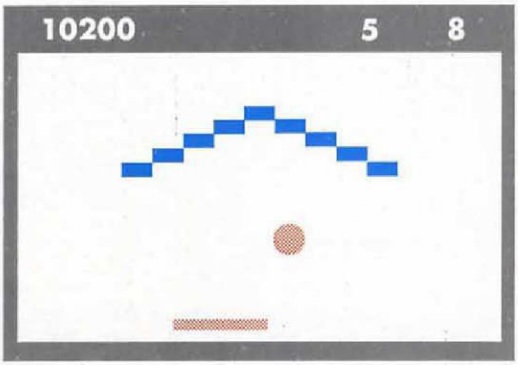
PAR FRANÇOIS LE GRIGUER

JOYSTICK-BREAKER

TU VIENS DE TERMINER JOYSTICK-ATTACK ET, POUR CONTINUER TON AVENTURE INFORMATIQUE IL FAUT, IMMÉDIATEMENT, ABORDER UN NOUVEAU PROGRAMME, PEUT-ÊTRE UN PEU PLUS DIFFICILE, MAIS SURTOUT CONTENANT DES PROBLÈMES NOUVEAUX À RÉSOUDRE. PARMI BEAUCOUP DE THÈMES POSSIBLES J'AI RETENU UN CASSE-BRIQUES, PAS DU TOUT PAR HASARD, MAIS POUR LA RÉFLEXION NÉCESSAIRE À SON DÉVELOPPEMENT.SANS IMAGINER DE RIVALISER AVEC ARKANOID TU VAS VOIR QUE LE SUJET EST VRAIMENT PASSIONNANT. CETTE NOUVELLE INITIATION VA SE DÉROULER EXACTEMENT SUIVANT LE MÊME PLAN QUE LA PRÉCÈDENTE: -L'EXPOSÉ DES MODULES À RÉALISER AVEC LES IDÉES, LES FICHIERS NÉCESSAIRES ET LES ORDINOGRAMMES QUI TE PERMETTENT DE PRÉPARER TON PROGRAMME. -LA SEMAINE SUIVANTE, LE LISTING BASIC AVEC TOUTES LES EXPLICATIONS D'ACCOMPAGNEMENT POUR CORRIGER OU MIEUX COMPRENDRE LA SÉQUENCE. DE SEMAINE EN SEMAINE TON PROGRAMME SE CONSTRUIT AVEC, EN PLUS LA RUBRIQUE INDISPENSABLE AUX DÉBUTANTS: «ET POUR QUELQUES DEFINITIONS DE PLUS...» DANS LAQUELLE ON ABORDE LES BASES DE L'INFORMATIQUE ET TOUTES LES DÉFINITIONS NÉCESSAIRES POUR LA PRÉPARATION DES SÉQUENCES.

J'APPRENDS

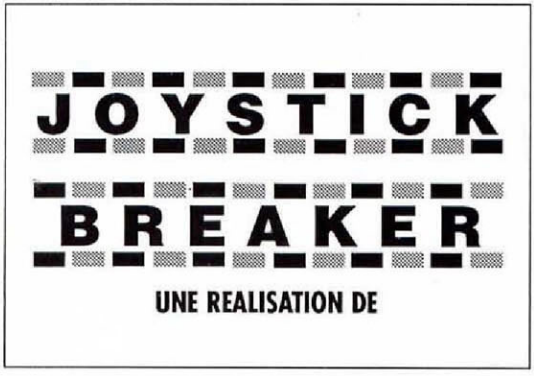


SCENARIO
COULEURS ET BRIQUES
GENERIQUE
REPERES DU JEU
REDEFINITION DES TABLEAUX
PAGE DES SCORES
DEBUT DE PARTIE
GESTION DE LA BALLE
SCORE ET BONUS
GAME OVER
DEPLACEMENT DE LA RAQUETTE
SONS
COMPLEMENTS

UN TITRE QUI CASSE DES BRIQUES

Tu dois aborder un nouveau jeu avec une imagination débordante, des idées plein la tête tu plonges dans la définition du scénario. Le titre vient souvent beaucoup plus tard mais pour un casse-briques il n'y a que l'embarras du choix. Le titre doit être le plus court possible, évoquer le thème du jeu et avoir, si possible, une consonnance anglaise. C'est ce qui marche le mieux pour les jeux. Je connais bien le sujet étant moi-même auteur de jeux. Alors pourquoi ne pas suivre la même démarche et, je te propose: **JOYSTICK-BREAKER** un titre qui casse des briques, tout à fait dans la lignée des casse-briques célèbres. Après le titre, les détails de notre scénario en restant modeste, n'oublie pas que le BASIC est très pratique quant à son écriture mais, la place occupée en mémoire et sa lenteur d'exécution fixent très vite les limites d'un programme. Que nous faut-il pour réaliser un casse-briques?

- des briques, une raquette et une balle (élémentaire mon cher WATSON!!!)
- des tableaux (le plus possible pour ne pas être monotone)
- une page de présentation - une page de scores (pour les meilleurs !)



PLUS FORT QUE LES PROS

Dans un domaine au moins tu vas faire plus fort que beaucoup de professionnels avec 4Ø tableaux différents et tous redéfinissables. Génial non! Encore un petit plus en prévoyant pour chaque tableau un nombre de balles maximum suivant les difficultés ou la disposition des briques. Pour les briques tu commences avec trois dessins et, tu en ajouteras bien d'autres suivant tes gouts. Avec tout cela de la couleur, beaucoup de couleurs et, c'est le mode Ø qui nous donne un affichage de 16 couleurs à l'écran, toujours choisies parmi les 27 couleurs AMSTRAD. Tu te souviens des autres caractéristiques du mode Ø: 2Ø caractères par ligne et toujours 25 lignes en hauteur. Le nombre de lignes reste identique quel que soit le mode tandis que le nombre de caractères est deux fois moins important qu'en mode 1, ce qui donne une forme rectangulaire tout à fait adaptée aux briques.

TONS	NUMÉROS DE COULEURS					
gris	13					
rouge	6	7				
bleu	2	14	20			
vert	9	10	18	21	22	
jaune	15	25				
noir	0					
marron	16					
blanc	26					

16 couleurs au total INK Ø à INK 15

Une répartition que tu peux modifier

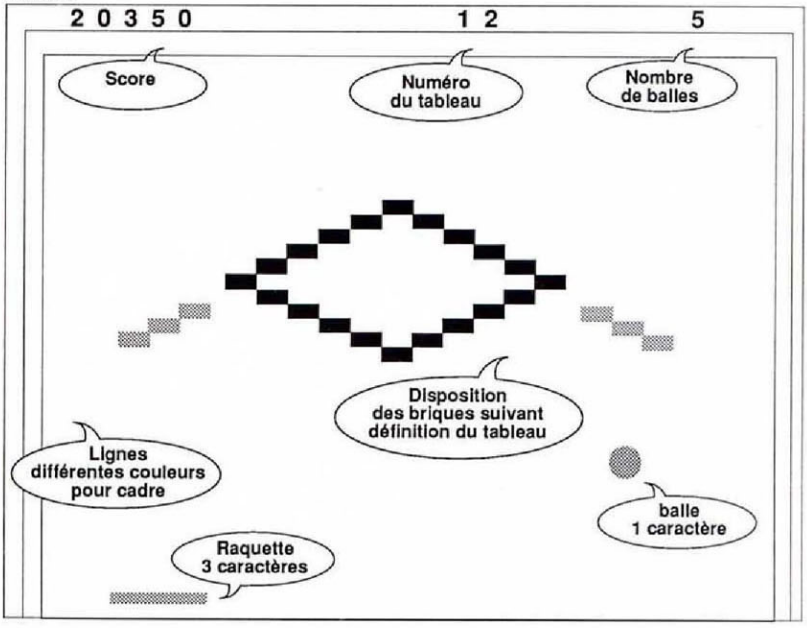
DES COULEURS ET DES BRIQUES

Tu entres maintenant dans le premier module à réaliser: la sélection des 16 couleurs et leur déclaration en INK de Ø à 15. Il s'agit de faire un choix assez varié dans les nuances. Les numéros sont incorporés au programme sous la forme de DATA et, une petite boucle initialise INK à partir de la lecture des DATA.

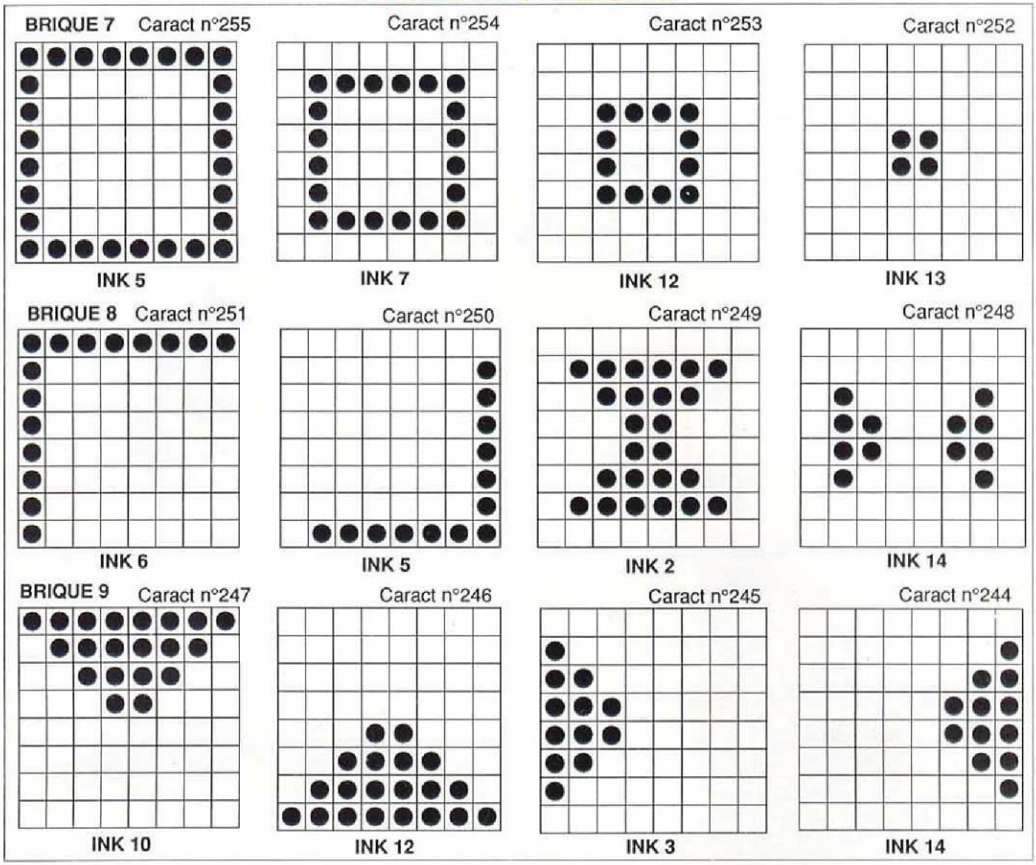
Toujours dans notre premier module, le dessin de chacune des trois briques, numérotées 7 8 9, par l'utilisation de caractères redéfinis, une méthode qui a été largement développée dans JOYSTICK numéro 5. Une précision importante reste à ajouter. L'affichage d'un caractère s'effectue avec une seule couleur (suivant PEN). Les briques, vont comporter plusieurs couleurs, donc plusieurs caractères. Chaque caractère avec un dessin particulier, affiché avec une couleur différente, va se superposer au précédent en utilisant le mode transparent (transparent = pas d'affichage des pixels à Ø). Les dessins sont constitués de 4

UN TABLEAU A L'ECRAN

Avant d'entamer le détail des modules à préparer, je voudrais te montrer tout de suite ce que sera le jeu avec un tableau à l'écran. L'affichage fera partie de la séquence «Début de Partie».



DESSIN DES BRIQUES



éléments (4 caractères par brique).
Le tableau «Dessin des Briques» récapitule toutes les données pour déclarer les caractères par l'instruction SYMBOL, après avoir codifié chacune des lignes, et écrire les 3 sous-programmes pour l'affichage de nos briques. Si tu débutes ne t'inquiètes pas, tu trouveras, la semaine prochaine, le listing avec tous les commentaires.

DES BRIQUES DANS LE GENERIQUE

Second module à préparer c'est le générique qui contient le titre et, ton nom pour la réalisation. Ce module, très simple, ne nécessite pas d'ordinogramme, une simple énumération des opérations nous suffit. Puisqu'il s'agit d'un casse-briques utilise des briques pour encadrer le titre. Une première ligne de briques, J O Y S T I C K sur la seconde, suivi d'une nouvelle ligne de briques. Tu renouvelles la même chose pour B R E A K E R. Ensuite afin de varier la présentation tu traces un cadre pour entourer «une réalistion de .....». En fin d'affichage tu n'oublies pas de prévoir une boucle de tempo qui maintient à l'écran, quelques instants, ton générique et le module est terminé. Simple non?

DES REPERES POUR LE JEU

Sur le papier, dessiner un écran avec ses bordures, des briques un peu partout, une raquette et une balle c'est à la portée de tout le monde. Il faut maintenant réfléchir un peu sur la méthode qui va nous permettre de gérer cet ensemble, d'en connaitre les limites, de situer les obstacles. Lorsque la balle se déplacera il sera nécessaire de savoir ce qu'e.. touche, une brique ou la bordure, la raquette ou rien, pour définir la suite, une brique détruite, un changement de trajectoire, ou encore la perte de la balle. Un fichier doit nous permettre de répondre à tous ces besoins, tabl comme tableau, avec une capacité de 2Ø sur l'axe des x et, de 25 sur l'axe des y (2Ø carcatères sur 25 lignes comme l'écran), un fichier que tu déclares par: DIM tabl(2Ø,25) pour son nom et sa capacité (séquence «Ini-

CODES EN TABL

Type	Code	Observations
Bord écran Haut	1	Lignes 1 et 2 (y=1 ou 2)
Bord écran Droit	2	Colonne 20 (x=20)
Bord écran gauche	4	Colonne 1 (x=1)
Bord écran bas	6	ligne 25 (y=25)
Raquette	3	ligne 25 les 3 caract. occupés par la raquette
Briques	7	suivant la position de chaque brique en x et y
	8	
	9	

tialisation Fichier»). Maintenant chaque élément de la surface de jeu va porter un code qui, mémorisé en tabl, indique sa présence et son type, récapitulé dans le tableau «Codes en tabl». Le fichier tabl, mis à jour à chaque changement de tableau, pourra être interrogé en fonction de la position de la balle.

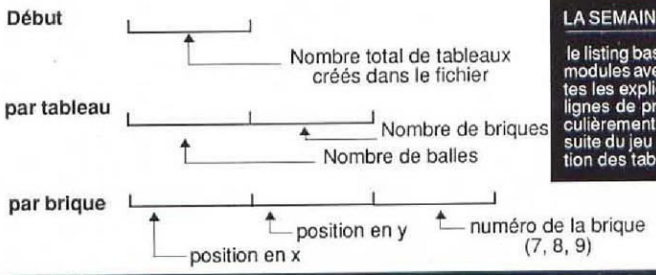
JOYSTICK-BREAKER ET LES 4Ø TABLEAUX

L'organisation de nos 4Ø tableaux doit être prévue dès le début par le chargement d'un fichier particulier nommé «tableaux», un fichier qu'il faut détailler. Tu vas l'exploiter octet par octet, ce n'est plus le programme Basic qui se charge des zones, mais toi qui calcule les adresses des différentes données. La formule la meilleure consiste, sur du papier quadrillé, à tracer le dessin d'enregistrement, avec 4 carreaux par octet, en indiquant le contenu.

Chaque tableau contient 5Ø briques maximum: 5Ø multiplié par 3 octets par brique donne 15Ø octets, en y ajoutant les 2 octets d'identification du tableau, ceci représente 152 octets par tableau. Tu multiplies 152 par 4Ø tableaux, plus l'octet de début et tu trouves la capacité totale de ton fichier «tableaux» soit 6Ø81 octets.

Le processus que tu viens de suivre, dans la description d'un fichier, est très important, surtout si tu rêves de pratiquer un jour l'Assembleur, un langage qui exige ce découpage, octet par octet, pour toutes les données.

FICHIER TABLEAUX



JOYSTICK-BREAKER prend forme, tu commences à voir de quoi il est fait, comment il fonctionnera. D'ici la semaine prochaine tu prépares ton générique, l'initialisation des couleurs, la redéfinition des caractères et la séquence d'affichage des briques.

Bien Joystiquement vôtre
François LE GRIGUER

LA SEMAINE PROCHAINE

le listing basic des premiers modules avec, toujours, toutes les explications pour les lignes de programme particulièrement importantes la suite du jeu avec la redéfinition des tableaux.

POUR QUELQUES DEFINITIONS DE PLUS...

L'instruction SYMBOL permet de donner une nouvelle définition à un caractère. Les numéros de 33 à 255 sont accessibles. Dans le tableau «Dessin des Briques» les matrices 8 lignes par 8 carreaux représentent les 64 points (pixels) de chaque caractère. Chaque ligne est à codifier pour indiquer les pixels à afficher. Ceci se fait de deux manières: soit par l'expression binaire Ø ou 1 (Ø=éteint et 1=allumé), soit par l'addition de la valeur de chaque pixel. Le petit schéma suivant te donne la valeur de chaque bit, à l'intérieur d'un octet, suivant les puissances de 2:

128	64	32	16	8	4	2	1	**1 octet = 8 bits**

Chacun des bits, s'il est égal à 1, renvoie la valeur qui correspond à son rang à l'intérieur de l'octet. L'addition de toutes ces valeurs donne la valeur totale de l'octet.
Pour la petite histoire, bit vient de Binary Digit, 8 bits associés forment 1 octet et 1 Ko contient 1Ø24 octets (1 K = 2 puissance 1Ø). Enfin 256 combinaisons sont possibles dans 1 octet (valeur Ø à 255). Si l'interprétation du contenu est alphanumérique la valeur correspond à un numéro de caractère (codification ASCII ou extension), sinon il s'agit d'une valeur numérique directement. Chaque octet est adressable individuellement suivant sa place en mémoire et la fameuse instruction POKE, que tu pratiques certainement depuis longtemps, permet de modifier le contenu de n'importe lequel des octets de la mémoire. L'inverse de POKE, c-a-d PEEK nous donne la valeur de l'octet adressé. PEEK et POKE sont les amis inséparables des allumés de trucs et astuces et, de JEUX...CRACK en particulier.

