Programmez votre premier jeu d'arcade

A dater d'aujourd'hui, nous passons à la vitesse supérieure ! Plutôt que de persister à détailler le basic instruction par instruction, nous nous proposons de construire un jeu. Grâce à ce cours, réalisez votre "Word Invaders".

Michel Maigrot

Le but global de l'opération est de bombarder depuis votre base, les lettres apparaissant au bas de l'écran jusqu'à ce que le mot donné en haut à gauche de l'écran soit intégralement reproduit. Votre base peut se déplacer latéralement comme verticalement avec les flèches du curseur. Chaque déplacement entraîne une certaine consommation de fuel.
La touche <COPY> permet de lancer une bombe. Si le projectile touche une lettre et que celle-ci permet de compléter le mot donné, vous marquez 50 points. Si vous arrivez à réécrire le mot complet, vous avez gagné.
Ce n'est simple qu'en apparence, la forte consommation de votre base compromet gravement vos chances de succès ! De plus, les lettres à bombarder ont la bougeotte et disparaissent périodiquement pour être remplacées par d'autres. Le temps est aussi important, le score et le fuel diminuent régulièrement même si vous ne bougez pas. Une bombe perdue coûte 25 points de score, toucher une lettre qui ne fait pas partie du mot vous en enlève 50 ! Pour obtenir le tiret de LOGI'STAR-RUN'STAR, il vous faudra impérativement bombarder un joker ("*") mais pas avant d'avoir inscrit toutes les lettres à la gauche de celui-ci...

Caractères spéciaux et modifications

Le "A" modifie votre quantité de fuel de 1 à 200. Parfois même en négatif !
Le "?" change le score de la même manière que "@" le fuel.
Lorsque l'un des deux est touché par la bombe, le résultat apparaît en rouge sur l'écran.
Le "*" est le joker. Il ne vous veut que du bien. Il complète le caractère manquant le plus à gauche du mot (quel qu'il soit), vous compte 100 points et ajoute 200 litres à votre réserve de fuel.
Notez que toute lettre entrée dans l'ordre de reconstitution du mot (de gauche à droite) vous vaut 100 points au lieu de 50 dans le désordre. La possibilité de passer d'un bord à l'autre de l'écran économise votre fuel.
Si le fuel baisse trop vite à votre goût, allez aux lignes 210-1230. Si les jokers sont trop rares : voyez la ligne 200. Pour changer la chaîne de caractères, regardez la ligne 180.
Vous ne voulez pas de signe négatif lorsque vous touchez les "?" ou "A" : trafiquez vous-même les lignes 1710 et 1810.

Commentaires détaillés ligne à ligne

60 On réserve un peu de place en mémoire pour une petite routine en LM incluse au programme et implantée en ligne 3300.

70 Toutes les variables sont signées de -32768 à 32768 ou de 0 à 65536 pour des valeurs absolues. L'avantage du DEF-INT est d'accélérer le déroulement général du programme.

80-100 Modification des matrices des caractères, 253, 254, 255. Chaque matrice de l'Amstrad est composé de 8 octets. Chaque bit mis correspond à un point allumé, éteint sinon. Voyez ci-dessous à quoi correspondent les valeurs transmises à la commande SYMBOL.

120 On met ces trois caractères dans une chaîne. PRINT base$ affichera l'ensemble complet.

130 Donner à la chaîne bomb$ la valeur du caractère numéro &FC.

140 L'instruction DIM réserve une place en RAM pour 35 caractères et 35 positions X,Y pour chacun. Pour avoir plus ou moins de caractères au bas de l'écran, changer la valeur de NBCAR.

150 Notre base occupe 3 caractères, il est important d'en tenir compte pour les tests de sortie d'écran ! Changez cette valeur pour voir le résultat...

160 Cette ligne conditionne le mode écran. La remplacer par xmax=20-(lbase-1):MODE 0, pour faire fonctionner le programme en mode 0. xmax=80-(lbase-1):MODE 2 fera tourner le programme en mode 2 mais il faudra de plus remplacer toutes les commandes PEN 2, PEN 3 par PEN 3 par PEN 1 car seules deux couleurs sont possibles dans ce mode. Le jeu sera plus facile en mode 0 et très difficile en mode 2.

170 On initialise la position d'affichage de la base. La bombe à larguer devant suivre la base, on lui donne la même position avec X+1 pour centrer la position du largage sur le second caractère de la base.

180 C'est ici que l'on fixe la chaîne de caractères à reproduire. La longueur du texte ne doit pas excéder 20,40,80 caractères selon le mode choisi. Il est impératif de tout composer en majuscules ! N'oubliez pas que tout caractère n'étant pas une lettre de A à Z nécessite un joker pour le compléter. Plus la chaîne est longue et, plus le jeu est difficile. L'instruction LEN(chaîne$) permet de connaître le nombre de caractères composant une chaîne quelconque. Exemple d'usage : A$="Essai":? len(a$):a=LEN(A$):if len(A$)=5 the PRINT "J'en ai 5 ..."

