, les deux joueurs se déplacent de la case qu'ils occupent à l'une des cases voisines, et cela dans n'importe quelle direction... à la seule condition de ne pas sortir du damier (fig. 2). La victoire est acquise dès qu'un joueur parvient à capturer son adversaire en venant se placer sur la case occupée par ce dernier, mais la recherche de cette case se fait à l'aveuglette : la seule indication dont dispose le joueur est en effet la direction dans laquelle son adversaire vient de se déplacer.

Fig. 2 : les huit déplacements possibles



Afin que cette sorte de cache-cache ne dure pas trop longtemps, on limite le nombre total de coups alloués aux deux joueurs. Un affichage particulier annonce le match nul lorsque les deux adversaires ont dépassé ce total. Et si l'un des deux réussit à capturer l'autre, sa position se met à clignoter.

Un combat singulier : deux pions sur un damier

Pour repérer les pièces, nous conserverons la solution retenue pour le jeu du loup et des agneaux : deux mémoires (que nous appellerons par exemple M1 et M2) enregistreront la position respective de chaque joueur. Ainsi la position du joueur 1 en colonne 4 et ligne 7 sera signalée par le nombre décimal 4,7 stocké dans la mémoire M1 ; de même, 8,4 en mémoire M2 signifiera que le

joueur 2 se trouve à l'intersection de la colonne 8 et de la ligne 4.

La meilleure solution consiste évidemment à indiquer à la calculatrice *en une seule entrée* quel est le joueur qui se déplace et dans quelle direction il le fait. On devrait y parvenir assez facilement. En effet, si la mémoire disponible le permet, les deux joueurs déplaçant leur pièce obligatoirement l'un après l'autre (il leur est interdit de « passer »), il suffit en fait d'indiquer quel est celui qui commence : la machine assurera automatiquement la gestion du numéro du joueur. Mais pour l'instant laissons de côté cet aspect de la question et voyons comment prendre en compte le déplacement des pièces.

Il y a huit directions possibles, et la solution consistant à utiliser huit étiquettes différentes paraît très lourde. Nous allons donc affecter à chacune des huit directions possibles un code numérique convenu. Le code proposé est représenté à la figure 3 (bien entendu, on aurait pu retenir un autre code). La direction NE sera donc codée 1, la direction E sera codée 2,

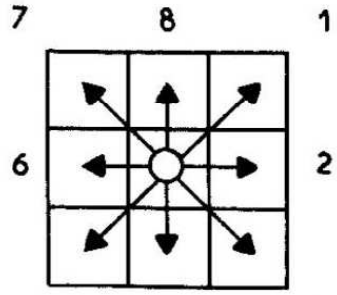


Fig. 3 : de 1 à 8, les codes correspondant à chacun des déplacements possibles

la direction SE sera codée 3, et ainsi de suite en tournant dans le sens des aiguilles d'une montre.

Ceci étant, nous allons représenter dans un même tableau le code des différents déplacements, les déplacements réellement effectués sur le damier et les modifications qu'ils apportent au contenu de la mémoire M1 ou M2 où la position du joueur 1 ou 2 est notée sous la forme x, y ; à la droite du tableau enfin, on notera quelle est la transformation à effectuer pour passer d'un code au suivant. Or on observe une certaine régularité dans ces transformations (− 0,1 puis − 0,1 ; − 1 puis − 1 ; + 0,1 puis + 0,1 ; + 1 et +1) et cela laisse entrevoir une façon de simplifier la programmation du jeu. En effet, si l'on utilise deux mémoires supplémentaires M4 et M5 et si l'on prévoit que le contenu de M5 sera divisé par 10, on obtient la solution décrite par l'organigramme de la