Après le courrier reçu, il parait nécessaire de reparler des instructions de boucle: FOR NEXT. La meilleure méthode étant l'exemple en voici quelques-uns, avec leurs commentaires.

```
1Ø FOR x=1 TO 15
2Ø LOCATE x,1Ø:PRINT "*";
3Ø NEXT x
```

Cette boucle de programme va se répéter 15 fois (1 à 15), et affiche le symbole «*» de la position 1 à la position 15 (x prend successivement les valeurs 1 à 15) et ceci en ligne 1Ø (déclaré en LOCATE).

```
1Ø FOR y=25 TO 1 STEP -1
2Ø LOCATE 8,y:PRINT «joystick»;
3Ø NEXT y
```

Ici répétition de la boucle 25 fois (de 25 à 1) mais dans un sens décroissant par STEP -1 (STEP = la définition d'un pas et la valeur associée indique le nombre à ajouter ou à déduire à chaque passage en NEXT). Le résultat de cette boucle: affichage de «joystick» du bas de l'écran vers le haut à partir de la position 8 (en LOCATE).
Un dernier exemple pour l'initialisation des INK à partir des numéros de couleurs écrits en DATA:

```
1Ø FOR j=Ø TO 15
2Ø READ c
3Ø INK j,c
4Ø NEXT j
```

La variable j prend successivement les valeurs de Ø à 15, correspondant aux numéros d'INK. Avec READ c, c contient le numéro de couleur existant en DATA. A chaque READ l'adresse en DATA passe à la donnée suivante. INK j,c initialise INK (numéro = j). Cette boucle répétée 16 fois remplace l'écriture longue, fastidueuse, aberrante, tout à fait inadmissible! (on se calme) de: READ c:INK Ø,c:READ c:INK 1,c etc... jusqu'à READ c:INK 15,c
Bien vu? OK ! Toutes ces lignes vont te paraitre complètement superflues à toi l'as de la programmation, tu riquanes, mais pense un peu à tes débuts, eh oui, d'autres commencent, bienvenue aux débutants!
Et pour conclure cette rubrique un rappel important sur l'organisation générale d'un programme. Celui-ci doit être structuré, c-a-d suivre un ordre rigoureux avec, un programme principal et, des séquences indépendantes écrites comme des sous-programmes, en y ajouter beaucoup de commentaires, la seule formule pour pouvoir, à tout moment, reprendre le listing pour des modifications.

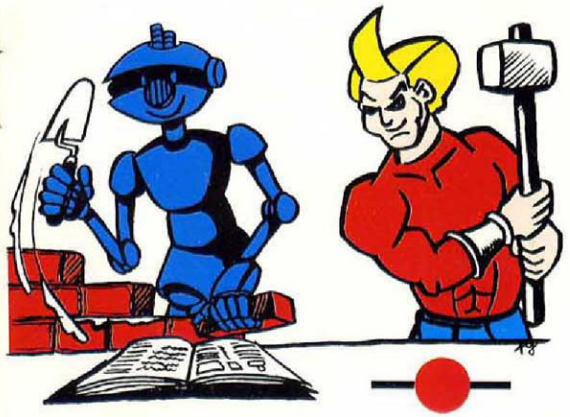
PAR FRANÇOIS LE GRIGUER

JOYSTICK-BREAKER

DES COMMENTAIRES DANS LE BASIC

AUJOURD'HUI TU SAISIS OU TU CORRIGES, S'IL EST DEJA ECRIT, LE PREMIER LISTING BASIC DE TON CASSE-BRIQUES. TOUTES CES INSTRUCTIONS DOIVENT ETRE BIEN COMPRISES EN LISANT LES EXPLICATIONS QUI LES ACCOMPAGNENT ET, EN REPRENANT L'EXPOSE DES MODULES DE LA SEMAINE DERNIERE. ENSUITE, COMME PREVU, TU ABORDES LA REDEFINITION DES TABLEAUX, UNE SEQUENCE PARTICULIEREMENT INTERESSANTE QUI TE PERMETTRA DE CREER TES 40 TABLEAUX.

J'APPRENDS



Dans la précédente initiation les explications des lignes figuraient dans un paragraphe séparé. Pour améliorer la présentation et la facilité d'utilisation, tous ces commentaires se trouvent directement intégrés au listing et, écrits en italique, pour ne pas les confondre avec les instructions du programme. Un commentaire en italique concerne toujours la ou les lignes précédentes. Alors ne les saisis surtout pas, tu aurais des problèmes. Souviens-toi également que tous les mots en majuscule dans le listing sont des mots-clé du BASIC et, que la formule la plus sure est de sauvegarder ton listing dès la fin de la saisie par :

```
SAVE"BREAKER
10 ''''''''''''''''''''''''''''
20 '    JOYSTICK-BREAKER
30 ''''''''''''''''''''''''''''
40 'un programme de .....
50 'decembre 1988
60 '
70 ''''''''''''''''''''''''''''
```

Voici le premier texte en italique. Il s'agit donc d'explications qui t'aident à comprendre le Basic mais, qui ne doivent, en aucun cas, faire partie de ton listing.
Toutes les lignes commençant par ' sont des commentaires écrits par le programmeur mais ignorés lors du traitement.

```
80 '
90 'codes collisions en tabl
100 'haut =1
110 'droite =2
120 'raquette=3
130 'bas(perte balle)=6
140 'gauche=4
150 'briques=7,8,9
160 '
```

Il s'agit du rappel du "Tableau des codes en tabl" défini la semaine dernière

```
170 '-------------------------
180 'PROGRAMME PRINCIPAL
190 '-------------------------
200 '152 octets reserves par tableau =50 briques maxi
210 'nombre maxi tableaux 40*152=6080 octets
220 ' structure tableau
230 'nb balles/nb briques/brique:x,y,type
240 'en ad-1 numero tableau maxi
250 '
```

Un rappel de la composition du fichier "Tableaux"

```
260 ad=33000:MEMORY ad-2:LOAD"TABLEAUX",ad-
1:maxt=PEEK(ad-1)
```

chargement du fichier "Tableaux". ad=adresse mémoire du premier tableau (tableau 1). ad-1=adresse contenant le nombre total de tableaux. MEMORY déclare l'adresse limite utilisable par le BASIC. LOAD suivi du nom du fichier et de l'adresse mémoire charge le fichier "Tableaux" à partir de 32999 (ad-1). maxt contient le nombre total de tableaux. Le fichier "Tableaux" n'existant pas actuellement tu vas le créer par: POKE 33000,0:SAVE"tableaux",b,33000,1
Un fichier généré en Binaire et qui contient 0 en nombre total de tableaux. Si tu travailles avec une cassette ce fichier doit toujours se trouver après le programme et, tu prévois 2 cassettes: l'une pour le programme que tu charges par LOAD et l'autre, pour le fichier, que tu places avant de faire RUN. "Tableaux" sera enregistré automatiquement par la séquence de redéfinition des tableaux.

```
270 'init couleurs et briques
280 GOSUB 3410
```

```
29Ø 'generique
3ØØ GOSUB 53Ø
31Ø 'init des fichiers
32Ø GOSUB 148Ø
33Ø RUN
```
Un listing bien structuré, c-a-d bien organisé, pour rester compréhensible et simple à modifier, comporte un programme principal qui gère des séquences écrites comme des sous-programmes. Appel par GOSUB suivi du numéro de ligne. Ces séquences peuvent faire appel à des sous-programmes, également écrits à part.
```
45Ø '**************************
46Ø '**************************
47Ø '  sequences
48Ø '**************************
49Ø '**************************
5ØØ '-------------------------
51Ø '     generique
52Ø '-------------------------
53Ø MODE Ø:PRINT CHR$(22)+CHR$(1);
```
Fonction PRINT qui donne un affichage transparent nécessaire à l'affichage des briques.
```
54Ø y=1Ø:FOR x=3 TO 17:LOCATE x,y:GOSUB 445Ø:NEXT x
55Ø y=12:FOR x=3 TO 17:LOCATE x,y:GOSUB 445Ø:NEXT x
56Ø y=14:FOR x=3 TO 17:LOCATE x,y:GOSUB 447Ø:NEXT x
57Ø y=16:FOR x=3 TO 17:LOCATE x,y:GOSUB 447Ø:NEXT x
```
Les lignes 54Ø à 57Ø assurent l'affichage des lignes de briques pour encadrer le titre. Les sous-programmes des briques 7,8 et 9 sont en lignes 445Ø,447Ø et 449Ø.
```
58Ø LOCATE 3,11:PEN 13:PRINT"J O Y S T I C K"
59Ø LOCATE 4,15:PRINT"B R E A K E R"
6ØØ '  trace du cadre
61Ø MOVE 64,5Ø:DRAWR 476,Ø,2
62Ø DRAWR Ø,8Ø,2:DRAWR -476,Ø,2:DRAWR Ø,-8Ø,2
```
Un tracé effectué en mode graphique qui est développé dans la rubrique "POUR QUELQUES DEFINITIONS DE PLUS..." "POUR QUELQUES DEFINITIONS DE PLUS..." afin de ne pas trop encombrer le listing. MOVE positionne le curseur graphique en x=64 et y=5Ø. DRAWR trace une droite horizontale à partir de la position du curseur et avec une longueur en x de 476 et l'INK 2. C'est un tracé en coordonnées relatives. Le DRAWR suivant trace une droite verticale avec une longueur en y de 8Ø, INK 2. Les 2 DRAWR suivants reprennent les valeurs précédentes en négatif pour refermer le dessin du rectangle.
```
63Ø LOCATE 5,18:PRINT"realisation"
64Ø LOCATE 1Ø,19:PRINT"de"
65Ø '  tu places ton nom entre les guillemets
66Ø LOCATE 5,21:PRINT"                "
67Ø FOR j=1 TO 9ØØ:NEXT:RETURN
145Ø '-------------------------
146Ø '  init des fichiers
147Ø '-------------------------
148Ø DIM tabl(2Ø,25)
```
Déclaration du fichier tabl prévu la semaine dernière pour gérer les 2Ø caractères sur 25 lignes de la surface de l'écran de jeu.
```
149Ø RETURN
15ØØ '-------------------------
329Ø '**************************
33ØØ '     sous-programmes
331Ø '**************************
```
Tu remarques que la numérotation des lignes ne se suit pas. C'est tout à fait volontaire, d'autres lignes viendront s'insérer avec les modules suivants.
```
338Ø '-------------------------
339Ø '  init des couleurs et briques
34ØØ '-------------------------
341Ø RESTORE 365Ø
```
Les numéros de couleurs sélectionnés dans notre Tableau des couleurs figurent en DATA à partir de la ligne 365Ø.
```
343Ø READ c:INK j,c
344Ø NEXT
```
Boucle d'initialisation des INK par lecture des DATA.
```
345Ø MODE Ø:CLS
346Ø '  brique 7
347Ø SYMBOL 255,255,129,129,129,129,129,255
348Ø SYMBOL 254,Ø,126,66,66,66,66,126
349Ø SYMBOL 253,Ø,Ø,6Ø,36,36,6Ø
35ØØ SYMBOL 252,Ø,Ø,Ø,24,24
351Ø '  brique 8
352Ø SYMBOL 251,255,128,128,128,128,128,128
353Ø SYMBOL 25Ø,Ø,1,1,1,1,1,127
354Ø SYMBOL 249,Ø,126,6Ø,24,24,6Ø,126
355Ø SYMBOL 248,Ø,Ø,66,1Ø2,1Ø2,66
356Ø '  brique 9
357Ø SYMBOL 247,255,126,6Ø,24
358Ø SYMBOL 246,Ø,Ø,Ø,Ø,24,6Ø,126,255
359Ø SYMBOL 245,Ø,128,192,224,224,192,128
36ØØ SYMBOL 244,Ø,1,3,7,7,3,1
361Ø RETURN
```
Les lignes 347Ø à 36ØØ reprennent la codification du Dessin des Briques,par caractère et par ligne, pour leur redéfinition.
```
362Ø '
363Ø 'data des couleurs
364Ø 'gris
365Ø DATA 13
366Ø 'rouges
367Ø DATA 6,7
368Ø 'bleus
369Ø DATA 2,14,2Ø
37ØØ 'verts
371Ø DATA 9,1Ø,18,21,22
372Ø 'jaunes
373Ø DATA 15,25
374Ø 'noir
375Ø DATA Ø
376Ø 'marron
377Ø DATA 16
378Ø 'blanc
379Ø DATA 26
441Ø '-------------------------
442Ø 'sous-prog affichage des briques
443Ø '-------------------------
```
Trois sous-programmes qui vont être utilisés par différentes séquences et, déjà par le générique.
```
444Ø 'afficher brique type 7
445Ø PEN 5:PRINT CHR$(255)
:LOCATE x,y:PEN 7:PRINT CHR$(254):PEN 12:LOCATE x,y
:PRINT CHR$(253):PEN 13:LOCATE x,y:PRINT CHR$(252):
:PRINT CHR$(253):PEN 13:LOCATE x,y:PRINT
CHR$(252):RETURN
446Ø 'afficher brique type 8
447Ø PEN 6:PRINT CHR$(251)
:LOCATE x,y:PEN 5:PRINT CHR$(25Ø):PEN 2:LOCATE x,y
:PRINT CHR$(249):PEN 14:LOCATE x,y:PRINT
CHR$(248):RETURN                NT CHR$(249):LOCATE
x,y:PEN 14:PRINT CHR$(248):R
448Ø 'afficher brique type 9
449Ø PEN 1Ø:PRINT CHR$(247)
:LOCATE x,y:PEN 12:PRINT CHR$(246):PEN 3:LOCATE x,y
:PRINT CHR$(245):LOCATE x,y:PEN 14:PRINT CHR$(244)
:PRINT CHR$(245):PEN 14:LOCATE x,y:PRINT
CHR$(244):RETURN
45ØØ '-------------------------
```
Avant d'entrer dans l'un des sous-programmes x et y doivent déclarés. Ensuite PEN donne le numéro d'INK à utiliser, PRINT indique le caractère et, LOCATE se répète pour rester sur les mêmes coordonnées.

UNE REDEFINITION A TOUT CASSER

Maintenez-vous que les 4Ø tableaux sont redéfinissables? Oui Votre Honneur et, nous entrons immédiatement dans tous les détails.

Ce module s'insère entre le générique et la page des scores et, intervient une seule fois lors du chargement du programme. Sur une première page d'écran tu prévois le choix: redéfinition ou suite en jeu.

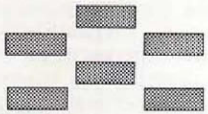
En composant J tu enregistres le fichier "Tableaux" pour le remettre à jour et, tu reviens dans le programme principal (attention, si tu travailles avec une cassette, celle-ci doit être ramenée en début de

REDEFINITIONS

T tableau
J jeu

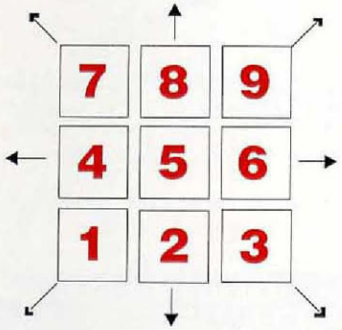
bande et en position RECORD). La composition de T t'envoie dans la redéfinition d'un tableau.
Le numéro de tableau doit être > Ø et < que le nombre de tableaux + 2, pour ne pas avoir de trous dans le fichier. Le nombre de balles, supérieur à zéro, ne dépasse pas une limite maxi que tu fixes, par exemple à 15. Les messages de bas d'écran concernent l'insertion

Tableau :
Nombre de balles :