190 On recopie le nombre de caractères dans la variable "tot" et on initialise la variable "JEU$". L'instruction STRING$ permet de créer une chaîne de N caractères identiques. STRING$(n,caractere) : n peut être un nombre, une expression mathématique, une fonction. Exemple : STRING$ ((112+2)/4,"L"), STRING$(LEN(A$),CHR$(45)), etc. Limites : l'expression du nombre de caractères ne doit pas être plus grand que 255 et donner un résultat entier. A l'initialisation, "jeu$" sera rempli de ".". Pour cette raison, ne mettez jamais un point dans "chaine$".

200 "rang" mémorise la position du prochain caractère à trouver. "jok!" conditionne la fréquence d'apparition des jokers. Le "!" qui suit la variable la définit comme réelle.

210-230 On affiche tous les paramètres du jeu en haut de l'écran. LOCATE place le curseur de texte sur la ligne et colonne spécifiées. Si les paramètres transmis à LOCATE X,Y sont incohérents, on recevra le message "improper argument". Les valeurs correctes sont : X de 1 à 20 en mode 0, 1 à 40 en mode 1, 1 à 80 en mode 2. Y de 0 à 25 dans tous les modes. Si votre fuel baisse trop vite, changez en ligne 210.

Gestion des caractères

500 Exécution du sous-programme en ligne 3000 qui mémorise et affiche des "nbcar" caractères de façon aléatoire, la base est affichée ensuite.
510 Cette commande est l'une des plus puissantes du CPC. Son maniement entraînant des problèmes complexes, elle sera détaillée dans le prochain Runstar. Pour l'instant, contentons-nous de savoir que la ligne 500 interrompt le programme principal toutes les 0,8 secondes pour exécuter le sous-programme en ligne 3200. Remplacer la valeur 40 par une autre, accélèrera ou ralentira la fréquence de remplacement des lettres au bas de l'écran.

Test du clavier et déplacement de la base

1000 Pour provoquer le déplacement d'un motif, il faut : l'afficher, calculer sa nouvelle position, l'effacer à son ancienne position, l'afficher à la nouvelle. On mémorise donc dans "xb", "yb" la position qui doit être effacée.
1000-1040 La fonction INKEY (touche) permet de savoir si une touche donnée est pressée ou non. Consultez votre manuel pour connaître le numéro de toutes les touches du clavier. Ne confondez pas INKEY avec INKEY$. INKEY travaille sur le numéro physique de la touche, alors que INKEY$ se sert de son code ASCII.
1050 Si la position verticale de la base dépasse les limites fixées, on annule le mouvement vertical.
1060 Si COPY est pressé et qu'il n'y a pas de bombardement en cours, on signale dans le flag 'largue' qu'une bombe vient d'être lancée. La position de la bombe est initialisée au centre et en dessous de la base.
1070 Si la base ne s'est pas déplacée, on ignore ce qui suit pour éviter un clignotement permanent du motif.
1210 On prend la couleur du papier pour effacer. CALL &BD19 appelle une routine du système qui synchronise l'affichage avec le balayage du rayon cathodique. DI vous sera expliqué avec EVERY.
1220 On affiche la base en PEN 2 à sa nouvelle position. EI sera commenté avec EVERY.
1230 Le déplacement entraîne une baisse du fuel. Plus vous volez bas, plus vous consommez. Modifiez la formule à votre gré pour compliquer ou faciliter la tâche.

Gestion de la bombe

1300 S'il n'y a pas de bombe lancée, on ignore.
1310 La bombe ne pouvant que descendre, on efface la bombe et on ajoute 1 à sa situation verticale.
1320 On vérifie que la bombe est toujours dans l'écran.
1330 La bombe sort de l'écran. Le flag est annulé, on met une nouvelle bombe dans la base et on vous enlève 10 points pour vous apprendre à viser juste !
1340 Son grave et suite du programme.
1350 La bombe est toujours dans l'écran, on l'affiche.

