Shoot again

N° 41

Par André Chéramy , Fabrice Francès et Dominique Pessan

Un shoot again avec trois auteurs ? Mais que sont venu faire André et Fabrice dans cette galère ? Simplement, me donner un petit coup de main. Them à été pour moi, et de loin, le jeu le plus difficile (et le plus long) à adapter au joystick.

La version que je possédais refusait toute adaptation et plantait systématiquement. Protection du soft ? Mauvais transfert ? Je ne sais pas. En tout cas, André à volé à mon secours en me fournissant une version « sûre » du jeu.

Quand à Fabrice, c'est lui qui a « tenté » de m'expliquer le fonctionnement de la lecture du clavier par le programme. Etrange lecture, jamais rencontrée auparavant. Les explications de Fabrice sont retranscrites telles quelles un peu plus loin.

En plus d'un scan bizarre du clavier, chacun des cinq tableaux du jeu utilise des touches différentes pour le même type d'action. Par exemple, pour tirer, selon le tableau c'est <Shift Gauche>, <Ctrl> ou bien <A>.

Première idée : Je suis tout d'abord parti sur l'idée de modifier le jeu lui même en rationalisant les touches gérées pour avoir une gestion simple du JS. Mauvaise piste, je ne suis arrivé qu'a des plantages successifs.

Deuxième idée : Une routine JS « compliquée » qui « saurait » d'où elle a été appelée et testerait la position correspondante du JS. (En clair, si l'endroit d'appel correspondait au test de mouvement à gauche, on ne testerait que la position gauche du JS). C'est grâce à l'explication de Fabrice que cette idée m'est venue car je n'avais pas « vu » que le résultat du test du clavier était « caché » dans Carry (C) . Une fois qu'on a compris cela, tout devient plus simple. Si le JS est dans la position testée, il suffit de faire monter Carry et de retourner au programme principal.

Pour savoir d'où le programme appelle, j'ai tenté d'utiliser la pile. Normalement, PLA, PLA, après l'appel d'un sous programme, vous donne l'adresse qui suit celle de l'appel . Le premier PLA, c'est la partie basse de l'adresse et, chance, vous le constaterez sur le listing qui suit, les 19 appels ont tous (sauf deux) une partie basse différente. Le problème avec la pile c'est que la moindre erreur ne pardonne pas, j'ai du en commettre trop ☺ … J'ai abandonné. (mais normalement, cela devrait marcher)

La troisième idée, je l'avais eu avant la deuxième, mais comme elle est moins élégante, je ne me suis résolu à la mettre en œuvre, que par défaut, après avoir tenté sans succès de mettre au point la précédante. C'est tout simple, cette fois chacun des 19 appels est orienté (par 19 DOKE) vers la bonne routine JS. C'est pas beau, mais ça marche !

Reprenons le fil habituel de nos shoot again, en faisant,

L'état des lieux

Sur la disquette d'André, le jeu est composé de deux fichiers :

THEM.COM $501-$25C0 chargeur Basic + LM
THEM.BIN $600-$5660 LM

La gestion du clavier se trouve dans le fichier THEM.BIN , elle est « explosée » en 19 appels, les voici :

Tableau 4 (le tapis roulant)

```
3582  A910    LDA $10         test <Ctl>
3584  8D0204  STA #0402       tir haut
3587  A902    LDA $02
3589  207920  JSR #2079
358C  9034    BCC #35C2

35C2  A910    LDA $10         <shiftG>
35C4  8D0204  STA #0402       tir bas
35C7  A904    LDA $04
35C9  207920  JSR #2079
35CC  90EE    BCC #35BC

35BC  A901    LDA $01
35BE  8D2404  STA #0424
35C1  60      RTS
```