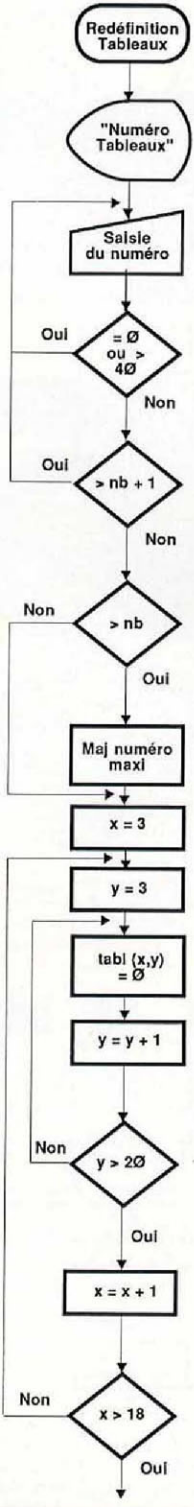


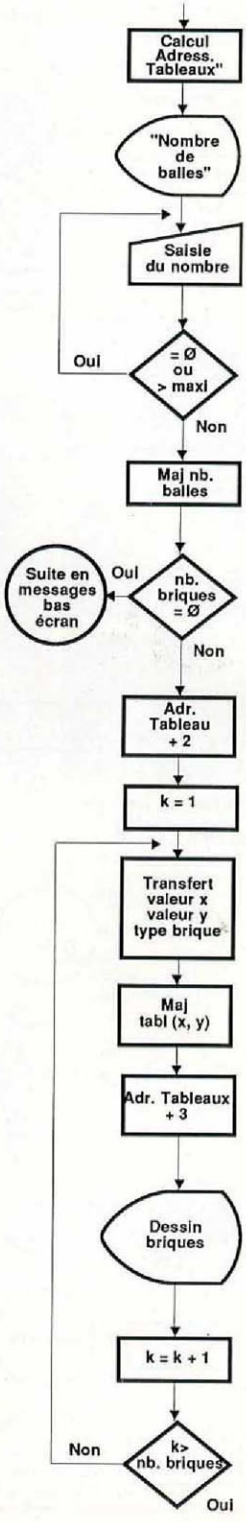
7 / 8 / 9 ou S
Fin par F

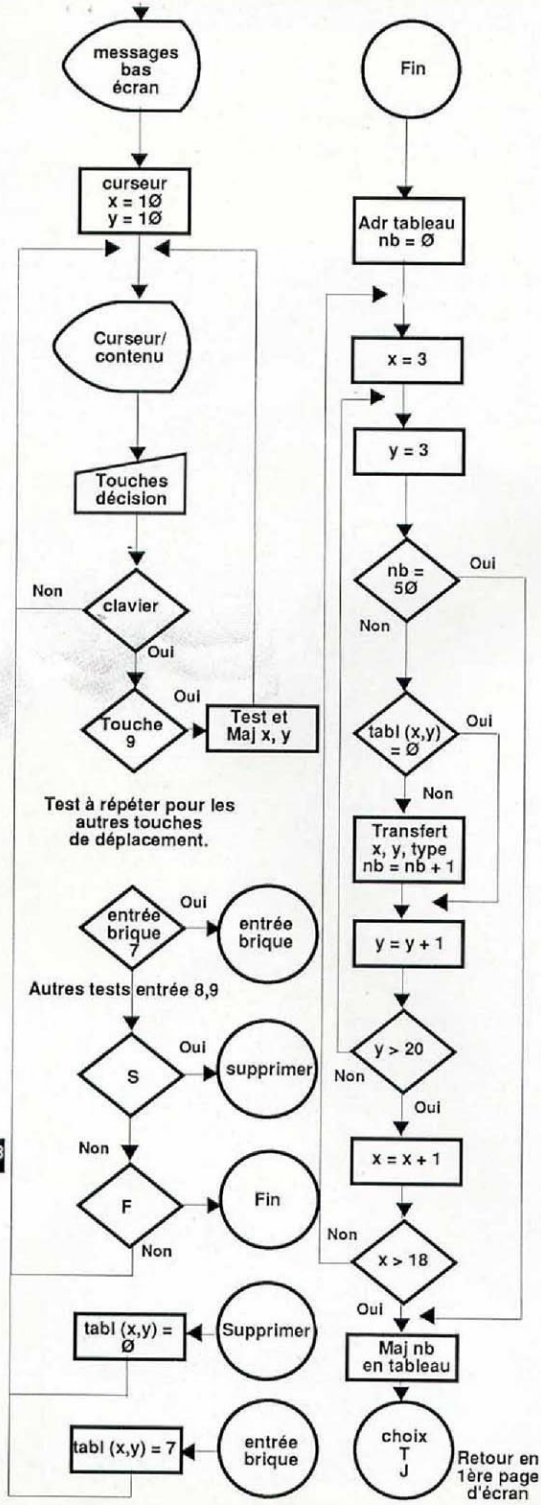
ou la suppression d'une brique, ou encore, la décision de terminer. La partie centrale représente le tableau tel qu'il apparaitra dans le jeu, avec toutes les briques dans la position que tu auras prévue. Il y a donc transfert du fichier "Tableaux" vers le fichier "tabl" qui gère la surface de jeu. Et avant tout tu calcules l'adresse du tableau avec une formule toute simple:
adt = ad + ((nt-1)*lg)
ad = adresse en "tableaux" du tableau numéro 1
nt = numéro du tableau composé
lg = longueur (nombre d'octets par tableau)
Le transfert par lui-même, une opération très simple, chaque brique entre en tabl avec son type 7, 8 ou 9 et, suivant ses coordonnées (valeurs de x et y). A partir de tabl il te reste à afficher les briques. Maintenant tu positionnes, au centre de l'écran, un curseur de la taille d'un caractère, qui clignote par affichage curseur et zone recouverte en alternance. Les coordonnées x et y de ce curseur vont évoluer en fonction des déplacements. La position étant connue, l'insertion ou la suppression devient élémentaire par une simple mise à jour de tabl.
Pour les déplacements tu peux utiliser les touches du pavé numérique.



Voilà un petit moment qu'il n'y avait pas eu d'ordinogramme, et celà nous manquait. D'accord!







POUR QUELQUES DEFINITIONS DE PLUS...

L'affichage à l'écran s'effectue, soit par le MODE caractères que tu connais (x=1 à 2Ø et y=1 à 25), soit par le mode graphique qui gère non plus des caractères mais des pixels (points). La redéfinition des caractères pour le dessin des briques, vue la semaine dernière, illustre très bien cette notion graphique, avec les 8 points par ligne et les 8 lignes. Dans ce mode l'écran est constitué de 64Ø pixels par ligne et de 2ØØ lignes, numérotées de Ø à 399 de 2 n 2 en 2 (25 lignes de caractères, chacun composé de 8 lignes = 2ØØ). Pour les 64Ø pixels il faut résonner en octets (voir le numéro 6 pour la définition de l'octet):

- une ligne d'écran = 8Ø octets (toujours identique quel que soit le MODE).
- le MODE 2 donne autant de caractères que d'octets, 8Ø caractères par ligne, donc chacun des bits représente un pixel (8 bits * 8Ø octets = 64Ø pixels).
- le MODE 1 donne deux fois moins de caractères que d'octets, 4Ø caractères par ligne, le nombre de pixels est donc de 32Ø. Chaque caractère à l'écran couvre 2 octets par ligne, toujours sur 8 lignes. Pour progresser de 1 en x il faut ajouter 2.
- le MODE Ø, réduit à 2Ø caractères par ligne, ne représente plus que 16Ø pixels par ligne. Un caractère couvre 4 octets par ligne (sur 8 lignes) et pour progresser de 1 en x tu ajoutes 4.

La capacité totale de l'écran se calcule tout simplement par 8Ø octets * 2ØØ lignes = 16ØØØ octets + 384 octets inutilisés = 16384 octets soit 16 Ko.

Encore un point qui concerne l'origine de x,y placée en bas et à gauche de l'écran. L'instruction ORIGIN permet de définir l'origine à une autre position. Par exemple: ORIGIN Ø,399 situe x et y à partir du coin haut et gauche.

Toujours pour t'aider, et cette fois dans les LIST que tu lances, passe en MODE 2 pour plus de clarté dans l'affichage des listings à l'écran.

Le Relatif et l'Absolu: deux notions à bien préciser avec une utilisation immédiate, DRAWR dans le tracé du cadre du générique. Une valeur absolue se suffit à elle même, elle est exploitée directement. La valeur relative se réfère à une valeur d'origine et s'y ajoute.

Je te l'avais bien dit que ça devait se corser, tout en devenant de plus en plus passionnant. Si l'informatique restait trop simple il n'y aurait plus de programmeurs. Dans tous les cas une excellente formule pour entretenir tes neurones.

Bien Joystiquement votre
François LE GRIGUER

LA SEMAINE PROCHAINE

Un super listing des redéfinitions à ne pas manquer et, la suite de JOYSTICK-BREAKER avec la page de scores.

PAR FRANÇOIS LE GRIGUER

JOYSTICK-BREAKER

SALUT A TOI L'AVENTURIER DE L'INFORMATIQUE, COMME CHAQUE SEMAINE ON SE RETROUVE, DES IDEES PLEIN LA TETE, LE SAC A DOS REMPLI DE BASIC ET DEUX OU TROIS ORDINOGRAMMES A LA MAIN, NOUS VOILA, UNE FOIS ENCORE, PRETS A AFFRONTER LES SEQUENCES ET LES SOUS-PROGRAMMES QUI NE DOIVENT PLUS RESSEMBLER A UN TERRIBLE LABYRINTHE MAIS, AU CONTRAIRE, A UN TERRAIN CONNU DANS LEQUEL TU EVOLUES TOUT A FAIT RELAX. LA REDEFINITION DES TABLEAUX VA TOURNER, LE LISTING BASIC EST LA, AVEC TOUS LES COMMENTAIRES SI TU DEBUTES ET, TU VAS AVOIR TOUS LES ELEMENTS POUR PREPARER UNE PAGE DE SCORES AINSI QUE LE DEBUT DE PARTIE.

TON PROGRAMME DE REDEFINITIONS

Un nouveau listing, qui complète ton programme «BREAKER» saisi la semaine dernière que tu commences par charger en mémoire par LOAD. Fais attention aux numéros de lignes qui ne se suivent pas toujours et, ceci pour permettre la saisie de séquences ultérieures. Ne saisis pas les textes en italique qui ne sont que des explications supplémentaires. Certaines lignes BASIC, coupées pour des raisons de largeur de texte, doivent être saisies à suivre avant de faire ENTER de fin de ligne. Une nouvelle ligne commence obligatoirement par un numéro. Utilise le plus possible la fonction AUTO n,1Ø qui numérote les lignes de 1Ø en 1Ø à partir de n (premier numéro de ligne à entrer).

```
33Ø 'redefinitions
34Ø GOSUB 71Ø
35Ø RUN
```

Les lignes 33Ø à 35Ø concernent le programme principal avec RUN qui relance le traitement en attendant les séquences suivantes du jeu.

```
68Ø '-----------------------
69Ø'   redefinition de tableaux
7ØØ'-----------------------
```

Il s'agit de la nouvelle séquence développée la semaine dernière. Tu dois rapprocher l'ordinogramme des instructions du listing. L'ordinogramme te donne le plan, l'ordre de déroulement du traitement, et le listing BASIC est un langage d'écriture, avec ses règles et sa syntaxe, compréhensible par ton ordinateur.

```
71Ø MODE 1:x=9:PEN 3:LOCATE x,5
:PRINT»REDEFINITIONS TABLEAUX»
72Ø LOCATE x+5,8:PRINT»tableau  T»
73Ø LOCATE x+5,1Ø:PRINT»jeu     J»
74Ø a$=INKEY$:IF a$=»» THEN 74Ø
75Ø IF a$=»j» OR a$=»J» THEN
SAVE»tableaux»,b,ad-1,61ØØ:MODE Ø:RETURN
76Ø IF a$=»t» OR a$=»T» THEN 79Ø ELSE 72Ø
```

Les lignes 71Ø à 76Ø déterminent le choix de la première page d'écran, la touche T vers la redéfinition d'un tableau et J enregistre le fichier «Tableaux» pour terminer la séquence.

```
77Ø 'entree/modification numero tableau
78Ø '
79Ø MODE Ø:LOCATE 1,1:INPUT»tableau»;n
8ØØ IF n=Ø OR n>4Ø OR  n>maxt+1 THEN 71Ø
81Ø IF n>maxt THEN POKE ad-1,n:maxt=n
```

La seconde page commence par la composition du numéro de tableau à redéfinir avec, le test = Ø et > numéro maxi + 1, et la mise à jour si > numéro maxi.

```
82Ø ' vider tabl
83Ø FOR x=3 TO 18:FOR y=3 TO 2Ø:tabl(x,y)=Ø :NEXT y:NEXT x
```

L'ensemble de la surface d'affichage des briques en tabl est remis à zéro, colonnes 3 à 18 pour x et, lignes 3 à 2Ø pour y.

```
84Ø '
85Ø 'calcul adresse tableau (n-1)*152 octets
86Ø ad1=ad+(n-1)*152
87Ø INPUT»balles»;b:IF b=Ø OR b>2Ø THEN LOCATE 1,2
:GOTO 87Ø
88Ø POKE ad1,b
```

Un tableau est prévu pour 5Ø briques et chaque brique occupe 3 octets, donc 15Ø octets plus 2 octets pour le nombre de balles et le nombre de briques. ad1 contient l'adresse de début du tableau.

```
89Ø 'chargement du tableau et affichage
9ØØ PRINT CHR$(22)+CHR$(1)
```

Passage en mode d'affichage transparent. Les caractères peuvent se superposer pour les briques.

```
91Ø j=PEEK(ad1+1):IF j=Ø THEN98Ø
92Ø ad2=ad1+2
93ØFORk=1 TOj:x=PEEK(ad2):y=PEEK(ad2+1):b=PEEK(ad2+2)
94Ø tabl(x,y)=b:ad2=ad2+3
95Ø LOCATE x,y:ON b-6 GOSUB 445Ø,447Ø,449Ø
96Ø NEXT k
```

j = nombre de briques. ad2 = adresse de la 1 ère brique valeur de x. ad2+1 = valeur de y. ad2+2 = type de la brique 7,8 ou 9 qui est transféré en tabl(x,y). L'adresse ad2 progresse de 3 octets à chaque passage. L'affichage de la brique s'effectue par les sous-programmes déjà créés avec -6 sur le type de brique pour ramener les valeurs 7,8,9 à 1,2,3. ON b-6 GOSUB exécute le sous-programme de la brique 7 en ligne 445Ø ou de la 8 en 447Ø ou, encore de la 9 en 449Ø, suivant la valeur de b.

```
97Ø '
98Ø nb=j
99Ø '  messages bas ecran
1ØØØ LOCATE 1,25:PRINT»fin     F»
1Ø1Ø LOCATE 1,23:PRINT»7/8/9 ou Supprimer»;
1Ø2Ø '
1Ø3Ø 'positionner curseur centre tableau
1Ø4Ø 'test touche deplacement/fonction
1Ø5Ø x=1Ø:y=1Ø
```

Coordonnées du curseur en position de départ au centre de l'écran.

```
1Ø6Ø GOSUB 134Ø
```
Vers sous-programme affichage du curseur
```
1Ø7Ø IF INKEY$=»» THEN
1Ø6Ø 1Ø8Ø IF INKEY(3)>-1 THEN 125Ø
1Ø9Ø IF INKEY(5)>-1 THEN 126Ø
11ØØ IF INKEY(13)>-1 THEN 127Ø
111Ø IF INKEY(1Ø)>-1 THEN 128Ø
112Ø IF INKEY(11)>-1 THEN 129Ø
113Ø IF INKEY(4)>-1 THEN 13ØØ
114Ø IF INKEY(14)>-1 THEN 131Ø
115Ø IF INKEY(2Ø)>-1 THEN 133Ø
```
Premier groupe de tests pour le déplacement du curseur par les touches du pavé numérique. La fonction INKEY, accompagnée du numéro de la touche (chaque touche est associée à un numéro suivant un tableau de ton manuel Amstrad), renvoie une valeur qui indique son état (-1 = non utilisée et toutes les autres valeurs = oui). Si oui suite en test des coordonnées du curseur pour décider si le déplacement est possible.
```
116Ø ' entree brique ou suppression
117Ø IF INKEY(41)>-1 THEN tabl(x,y)=7:GOTO 1Ø6Ø
118Ø IF INKEY(4Ø)>-1 THEN tabl(x,y)=8:GOTO 1Ø6Ø
119Ø IF INKEY(33)>-1 THEN tabl(x,y)=9:GOTO 1Ø6Ø
12ØØ IF INKEY(6Ø)>-1 THEN tabl(x,y)=Ø:GOTO 1Ø6Ø
```
Les touches 7,8 et 9 du clavier alpha (numéros 41,4Ø,33) entrent une nouvelle brique à la position du curseur. Transfert de 7,8 ou 9 en tabl(x,y). La touche S (numéro 6Ø) supprime la brique tabl(x,y) = Ø.
```
121Ø ' fin de saisie touche f
122Ø IF INKEY(53)>-1 THEN 139Ø
```
La touche F a le numéro 53 et envoie en fin du tableau.
```
123Ø GOTO 1Ø6Ø
124Ø ' deplacement curseur
125Ø IF x=18 OR y=3 THEN 1Ø6Ø ELSE x=x+1 :y=y-1:GOTO 117Ø
126Ø IF x=18 OR y=2Ø THEN 1Ø6Ø ELSE x=x+1 :y=y+1
:GOTO 117Ø
127Ø IF x=3 OR y=2Ø THEN 1Ø6Ø ELSE x=x-1 :y=y+1:
GOTO 117Ø
128Ø IF x=3 OR y=3 THEN 1Ø6Ø ELSE x=x-1 :y=y-1:GOTO 117Ø
129Ø IF y=3 THEN 1Ø6Ø ELSE y=y-1:GOTO 117Ø
13ØØ IF x=18 THEN 1Ø6Ø ELSE x=x+1:GOTO 117Ø
131Ø IF y=2Ø THEN 1Ø6Ø ELSE y=y+1:GOTO 117Ø
132Ø '
133Ø IF x=3 THEN 1Ø6Ø ELSE x=x-1:GOTO 117Ø
```
Les lignes 125Ø à 133Ø controlent les limites de déplacement du curseur et exécutent la mise à jour de x et y suivant la direction.
```
134Ø LOCATE x,y:PEN 3:PRINT CHR$(143):LOCATE x,y
135Ø IF tabl(x,y)=Ø THEN PEN Ø:PRINT CHR$(143) :RETURN
136Ø ON tabl(x,y)-6 GOSUB 445Ø,447Ø,449Ø
137Ø RETURN
```
Sous-programme d'affichage du curseur ou de la brique pour assurer le clignotement.
```
138Ø ' mise a jour donnees en fichier tableaux
```
Entrée en ligne 139Ø par la touche F.
```
139Ø ad2=ad1+2:nb=Ø
```
ad2 = adresse début de la première brique en tableau. nb va permettre de controler le nombre total de briques maxi 5Ø.
```
14ØØ FOR x=3 TO 18:FOR y=3 TO 2Ø
141Ø IF nb=5Ø THEN LOCATE 2,21:PEN 3:PRINT»plus de 5Ø
briques»;:FOR j=1 TO 6ØØ :NEXT :GOTO 144Ø
142Ø a=tabl(x,y):IF a<>Ø THEN POKE ad2,x :POKE ad2+1,y:POKE
ad2+2,a:ad2=ad2+3:nb=nb+1 nb+1
```
Mise à jour de Tableaux à partir de tabl(x,y).
```
143Ø NEXT y:NEXT x
144Ø POKE ad1+1,nb:GOTO 71Ø
145Ø '
```
Transfert en ad1+1 du nombre de briques du tableau.