Lorsque la bombe touche une lettre

1400 Si la bombe est tout en bas, il n'y a rien à tester.
1410 On utilise la routine langage machine implantée en 40000. Cette routine vous donne dans la variable n, la valeur du caractère ASCII lu aux coordonnées X,Y données comme paramètres. Le test se fait sur la ligne située directement sous la position actuelle de la bombe. Notez au passage que l'instruction IF accepte les parenthèses pour tester une condition en bloc. Si aucun caractère valable n'est reconnu : circulez, il n'y a rien à voir.
1500-1500 Annulation de la bombe. Effacement.
1520 Le caractère lu est mis en ASCII.
1600 Test : est-ce le joker ?
1610 La fonction MID$, permet de manipuler les chaînes. Exemples : a$ = "ABCDEFGHIJ" PRINT MID$(a$,3,4) donne CDEF \ b$ = MID$(a$,1,3) PRINT b$ donne ABC \ MID$ (a$,2,3) = "XXX" PRINT a$ donne AXXXEFGHIJ. Le premier terme dans les parenthèses est la chaîne à traiter. Le second présente le point de départ du traitement. Il doit être situé dans la chaîne, à savoir : > 0 et < = Longueur chaîne. Le troisième est la longueur à extraire ou traiter. Dans la ligne 1610, MID$ permet d'extraire de la chaîne à reconstituer le caractère pointé par rang et de le stocker dans a$. Le joker vous vaut 200 l de fuel supplémentaires et on saute dans la routine qui complète le mot en cours.
1700 Ce n'était pas le joker. On teste le fuel.
1710 La fonction RND renvoie un nombre aléatoire compris entre 0 et 0.99999999. Selon sa valeur, il vous sera ajouté ou retranché des litres de fuel.
1720 La valeur du bonus (ou malus) est calculée de manière aléatoire. On affiche en PEN 3.
1730 Son aigu et suite des opérations.
1800 Ni joker, ni fuel. Test du score.1810-1830 : Comme 1700-1730, pour le score.
1900 Ni joker, ni fuel, ni score. Voir si on peut compléter le mot. Teste d'abord si le caractère touché correspond au caractère manquant le plus à gauche du mot. Si ce n'est pas le cas on saute en 2000. - Lettre dans l'ordre -
1940 On a touché une lettre dans l'ordre par "rang". MID$ permet de compléter le mot, l'exploit vous ajoute 100 points. Le rang actuel est copié dans n.
1950 On avance rang de 1 pour le caractère suivant. La fonction MID$ dans la boucle WHILE WEND teste le mot en cours de reconstitution da la gauche vers la droite. Tant que le point n'est pas détecté, "rang" est incrémenté. Dans le cas où le mot serait enfin complété, le point ne sera jamais rencontré et l'incrémentation de "rang" finira par provoquer un "improper argument" dans MID$. Pour cette raison, on teste "rang < lch" à chaque tour de boucle.
1960 Aller à la suite des opérations.- Voir si lettre dans le désordre -
2000 Rechercher sur la longueur totale de la chaîne.
2010 Le caractère retenu est-il présent dans la chaîne à la position n ?
2020 Si oui, manque t-il à la même position dans le mot à compléter ?
2030 Si oui, aux 2 questions précédentes, c'est bon : 50 points de plus et on va ranger la trouvaille.
2040 Pas bon, au suivant.
2050 Ce caractère ne manquait pas ! 25 points de pénalité, son et suite. - Lettre trouvée : Compléter le mot et voir si fini -
2100 On complète le mot avec le caractère a$ à la position n.
2100 On affiche le mot complété, on fait BIP, le total à reconstituer est décrémenté de 1. S'il tombe à zéro, c'est gagné et on saute à la fin du jeu.