Tableau 5 (les moitiés d'os)

```
3E25  AD2904  LDA #0429
3E28  F003    BEQ #3E2D
3E2A  4CD83E  JMP #3ED8
3E2D  A910    LDA $10         test <Ctl>
3E2F  8D0204  STA #0402       à gauche
3E32  A902    LDA $02
3E34  207920  JSR #2079
3E37  901B    BCC #3E54

3E54  A901    LDA $01
3E56  8D2704  STA #0427
3E59  A920    LDA $20         test <A>
3E5B  8D0204  STA #0402       à droite
3E5E  A906    LDA $06
3E60  207920  JSR #2079
3E63  B007    BCS #3E6C
……
3E84  A920    LDA $20         test <Ret>
3E86  8D0204  STA #0402       validation
3E89  A907    LDA $07
3E8B  207920  JSR #2079
3E8E  B006    BCS #3E96

3ED8  A910    LDA $10         test <Ctl>
3EDA  8D0204  STA #0402       à gauche
3EDD  A902    LDA $02
3EDF  207920  JSR #2079
3EE2  9047    BCC #3F2B
……
3F2B  A920    LDA $20         test <A>
3F2D  8D0204  STA #0402       à droite
3F30  A906    LDA $06
3F32  207920  JSR #2079
3F35  9043    BCC #3F7A
……
3F7A  A920    LDA $20         test <Ret>
3F7C  8D0204  STA #0402       validation
3F7F  A907    LDA $07
3F81  207920  JSR #2079
3F84  B006    BCS #3F8C
……
```

Tableau 1 (le labyrinthe)

```
4793  AD1F04  LDA #041F
……
479E  A902    LDA $02         test <I>
47A0  8D0204  STA #0402       en haut
47A3  A905    LDA $05
47A5  207920  JSR #2079
47A8  9008    BCC #47B2
……
47B2  A901    LDA $01         test <J>
47B4  8D0204  STA #0402       à gauche
47B7  207920  JSR #2079
47BA  9008    BCC #47C4
……
47C4  A901    LDA $01         test <K>
47C6  8D0204  STA #0402       à droite
47C9  A903    LDA $03
47CB  207920  JSR #2079
47CE  9008    BCC #47D8
……
47D8  A901    LDA $01         test <M>
47DA  8D0204  STA #0402       en bas
47DD  A902    LDA $02
47DF  207920  JSR #2079
47E2  9008    BCC #47EC

4CCF  A910    LDA $10         <shiftG>
4CD1  8D0204  STA #0402       bouclier
4CD4  A904    LDA $04
4CD6  207920  JSR #2079
4CD9  903A    BCC #4D15
```

Tableau 3 (le cercle mortel)

```
5086  A910    LDA $10         <shiftG>
5088  8D0204  STA #0402       tir
508B  A904    LDA $04
508D  207920  JSR #2079
5090  B006    BCS #5098

51FB  AD1F04  LDA #041F
……
5210  A902    LDA $02         test <I>
5212  8D0204  STA #0402       en haut
5215  A905    LDA $05
5217  207920  JSR #2079
521A  9007    BCC #5223
……
5223  A901    LDA $01         test <J>
5225  8D0204  STA #0402       à gauche
5228  207920  JSR #2079
522B  900A    BCC #5237
……
5237  A901    LDA $01         test <K>
5239  8D0204  STA #0402       à droite
523C  A903    LDA $03
523E  207920  JSR #2079
5241  900C    BCC #524F
……
524F  A901    LDA $01         test <M>
5251  8D0204  STA #0402       en bas
```

```
5254  A902    LDA $02
5256  207920  JSR #2079
5259  900A    BCC #5265
```

Tableau 2 (les blocs coulissants)

```
5482  AD1F04  LDA #041F
5485  F001    BEQ #5488
5487  60      RTS
5488  A910    LDA $10         test <Ctl>
548A  8D0204  STA #0402       stop bloc
548D  A902    LDA $02
548F  207920  JSR #2079
5492  B006    BCS #549A
```

C'est la routine en $2079 qui m'a posée un problème de compréhension, voici, « en direct », les explications de Fabrice