Maintenant que tu as saisis ton listing tu fais, sans attendre, SAVE»BREAKER» pour ne pas risquer de le perdre. Ensuite, par RUN tu controles si ton programme se déroule correctement, sans message d'erreur, en testant l'ensemble de la redéfinition de plusieurs tableaux. Tu définis un tableau, tu sors par F et, tu le rappelle sous le même numéro pour retrouver ta définition. Les erreurs sont toujours dues à des fautes de frappe et, en particulier:

- manque du séparateur (:) entre 2 instructions à l'intérieur d'une ligne.
- composition d'un o au lieu d'un zéro.
- numéro de ligne erroné après un GOTO ou un GOSUB.
- mot-clé mal orthographié.
- espace manquant.

QUI EST LE MEILLEUR???

Un jeu sans page de scores, c'est dommage. Tu es toujours super content de rentrer ton prénom au-dessus des copains, de voir qui est le meilleur, pas vrai!! Huit lignes c'est une bonne moyenne avec une zone de 1Ø caractères qui permet d'entrer la plupart des prénoms. A l'origine le programme contient 8 noms assortis de scores plus ou moins élevés, décrits en ligne de DATA, par ordre décroissant. Tu déclares 2 fichiers:

- nom$ contenant les 8 noms (le symbole $ indique un fichier alphanumérique).
- score avec les 8 valeurs de scores du plus élevé au plus petit.

Une première étape, en début de traitement, initialise nom$ et score par le transfert des DATA. Les deux fichiers en place, à chaque fin de partie tu vas comparer le score réalisé par le joueur avec chacune des lignes en partant du haut. Si le score en cours est plus grand que celui du fichier, le joueur va entrer son nom sur la ligne après avoir opéré le décalage des 2 fichiers d'une ligne vers le bas et, la suppression de la dernière. Ce décalage n'aura pas lieu si le joueur entre à la 8 ème ligne. Cette opération réalisée, il reste à prévoir et à controler la saisie du nouveau nom avec le décompte des caractères (maxi 1Ø). Tout celà, tu en trouves le résumé dans l'ordinogramme.

L'initialisation des fichiers nom$ et score nécessite une ou deux boucles de programme et, celà dépend de l'écriture des lignes de DATA. Une première ligne avec les 8 prénoms, suivie d'une seconde avec les 8 scores t'oblige à écrire 2 boucles. C'est sous cette forme que je l'ai prévu pour te permettre de bien séparer les opérations. Mais comme en programmation le plus court représente toujours la meilleure formule tu pourrais écrire:

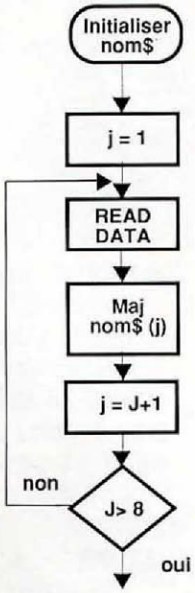
```
DATA
SULYANE,55ØØØ,MICHEL,43ØØØ
etc.....
```

Une seule boucle suffit alors avec:

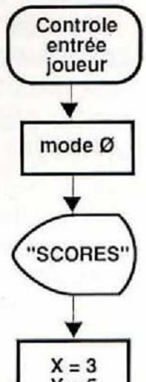
```
READ nom$(j),score(j)
```

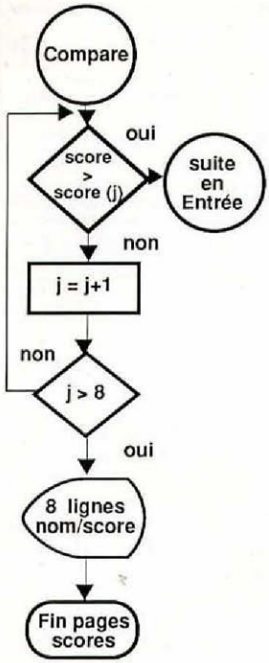
L'instruction READ permet la lecture de plusieurs DATA à suivre.

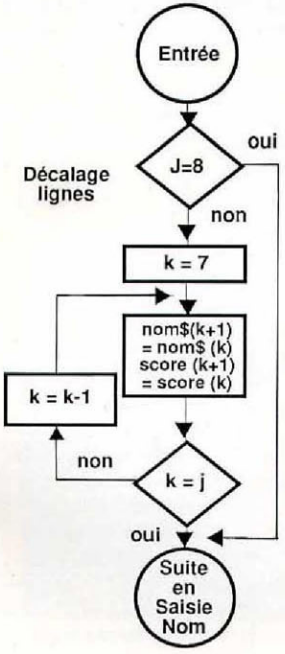
Pour l'affichage de chaque ligne de score tu dois tenir compte de la valeur pour conserver un même alignement. Tu testes < que 1ØØØ (= 3 caractères), < que 1ØØØØ (= 4 caract.) et < que 1ØØØØØ (= 5 caract.), sinon 6 caractères. La valeur de x en LOCATE sera modifiée suivant le résultat du test.

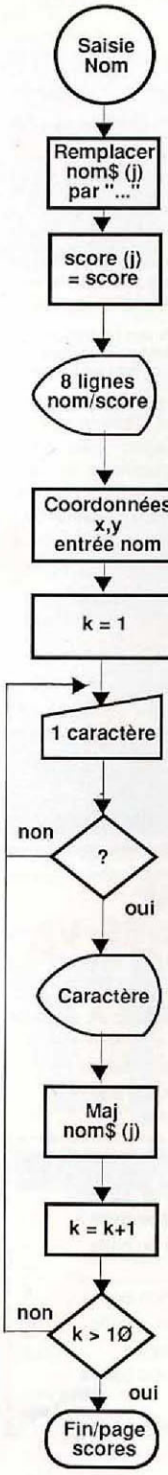


Pour bien décomposer les opérations une 2e boucle est à prévoir pour initialiser score.









POUR QUELQUES DEFINITIONS DE PLUS...

Les ordinogrammes c'est bien, celà fait très technique, mais tu as du mal à les interpréter. Tout d'abord ils ne sont vraiment utiles que pour traiter des séquences particulièrement compliquées, mais pour débuter, il s'agit d'une aide, d'un shéma qui te donne exactement l'ordre des opérations. Les symboles utilisés correspondent à une définition internationale et, nous allons revoir un par un les six dessins utilisés ici (il y en a beaucoup d'autres).

Le symbole d'un début ou d'une fin de séquence qui contient en texte le nom de la séquence.

Le cercle indique une sortie vers un point du programme. Un autre cercle correspondra au point d'entrée. Dans le premier tu indiques «suite en entrée» et, dans le second «entrée», comme dans l'ordinogramme de la page des scores.

Le rectangle représente une opération, un traitement, avec les noms des constantes, variables ou fichiers concernés.

Le losange correspond à un test, c-à-d, une question concernant le contenu d'une donnée par rapport à une valeur immédiate ou, au contenu d'une autre donnée. C'est le type d'instruction le plus fréquent dans un programme. Les tests possibles sont nombreux: égal à, plus petit que, plus grand que, négatif, positif, différent de, compris entre, contenant, commençant par, finissant par, multiple de etc.... et, tout ceci peut, bien entendu être lié à une suite de test avec AND, OR, NOT. Dans tous les cas il n'y a que deux réponses possibles: OUI ou NON qui conditionnent la suite du traitement. Le losange se prête très bien à cette représentation avec, sur l'une des pointes la sortie OUI, et sur une autre la sortie NON.

Ceci matérialise l'affichage à l'écran des données indiquées.

Cette forme indique une saisie de données au clavier

L'ordinogramme t'évite de rédiger des pages de texte qui ne seront jamais très claires. Certains symboles vont avoir une traduction quasi-immédiate en BASIC, par exemple le losange, qui génère automatiquement, dans ton programme, IF ..... THEN ..... ELSE (en français SI .... ALORS ..... SINON), ou encore les débuts et fin de boucles: FOR ... TO et NEXT. Enfin la lecture d'un ordinogramme reste, rapide et claire, par les droites de liaison et les flèches de direction.

UN DEBUT DE PARTIE MUSCLE

Cette nouvelle séquence contient beaucoup de détails puisqu'il faut définir ici tous les éléments nécessaires au bon fonctionnement du jeu. Jusqu'à présent nous avons préparé les modules annexes qui interviennent avant le début d'une partie. Voici la petite check-list que l'on va reprendre point par point:

- tracer le cadre
- afficher le score, le numéro de tableau et le nombre de balles
- positionner la raquette
- initialiser le fichier tabl
- initialiser toutes les constantes et variables
- placer la balle sur la raquette

Au-delà il s'agira du jeu avec une balle et une raquette en mouvement. Donc ton casse-briques se précise avec cette nouvelle page d'écran, la dernière, mais aussi la plus complexe.

Le mode adopté (MODE Ø) t'apporte 16 couleurs et, tu vas en utiliser 8 pour le cadre de jeu. Ce cadre concerne les deux cotés et le haut de l'écran. Le bas reste vide, réservé au déplacement de la raquette. En utilisant les instructions graphiques MOVE et DRAWR, déjà vues dans le générique, tu traces 8 lignes ayant chacune une couleur différente. Rien de particulièrement compliqué, tu prévois une longueur de tracé qui doit diminuer, en hauteur comme en largeur (y et x), après chaque affichage, et toujours en utilisant une boucle de programme avec 8 passages. Un tout petit peu de réflexion et tu es certainement capable d'écrire ces quelques lignes de BASIC. Comme toujours, si tu n'y arrives pas, la semaine prochaine tu auras le listing avec un maximum d'explications.

Les affichages se superposent au cadre en partie haute. Regarde les dessins publiés dans le numéro 6 pour la disposition du score, du numéro de tableau et du nombre de balles en cours.

Le positionnement de la raquette fait appel, pour la définition de x, à

un sous-programme de tirage d'un nombre aléatoire, c-a-d tiré au hasard, à partir du temps. Le temps correspond au nombre de 3ØØ/1ØØ de secondes passés depuis la mise sous tension de ton **AMSTRAD**. La balle vient se placer, en ligne 24 (yb=24), sur le premier carcatère de la raquette ou, au contraire sur le 3ème, suivant xr. La direction db sera 4 ou 1.
Avant de voir l'initialisation de tabl je te propose de bien analyser le tableau récapitulatif de toutes les constantes et variables nécessaires au jeu. Ce tableau donne, pour chaque étiquette, son nom et son initialisation, la séquence qui effectue sa mise à jour et, son utilisation. Les noms utilisés sont définis par le programmeur, ils doivent être assez courts pour ne pas trop ralentir le traitement. Si tu prends sco pour gérer le score, tu aurais pu choisir sc ou encore scor, c'est vraiment comme tu veux. La seule contrainte qui existe c'est d'utiliser toujours le label déclaré au début dans la suite du programme, évident mais oublié parfois. Ce tableau des constantes et des variables devient donc ta bible de référence pour toute la suite du programme.

L'initialisation de tabl se déroule de la façon suivante:

- vider tabl c-à-d mettre Ø sur les 2Ø positions des 25 lignes

CONTENU DU FICHIER

```
1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
4 Ø Ø Ø Ø Ø Ø Ø Ø Ø Ø Ø Ø Ø Ø Ø Ø Ø Ø 2
4 Ø Ø Ø Ø Ø Ø Ø Ø 7 Ø Ø Ø Ø Ø Ø Ø Ø Ø 2
4 Ø Ø Ø Ø Ø Ø Ø 7 Ø 7 Ø Ø Ø Ø Ø Ø Ø Ø 2
4 Ø Ø Ø Ø Ø Ø 7 Ø Ø Ø 7 Ø Ø Ø Ø Ø Ø Ø 2
4 Ø Ø Ø Ø 8 8 Ø Ø Ø Ø Ø 8 8 Ø Ø Ø Ø Ø 2
4 Ø Ø Ø 8 8 Ø Ø Ø Ø Ø Ø Ø 8 8 Ø Ø Ø Ø 2
4 Ø Ø Ø 8 8 Ø Ø Ø Ø Ø Ø Ø 8 8 Ø Ø Ø Ø 2
4 Ø Ø Ø 9 9 9 9 9 9 9 9 9 9 9 Ø Ø Ø Ø 2
4 Ø Ø Ø Ø Ø Ø Ø Ø Ø Ø Ø Ø Ø Ø Ø Ø Ø Ø 2
4 Ø Ø Ø Ø Ø Ø Ø Ø Ø Ø Ø Ø Ø Ø Ø Ø Ø Ø 2
4 Ø Ø Ø Ø Ø Ø Ø Ø Ø Ø Ø Ø Ø Ø Ø Ø Ø Ø 2
4 Ø Ø Ø Ø Ø Ø Ø Ø Ø Ø Ø Ø Ø Ø Ø Ø Ø Ø 2
4 Ø Ø Ø Ø Ø Ø Ø Ø Ø Ø Ø Ø Ø Ø Ø Ø Ø Ø 2
4 Ø Ø Ø Ø Ø Ø Ø Ø Ø Ø Ø Ø Ø Ø Ø Ø Ø Ø 2
4 Ø Ø Ø Ø Ø Ø Ø Ø Ø Ø Ø Ø Ø Ø Ø Ø Ø Ø 2
4 Ø Ø Ø Ø Ø Ø Ø Ø Ø Ø Ø Ø Ø Ø Ø Ø Ø Ø 2
4 Ø Ø Ø Ø Ø Ø Ø Ø Ø Ø Ø Ø Ø Ø Ø Ø Ø Ø 2
4 Ø Ø Ø Ø Ø Ø Ø Ø Ø Ø Ø Ø Ø Ø Ø Ø Ø Ø 2
4 Ø Ø Ø Ø Ø Ø Ø Ø Ø Ø Ø Ø Ø Ø Ø Ø Ø Ø 2
4 Ø Ø Ø Ø Ø Ø Ø Ø Ø Ø Ø Ø Ø Ø Ø Ø Ø Ø 2
4 Ø Ø Ø Ø Ø Ø Ø Ø Ø Ø Ø Ø Ø Ø Ø Ø Ø Ø 2
4 Ø Ø Ø Ø Ø Ø Ø Ø Ø Ø Ø Ø Ø Ø Ø Ø Ø Ø 2
4 Ø Ø Ø Ø Ø Ø Ø Ø Ø Ø Ø Ø Ø Ø Ø Ø Ø Ø 2
6 6 6 6 6 3 3 3 6 6 6 6 6 6 6 6 6 6 6 6
```

- mettre à jour le numéro de tableau en cours et calculer son adresse

- charger les codes de bordure haut, gauche, droite et bas (1,4,2 et 6)

- transférer les briques du tableau en tabl suivant x, y et type (7,8,9)

- afficher les briques à l'écran

Pour matérialiser le fichier tabl je t'ai préparé un petit tableau de 2Ø colonnes sur 25 lignes dans lequel tous les codes sont en place. N'oublie pas que ce fichier représente, de façon interne, la disposition exacte de la surface de jeu, ce qui permet à l'ordinateur d'avoir, lui aussi, une vue du jeu par l'intermédiaire du test de contenu autour de la balle. Le fichier tabl doit représenter sous la forme de codes ce que tu vois à l'écran, les bordures, les briques et la raquette. La balle n'entre pas dans le fichier et, ce sont ses coordonnées x et y qui vont la situer. A titre d'exemple, j'ai ajouté plusieurs briques et la raquette.

Toutes ces opérations, prises individuellement, ne posent pas de problèmes particuliers, pense à être toujours très ordonné, c'est la clé du succés en informatique.

Un programme bien chargé, **JOYSTICK-BREAKER** prend du volume, mais pas la tête! fais quand même attention!! Maintenant si tu es complètement égaré et, que tu commences à confondre le tube de mayonnaise avec le dentifrice, calme-toi et installe-toi à la porte de ton marchand de journeaux pour plonger sur le **JOYSTICK-Hebdo** numéro 9, dès son arrivée.

Bien Joystiquement votre.
François LE GRIGUER

LA SEMAINE PROCHAINE

Tous les listings avec un max de commentaires.
Le controle de la balle.

TABLEAU DES CONSTANTES ET DES VARIABLES

NOM	INITIALISATION	MISE À JOUR	OBSERVATIONS
sco	début partie	gestion balle	valeur du score
xr	position raquette	gestion raquette	valeur x raquette
yr	« «	toujours 25	valeur y raquette
raq$	dessin raquette		3 car (2Ø8)
raqz$	effacem "		2 car(143)
nt	initialisation tabl	changt. tableau	numéro tableau en cours
xb	suivant x raquette	gestion balle	valeur x balle
yb	en ligne 24	gestion balle	valeur y balle
bal$	début de partie		dessin balle caract.(231)
db	1 ou 4 suiv xb	gestion balle	direction balle (1,2,3,4)
nb	suivant contenu tableau	gestion balle	nombre de balles
nbo	= nb	nb balles	départ pour bonus
nbr	suivant contenu tableau	gestion balles	nombre de briques
nbro	= nbr		
ad1	initilisation tabl	adresse du tableau	
maxt	«		numéro tableau maxi

HAPPY NEW YEAR!!! QUE L'ANNEE 89 T'APPORTE UN MAXIMUM DE REUSSITE, DANS TOUS LES DOMAINES, MAIS SURTOUT, QUE TU DEVIENNES UN AS DE LA PROGRAMMATION!!! EN BASIC DEJA, MAIS POURQUOI PAS AUSSI EN ASSEMBLEUR. NOUS EMBARQUONS POUR UNE NOUVELLE ANNEE, A BORD D'UN FABULEUX VAISSEAU, REMPLI D'ORDINATEURS, TOUS LES ALLUMES DE LA MICRO SONT A BORD, LEUR JOYSTICK BIEN EN MAIN, PRETS A DETRUIRE TOUS LES OBSTACLES, POUR DECOUVRIR LES GALAXIES LOINTAINES DE L'INFORMATIQUE ET, CHAQUE PLANETE VA NOUS APPORTER SON LOT DE CONNAISSANCES NOUVELLES. FOUR..... THREE..... TWO..... ONE..... ZERO..... FIRE !!!!!