Sortie de la boucle commencée en ligne 1000

2150 "flagnew" est mis par la routine sous interruption (EVERY). Si cette routine a été activée, un caractère a changé.
2200 "Flagnew" est annulé. "xket1" et "ylet1" correspondent à la position du caractère supprimé par la routine EVERY, on efface ce caractère.
2210 "xlet(a),ylet(a),lettre$(a)" sont des paramètres du caractère de remplacement que l'on affiche.
2300 Ne trainez pas ! Tous les 20 tours de boucle, vous perdez des points et du fuel...
2310 Affichage du score et du niveau du fuel.
2320 S'il reste du fuel on continue, sinon c'est perdu. Nous arrêtons ici ce premier jeu d'arcade que nous nous proposons d'apprendre à faire vous-même. N'oubliez pas : le prochain Run'Star vous parlera des sous-programmes et vous expliquera en détail le rôle et le fonctionnement des instructions EVERY, DI et EI.

```
10 '
20 '* WORD INVADER *
30 '
40 '- Initialisation -
50 '
60 MEMORY 39999:GOSUB 3300
70 DEFINT a-z
80 SYMBOL 253,&X1,&X11,&X11000111,&X11111111,&X11111111,&X11000111,&X1,0
90 SYMBOL 254,&X11111111,&X100100,&X10011001,&X11111111,&X10111101,&X11000,&X100
00001
100 SYMBOL 255,&X10000000,&X11000000,&X11100011,&X11111111,&X11111111,&X11100011
,&X10000000,0
110 '
120 DIM lettre$(35),xlet(35),ylet(35)
130 BASE$=CHR$(253)+CHR$(254)+CHR$(255)
140 bomb$=CHR$(&FC)
150 lbase=3
160 xmax=40-(lbase-1):MODE 1
170 xbase=xmax/2:ybase=6:largue=0:xlargue=xbase+1:ylargue=ybase
180 nbcar=35:chaine$="RUNSTAR-LOGISTAR":lch=LEN(chaine$)
190 tot=lch:jeu$=STRING$(lch,".")
200 rang=1:jok!=0.025
210 PEN 1:LOCATE 1,1:PRINT chaine$:PEN 3:PRINT jeu$:score=0:fuel=5000
220 PEN 2:LOCATE 1,3:PRINT"SCORE";score:LOCATE 1,4:PRINT"FUEL";fuel
230 LOCATE 13,3:PRINT"HI";hisco;
240 '
250 '- 1ers affichages -
260 '
500 GOSUB 3000:PEN 2:LOCATE xbase,ybase:PRINT base$
510 EVERY 40,0 GOSUB 3200
520 '
530 '- Boucle principale . Test du clavier pour deplacement base -
540 '
1000 xb=xbase:yb=ybase
1010 IF INKEY(1)=0 THEN xbase=xbase+1:IF xbase>xmax THEN xbase=1
1020 IF INKEY(8)=0 THEN xbase=xbase-1:IF xbase<1 THEN xbase=xmax
1030 IF INKEY(0)=0 THEN ybase=ybase-1
1040 IF INKEY(2)=0 THEN ybase=ybase+1
1050 IF ybase<6 OR ybase>18 THEN ybase=yb
   1060 IF INKEY(9)=0 AND largue=0 THEN largue=1:xlargue=xbase+1:ylargue=ybase+1
1070 IF ybase=yb AND xbase=xb THEN 1300
1080 '
1090 '- Deplacer base -
1100 '
1200 IF ybase<6 OR ybase>18 THEN ybase=yb:GOTO 1300
1210 PEN 0:CALL &BD19:DI:LOCATE xb,yb:PRINT base$;
1220 PEN 2:LOCATE xbase,ybase:PRINT base$;:EI
1230 fuel=fuel-20*SQR(ybase+1-6)
1240 '
1250 '- Bombe lancee -
1260 '
1300 IF largue=0 THEN 2150
1310 PEN 0:LOCATE xlargue,ylargue:PRINT bomb$;:ylargue=ylargue+1
1320 IF ylargue<26 THEN 1350
1330 largue=0:xlargue=xbase+1:ylargue=ybase+1:score=score-10
1340 GOSUB 3500:GOTO 2150
1350 PEN 3:LOCATE xlargue,ylargue:PRINT bomb$;
1360 '
1370 '- Bombe touche lettre ? -
1380 '
1400 IF ylargue=25 THEN 2150
1410 n=0:CALL 40000,xlargue,ylargue+1,an:IF (n<63 OR n>90)AND n<>&2A THEN 2150
```

```
1420 '
1430 '- OUI : Annuler bombe et Interpreter -
1440 '
1500 largue=0:PEN 0:LOCATE xlargue,ylargue:PRINT bomb$;CHR$(8);CHR$(10);" "
1510 xlargue=xbase+1:ylargue=ybase+1
1520 a$=CHR$(n)
1530 '
1540 '- Joker -
1550 '
1600 IF a$<>"*"THEN 1700
1610 a$=MID$(chaine$,rang,1):fuel=fuel+200:GOTO 1940
1620 '
1630 '- à = Fuel ? -
1640 '
1700 IF a$<>"à"THEN 1800
1710 IF RND(1)<0.5 THEN n=-1 ELSE n=1
1720 bon=RND(1)*200*n:fuel=fuel+bon:LOCATE 11,4:PEN 3:PRINT"FUEL ";bon
1730 GOSUB 3600:GOTO 2130
1740 '
1750 '- ? =Score ? -
1760 '
1800 IF a$<>"?"THEN 1900
1810 IF RND(1)<0.5 THEN n=-1 ELSE n=1
1820 bon=RND(1)*200*n:score=score+bon:LOCATE 11,4:PEN 3:PRINT"SCORE";bon
1830 GOSUB 3600:GOTO 2130
1840 '
1850 '- Voir si la lettre touchee est utile -
1860 '
1900 IF a$<>MID$(chaine$,rang,1)THEN 2000
1910 '
1920 '- Lettre dans l'ordre -
1930 '
1940 MID$(jeu$,rang,1)=a$:score=score+100:n=rang
1950 rang=rang+1:WHILE MID$(jeu$,rang,1)<>"." AND rang<lch:rang=rang+1:WEND
1960 GOTO 2100
1970 '
1980 '- Voir si lettre dans le desordre -
1990 '
2000 FOR n=1 TO lch
2010 IF MID$(chaine$,n,1)<>a$ THEN 2040
2020 IF MID$(jeu$,n,1)<>"."THEN 2040
2030 score=score+50:GOTO 2100
2040 NEXT
2050 score=score-25:GOSUB 3500:GOTO 2150
2060 '
2070 '- Lettre trouvee : Completer le mot et voir si fini -
2080 '
2100 MID$(jeu$,n,1)=a$
2110 LOCATE 1,2:PEN 3:PRINT jeu$;CHR$(7);:tot=tot-1:IF tot=0 THEN 3410
2120 '
2130 '- Deplacer une lettre s'il y-a lieu -
2140 '
2150 IF flagnew=0 THEN 1000
2160 '
2200 flagnew=0:PEN 1:LOCATE xlet1,ylet1:PRINT" "
2210 LOCATE xlet(a),ylet(a):PRINT lettre$(a);
2220 '
2230 '- Diminuer et afficher score s'il y-a lieu -
2240 '
2300 cnt=cnt+1:IF cnt=20 THEN score=score-1:fuel=fuel-10:cnt=0
2310 LOCATE 6,3:PRINT score:LOCATE 5,4:PRINT fuel
2320 IF fuel>1 THEN 1000 ELSE 3400
```

COIN DES BIDOUILLEURS : Lilian se venge

Deuxième partie

Cette rubrique, débutée dans Run'Star 4, est là pour vous aider. Si vous voulez que votre micro n'ait plus de secret pour vous, suivez nos indications.