---

Bon on y va, donc…
Si le clavier était parfait, le fait de tester plusieurs colonnes à la fois donnerait toujours un résultat qui est un 'OU' de toutes les touches testées dans la ligne… Autrement dit, si tu forces un 0 sur toutes les colonnes, tu détectes une touche enfoncée dès qu'au moins une touche sur les 8 de la ligne est enfoncée.

```
5210  A902    LDA $02         On s'intéresse à la colonne 1
5212  8D0204  STA #0402
5215  A905    LDA $05         et à la ligne 5
5217  207920  JSR #2079
521A  9007    BCC #5223

La routine en $2079 teste une touche en vérifiant que les autres touches de la
ligne ne sont pas enfoncées.
Paramètres  :     ligne dans A / colonnes dans $402
Résultat    :     Carry

2079  08      PHP
207A  78      SEI
207B  8D0003  STA #0300       positionne la ligne, normal…
207E  AD0204  LDA #0402
2081  8D0E04  STA #040E
2084  A90E    LDA $0E
2086  20EF1C  JSR #1CEF       positionne les colonnes qui ne nous intéresse pas
                              (dans ton exemple, toutes les colonnes sauf la 1)
2089  AD0003  LDA #0300
208C  2908    AND $08
208E  D02B    BNE #20BB       si une des touches est enfoncée
2090  AD0E04  LDA #040E       alors on sort avec C=0
2093  4907    EOR $07         on inverse les trois premières colonnes testées,
                              on continue à tester les 5 autres
2095  2907    AND $07         Là, je dois avouer que c'est curieux, j'aurais
                              simplement fait un EOR #FF (cf remarque finale)
2097  8D0204  STA #0402
209A  A90E    LDA $0E
209C  20EF1C  JSR #1CEF       On positionne les nouvelles colonnes, ce sont
                              celles qui nous intéressent en fait
209F  AD0003  LDA #0300
20A2  2908    AND $08
20A4  F015    BEQ #20BB       Si rien ne semble enfoncé, on sort avec C=0
20A6  AD0E04  LDA #040E
20A9  8D0204  STA #0402
20AC  A90E    LDA $0E
20AE  20EF1C  JSR #1CEF       on re teste les colonnes comme au début, au cas où
                              ça aurait changé depuis
20B1  AD0003  LDA #0300
20B4  2908    AND $08
20B6  D003    BNE #20BB       et on sort avec C=0 si une des touches est
                              enfoncée
20B8  28      PLP
20B9  38      SEC             si on arrive là, c'est que la colonne qui nous
```

```
                                  intéresse était enfoncée, et pas ses copines de la
                                  même ligne
20BA  60      RTS
20BB  28      PLP
20BC  18      CLC
20BD  60      RTS
```

Effectivement, donc, ce EOR#7 / AND#7 au lieu d'un EOR#FF est bizarre… J'ai dis au début que si le clavier était parfait, alors il réagirait ainsi…
Mais tu sais qu'il peut y avoir des ratés quand on utilise simultanément certains ensembles de touches, je suppose que c'est pour éviter ces problèmes que la routine est si compliquée (à moins qu'il ne s'agisse d'un bug : j'en ai déjà rencontrés…)
Il faudrait avoir un data sheet du petit multiplexeur du clavier pour en apprendre plus long sur ces problèmes de touches simultanées…

La routine en $1CEF programme un registre du PSG.
Paramètres : [A] le numéro de registre du PSG
: [$402] la valeur à écrire

La méthode utilisée

Le principe a été évoqué en début d'article, alors entrons tout de suite dans le concret
La routine JS est implantée en $7000 c'est la classique routine de G. BERTIN, il n'y a que la fin qui est un peu changée.


```
7000  … … …
… … …
7032  68      PLA

7033  8D0303  STA #0303
7036  68      PLA
7037  8D0103  STA #0301
703A  68      PLA
703B  A8      TAY
703C  68      PLA

703D  8500    STA #00       on stocke en $00 le numéro de ligne de la matrice clavier
703F  AD4770  LDA #7047
7042  60      RTS
```

Voici les routines de gestions des cinq actions possibles avec le JS

routine de test si <haut>

```
7060  200070  JSR #7000 app.routine JS
7063  C901    CMP $01   haut haut
7065  F00D    BEQ #7074
7067  C908    CMP $08   haut gauche
7069  F009    BEQ #7074
706B  C902    CMP $02   haut droite
706D  F005    BEQ #7074
706F  A500    LDA #00   récup.n° ligne
7071  4C7920  JMP #2079 test clavier
7074  38      SEC       si haut, C=1
7075  60      RTS       et retour
```

routine de test si <bas>

```
7076  200070  JSR #7000 app.routine JS
7079  C905    CMP $05   bas bas
707B  F00D    BEQ #708A
707D  C904    CMP $04   bas droite
707F  F009    BEQ #708A
7081  C906    CMP $06   bas gauche
7083  F005    BEQ #708A
7085  A500    LDA #00   récup.n° ligne
7087  4C7920  JMP #2079 test clavier
708A  38      SEC       si bas, C=1
708B  60      RTS       et retour
```