J'APPRENDS

JOYSTICK-BREAKER

TOUJOURS PLUS DE BASIC

Après une mise à feu impeccable, nous sommes déjà sur la bonne trajectoire avec les deux séquences, vues la semaine dernière, qui se traduisent aujourd'hui par un nouveau listing: la page des scores suivie de l'initialisation du début de partie. Je t'ajoute toujours beaucoup d'explications en italique pour t'aider si tu as peu d'expérience. La saisie d'un listing demande de l'attention, une virgule oubliée, une parenthèse en moins ou en trop et, c'est le message d'erreur garanti, pas la moindre fantaisie possible.

```
35Ø ' page score
36Ø GOSUB 153Ø
37Ø 'init debut de partie
38Ø GOSUB 198Ø 39Ø END
```

Instructions de complément pour le programme principal. ATTENTION à partir de maintenant ton programme ne peut plus tourner seul, certaines séquences comme l'initialisation de la balle fait appel à la gestion de la balle. Donc ne lance pas de RUN et, attends les prochains modules

```
15ØØ '————————————————
151Ø '   page de score
152Ø '————————————————
153Ø DIM nom$(8):DIM score(8)
```

Déclaration des fichiers avec leur label et leur DIMension.

```
154Ø RESTORE 155Ø
```

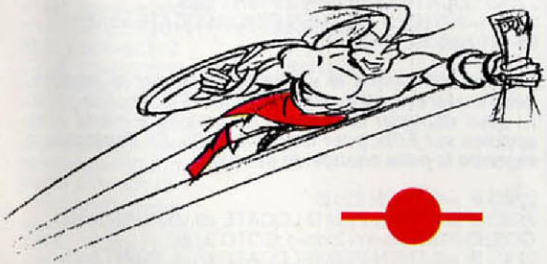
Indique le premier numéro de ligne DATA à utiliser pour le prochain READ.

```
155Ø DATA BREAKER,SULYANE,MICHEL,SERVANE,
ERIC,GWENOLA,JEANINE,CLAUDE
156Ø DATA 55Ø7ØØ,13Ø1ØØ,1152ØØ,11Ø1ØØ,
7Ø3ØØ,69ØØØ,51ØØØ,45ØØØ
```

Ces 2 lignes contiennent les 8 noms et les 8 scores qui vont s'afficher à la première page de score. Tu peux les modifier en gardant, dans tous les cas, 8 données. Un petit rappel: pour des raisons de format de nos colonnes certaines lignes BASIC sont tronquées, mais tu dois les saisir à suivre sans passer à la ligne. Tu fais ENTER juste avant le numéro de ligne suivant

```
157Ø FOR j=1 TO 8:READ nom$(j):NEXT
158Ø FOR J=1 TO 8:READ score(j):NEXT
```

Les 2 boucles entrent en fichier les DATA. A chaque READ le pointeur DATA avance à la donnée suivante suivant le séparateur virgule (,). Le 1er READ transfère en nom$(1), BREAKER, le 2ème transfère en nom$(2), SULYANE et ainsi de suite.



```
159Ø '   debut affichage page
16ØØ CLS:PEN 1:LOCATE 5,2:PRINT «S C O R E «
161Ø x=3:y=5
162Ø FOR j=1 TO 8:IF sco>score(j) THEN 167Ø
```

Comparaison du score réalisé par le joueur (sco) avec le contenu du fichier score. Si sco est plus grand le joueur va entrer sur la ligne dont le numéro correspond à la valeur de j

```
163Ø NEXT:GOSUB 183Ø:GOTO 181Ø
```

Le joueur n'entre pas dans les scores. Affichage des scores et fin de séquence.

```
164Ø '
165Ø '   decaler les lignes de score
166Ø '
167Ø IF j=8 THEN 171Ø
```

Controle du numéro de ligne d'entrée. Si = 8 alors pas de décalage des lignes.

```
168Ø FOR k=7 TO j STEP -1
169Ø nom$(k+1)=nom$(k):score(k+1)=score(k)
17ØØ NEXT k
```

Boucle de décalage des lignes, de la 7 vers la 8, de la 6 vers la 7 etc... et celà suivant la valeur de j. Si j=7 une seule opération

```
171Ø nom$(j)=».....»
```

Le nom en fichier est remplacé par une suite de points.

```
172Ø IF INKEY$<>»» THEN 172Ø
173Ø score(j)=sco
174Ø GOSUB 183Ø
```

Entrée score joueur (sco) en fichier et affichage page scores.

```
175Ø LOCATE x;5+(j-1)*2:nom$(j)=»»
```

Se positionner sur le 1er caractère du nom à composer et remplacer, en nom$ les points par vide («»)

```
176Ø FOR k=1 TO 1Ø
177Ø n$=INKEY$:IF n$=»» THEN 177Ø
178Ø PRINT n$;
179Ø nom$(j)=nom$(j)+n$
18ØØ NEXT k
```

Boucle de programme qui controle la saisie du nom du joueur avec 1Ø caractères. Caractère par caractère mise à jour de nom$. N'oublie pas que j contient toujours le numéro de la ligne d'entrée.

```
181Ø FOR j=1 TO 5ØØ:NEXT
182Ø IF INKEY$=»» THEN 182Ø ELSE RETURN
```

Boucle de tempo (temporisation) qui tourne sur elle-même 5ØØ fois et attente d'une touche pour sortir de la séquence par RETURN.

```
183Ø FOR m=1 TO 8
184Ø LOCATE x,y
185Ø PRINT nom$(m)
186Ø LOCATE 13,y
187Ø IF score(m)>99999 THEN LOCATE x+9,y
188Ø IF score(m)<1ØØØØ THEN LOCATE x+11,y
189Ø IF score(m)<1ØØØ THEN LOCATE x+12,y
19ØØ PRINT score(m)
191Ø y=y+2
192Ø NEXT m
193Ø RETURN
```

Les lignes 183Ø à 193Ø constituent un petit sous-programme, pour l'affichage des 8 lignes de la page de scores, avec controle de la valeur du score, c-a-d, du nombre de caractères et modification éventuelle de la valeur x (alignement de l'affichage).

```
194Ø '————————————————
195Ø '   debut de partie
196Ø '————————————————
```

```
197Ø 'trace de la page jeu
198Ø GOSUB 381Ø
```

Tracé de l'ensemble de la page de jeu à partir de la ligne 381Ø.

```
199Ø 'score
2ØØØ sco=Ø:GOSUB 4Ø1Ø
```

Initialiser le score du joueur et afficher (sous-programme en 4Ø1Ø).

```
2Ø1Ø '
2Ø2Ø 'init position raquette
2Ø3Ø mini=2:maxi=15:GOSUB 335Ø
```

Tirage d'un nombre aléatoire en 335Ø.

```
2Ø4Ø xr=nombre:yr=25
```

Initialisation des coordonnées x et y de la raquette

```
2Ø5Ø 'init raquette
2Ø6Ø raq$=CHR$(2Ø8)+CHR$(2Ø8)+CHR$(2Ø8)
```

Le dessin de la raquette est constitué de 3 caractères numéro 2Ø8.

```
2Ø7Ø tabl(xr,yr)=3
2Ø8Ø tabl(xr+1,yr)=3
2Ø9Ø tabl(xr+2,yr)=3
```

Mise à jour de tabl pour les 3 positions occupées par la raquette

```
21ØØ GOSUB 3Ø9Ø
211Ø raqz$=CHR$(143)+CHR$(143)
```

Définition de raqz$ pour l'effacement de la raquette dans ses déplacements avec 2 caracères (déplacement prévu).

```
212Ø '
213Ø 'init tableau
214Ø nt=Ø:ad1=ad:GOSUB 4Ø6Ø
215Ø '
216Ø 'init balle
217Ø yb=24:bal$=CHR$(231):xb=xr:db=4
```

La balle, caractère 231, se positionne en ligne 24, au-dessus de la raquette, sur le coté gauche de celle-ci, et la direction initiale est 4.

```
218Ø IF xb>18 THEN xb=18:db=4
219Ø IF xb<3 THEN xb=3:db=1
```

Ajustement de la position de la balle en fonction des coordonnées pour éviter de voir la balle dans la bordure du tableau.

```
22ØØ LOCATE xb,yb:PEN 2:PRINT bal$
221Ø j=JOY(Ø):IF j>8 THEN PEN Ø:LOCATE xb,yb:
PRINT bal$:RETURN
```

L'ordre JOY donne la valeur envoyée par le joystick, cette valeur correspond à la position. En début de partie tu peux déplacer la raquette de gauche à droite et tu appuies sur FIRE pour lancer la balle. En déplaçant la raquette la balle change de position.

```
222Ø IF j=Ø THEN 221Ø
223Ø IF j=4 THEN PEN Ø:LOCATE xb,yb:PRINT bal$:
GOSUB 318Ø:xb=xr+2:db=1:GOTO 218Ø
224Ø IF j=8 THEN PEN Ø:LOCATE xb,yb:PRINT bal$:
GOSUB 312Ø:xb=xr:db=4:GOTO 218Ø
225Ø GOTO 221Ø
```

Les lignes 222Ø à 224Ø testent le déplacement de la raquette: 4= gauche et 8= droite. Si gauche la balle passe à droite (xb=xr+2) et la direction = 1. Si droite la balle se place à gauche (xb=xr) et la direction = 4.

```
226Ø '-------------------------
232Ø '-------------------------
233Ø '  tirage aleatoire
234Ø '-------------------------
235Ø RANDOMIZE TIME
236Ø nombre=INT(RND*(maxi-mini+1))+mini
237Ø RETURN
238ØØ '-------------------------
381Ø '  trace page jeu
382Ø '-------------------------
383Ø '  cadre
384Ø '
385Ø CLS
386Ø x=Ø:xr=638:yr=366:pr=4
387Ø FOR j=1 TO 8:GOSUB 39ØØ:NEXT
388Ø RETURN
389Ø '
39ØØ MOVE x,32
391Ø DRAWR Ø,yr,j
392Ø DRAWR xr,Ø,j
393Ø DRAWR Ø,-yr,j
394Ø x=x+pr
395Ø yr=yr-pr
396Ø xr=xr-pr*2
397Ø RETURN
```

Les lignes 385Ø à 397Ø concernent le tracé du cadre avec 8 lignes graphiques de couleurs différentes.

```
398Ø '-------------------------
399Ø '   afficher score
4ØØØ '-------------------------
4Ø1Ø LOCATE 2,1:PEN 15:PRINT sco
4Ø2Ø RETURN
4Ø3Ø '-------------------------
4Ø4Ø '   init d"un tableau
4Ø5Ø '-------------------------
4Ø6Ø '  vider tabl
4Ø7Ø FOR x=1 TO 2Ø:FOR y=1 TO 25
4Ø8Ø tabl(x,y)=Ø
4Ø9Ø NEXT y:NEXT x
41ØØ ' mise a jour numero tableau
411Ø nt=nt+1:IF nt>maxt THEN nt=1:ad1=ad
```

Le numéro de tableau progresse de 1. Controle si plus grand que le maxi retour à

```
412Ø ad1=ad+(nt-1)*152
```

Calcul de l'adresse du tableau en fichier Tableau.

```
413Ø PRINT CHR$(22)+CHR$(Ø);
414Ø LOCATE 12,1:PEN 1Ø:PRINT nt;
415Ø '
416Ø 'nombre balles
417Ø nb=PEEK(ad1):GOSUB 453Ø:nbo=nb
418Ø 'nombre briques
419Ø j=PEEK(ad1+1):nbr=j:nbro=nbr
42ØØ ad1=ad1+2
421Ø 'chargement de tabl et affichage
422Ø PRINT CHR$(22)+CHR$(1);
423Ø FOR k=1 TO j
424Ø x=PEEK(ad1)
425Ø y=PEEK(ad1+1)
426Ø typ=PEEK(ad1+2)
427Ø ad1=ad1+3
428Ø tabl(x,y)=typ
429Ø LOCATE x,y
43ØØ 'afficher brique
431Ø ON tabl(x,y)-6 GOSUB 445Ø,447Ø,449Ø
432Ø NEXT k
433Ø '
434Ø PRINT CHR$(22)+CHR$(Ø);
435Ø 'remplir tabl avec les codes 1,2,4,6
436Ø FOR x=1 TO 1:FOR y=1 TO 24:tabl(x,y)=4:
NEXT y:NEXT x
437Ø FOR x=2Ø TO 2Ø:FOR y=1 TO 24:tabl(x,y)=2:
NEXT y:NEXT x
438Ø FOR x=1 TO 2Ø:FOR y=25 TO 25:tabl(x,y)=6:
NEXT y:NEXT x
439Ø FOR x=1 TO 2Ø:FOR y=1 TO 2:tabl(x,y)=1:
NEXT y:NEXT x
44ØØ RETURN
```

Ce sous-programme «Initialisation d'un tableau» a été largement expliqué ainsi que le contenu du fichier tabl, dans le numéro 7.

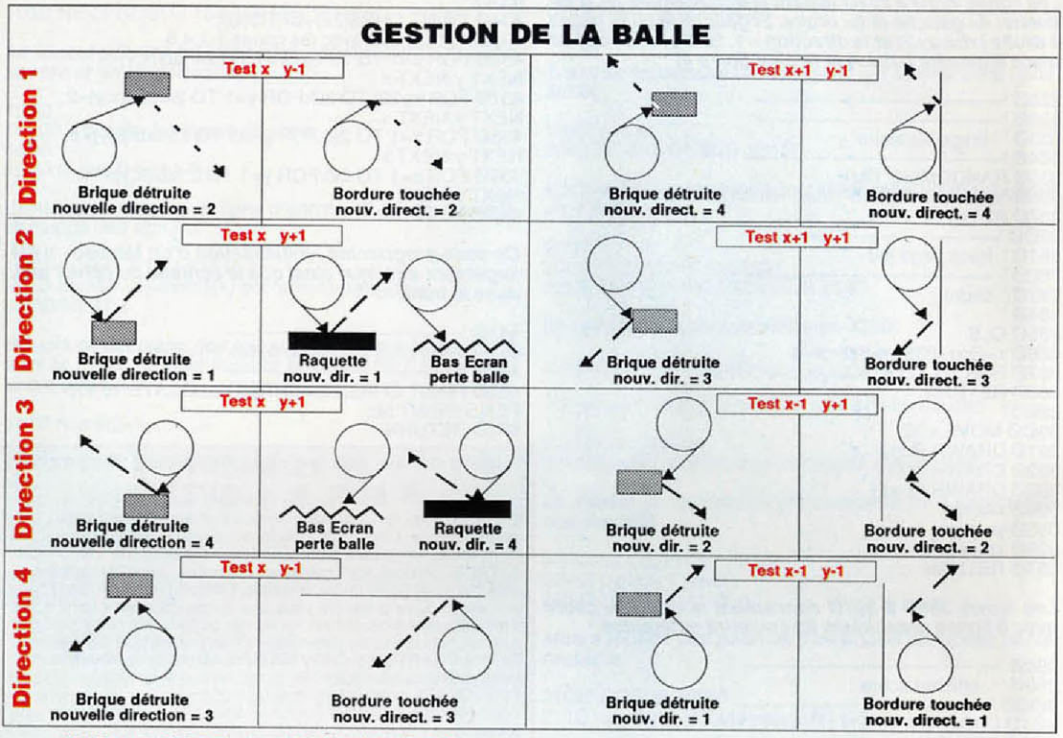
```
441Ø '-------------------------
451Ø 'sous-prog affiche nb balles en cours
452Ø '-------------------------
453Ø PRINT CHR$(22)+CHR$(Ø);:LOCATE 16,1:
PEN 5:PRINT nb;
454Ø RETURN
```

JE TE RECOIS 5 SUR 5....OVER....

Des messages en quantité, qui nous parviennent de partout. Les aventuriers de JOYSTICK-Hebdo, en pleine activité, transmettent leurs idées et posent des questions à propos de JOYSTICK-ATTACK, certains sont même en difficultés. Tous les problèmes sont dus à des petits oublis en saisie. Le listing ATTACK, paru dans le numéro 5, se présente sous une forme compacte car il s'agit d'une récapitulation de tous les listings partiels des numéros 2,3 et 4 et, les caractères de petite taille s'interprètent plus difficilement.
Patrick CHARTON et **Garry MAZUELAS** ont des problèmes avec la ligne 96Ø qui doit s'écrire:
96Ø LOCATE posx(j),posy(j)
Pour **Nicolas POURSIN** voici la ligne 78Ø, si tu ouvres une parenthèse il faut absolument la fermer:
78Ø nombre=INT(RND*(maxi-mini+1))+mini
Pascale BOUCHER s'interroge sur le contenu des DATA de la ligne 45Ø. Tu prévois les 16 numéros de couleurs, choisis parmi les 27 couleurs AMSTRAD. Ces couleurs permettent d'initialiser les 16 INK (Ø à 15) avec la boucle des lignes 53Ø à 56Ø.
Toutes les rubriques J'APPRENDS se suivent (et ne se ressemblent pas!) et forment un ensemble. Pour bien suivre et comprendre JOYSTICK-ATTACK les numéros 1 à 5 sont indispensables, s'il t'en manque un tu ne peux plus suivre, c'est le cas pour **Stéphane DECOMBIS**. A partir du numéro 6, tu as pris un nouveau départ avec JOYSTICK-BREAKER.
Si tu possèdes un AMSTRAD 164Ø, comme **Stéphane MALGOUIRES**, tu ne peux saisir un listing destiné aux AMSTRAD CPC. Il faut l'adapter.
Une très bonne question d' **Alexis GUTIERREZ**: comment fonctionne le tirage d'un nombre aléatoire (tiré au hasard), le sous-programme des lignes 77Ø à 79Ø de JOYSTICK-ATTACK. Pour pouvoir comprendre le contenu je te donne une petite séquence qui décompose les étapes du calcul et les affiche à l'écran: RND, RND*(maxi-mini+1), INT(résultat) et le nombre final. Dans (maxi-mini+1) si tu supprimes le +1 tu ne pourras jamais obtenir en résultat le maxi, tu vas pouvoir le controler par toi-même.

```
1Ø CLS:mini=1:maxi=3
2Ø RANDOMIZE TIME
3Ø a=RND
4Ø b=a*(maxi-mini+1)
5Ø nb=INT(b)+mini
6Ø LOCATE 5,12:PRINT a;b;INT(b);nb
7Ø IF INKEY$<>»» THEN 1Ø ELSE 7Ø
```

Vous êtes SUPER avec un wagon de suggestions pour la suite:
Jacques GOMEZ voudrait étudier un programme destiné à tenir l'historique du LOTO, **Bernard MANSARD** imagine un programme de gestion, **Thomas BRESSE** souhaite plus d'explications dans les listings, **Vincent DEMAREZ** demande un tableau de conversion BASIC pour les autres matériels, **Daniel EXCOFFIER** attend avec impatience une gestion de fichier.
Je vous ai compriiis!!!!!
Salut au futur ingénieur en informatique **Nicolas BERNARD**, quand tu veux pour les idées. Et comme les compliments font toujours très chaud au coeur , je termine avec un grand merci particulier à **Patricia BOUVARD**, **Pascal DRIDI**, **Stéphane CARRÉ**, **Bruce, Konrad** et les autres...