Lilian Margerie

S.O.S. vecteurs systèmes !

Pour les non-habitués de la programmation Assembleur, les vecteurs sont peu, voir complètement inconnus. Et pourtant, leur utilisation, même pour un utilisateur du Basic, peut être des plus bénéfiques. Nous allons voir ce que l'on peut tirer de ces adresses utilisées par l'ordinateur allouant un gain de temps certain. Donc, en route pour un voyage au centre de la mémoire vive du CPC. La première adresse que nous allons voir peut rendre d'innombrables services dans un programme. Il arrive souvent que, lors de l'exécution d'un programme, on demande à l'utilisateur de frapper une touche pour continuer. La ligne est du style :

```
...
100 WHILE A$="":A$=INKEY$:WEND
...
```

ou encore :

```
...
210 A$=INKEY$
220 IF A$="" THEN GOTO 210
...
```

Or, ces lignes peuvent être remplacées par un CALL &BB18. Ce Call attendra que l'utilisateur frappe une touche du clavier. Vous pouvez constater qu'il prend beaucoup moins de place en mémoire. Il arrive parfois que lors d'une présentation ou de l'exécution d'un jeu à programmer, on ne veuille pas que l'utilisateur puisse arrêter le programme en frappant deux fois la touche <ESC>. La première solution est d'utiliser la fonction Basic ON BREAK GOSUB. Le programme serait donc :

```
10 ON BREAK GOSUB 1000
20 ...
...
1000 RETURN
```

Ce programme empêche l'arrêt du programme uniquement. Or, il suffit de taper : POKE &BDEE,&C9. Cela aura pour conséquence de déconnecter complètement la touche <ESC> pendant l'exécution d'un programme. De plus, il déconnecte la remise à zéro qui est habituellement effectuée lorsque l'on frappe simultanément les touches <CTRL> + <SHIFT> + <ESC>. Son emploi est donc des plus intérressants. On peut faire varier l'emploi, en faisant :

```
POKE &BDEF,0:POKE &BDF0,0
```

Après avoir tapé ces deux Pokes, une remise à zéro de l'ordinateur s'effectuera lorsque l'on pressera la touche <ESC>.

Le balayage vidéo, la propreté à moindre prix !

Voici un call un peu particulier qui demande des explications approfondies. C'est le test de la position du balayage vidéo. What is it ? comme dirait nos amis belges. Dans le ventre de notre cher CPC, existe un microprocesseur qui répond au joli nom de Gate Array. Or, cette "puce" est de bonne compagnie car c'est elle qui s'occupe de la gestion de l'image vidéo de l'Amstrad. Par exemple, quand vous changez la couleur du fond de l'écran ou encore quand vous tapez un caractère au clavier, c'est le Gate Array qui s'occupe de l'affichage à l'écran. Alors, quel est le déroulement ? Eh bien, vous savez peut-être que la mémoire écran du CPC se trouve, au départ, à l'adresse &C000 et se termine en &FFFF. Donc, elle a une taille de 16384 octets. Quand elle dit à l'ordinateur d'afficher un texte à l'écran avec un PRINT, par exemple, il stocke le caractère dans la mémoire écran sous forme d'octets. Seulement, il faut afficher lesdits octects à l'écran. Et c'est là que notre bon Gate Array entre en compte. Tout les 1/50e de seconde, il va chercher les données se trouvant dans la mémoire écran afin de les afficher à l'écran. Bien sûr, l'affichage ne se fait pas n'importe comment. Il s'effectue comme l'affichage d'une image de télévision. C'est-à-dire de haut en bas. On parle alors de balayage vidéo. Le balayeur vidéo (concierge dans ces heures perdues) part tranquillement du haut de l'écran et affiche les lignes qui composent l'écran. Il affiche également la bordure. En clair, le Gate Array copie le contenu de la mémoire écran sur le moniteur. Le balayage

étant effectué très rapidement, il est souvent très utile de savoir où il se trouve. Tout programmeur faisant des animations, des scrollings et toutes les autres choses qui bougent à l'écran, est obligé d'attendre le début du balayage vidéo avant de commencer les différents affichages. Pour cela, il existe un moyen d'attendre le début du balayage vidéo. Sur 6128, il existe la fonction Basic FRAME. Sur tous les CPC, le meilleur moyen est de faire un CALL &BD19. Ce call permet de dire au Gate Array : "On attend que tu sois prêt à démarrer et tu nous préviens quand tu l'es". Pour ceux qui programment en Assembleur, voici la routine langage-machine du CALL &BD19 en ROM :