Code du déplacement	Déplacement réel sur le damier		Modifications de M_1 ou M_2	Différences entre les modifications successives
	x	y		
1	+ 1	+ 1	+ 1,1	
2	+ 1	0	+ 1,0	− 0,1
3	+ 1	− 1	+ 0,9	− 0,1
4	0	− 1	− 0,1	− 1
5	− 1	− 1	− 1,1	− 1
6	− 1	0	− 1,0	+ 0,1
7	− 1	+ 1	− 0,9	+ 0,1
8	0	+ 1	+ 0,1	+ 1
1	+ 1	+ 1	+ 1,1	+ 1

Exemple de partie :

0.0 ST0 1 position de départ du premier joueur
9.9 ST0 2 position de départ du second joueur
20 ST0 0 nombre maximum de coups
2nd Fix 1 format d'affichage

- Premier joueur : 1 SBR 1
  Affichage (pause) : 1.1 — nouvelle position du joueur 1
  Affichage: 1.0 — direction du déplacement qu'il vient d'effectuer

Le premier joueur passe la machine au deuxième qui apprend que son adversaire vient de se déplacer dans la direction 1.

- Deuxième joueur : 4 SBR 2
  Affichage (pause) : 9.8 — position du joueur 2
  Affichage : 4.0 — direction du déplacement.

Le deuxième joueur repasse la machine au premier qui apprend que son adversaire vient de se déplacer dans la direction 4.

etc..

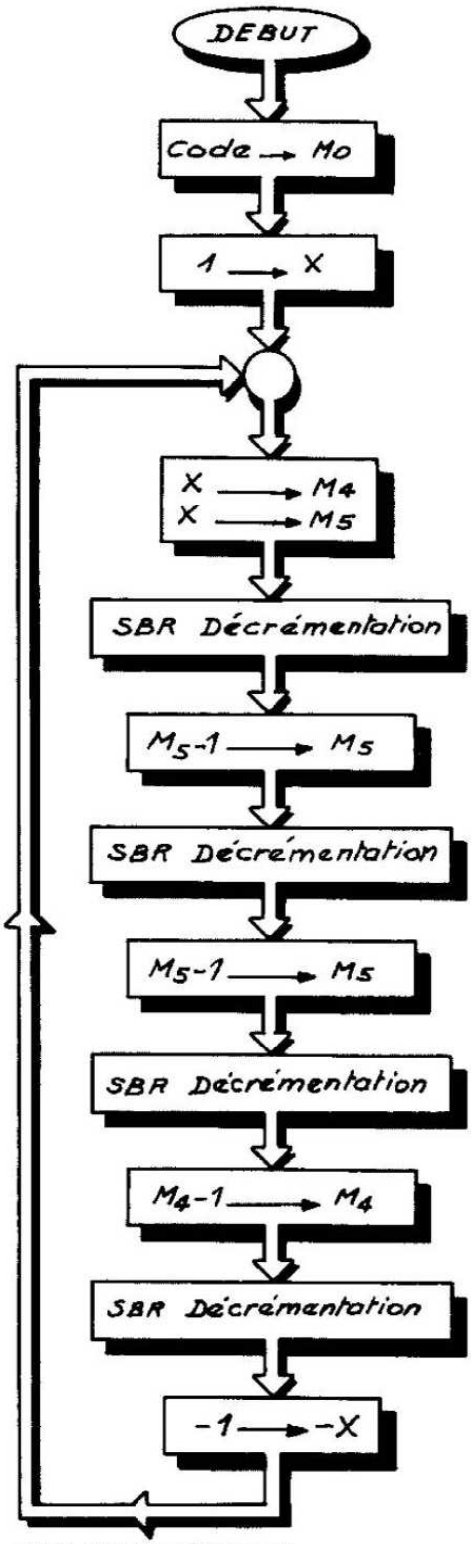
Nota : au début de chaque partie, bien entendu, les deux adversaires choisissent de se placer sur la case de leur choix sans indiquer ce choix à l'autre.

figure 4 (boucle principale et sous-programme de décrémentation). En procédant de la sorte, on enlèvera 1 à M4 ou M5 si le déplacement est codé 1, 2, 3 ou 4, et s'il est codé 5, 6, 7 ou 8, on ajoutera 1 en M4 ou M5.