UNE BALLE A CONTROLER

Tu appuies sur FIRE et ta balle part, en direction 1 ou 4, suivant l'initialisation, tu dois maintenant la controler, c-à-d, la faire progresser en testant, à chaque déplacement la surface de jeu. Avec 4 directions ta balle évolue toujours en diagonale.

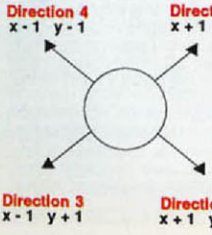
N'oublie pas que nous sommes en mode caractère et, que l'origine des coordonnées se trouve en haut et à gauche, pour x tout est normal avec plus vers la droite, mais y se trouve inversé, plus vers le bas et moins vers le haut. Mieux vaut toujours une image plutot que de longues phrases et, le grand tableau résume les différentes rencontres possibles de la balle en fonction de sa direction. L'ordre des tests reste toujours le même à partir de xb et yb, les coordonnées de la balle:

1) contenu en y + ou - 1
2) contenu en x + ou - 1 et en y + ou - 1

Si le résultat des 2 tests = Ø la voie est libre, ta balle continue dans sa direction. Si au moins l'un des tests est différent de Ø, il s'agit d'un obstacle:

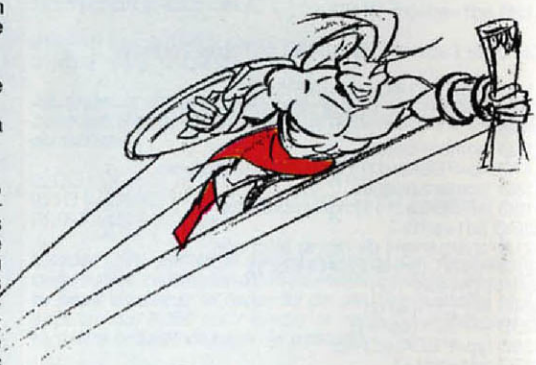
1 = la bordure en haut d'écran
2 = la bordure droite de l'écran
3 = la raquette
4 = la bordure gauche de l'écran
6 = le bas de l'écran et la balle est perdue
7, 8, 9 = une brique

Chaque direction ne peut, bien entendu, rencontrer l'ensemble des codes. Ces codes, positionnés dans le fichier tabl en initialisation du tableau, évoluent pour les briques, à chaque destruction remplacement par Ø, et pour la raquette. Essaie d'écrire cette séquence, avec tous les tests de collision, et la semaine prochaine on reprend avec les suites possibles: perte de la balle et destruction des briques.



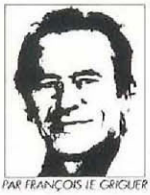
Est-ce que tu sens bien ton jeu? Quelques problèmes? Ne t'inqiète pas, ça va s'arranger avec la suite, mais si tu patines vraiment de trop écris-moi très vite. Tu vois que le listing contient beaucoup plus de commentaires, pour t'aider au maximum, comme tu me l'as demandé.

Bien Joystiquement Votre
François LE GRIGUER



LA SEMAINE PROCHAINE

Les listings + un lot d'explications + des ordinogrammes + la suite du jeu, plus et encore plus!! Et tout celà dans JOYSTICK-Hebdo numéro 1Ø!

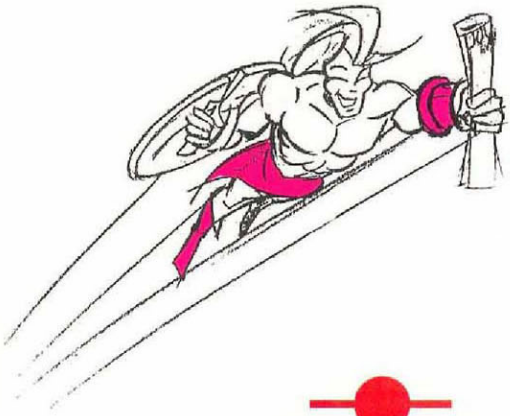


PAR FRANÇOIS LE GRIGUER

JOYSTICK-BREAKER

LA BALLE EN BASIC

JE TE SENS IMPATIENT, TU SAISIS LISTING SUR LISTING ET TU NE PEUX PAS ENCORE DETRUIRE LA MOINDRE BRIQUE. UNE SEMAINE, UNE TOUTE PETITE SEMAINE ET LE DERNIER MODULE INSDISPENSABLE, LA GESTION DE LA RAQUETTE, VA ETRE LA POUR REALISER UN SCORE D'ENFER. EN ATTENDANT, J'ESPERE QUE TU AS PREPARE 40 TABLEAUX AVEC DES DESSINS, TOUS PLUS FANTASTIQUES LES UNS QUE LES AUTRES ET PUIS, RIEN NE T'EMPECHE DE CREER DE NOUVEAUX MODELES DE BRIQUES AVEC DES TRACES ET DES COULEURS BIEN A TOI, JE T'EXPLIQUERAI COMMENT LES AJOUTER DANS TON PROGRAMME.



Tu charges "BREAKER", tu saisis le listing et, tu sauvegardes le tout. Mais comme tu es devenu un programmeur, tu dois comprendre le pourquoi de chaque ligne, terminée l'époque ou tu rentrais des listings sans chercher à comprendre. Maintenant les copains qui sont plantés viennent te voir, génial!! Attention, si tu rentres les lignes à une vitesse d'enfer n'oublie, quand même pas, de sauter tous les textes en italique, leur digestion par l'ordinateur ne se passerait pas très bien!! Bonjour les Syntax Error et Improper Argument.

```
39Ø 'gestion balle
4ØØ GOSUB 23ØØ
41Ø END
```

Lignes de complément pour le programme principal.

```
226Ø '---------------------------
227Ø '    gestion balle
228Ø '---------------------------
229Ø ' effacer balle
23ØØ LOCATE xb,yb:PEN Ø:PRINT bal$
```

La première opération: effacer la balle suivant ses coodonnées xb et yb en cours et avec PEN Ø, la couleur du fond.

```
231Ø ON db GOTO 234Ø,248Ø,263Ø,278Ø
```

A l'entrée dans cette séquence db contient la direction de la balle (1 à 4). ON db GOTO exécute un branchement à la ligne 234Ø si db=1, 248Ø si db=2 etc. .. Reprends le tableau de la gestion de la balle, paru dans le numéro 9, pour bien interpréter l'enchainement des tests de chaque direction.

```
232Ø '
233Ø '  direct 1
234Ø l=tabl(xb,yb-1)
```

Test du contenu de xb,yb-1. Les contenus possibles: Ø=rien, 1=bordure haute, sinon il s'agit d'une brique.

```
235Ø IF l=Ø THEN 239Ø
```

Si = Ø passage au second test.

```
236Ø IF l=1 THEN db=2:GOTO 248Ø
```

Si = 1 la balle rebondit sur la bordure et prend la direction 2. Suite en test de la direction 2.

```
237Ø 'brique
238Ø tabl(xb,yb-1)=Ø:LOCATE xb,yb-1:
GOSUB 3Ø1Ø:db=2:GOTO 248Ø
```

Une brique détruite entraine: la suppression en tabl, le passage dans la séquence "brique détruite", le changement de direction et la suite en traitement de la nouvelle direction 2.

```
239Ø c=tabl(xb+1,yb-1)
```

2ème groupe de tests pour la direction 1: xb+1,yb-1. Les contenus possibles: Ø=rien, 2=bordure gauche, sinon c'est une brique.

```
24ØØ IF c=Ø THEN 244Ø
```

Si = Ø se brancher en mise à jour des coordonnées, xb et yb, et affichage de la balle à sa nouvelle position.

```
241Ø IF c=2 THEN db=4:GOTO 278Ø
```

Si = 2 la balle rebondit et prend la direction 4. Suite en direction 4.

```
242Ø 'brique
243Ø tabl(xb+1,yb-1)=Ø:LOCATE xb+1,yb-1:
```

```
GOSUB 3Ø1Ø:db=4:GOTO 278Ø
```

La brique est détruite: mêmes opérations que précédemment.

```
244Ø xb=xb+1:yb=yb-1
245Ø LOCATE xb,yb:PEN 2:PRINT bal$:RETURN
```

Mise à jour des coodonnées pour la direction 1 et affichage de la balle, à la nouvelle position xb,yb et, avec l' INK 2. Fin de la séquence par RETURN.

```
246Ø '
```

La succession des tests reste identique pour les 3 autres directions, en y ajoutant le controle de la raquette et de la balle perdue pour les directions 2 et 3.

```
247Ø '  direct 2
248Ø l=tabl(xb,yb+1)
249Ø IF l=Ø THEN 254Ø
25ØØ IF l=3 THEN db=1:GOTO 234Ø
```

Si = 3 la balle rebondit sur la raquette.

```
251Ø IF l=6 THEN 292Ø
```

Si = 6 branchement en balle perdue, sinon il s'agit d'une brique.

```
252Ø 'brique
253Ø tabl(xb,yb+1)=Ø:LOCATE xb,yb+1:
GOSUB 3Ø1Ø:db=1:GOTO 234Ø
254Ø c=tabl(xb+1,yb+1)
255Ø IF c=Ø THEN 259Ø
256Ø IF c=2 THEN db=3:GOTO 263Ø
257Ø 'brique
258Ø tabl(xb+1,yb+1)=Ø:LOCATE xb+1,yb+1:
GOSUB 3Ø1Ø:db=3:GOTO 263Ø
259Ø xb=xb+1:yb=yb+1
26ØØ LOCATE xb,yb:PEN 2:PRINT bal$:RETURN
261Ø '
262Ø '  direct 3
263Ø l=tabl(xb,yb+1)
264Ø IF l=Ø THEN 269Ø
265Ø IF l=3 THEN db=4:GOTO 278Ø
266Ø IF l=6 THEN 292Ø
267Ø 'brique
268Ø tabl(xb,yb+1)=Ø:LOCATE xb,yb+1:
GOSUB 3Ø1Ø:db=4:GOTO 278Ø
269Ø c=tabl(xb-1,yb+1)
27ØØ IF c=Ø THEN 274Ø
271Ø IF c=4 THEN db=2:GOTO 248Ø
272Ø 'brique
273Ø tabl(xb-1,yb+1)=Ø:LOCATE xb-1,yb+1:
GOSUB 3Ø1Ø:db=2:GOTO 248Ø
274Ø xb=xb-1:yb=yb+1
275Ø LOCATE xb,yb:PEN 2:PRINT bal$:RETURN
276Ø '
277Ø '  direct 4
278Ø l=tabl(xb,yb-1)
279Ø IF l=Ø THEN 283Ø
28ØØ IF l=1 THEN db=3:GOTO 263Ø
281Ø 'brique
282Ø tabl(xb,yb-1)=Ø:LOCATE xb,yb-1:
GOSUB 3Ø1Ø:db=3:GOTO 263Ø
283Ø c=tabl(xb-1,yb-1)
284Ø IF c=Ø THEN 288Ø
285Ø IF c=4 THEN db=1:GOTO 234Ø
286Ø 'brique
287Ø tabl(xb-1,yb-1)=Ø:LOCATE xb-1,yb-1:
GOSUB 3Ø1Ø:db=1:GOTO 234Ø
288Ø xb=xb-1:yb=yb-1
289Ø LOCATE xb,yb:PEN 2:PRINT bal$:RETURN
```

Les 4 ensembles de tests devraient se regrouper sur un nombre beaucoup plus restreint de lignes mais, le but est de t'expliquer les opérations au pas-à-pas. La multiplication des lignes donne un listing plus clair, au détriment de la place mémoire et, de la rapidité du traitement.

```
29ØØ '-------------------------------
291Ø '  perte balle
292Ø nb=nb-1:GOSUB 453Ø
```

Nombre de balles -1 et affichage du nouveau nombre.

```
293Ø FOR j=1 TO 3ØØ:NEXT j
```

ITLD Boucle d'attente (tempo). ITLF

```
294Ø IF nb<>Ø THEN 217Ø
```

Les 2 symboles < et > associés donnent: différent de. A l'impression ces deux signes paraissent souvent former un losange. On aurait pu écrire également: IF nb>Ø THEN 217Ø, nb étant toujours positif. Le branchement s'effectue en 217Ø pour initialiser une nouvelle balle.

```
295Ø GOSUB 325Ø
```

Branchement dans le sous-programme "GAME OVER", n'ayant plus de balle la partie se termine.

```
296Ø GOSUB 16ØØ
```

Séquence de la page des scores.

```
297Ø GOSUB 197Ø
```

Séquence d'initialisation du début de partie.

```
298Ø RETURN
299Ø '  brique detruite
3ØØØ 'points=(nb briques tableau-nb balles perdues)
*num tableau
3Ø1Ø PEN Ø:PRINT CHR$(143)
```

Effacement de la brique à l'écran

```
3Ø2Ø sco=sco+(nbro-(nbo-nb))*nt
```

La mise à jour du score, avantage le bon joueur avec la prise en compte du nombre de balles perdues (nombre origine - nombre en cours) et, la multiplication par le numéro de tableau.

```
3Ø3Ø GOSUB 4Ø1Ø:nbr=nbr-1
```

Affichage du nouveau score et -1 en nombre de briques.

```
3Ø4Ø IF nbr<>Ø THEN RETURN
```

S'il reste des briques à détruire fin de la séquence, sinon changement de tableau.

```
3Ø5Ø '  fin tableau
3Ø6Ø IF nbo<>nb THEN 3Ø7Ø
```

Pour bénéficier du Bonus il ne faut pas avoir perdu une seule balle dans le tableau, donc nbo=nb (nombre balle origine = nombre en cours).

```
3Ø61 LOCATE 7,12:PEN 12:PRINT"BONUS";nbro*1ØØ*nt;
```

*Affichage de BONUS et de sa valeur: nombre de briques du tableau * 1ØØ * le numéro de tableau.*

```
3Ø62 sco=sco+(nbro*1ØØ*nt):GOSUB 4Ø1Ø
```

Mise à jour du score et affichage.

```
3Ø63 FOR j=1 TO 6ØØ:NEXT
```

Boucle de tempo (temps d'affichage de BONUS à l'écran).

```
3Ø64 LOCATE 7,12:PEN Ø:PRINT STRING$(12,CHR$(143));
```

Effacement de BONUS.

```
3Ø7Ø GOSUB 4Ø6Ø:GOTO 217Ø
```

Initialisation d'un nouveau tableau et branchement sur l'initialisation de la balle.

```
3Ø8Ø '-------------------------
```

Sauvegarde immédiatement ton nouveau listing par SAVE"breaker"

et, tu dois encore attendre le module de gestion de la raquette pour pouvoir lancer ton programme. Dès la semaine prochaine tu vas pouvoir jouer.

UNE BALLE DE PERDUE

dix de retrouvées!! enfin pas toujours. La séquence se trouve dans le listing mais on va refaire rapidement l'historique de la gestion de la balle. Dans le fichier "Tableaux" chaque tableau contient son nombre de balles, suivant la valeur entrée lors de la redéfinition. Ce nombre de balles est pris en charge avec l'initialisation du tableau et, affiché en haut de la surface de jeu. Les directions 2 et 3 testent la rencontre d'un code 6 en tabl et, si c'est OUI, tu décrémentes le nombre de balles en cours (nb-1). Tu testes la nouvelle valeur de nb, avec une réponse NON il suffit d'afficher nb et d'initialiser une nouvelle balle en ligne 24 (yb=24) sur la raquette. Avec une réponse OUI nb=Ø la partie s'achève par GAME OVER et, le traitement passe dans la page des scores, dans laquelle le joueur entre éventuellement. Ensuite une nouvelle partie commence.
A chaque brique détruite le fichier tabl est mis à jour avec un zéro qui remplace le code 7,8 ou 9. Tu controles si le nouveau nombre de briques, nbr, est différent de Ø (<>Ø), si OUI la partie continue il reste à mettre le score à jour. Si nbr=Ø le 'ableau se termine par un BONUS si le nombre de balles en cours reste identique au nombre de balles de début, initialisation d'un nouveau tableau et d'une balle, et on enchaine. Si tu as définis 4Ø tableaux l'initialisation progresse de 1 à 4Ø et retour à 1, sinon le retour à 1 s'effectue au-delà du nombre maxi.
Le mode de calcul de la mise à jour du score et l'interprétation du bonus que je te propose peuvent, bien entendu, se modifier selon tes critères personnels. La personnalisation de ton jeu te permet de tester tes connaissances, un excellent exercice que je te conseille. En plus tu pourrais nous faire profiter de tes idées en m'envoyant un petit listing, pourquoi pas!! Pour tes essais, tu enregistres ton listing sous un autre label et, tu le modifies. Si tu te plantes complètement tu auras toujours le programme d'origine.