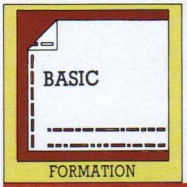
```
BALAI   LD B,&F5
        IN A,(C)
        RRA
        JR NC,BALAI
```

C'est tout. Il est donc plus rapide de l'écrire directement et cela fait gagner du temps en programmation Assembleur. Il ne vous reste plus qu'à faire comme la sorcière de Cauldron, enfourchez le balai magique !

```
10 'LISTING BASIC DU SCROLLING
20 FOR I=&A000 TO &A053
30 READ A$
40 POKE I,VAL("&"+A$)
50 NEXT
60 END
70 DATA FE, 1, C0, DD, 6E, 0, DD, 66, 1, 11,
   1, 0, CD, 1D, BC, DD, 21, 4C, A0, 6, 8, DD,
   74, 0, DD, 75, 1, CD, 26, BC, DD, 23, DD,
   23, 10, F1, F3, 6, F5, ED, 78, 1F, 30, FB,
   DD, 21, 4C, A0, 6, 8, C5, DD, 56, 0, DD,
   5E, 1, 62, 6B, 23
80 DATA 1A, 1, 4F, 0, ED, B0, 12, DD, 23, DD,
   23, C1, 10, E8, FB, C9, 0, 0, 0, 0, 0, 0, 0, 0
```

Word invaders (2ème partie) - Michel Maigrot

```
2330 '
2340 '*** SOUS PROGRAMMES ***
2350 '
2360 '- Initialiser la position et la valeur des lettres et afficher -
2370 '
3000 FOR i=0 TO nbcar
3010 GOSUB 3100:GOSUB 3150
3020 NEXT
3030 RETURN
3040 '
3050 '- Position et valeur aleatoire des caracteres -
3060 '
3100 xlet(i)=RND(1)*xmax+1base:IF xlet(i)<2 OR xlet(i)>xmax THEN 3100
3110 ylet(i)=RND(1)*25:IF ylet(i)<20 OR ylet(i)>25 THEN 3110
3120 car=RND(1)*90:IF car<63 THEN 3120 ELSE lettre$(i)=CHR$(car)
3130 IF RND(1)<jok! THEN lettre$(i)="*"
3140 RETURN
3150 PEN 1:LOCATE xlet(i),ylet(i):PRINT lettre$(i);
3160 RETURN
3170 '
3180 '- Sous programme 'EVERY' pour changer et deplacer une lettre -
3190 '
3200 flagnew=1:i=RND(9)*nbcar
3210 xlet1=xlet(i):ylet1=ylet(i):a=i:GOTO 3100
3220 '
3230 '- Implatation routine lire caractere -
3240 '
3300 FOR i=0 TO 29:READ a:POKE 40000+i,a:NEXT:RETURN
3310 DATA &fe,&03,&c0,&cd,&78,&bb,&e5,&dd,&6e,&02,&dd,&66,&04,&cd,&75,&bb
3320 DATA &cd,&60,&bb,&dd,&6e,&00,&dd,&66,&01,&77,&e1,&c3,&75,&bb
3330 '
3340 '-Fin de partie -
3350 '
3400 PEN 3:LOCATE 1,5:PRINT"PANNE SECHE !":GOTO 3420
3410 PEN 3:LOCATE 1,5:PRINT"SERIE REUSSIE !"
3420 PEN 2:LOCATE 1,6:PRINT"'F$ini 'R$eprise"
3430 IF score>hisco THEN hisco=score
3440 a$="":WHILE a$<>"R"AND a$<>"F":a$=UPPER$(INKEY$):WEND
3450 x=REMAIN(0):a=0:IF a$="R"THEN 150 ELSE END
3460 '
3500 SOUND 1,500,20,15:RETURN
3600 SOUND 1,200,20,15:RETURN
```



programmez votre : premier jeu d'arcade

Listing programme (voir page 31)

Voici le deuxième volet de ce jeu d'arcade "Word Invaders". A la fin de cet article, vous aurez réalisé votre premier jeu d'arcade.

Michel Maigrot

Les sous-programmes

3000 Tous les caractères de 0 à "nbcar"
3100 Fixer aléatoirement la colonne d'affichage, vérifier si elle est compatible avec les coordonnées écran (la bombe ne peut être lâchée dans les 2 colonnes extrêmes).
3110 La même pour ligne d'affichage entre 20 et 25.
3120 La même pour un caractère entre "A" et "Z".
3130 Reprise aléatoire et remplacement par joker si il y a lieu. 3150 Afficher une lettre en PEN 1.

Sous programme EVERY pour changer et déplacer une lettre

3200 Mise du flag et déterminer la lettre à changer.
3210 Mémoriser la position de l'ancienne lettre pour pouvoir l'effacer et aller en 3100 pour affecter les nouvelles valeurs. C'st à la fin de la routine 3100 que se trouve le RETURN qui semble manquer ici.