routine de test si gauche

```
708C  200070  JSR #7000 app.routine JS
708F  C907    CMP $07   gauche gauche
7091  F00D    BEQ #70A0
7093  C906    CMP $06   gauche bas
7095  F009    BEQ #70A0
7097  C908    CMP $08   gauche haut
7099  F005    BEQ #70A0
709B  A500    LDA #00   récup.n° ligne
709D  4C7920  JMP #2079 test clavier
70A0  38      SEC       si gauche,C=1
70A1  60      RTS       et retour
```

routine de test si droite

```
70A2  200070  JSR #7000 app.routine JS
70A5  C903    CMP $03   droite droite
70A7  F00D    BEQ #70B6
70A9  C902    CMP $02   droite haut
70AB  F009    BEQ #70B6
70AD  C904    CMP $04   droite bas
70AF  F005    BEQ #70B6
70B1  A500    LDA #00   récup.n° ligne
70B3  4C7920  JMP #2079 test clavier
70B6  38      SEC       si droite,C=1
70B7  60      RTS       et retour
```

routine de test si tir

```
70B8  200070  JSR #7000 app.routine JS
70BB  AD4670  LDA #7046 image tir duJS
70BE  C900    CMP $00   si 0, tir
70C0  F005    BEQ #70C7
70C2  A500    LDA #00   récup.n° ligne
70C4  4C7920  JMP #2079 test clavier
70C7  38      SEC       si tir, C=1
70C8  60      RTS       et retour
```

Chacune de ces cinq routines est ensuite appelée, de manière sélective, comme dans l'exemple qui suit.

```
3582  A910    LDA $10
3584  8D0204  STA #0402
3587  A902    LDA $02
3589  206070  JSR #7060
… … …
```

La marche à suivre

1- Entrez et sauvegardez le listing suivant

```
10 A=#7000:F=#70C8:L=100:REPEAT:FOR A=A TO A+15:READ C$
20 K=VAL("#"+C$):S=S+K+65536*(S+K>65535):IF A<=F THEN POKE A,K
30 NEXT:READ D$:IF S=VAL("#"+D$) THEN L=L+5:UNTIL A>F:END
40 PING:PRINT"Erreur ligne";L
100 DATA 48,98,48,AD,01,03,48,AD,03,03,48,A9,C0,8D,03,03,0518
105 DATA A9,40,8D,01,03,AD,01,03,A8,29,20,8D,46,70,98,4A,0A59
110 DATA 29,0C,8D,2A,70,98,29,03,18,69,0C,A8,B9,48,70,8D,0FAC
115 DATA 47,70,68,8D,03,03,68,8D,01,03,68,A8,68,85,00,AD,1501
120 DATA 47,70,60,EA,EA,EA,EA,EA,00,00,00,00,00,02,08,01,1AB5
125 DATA 00,04,06,05,00,03,07,00,00,FF,00,00,00,00,00,00,1BCD
130 DATA 20,00,70,C9,01,F0,0D,C9,08,F0,09,C9,02,F0,05,A5,2253
135 DATA 00,4C,79,20,38,60,20,00,70,C9,05,F0,0D,C9,04,F0,27E8
140 DATA 09,C9,06,F0,05,A5,00,4C,79,20,38,60,20,00,70,C9,2D30
145 DATA 07,F0,0D,C9,06,F0,09,C9,08,F0,05,A5,00,4C,79,20,334C
150 DATA 38,60,20,00,70,C9,03,F0,0D,C9,02,F0,09,C9,04,F0,39BE
155 DATA 05,A5,00,4C,79,20,38,60,20,00,70,AD,46,70,C9,00,3EA1
160 DATA F0,05,A5,00,4C,79,20,38,60,00,00,00,00,00,00,00,41B8
```

2 - RUN
3 - THEM.BIN,N
4 - DOKE#358A,#7060 :DOKE#35CA,#7076 : DOKE#3E35,#708C :DOKE#3E61,#70A2
5 - DOKE#3E8C,#70B8 :DOKE#3EE0,#708C :DOKE#3F33,#70A2 :DOKE#3F82,#70B8
6 - DOKE#47A6,#7060 :DOKE#47B8,#708C :DOKE#47CC,#70A2 :DOKE#47E0,#7076
7 - DOKE#4CD7,#70B8 :DOKE#508E,#70B8 :DOKE#5218,#7060 :DOKE#5229,#708C
8 - DOKE#523F,#70A2 :DOKE#5257,#7076 :DOKE#5490,#70B8
9 - SAVEO « THEM.BIN »,A#600,E#70C8,AUTO

à bientôt…