POUR QUELQUES DEFINITIONS DE PLUS...

Tu sais maintenant réaliser un dessin, en superposant ou, en juxtaposant des caractères redéfinis par l'instruction SYMBOL. Tu l'utilises pour le dessin des briques mais, ton AMSTRAD CPC, permet d'améliorer encore cette technique par l'utilisation de caractères spéciaux. Les lignes qui suivent représentent les ordres Basic pour l'affichage de la brique numéro 7, avec PEN pour le numéro d'INK à utiliser, LOCATE qui redonne la position d'affichage et PRINT suivi du numéro de carcatère à afficher et, tout ceci, suivant le listing publié dans JOYSTICK-Hebdo numéro 7:

```
445Ø PEN 5:PRINT CHR$(255):
PEN 7:LOCATE x,y:PRINT CHR$(254):
PEN 12:LOCATE x,y:PRINT CHR$(253):
PEN 13:LOCATE x,y:PRINT CHR$(252):RETURN
```

Ce groupe d'instructions peut être remplacé par:

```
445Ø PRINT CHR$(15);CHR$(5);CHR$(255);
CHR$(15);CHR$(7);CHR$(8);CHR$(254);
CHR$(15);CHR$(12);CHR$(8);CHR$(253);
CHR$(15);CHR$(13);CHR$(8);CHR$(252);:RETURN
```

Ici seul reste PRINT, PEN et LOCATE ont disparus, pourquoi et comment? avec les fameux caractères spéciaux:
- CHR$(15) correspond à PEN et le CHR$() suivant indique le numéro d'INK
- CHR$(8) déplace la position d'affichage d'un caractère en arrière et remplace LOCATE pour le retour sur la position de la brique

Il s'agit donc d'une simple énumération de caractères, séparés par le point virgule comme dans toute succession d'édition. L'affichage peut se préparer par la déclaration de sous-ensembles, comme par exemple, la décomposition d'une brique en 4 parties:

```
531 b7Ø$=CHR$(15)+ CHR$(5)+ CHR$(255)+CHR$(8)
532 b71$=CHR$(15)+ CHR$(7)+CHR$(254)+ CHR$(8)
533 b72$=CHR$(15)+CHR$(12)+CHR$(253)+CHR$(8)
534 b73$=CHR$(15)+CHR$(13)+CHR$(252)
```

Les 4 sous-ensembles définis en début de générique permettent alors un affichage plus simple dans le sous-programme de la brique 7:

```
445Ø PRINT b7Ø$;b71$;b72$;b73$:RETURN
```

D'autres caractères spéciaux sont disponibles et, en particulier:
- CHR$(9) déplace la position d'affichge d'une position en avant
- CHR$(1Ø) déplace d'une ligne vers le bas
- CHR$(11) une ligne vers le haut
- CHR$(22)+CHR$(Ø) rétablit le mode d'affichage normal
- CHR$(22)+CHR$(1) active le mode d'affichage transparent

Les autres caractères, décrits dans ton manuel d'utilisation, présentent moins d'intérêt.
Cette rubrique, destinée aux débutants avides de découvertes, traite des bases générales de l'informatique comme de particularités liées au développement des sujets sur AMSTRAD CPC et, pour satisfaire tout le monde je te propose de parler des bases de numération.
Un souvenir scolaire plus ou moins lointain, ou encore plus ou moins bon! peut importe, mais en informatique tu dois connaitre deux bases: le binaire et l'hexadécimal. Nous apprenons tous à compter en décimal, c-à-d, en base 1Ø avec 1Ø symboles de Ø à 9, sans oublier le zéro, ce n'est pas 1, 2, 3... mais Ø, 1, 2, 3... Arrivé à 1Ø tu changes de colonne, en inscrivant un Ø dans la colonne des unités et 1 dans la suivante, la colonne des dizaines. Chaque colonne a donc une valeur bien définie (unités, dizaines, centaines, milliers etc...):

```
etc... 1ØØØ 1ØØ  1Ø   1
```

L'interprétation du contenu de chacune de nos colonnes consiste à le multiplier par la valeur de sa colonne pour reconstituer le nombre représenté:

```
2  5  3  7

2 multiplié par 1ØØØ = 2ØØØ
5   "    "  1ØØ =  5ØØ
3   "    "  1Ø =  3Ø
7   "    "   1 =   7
```

Le total est bien de 2537. Le principe reste exactement le même pour les autres bases. La base 16, l'hexadécimal, comporte 16 symboles: Ø, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E, F. Les lettres A à F évitent toute confusion pour les valeurs de 1Ø à 15. Les colonnes ont cette fois comme valeur:

```
etc... 4Ø96  256   16    1

       F     F     F     F
```

Ce nombre FFFF, en hexadécimal, se traduit en décimal par:

```
F multiplié par 4Ø96 = 6144Ø
F   "    "   256 =  384Ø
F   "    "    16 =   24Ø
F   "    "     1 =   15
```

Le total donne 65535 en décimal, représente l'adresse la plus haute pour un CPC 464, soit 64 Ko (64 * 1Ø24 = 65536 donc Ø à 65535 = 64 Ko).
A l'inverse pour convertir un nombre décimal en hexadécimal, le diviser par 16, le nombre en hexadécimal prend successivement le reste de chaque division:

3Ø6 divisé par 16 = 19 reste 2
19 " " 16 = 1 reste 3
La valeur hexadécimale est donc de: 132. Ce résultat se controle par:
(1*256) + (3*16) + (2*1) = 3Ø6 décimal

Dans un prochain numéro je te donnerai une table de conversion pour éviter tous ces calculs et, on parlera de la base binaire.

Comment vas l'as de la programmation? Bof! Un peu dur! Super! Quelle que soit ta réponse tu es le plus génial puisque tu lis JOYSTICK-Hebdo. Encore huit jours et tu vas pouvoir épater tout le monde, en leur expliquant comment tu as programmé ton casse-briques, et celà vaut la peine de se donner un peu de mal.

Bien Joystiquement Votre
François LE GRIGUER

LA SEMAINE PROCHAINE

Le listing complet de JOYSTICK-BREAKER, avec la gestion de la raquette et le complément du programme principal.

UNE RAQUETTE SUPER RAPIDE

Le dernier élément indispensable à ton casse-briques, et pas le moindre, une raquette en bas d'écran, en ligne 25, que tu déplaces au joystick, une raquette super rapide pour suivre les mouvements de la balle. Toutes les constantes et variables sont déjà définies et, il te reste à tester l'ordre joystick avec seulement 2 valeurs possibles: 4 = gauche et 8 = droite.
JOY(Ø)=4 ou JOY(Ø)=8
La séquence reste très simple à écrire. La valeur de déplacement est fixée à + ou - 2 pour la mise à jour de xr, donc plus rapide que la balle. Les opérations à réaliser se résument simplement:

- test pour le branchement en déplacement gauche ou droite
- test de xr pour rester à l'intérieur des limites de l'écran
- effacer 2 caractères de la raquette (2 premiers ou 2 derniers)
- mettre à jour tabl avec le code 6 pour les 2 caractères effacés
- mettre à jour xr par + ou - 2 et tabl avec le code 3 (nouvelle position)
- afficher la raquette dans sa nouvelle position

Quoi d'autre? Rien, tu écris la séquence et tu controles si la balle rebondit bien sur la raquette. Si par hasard elle passe à travers ce sont les codes 3 et 6 qui se positionnent mal en tabl. La gestion de la raquette, prévue au joystick, peut se programmer au clavier en remplaçant l'intruction JOY par un test de touche.

TABLEAU RECAPITULATIF DE LA GESTION DE LA BALLE

Direction		Position à tester	Code en fichier tabl	Changement de direction	brique détruite	m.a.j. des coordonnées	afficher la balle	suite en
1	1er test	xb, yb - 1	Ø 1 7, 8, 9	2 2	oui			2e test Test Direct. 2 Test Direct. 2
	2e test	xb + 1, yb - 1	Ø 2 7, 8, 9	4 4	oui	xb + 1 yb - 1	oui	Fin séquence Test Direct. 4 Test Direct. 4
2	1er test	xb, yb + 1	Ø 7, 8, 9 3 6	1 1	oui			2e test Test Direct. 1 Test Direct. 1 Balle perdue
	2e test	xb + 1, yb + 1	Ø 2 7, 8, 9	3 3	oui	xb + 1 yb + 1	oui	Fin séquence Test Direct. 3 Test Direct. 3
3	1er test	xb, yb + 1	Ø 7, 8, 9 3 6	4 4	oui			2e test Test Direct. 4 Test Direct. 4 Balle perdue
	2e test	xb - 1, yb + 1	Ø 4 7, 8, 9	2 2	oui	xb - 1 yb + 1	oui	Fin séquence Test Direct. 2 Test Direct. 2
4	1er test	xb, yb - 1	Ø 1 7, 8, 9	3 3	oui			2e test Test Direct 3 Test Direct. 3
	2e test	xb - 1, yb - 1	Ø 4 7, 8, 9	1 1	oui	xb - 1 yb - 1	oui	Fin séquence Test Direct. 1 Test Direct. 1

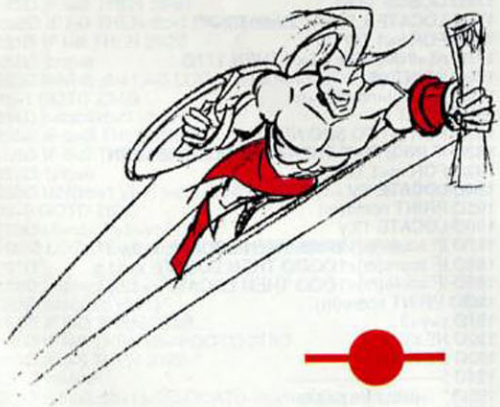
PAR FRANÇOIS LE GRIGUER

JOYSTICK-BREAKER

UN LISTING POUR BEAUCOUP PLUS DE CENT BRIQUES

DEPUIS LE NUMERO 6 TU CONSTRUIS, SEMAINE APRES SEMAINE, TON CASSE-BRIQUES. AUJOURD'HUI C'EST SUPER!! LE PROGRAMME FONCTIONNE COMPLETEMENT. UN PEU COMME UN PUZZLE LES PIECES S'ASSEMBLENT, CERTAINES DU PREMIER COUP, D'AUTRES AVEC UN PEU PLUS DE DIFFICULTES ET, L'IDEAL SERAIT DE POUVOIR RECOMMENCER LES YEUX FERMES, SANS AIDE. APRES LE LISTING INTEGRAL, IL NOUS RESTE ENCORE A L'AMELIORER MAIS, DES LA SEMAINE PROCHAINE, A BORD DU VAISSEAU SPATIAL "JOYSTICK-HEBDO", NOUS ALLONS, TOUS ENSEMBLE, EXPLORER UNE NOUVELLE PLANETE.