Implantation routine lire caractère

3300-3320 Implantation des codes machine de la routine "lire caractère" utilisée par le test de la bombe. Ces 3 lignes peuvent être incorporées à n'importe quel autre programme Basic. Si l'on souhaite une autre adresse d'implantation, il suffit de changer 40000 par une autre valeur.
3400-3450 Fin de partie et calcul du HI-SCORE. REMAIN sera expliqué avec EVERY.
3500-3510 Sonorisation qui aurait avantage à être plus sophistiquée !

EVERY

Plutôt mal aimée des débutants car très mal décrite dans les manuels, cette commande n'en reste pas moins une des plus puissantes du Basic CPC.
Syntaxe : EVERY temps, chronomètre GOSUB linge.
On notera l'absence de ":" avant GOSUB qui est tout à fait normale.
"temps" : est la fréquence d'appel du sous-programme désigné par GOSUB ligne. Cette valeur s'exprime en 1/50èmes de secondes.
"chronmètre" : est le numéro d'ordre de l'une des horloges internes du CPC et peut varier de 0 à 3.
Effets : Le sous-programme GOSUB sera régulièrement exécuté tous les "temps"*1/50. C'est-à-dire que le délai fixé étant écoulé, le programme principal s'interrompt (quelque soit l'opération en cours) et ne reprend son cours normal qu'une fois la routine EVERY terminée.

```
Exemple :
10 EVERY 50,0 GOSUB 100
20 t=0
30 LOCATE 2,2:PRINT t
40 GOTO 30
50 END
100 t=t+1
110 RETURN
```

Ce programme compte les secondes écoulées et les affiches. Il ne s'arrête que si l'on appuie sur BREAK.

EFFETS PERNICIEUX DE EVERY

Lorsque l'on souhaite un véritable usage de EVERY, il faut tenir compte des problèmes suivants :

```
Tapez ce qui suit et faites RUN :
10 EVERY 10 GOSUB 70
20 '
30 FOR I=1 TO 80
40 PRINT I;
50 NEXT
60 END
70 I=3:RETURN
```

Cette boucle qui ne veut plus stopper est des plus irritantes, c'est pourtant tout à fait normal car EVERY modifie périodiquement la valeur de "i" utilisée par la boucle FOR — NEXT. C'est le bogue classique dans un programme sous interruptions.

```
Ce qui suit est des plus vicieux :
10 EVERY 10 GOSUB 60
20 '
30 LOCATE 10,10:PRINT"PROGRAMME
   PRINCIPAL"
40 GOTO 30
50 '
60 LOCATE 10,20:PRINT"EVERY"
70 RETURN
```

Laissez tourner ce programme quelques secondes et observez ce qui arrive au texte "PROGRAMME PRINCIPAL" !
Explication : Tôt ou tard, l'interruption se produit entre l'instruction LOCATE 10,10 et PRINT"PROGRAMME PRINCIPAL" de la ligne 20. Si l'on décompose les opérations on obtient : LOCATE 10,10 interruption et GOSUB 1000 où l'on trouve : LOCATE 10,20:PRINT"EVERY" qui sont exécutés DEPLACENT LE CURSEUR TEXTE ! Ceci fait, fin de la routine d'interruption et reprise sur PRINT"PROGRAMME PRINCIPAL" qui va s'afficher là où EVERY a laissé le curseur... Ceci s'applique aussi aux instructions MOVE,DRAW,PLOT, etc... Le problème étant posé, voyons les remèdes qui sont au nombre de deux :
— 1— Vous n'avez pas envie de vous casser la tête, alors faites comme moi dans "Word invaders", n'utilisez pas, dans EVERY, de commande déplaçant le curseur, n'employez pas de noms de variables utilisées par le programme principal.
EVERY ne servira qu'à modifier un "flag" qui sera interprété par le programme principal.

```
10 EVERY 10 GOSUB 70
20 '
30 FOR I=1 TO 80
40 PRINT I;
50 NEXT
60 N=3:RETURN
```

Ici, I devient N en ligne 70

```
10 EVERY 10 GOSUB 60
20 '
30 LOCATE 10,10:PRINT"PROGRAMME PRIN-
   CIPAL" 31 IF cnt=1 THEN cnt=0:LOCATE
   10,20:PRINT"EVERY"
40 GOTO 30
50 '
60 cnt=1 70 RETURN
```

Là, la commande directe est remplacée par un compteur "cnt" qui est testé et remis à zéro par le programme principal.
— 2— Utiliser DI & EI DI : Cette instruction ordonne au CPC la suspension de tout programme mis sous interruptions. Toute routine EVERY sera inhibée par DI jusqu'à la rencontre avec l'instruction EI qui rétablit le cycle normal des interruptions.
Une instruction DI judicieusement placée, permettra donc d'effectuer une opération complète dans le programme principal, sans être perturbé par une commande placée dans EVERY.

```
 10 EVERY 10 GOSUB 150
 20 DI
 30 FOR I=1 TO 80
 40 PRINT I;
 50 NEXT
 70 EI
 80 PRINT
 90 '
100 FOR I=1 TO 80
110 PRINT I;
120 NEXT
130 END
140 '
150 I=3:RETURN
```