Nous disposons donc d'un moyen commode de coder les déplacements. Reste, pour que ce procédé soit intéressant, à combiner le choix du déplacement avec celui du joueur de telle sorte qu'en une seule entrée le déplacement signalé soit bien attribué à celui des deux joueurs qui l'effectue. Les deux joueurs vont donc se voir attribuer un code de jeu, par exemple SBR 1 pour le premier joueur et SBR 2 pour le second.

Sur une machine dotée d'un affichage alphanumérique, on remplacerait cela par une question du type : « joueur n° 1, votre déplacement ? », la réponse à donner étant alors un code de 1 à 8. Mais nous conviendrons ici que la frappe de p SBR 1 signifie que le joueur 1 se déplace d'une case dans la direction p.

Fig. 4 : Organigramme du décodage

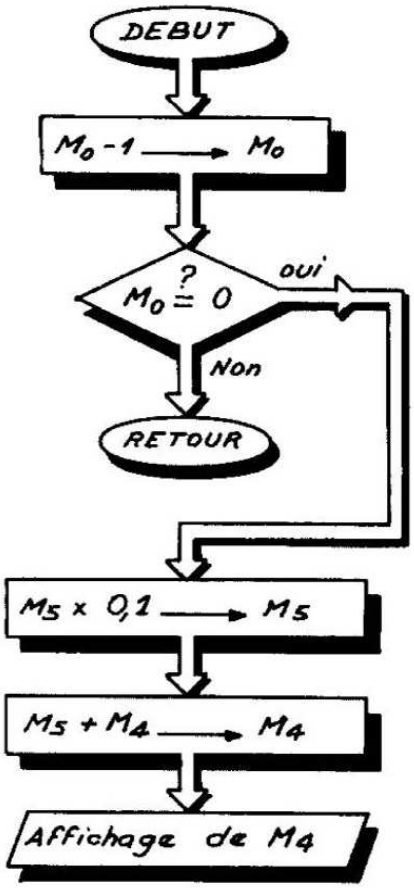


Prudence avec les sous-programmes

On peut être tenté de se dire que tout le décodage précédent sera considéré comme un sous-programme appelé pour imputer le coup du joueur 1 à la mémoire M1 et celui du joueur 2 à la mémoire M2. Mais cette solution, pour séduisante qu'elle soit, présente deux inconvénients.

Premier inconvénient : cela conduirait à faire appel à deux niveaux de sous-programme puisque le sous-programme « décodage » appellerait lui-même un deuxième sous-programme « décrémentation et test ».

Sous-programme de décrémentation et test de fin.



Le second inconvénient est beaucoup plus grave : la sortie de la boucle de décodage se ferait alors non pas dans la boucle elle-même, mais dans le sous-programme de décrémentation, ce qui revient à dire que l'instruction de retour pour le sous-programme « décodage » resterait en attente. Il faudrait donc pour revenir au programme principal, pouvoir enlever une adresse dans la pile de retour des sous programmes, ce qui n'est pas possible en général sur les ordinateurs de poche, TI 57 y compris.

Nous avons donc retenu une autre solution, moins élégante : nous utiliserons un drapeau pour signaler au programme de décodage si le résultat doit être affecté à la mémoire 1 (joueur 1) ou à la mémoire 2 (joueur 2).

Vous me direz, et vous avez raison, qu'il n'y a pas non plus de drapeaux sur la TI 57. (Pendant que l'on y est, j'ajouterai que cette bonne vieille machine ne connaît pas l'adressage indirect !) Mais s'il est impossible d'effacer la première adresse de retour de sous-programme, on peut en revanche simuler un drapeau. C'est ce que nous allons faire.