La première boucle de 20 à 50 s'exécute correctement, la seconde sans DI & EI...

```
10 EVERY 10 GOSUB 60 '
20 '
30 DI:LOCATE 10,10:écran, changer la
   valeur de NBCAR.
```

150 Notre base occupe 3 caractères, il est important d'en tenir compte pour les tests de sortie d'écran ! Changez cette valeur pour voir le résultat...
160 Cette ligne conditionne le mode d'écran. La remplacer par xmax=20-(lbase-1):MODE 0, pour faire fonctionner le programme en mode 0. xmax=80-(lbase-1):MODE 2 fera tourner le programme en mode 2 mais il faudra de plus remplacer toutes les commandes PEN 2, PEN 3 par PEN 1 car seules 2 couleurs sont possibles dans ce mode. 170 On initialise la position d'affichage de la base. La bombe à larguer devant suivre la base, on lui donne la même position avec X + 1 pour centrer la position du largage sur le second caractère de la base. 180 C'est ici que l'on fixe la chaîne de caractères à reproduire. La longueur du texte ne doit pas excéder 20,40,80 caractères selon le mode choisi. Il est impératif de tout composer en majuscules ! N'oubliez pas que R I=1 to 20:PRINT "A";:next:EI

Le facteur temps

Le temps d'exécution de la routine EVERY est vital ! Essayez :

```
10 EVERY 10 GOSUB 60
20 '
30 a=0 40 a=a+1=LOCATE 10,10:PRINT
   a;:GOTO 40
50 '
60 FOR tp=1 TO 5000:NEXT:a=0
70 a=0:PRINT CHR$(7);
80 RETURN
```

La boucle d'attente en ligne 60 dépasse largement le temps de répétition de la routine EVERY. Conséquence, celle-ci est sollicitée en permanence et le programme principal ne peut plus s'exécuter !
Changez la ligne 10 en EVERY 10000 GOSUB 60, attendez un peu après RUN pour voir le résultat. En principe, une routine sous interruption doit être aussi brève que possible, il ne vaut rien de mettre un programme complet sous EVERY !
Se contenter d'y modifier des données qui seront interprétées par le programme principal est satisfaisant dans 99% des cas ! Ce qui suit est un exemple de ce qu'il faut faire :

```
 10 EVERY 8 GOSUB 90
 20 '
 30 MODE 0
 40 p=1:p1=10:y=1
 50 WHILE y1=y:WEND
 60 DI:y1=y:LOCATE 1,y:EI:PEN p:PAPER
    p1:PRINT" EVERY CHANGE"
 70 GOTO 50
 80 '
 90 p=p+1:IF p>15 THEN p=1
100 p1=p1+1:IF p1>15 THEN p1=1
110 y=y+1:IF y>25 THEN y=1
120 RETURN
```

Il est pratique d'effectuer les tests sous EVERY, toutefois, si l'on diminue la temporisation en dessous de 5, le programme principal commencera à sauter des lignes... Si la temporisation est beaucoup plus importante, la boucle du programme principal (50-70) s'exécutera plusieurs fois avec les mêmes valeurs de p,p1,y ce qui n'est pas vraiment nécessaire. En ligne 60, DI:y1=y:LOCATE 1,y:EI, permet de mémoriser la coordonnée d'affichage dans y1. De retour en ligne 50, la boucle WHILE y1=y:WEND, attend sagement que EVERY ait changé la valeur de 'y' avant d'effectuer un nouvel affichage. C'est une bonne méthode pour synchroniser un programme avec une routine d'interruption quelconque.

Syntaxe : x=REMAIN(chronomètre) ou PRINT REMAIN(chromètre).

Ceci nous donne le temps qu'il reste avant le prochain appel à une routine EVERY ; ce qui est tout à fait dépourvu d'intérêt. En revanche, cela a aussi pour effet de stopper définitivement l'opération EVERY activée par l'horloge donnée (0 à 3). C'est indispensable en cas de réinitialisation du programme principal par lui-même (dernières lignes de "Word invaders"). Faute de ce REMAIN, le cycle d'interruption se poursuivrait sans cesse et pertuberait la réinitialisation des variables lors de la reprise du jeu. Supprimez ce REMAIN et vous comprendrez très vite le problème... Pour mémoire, citons : AFTER temps, chronomètre GOSUB ligne. Qui est en tout point semblable à EVERY sauf que la routine désignée par GOSUB n'est appelée qu'une seule fois et ne se répète pas. Voilà, vous êtes désormais capable de réaliser votre propre jeu d'arcade, vous avez même le droit de vous faire votre menu et d'y ajouter vos initiales !