Pour différencier les entrées SBR 1 et les entrées SBR 2, nous ferons en sorte que le déplacement (chiffre de 1 à 8 entré au clavier) soit stocké en M7 et en M6 avec SBR 1. Si le déplacement est introduit avec SBR 2, le chiffre sera stocké en M6 et le programme donnera à M7 la valeur zéro.

Un combat singulier : deux pions sur un damier

Après le calcul du déplacement, il suffira de comparer les contenus de M6 et de M7 pour reconnaître une entrée faite par SBR 1 ou SBR 2.

On trouvera ci-dessous un programme complet utilisant cette solution. Il a été conçu pour TI 57.

Au début de la partie, chaque joueur doit indiquer sa position de départ sans la montrer à son adversaire. La machine doit être en Fix 1, et la mémoire 0 doit contenir le nombre maximum de coups. On pourra choisir par exemple :

- 0,0 STO 1 : position initiale du premier joueur ;
- 9,9 STO 2 : position initiale du second joueur ;
- 20 STO 0 : nombre maximum de coups ;
- 2nd Fix 1 : format de l'affichage.

Le premier joueur indique son déplacement en entrant n SBR 1 (n compris entre 1 et 8) puis il passe la calculatrice à son adversaire. L'affichage indique alors dans quelle direction le premier joueur s'est déplacé. Le second joueur en prend connaissance avant d'indiquer son déplacement : m SBR 2 (m compris entre 1 et 8) et il repasse la calculatrice au premier, etc.

Chacun doit veiller lui-même à ne pas sortir des limites du terrain. Quand l'un des deux parvient à capturer l'autre, il obtient l'affichage clignotant de sa position. Au contraire, si les joueurs ont épuisés les vingt coups autorisés sans aboutir à une capture, l'affichage du déplacement est suivi de l'indication 00, et la partie est alors considérée comme perdue (ou gagnée...) par les deux joueurs.

Pour entamer une autre partie, il suffit de redonner en M1 et M2 la position initiale des deux joueurs et de remettre dans M0 le nombre de coups autorisés. Si la partie précédente s'est soldée par un ex-aequo, on peut d'ailleurs se contenter de « recharger » M 0 en conservant la position occupée par les joueurs. Dans ce cas, on prolongera la partie en demandant seulement 10 STO 0 par exemple, ce qui donnera 10 coups de plus. On n'oubliera pas de presser sur CLR ou sur INV EE pour rétablir un affichage normal.

□ Jacques Deconchat

A l'aveuglette
Programme pour TI 57
Auteur Jacques Deconchat


```
00   33  4      RCL  4
01   34  1      SUM  1
02   33  1      RCL  1
03   51  6      GTO  6
04   86  1  2nd Lbl  1
05   32  7      STO  7
06   86  2  2nd Lbl  2
07   32  6      STO  6
08   38  0  2nd Exc  0
09   32  3      STO  3
10   01          1
11   86  8  2nd Lbl  8
12   32  4      STO  4
13   32  5      STO  5
14   61  7      SBR  7
15  -34  5      Inv SUM  5
16   61  7      SBR  7
17  -34  5      Inv SUM  5
18   61  7      SBR  7
19  -34  4      Inv SUM  4
20   61  7      SBR  7
21   84          +/-
22   51  8      GTO  8
23   86  7  2nd Lbl  7
24   56     2nd Dsz
25  -61         Inv SBR
26   01          1
27   00          0
28  -39  5  2nd Inv Prd  5
29   33  3      RCL  3
30   32  0      STO  0
31   33  5      RCL  5
32   34  4      SUM  4
33   33  6      RCL  6
34   66     2nd x = t
35   71          RST
36   33  4      RCL  4
37   34  2      SUM  2
38   33  2      RCL  2
39   86  6  2nd Lbl  6
40   36     2nd Pause
41   33  1      RCL  1
42   65          -
43   33  1      RCL  2
44   85          =
45   25          1/x
46   19     2nd Ct
47   33  6      RCL  6
48   56     2nd Dsz
49   81          R/S
```