J'APPRENDS



Je te propose le listing complet pour t'aider à ordonner toutes les séquences déjà publiées et, à controler tes saisies. Si tu as pris le jeu en marche ceci te donne les éléments qui te manquaient. La gestion de la balle, étudiée la semaine dernière, figure à partir de la ligne 3Ø8Ø et, les lignes 41Ø à 44Ø sont à ajouter dans le programme principal.
Certaines lignes BASIC sont coupées pour des raisons de mise en page mais, tu dois, dans tous les cas, les saisir à suivre et, ne faire ENTER que juste avant un nouveau numéro de ligne.

```
1Ø '**************************
2Ø '    JOYSTICK-BREAKER
3Ø '**************************

4Ø 'un programme de .....
5Ø 'decembre 1988
6Ø '
7Ø '**************************
8Ø '
9Ø 'codes collisions en tabl
1ØØ 'haut =1
11Ø 'droite =2
12Ø 'raquette=3
13Ø 'bas(perte balle)=6
14Ø 'gauche=4
15Ø 'briques=7,8,9
16Ø '
17Ø '--------------------------
18Ø 'PROGRAMME PRINCIPAL
19Ø '--------------------------

2ØØ '152 octets reserves par tableau =5Ø briques maxi
21Ø 'nombre maxi tableaux 4Ø*152=6Ø8Ø octets
22Ø ' structure tableau
23Ø 'nb balles/nb briques/brique:x,y,type
24Ø 'en ad-1 numero tableau maxi
25Ø '
26Ø ad=33ØØØ:MEMORY ad-2:LOAD"TABLEAUX",ad-1:
maxt=PEEK(ad-1)
27Ø 'init couleurs et briques
28Ø GOSUB 341Ø
29Ø 'generique
3ØØ GOSUB 53Ø
31Ø 'init des fichiers
32Ø GOSUB 148Ø
33Ø 'redefinitions
34Ø GOSUB 71Ø
35Ø 'page score
36Ø GOSUB 153Ø
37Ø 'init debut de partie
38Ø GOSUB 198Ø
39Ø 'gestion balle
4ØØ GOSUB 23ØØ
41Ø 'gestion raquette si joy=8=droite. si joy=4=gauche
42Ø ON JOY(Ø)+1 GOSUB 44Ø,44Ø,44Ø,44Ø,318Ø,44Ø,
44Ø,44Ø,312Ø
43Ø GOTO 4ØØ
44Ø RETURN

45Ø '**************************
46Ø '**************************
47Ø '  sequences
48Ø '**************************
49Ø '**************************
5ØØ '--------------------------
51Ø '     generique
52Ø '--------------------------
```

```
530 MODE 0:PRINT CHR$(22)+CHR$(1);
540 y=10:FOR x=3 TO 17:LOCATE x,y:GOSUB 4450:NEXT x
550 y=12:FOR x=3 TO 17:LOCATE x,y:GOSUB 4450:NEXT x
560 y=14:FOR x=3 TO 17:LOCATE x,y:GOSUB 4470:NEXT x
570 y=16:FOR x=3 TO 17:LOCATE x,y:GOSUB 4470:NEXT x
580 LOCATE 3,11:PEN 13:PRINT"J O Y S T I C K"
590 LOCATE 4,15:PRINT"B R E A K E R"
600 ' trace du cadre
610 MOVE 64,500:DRAWR 476,0,2
620 DRAWR 0,80,2:DRAWR -476,0,2:DRAWR 0,-80,2
630 LOCATE 5,18:PRINT"realisation"
640 LOCATE 10,19:PRINT"de"
650 ' entre ton nom
660 LOCATE 5,21:PRINT"                    "
670 FOR j=1 TO 900:NEXT:RETURN
680 '---------------------------
690 '     redefinition de tableaux
700 '---------------------------
710 MODE 1:x=9:PEN 3:LOCATE x,5:
PRINT"REDEFINITION TABLEAUX"
720 LOCATE x+5,8:PRINT"tableau   T"
730 LOCATE x+5,10:PRINT"jeu      J"
740 a$=INKEY$:IF a$="" THEN 740
750 IF a$="j" OR a$="J" THEN
SAVE"tableaux",b,ad-1,61000:MODE 0:RETURN
760 IF a$="t" OR a$="T" THEN 790 ELSE 720
770 'entree/modification numero tableau
780 '
790 MODE 0:LOCATE 1,1:INPUT"tableau";n
800 IF n=0 OR n>40 OR  n>maxt+1 THEN 710
810 IF n>maxt THEN POKE ad-1,n:maxt=n
820 ' vider tabl
830 FOR x=3 TO 18:FOR y=3 TO 20:tabl(x,y)=0:
NEXT y:NEXT x
840 '
850 'calcul adresse tableau (n-1)*152 octets
860 ad1=ad+(n-1)*152
870 INPUT"balles";b:IF b=0 OR b>20 THEN LOCATE 1,2:
GOTO 870
880 POKE ad1,b
890 'chargement du tableau et affichage
900 PRINT CHR$(22)+CHR$(1)
910 j=PEEK(ad1+1):IF j=0 THEN 980
920 ad2=ad1+2
930 FOR k=1 TO j:x=PEEK(ad2):y=PEEK(ad2+1):
b=PEEK(ad2+2)
940 tabl(x,y)=b:ad2=ad2+3
950 LOCATE x,y:ON b-6 GOSUB 4450,4470,4490
960 NEXT k
970 '
980 nb=j
990 '  messages bas ecran
1000 LOCATE 1,25:PRINT"fin      F"
1010 LOCATE 1,23:PRINT"7/8/9 ou Supprimer";
1020 '
1030 'positionner curseur centre tableau
1040 'test touche deplacement/fonction
1050 x=10:y=10
1060 GOSUB 1340
1070 IF INKEY$="" THEN 1060
1080 IF INKEY(3)>-1 THEN 1250
1090 IF INKEY(5)>-1 THEN 1260
1100 IF INKEY(13)>-1 THEN 1270
1110 IF INKEY(10)>-1 THEN 1280
1120 IF INKEY(11)>-1 THEN 1290
1130 IF INKEY(4)>-1 THEN 1300
1140 IF INKEY(14)>-1 THEN 1310
1150 IF INKEY(20)>-1 THEN 1330
1160 'entree brique ou suppression
1170 IF INKEY(41)>-1 THEN tabl(x,y)=7:GOTO 1060
1180 IF INKEY(40)>-1 THEN tabl(x,y)=8:GOTO 1060
1190 IF INKEY(33)>-1 THEN tabl(x,y)=9:GOTO 1060
1200 IF INKEY(60)>-1 THEN tabl(x,y)=0:GOTO 1060
1210 'fin de saisie touche f
1220 IF INKEY(53)>-1 THEN 1390
1230 GOTO 1060
1240 'deplacement curseur
1250 IF x=18 OR y=3 THEN 1060 ELSE x=x+1:y=y-1:
GOTO 1170
1260 IF x=18 OR y=20 THEN 1060 ELSE x=x+1:y=y+1:
GOTO 1170
1270 IF x=3 OR y=20 THEN 1060 ELSE x=x-1:y=y+1:
GOTO 1170
1280 IF x=3 OR y=3 THEN 1060 ELSE x=x-1:y=y-1:
GOTO 1170
1290 IF y=3 THEN 1060 ELSE y=y-1:GOTO 1170
1300 IF x=18 THEN 1060 ELSE x=x+1:GOTO 1170
1310 IF y=20 THEN 1060 ELSE y=y+1:GOTO 1170
1320 'affichage curseur/brique
1330 IF x=3 THEN 1060 ELSE x=x-1:GOTO 1170
1340 LOCATE x,y:PEN 3:PRINT CHR$(143):LOCATE x,y
1350 IF tabl(x,y)=0 THEN PEN 0:PRINT CHR$(143):
RETURN
1360 ON tabl(x,y)-6 GOSUB 4450,4470,4490
1370 RETURN
1380 'mise a jour donnees en fichier tableaux
1390 ad2=ad1+2:nb=0
1400 FOR x=3 TO 18:FOR y=3 TO 20
1410 IF nb=50 THEN LOCATE 2,21:PEN 3:
PRINT"plus de 50 briques";:
FOR j=1 TO 600:NEXT j:GOTO 1440
1420 a=tabl(x,y):IF a<>0 THEN POKE ad2,x:
POKE ad2+1,y:POKE ad2+2,a:ad2=ad2+3:nb=nb+1
1430 NEXT y:NEXT x
1440 POKE ad1+1,nb:GOTO 710
1450 '---------------------------
1460 '  init des fichiers
1470 '---------------------------
1480 DIM tabl(20,25)
1490 RETURN
1500 '---------------------------
1510 '    page de score
1520 '---------------------------
1530 DIM nom$(8):DIM score(8)
1540 RESTORE 1550
1550 DATA BREAKER,SULYANE,MICHEL,SERVANE,
ERIC,GWENOLA,JEANINE,CLAUDE
1560 DATA 550700,130100,115200,110100,
70300,69000,51000,45000
1570 FOR j=1 TO 8:READ nom$(j):NEXT
1580 FOR J=1 TO 8:READ score(j):NEXT
1590 '
1600 CLS:PEN 1:LOCATE 5,2:PRINT "S C O R E "
1610 x=3:y=5
1620 FOR j=1 TO 8:IF sco>score(j) THEN 1670
1630 NEXT:GOSUB 1830:GOTO 1810
1640 '
1650 '   decaler les lignes de score
1660 '
1670 IF j=8 THEN 1710
1680 FOR k=7 TO j STEP -1
1690 nom$(k+1)=nom$(k):score(k+1)=score(k)
1700 NEXT k
1710 nom$(j)="....."
1720 IF INKEY$<>"" THEN 1720
1730 score(j)=sco
1740 GOSUB 1830
1750 LOCATE x,5+(j-1)*2:nom$(j)=""
1760 FOR k=1 TO 10
1770 n$=INKEY$:IF n$="" THEN 1770
1780 PRINT n$;
1790 nom$(j)=nom$(j)+n$
1800 NEXT k
1810 FOR j=1 TO 500:NEXT
1820 IF INKEY$="" THEN 1820 ELSE RETURN
1830 FOR m=1 TO 8
1840 LOCATE x,y
1850 PRINT nom$(m)
1860 LOCATE 13,y
1870 IF score(m)>99999 THEN LOCATE x+9,y
1880 IF score(m)<10000 THEN LOCATE x+11,y
1890 IF score(m)<1000 THEN LOCATE x+12,y
1900 PRINT score(m)
1910 y=y+2
1920 NEXT m
1930 RETURN
1940 '---------------------------
1950 '    debut de partie
1960 '---------------------------
1970 'trace de la page jeu
1980 GOSUB 3810
```

```
199Ø 'score
2ØØØ sco=Ø:GOSUB 4Ø1Ø
2Ø1Ø '
2Ø2Ø 'init position raquette
2Ø3Ø mini=2:maxi=15:GOSUB 335Ø
2Ø4Ø xr=nombre:yr=25
2Ø5Ø 'init raquette
2Ø6Ø raq$=CHR$(2Ø8)+CHR$(2Ø8)+CHR$(2Ø8)
2Ø7Ø tabl(xr,yr)=3
2Ø8Ø tabl(xr+1,yr)=3
2Ø9Ø tabl(xr+2,yr)=3
21ØØ GOSUB 3Ø9Ø
211Ø raqz$=CHR$(143)+CHR$(143)
212Ø '
213Ø 'init tableau
214Ø nt=Ø:ad1=ad:GOSUB 4Ø6Ø
215Ø '
216Ø 'init balle
217Ø yb=24:bal$=CHR$(231):xb=xr:db=4
218Ø IF xb>18 THEN xb=18:db=4
219Ø IF xb<3 THEN xb=3:db=1
22ØØ LOCATE xb,yb:PEN 2:PRINT bal$
221Ø j=JOY(Ø):IF j>8 THEN PEN Ø:LOCATE xb,yb:
PRINT bal$:RETURN
222Ø IF j=Ø THEN 221Ø
223Ø IF j=4 THEN PEN Ø:LOCATE xb,yb:PRINT bal$:
GOSUB 318Ø:xb=xr+2:db=1:GOTO 218Ø
224Ø IF j=8 THEN PEN Ø:LOCATE xb,yb:PRINT bal$:
GOSUB 312Ø:xb=xr:db=4:GOTO 218Ø
225Ø GOTO 221Ø
226Ø '-------------------------
227Ø '    gestion balle
228Ø '-------------------------
229Ø ' effacer balle
23ØØ LOCATE xb,yb:PEN Ø:PRINT bal$
231Ø ON db GOTO 234Ø,248Ø,263Ø,278Ø
232Ø '
233Ø '   direct 1
234Ø l=tabl(xb,yb-1)
235Ø IF l=Ø THEN 239Ø
236Ø IF l=1 THEN db=2:GOTO 248Ø
237Ø 'brique
238Ø tabl(xb,yb-1)=Ø:LOCATE xb,yb-1:GOSUB 3Ø1Ø:
db=2:GOTO 248Ø
239Ø c=tabl(xb+1,yb-1)
24ØØ IF c=Ø THEN 244Ø
241Ø IF c=2 THEN db=4:GOTO 278Ø
242Ø 'brique
243Ø tabl(xb+1,yb-1)=Ø:LOCATE xb+1,yb-1:GOSUB 3Ø1Ø:
db=4:GOTO 278Ø
244Ø xb=xb+1:yb=yb-1
245Ø LOCATE xb,yb:PEN 2:PRINT bal$:RETURN
246Ø '
247Ø '   direct 2
248Ø l=tabl(xb,yb+1)
249Ø IF l=Ø THEN 254Ø
25ØØ IF l=3 THEN db=1:GOTO 234Ø
251Ø IF l=6 THEN 292Ø
252Ø 'brique
253Ø tabl(xb,yb+1)=Ø:LOCATE xb,yb+1:GOSUB 3Ø1Ø:
db=1:GOTO 234Ø
254Ø c=tabl(xb+1,yb+1)
255Ø IF c=Ø THEN 259Ø
256Ø IF c=2 THEN db=3:GOTO 263Ø
257Ø 'brique
258Ø tabl(xb+1,yb+1)=Ø:LOCATE xb+1,yb+1:GOSUB 3Ø1Ø:
db=3:GOTO 263Ø
259Ø xb=xb+1:yb=yb+1
26ØØ LOCATE xb,yb:PEN 2:PRINT bal$:RETURN
261Ø '
262Ø '   direct 3
263Ø l=tabl(xb,yb+1)
264Ø IF l=Ø THEN 269Ø
265Ø IF l=3 THEN db=4:GOTO 278Ø
266Ø IF l=6 THEN 292Ø
267Ø 'brique
268Ø tabl(xb,yb+1)=Ø:LOCATE xb,yb+1:GOSUB 3Ø1Ø:
db=4:GOTO 278Ø
269Ø c=tabl(xb-1,yb+1)
27ØØ IF c=Ø THEN 274Ø
271Ø IF c=4 THEN db=2:GOTO 248Ø
272Ø 'brique
273Ø tabl(xb-1,yb+1)=Ø:LOCATE xb-1,yb+1:GOSUB 3Ø1Ø:
db=2:GOTO 248Ø
274Ø xb=xb-1:yb=yb+1
275Ø LOCATE xb,yb:PEN 2:PRINT bal$:RETURN
276Ø '
277Ø '   direct 4
278Ø l=tabl(xb,yb-1)
279Ø IF l=Ø THEN 283Ø
28ØØ IF l=1 THEN db=3:GOTO 263Ø
281Ø 'brique
282Ø tabl(xb,yb-1)=Ø:LOCATE xb,yb-1:GOSUB 3Ø1Ø:
db=3:GOTO 263Ø
283Ø c=tabl(xb-1,yb-1)
284Ø IF c=Ø THEN 288Ø
285Ø IF c=4 THEN db=1:GOTO 234Ø
286Ø 'brique
287Ø tabl(xb-1,yb-1)=Ø:LOCATE xb-1,yb-1:GOSUB 3Ø1Ø:
db=1:GOTO 234Ø
288Ø xb=xb-1:yb=yb-1
289Ø LOCATE xb,yb:PEN 2:PRINT bal$:RETURN
29ØØ '-----------------------------------
291Ø '   perte balle
292Ø nb=nb-1:GOSUB 453Ø
293Ø FOR j=1 TO 3ØØ:NEXT j
294Ø IF nb<>Ø THEN 217Ø
295Ø GOSUB 325Ø
296Ø GOSUB 16ØØ
297Ø GOSUB 197Ø
298Ø RETURN
299Ø '  brique detruite
3ØØØ 'points=(nb briques tableau-nb balles perdues)
*num tableau
3Ø1Ø PEN Ø:PRINT CHR$(143)
3Ø2Ø sco=sco+(nbro-(nbo-nb))*nt
3Ø3Ø GOSUB 4Ø1Ø:nbr=nbr-1
3Ø4Ø IF nbr<>Ø THEN RETURN
3Ø5Ø '  fin tableau
3Ø6Ø IF nbo<>nb THEN 3Ø7Ø
3Ø61 LOCATE 7,12:PEN 12:PRINT"BONUS";nbro*1ØØ*nt;
3Ø62 sco=sco+(nbro*1ØØ*nt):GOSUB 4Ø1Ø
3Ø63 FOR j=1 TO 6ØØ:NEXT
3Ø64 LOCATE 7,12:PEN Ø:PRINT STRING$(12,CHR$(143));
3Ø7Ø GOSUB 4Ø6Ø:GOTO 217Ø
3Ø8Ø '-----------------------------
3Ø9Ø '    gestion raquette
31ØØ '-----------------------------
311Ø 'joy=8   a droite
312Ø IF xr>16 THEN RETURN
313Ø LOCATE xr,yr:PEN Ø:PRINT raqz$;:
LOCATE xr+2,yr:PEN 1Ø:PRINT raq$;
314Ø xr=xr+2
315Ø tabl(xr-1,yr)=6:tabl(xr-2,yr)=6:
tabl(xr+1,yr)=3:tabl(xr+2,yr)=3:RETURN
316Ø '
317Ø 'joy=4   a gauche
318Ø IF xr<3 THEN RETURN
319Ø LOCATE xr+1,yr:PEN Ø:PRINT raqz$;:
LOCATE xr-2,yr:PEN 1Ø:PRINT raq$;
32ØØ xr=xr-2
321Ø tabl(xr+3,yr)=6:tabl(xr+4,yr)=6:
tabl(xr,yr)=3:tabl(xr+1,yr)=3:RETURN
322Ø '-----------------------------
323Ø '    game over
324Ø '-----------------------------
325Ø CLS
326Ø LOCATE 6,12:PEN 5:PRINT"GAME OVER"
327Ø FOR j=1 TO 4ØØ:NEXT
328Ø RETURN
329Ø '*****************************
33ØØ '    sous-programmes
331Ø '*****************************
332Ø '-----------------------------
333Ø '   tirage aleatoire
334Ø '-----------------------------
335Ø RANDOMIZE TIME
336Ø nombre=INT(RND*(maxi-mini+1))+mini
337Ø RETURN
338Ø '-----------------------------
```

```
339Ø '   init des couleurs et briques
34ØØ '---------------------------
341Ø RESTORE 365Ø
342Ø FOR j=Ø TO 15
343Ø READ c:INK j,c
344Ø NEXT
345Ø MODE Ø:CLS
346Ø ' brique 7
347Ø SYMBOL 255,255,129,129,129,129,129,129,255
348Ø SYMBOL 254,Ø,126,66,66,66,66,126
349Ø SYMBOL 253,Ø,Ø,6Ø,36,36,6Ø
35ØØ SYMBOL 252,Ø,Ø,Ø,24,24
351Ø ' brique 8
352Ø SYMBOL 251,255,128,128,128,128,128,128
353Ø SYMBOL 25Ø,Ø,1,1,1,1,1,127
354Ø SYMBOL 249,Ø,126,6Ø,24,24,6Ø,126
355Ø SYMBOL 248,Ø,Ø,66,1Ø2,1Ø2,66
356Ø ' brique 9
357Ø SYMBOL 247,255,126,6Ø,24
358Ø SYMBOL 246,Ø,Ø,Ø,Ø,24,6Ø,126,255
359Ø SYMBOL 245,Ø,128,192,224,224,192,128
36ØØ SYMBOL 244,Ø,1,3,7,7,3,1
361Ø RETURN
362Ø '
363Ø 'data des couleurs
364Ø 'gris
365Ø DATA 13
366Ø 'rouges
367Ø DATA 6,7
368Ø 'bleus
369Ø DATA 2,14,2Ø
37ØØ 'verts
371Ø DATA 9,1Ø,18,21,22
372Ø 'jaunes
373Ø DATA 15,25
374Ø 'noir
375Ø DATA Ø
376Ø 'marron
377Ø DATA 16
378Ø 'blanc
379Ø DATA 26
38ØØ '---------------------------
381Ø '  trace page jeu
382Ø '---------------------------
383Ø '  cadre
384Ø '
385Ø CLS
386Ø x=Ø:xr=638:yr=366:pr=4
387Ø FOR j=1 TO 8:GOSUB 39ØØ:NEXT
388Ø RETURN
389Ø '
39ØØ MOVE x,32
391Ø DRAWR Ø,yr,j
392Ø DRAWR xr,Ø,j
393Ø DRAWR Ø,-yr,j
394Ø x=x+pr
395Ø yr=yr-pr
396Ø xr=xr-pr*2
397Ø RETURN
398Ø '---------------------------
399Ø '   afficher score
4ØØØ '---------------------------
4Ø1Ø LOCATE 2,1:PEN 15:PRINT sco
4Ø2Ø RETURN
4Ø3Ø '---------------------------
4Ø4Ø '   init d'un tableau
4Ø5Ø '---------------------------
4Ø6Ø ' vider tabl
4Ø7Ø FOR x=1 TO 2Ø:FOR y=1 TO 25
4Ø8Ø tabl(x,y)=Ø
4Ø9Ø NEXT y:NEXT x
41ØØ ' mise a jour numero tableau
411Ø nt=nt+1:IF nt>maxt THEN nt=1:ad1=ad
412Ø ad1=ad+(nt-1)*152
413Ø PRINT CHR$(22)+CHR$(Ø);
414Ø LOCATE 12,1:PEN 1Ø:PRINT nt;
415Ø '
416Ø 'nombre balles
417Ø nb=PEEK(ad1):GOSUB 453Ø:nbo=nb
418Ø 'nombre briques
419Ø j=PEEK(ad1+1):nbr=j:nbro=nbr
42ØØ ad1=ad1+2
421Ø 'chargement de tabl et affichage
422Ø PRINT CHR$(22)+CHR$(1);
423Ø FOR k=1 TO j
424Ø x=PEEK(ad1)
425Ø y=PEEK(ad1+1)
426Ø typ=PEEK(ad1+2)
427Ø ad1=ad1+3
428Ø tabl(x,y)=typ
429Ø LOCATE x,y
43ØØ 'afficher brique
431Ø ON tabl(x,y)-6 GOSUB 445Ø,447Ø,449Ø
432Ø NEXT k
433Ø '
434Ø PRINT CHR$(22)+CHR$(Ø);
435Ø 'remplir le tableau codes 1,2,4,6
436Ø FOR x=1 TO 1:FOR y=1 TO 24:tabl(x,y)=4:
NEXT y:NEXT x
437Ø FOR x=2Ø TO 2Ø:FOR y=1 TO 24:tabl(x,y)=2:
NEXT y:NEXT x
438Ø FOR x=1 TO 2Ø:FOR y=25 TO 25:tabl(x,y)=6:
NEXT y:NEXT x
439Ø FOR x=1 TO 2Ø:FOR y=1 TO 2:tabl(x,y)=1:
NEXT y:NEXT x
44ØØ RETURN
441Ø '-----------------------
442Ø 'sous-prog affichage des briques
443Ø '-----------------------
444Ø 'afficher brique type 7
445Ø PEN 5:PRINT CHR$(255):
LOCATE x,y:PEN 7:PRINT CHR$(254):
PEN 12:LOCATE x,y:PRINT CHR$(253):
PEN 13:LOCATE x,y:PRINT CHR$(252):RETURN
446Ø 'afficher brique type 8
447Ø PEN 6:PRINT CHR$(251):
LOCATE x,y:PEN 5:PRINT CHR$(25Ø):
LOCATE x,y:PEN 2:PRINT CHR$(249):
LOCATE x,y:PEN 14:PRINT CHR$(248):RETURN
448Ø 'afficher brique type 9
449Ø PEN 1Ø:PRINT CHR$(247):
LOCATE x,y:PEN 12:PRINT CHR$(246):
LOCATE x,y:PEN 3:PRINT CHR$(245):
LOCATE x,y:PEN 14:PRINT CHR$(244):RETURN
45ØØ '-----------------------
451Ø 'sous-prog affiche nb balles en cours
452Ø '-----------------------
453Ø PRINT CHR$(22)+CHR$(Ø);:LOCATE 16,1:
PEN 5:PRINT nb;
454Ø RETURN
```

Un programme peut toujours s'améliorer par l'adjonction d'éléments supplémentaires comme, par exemple, les sons. Il nous reste donc à faire un tour d'horizon des plus à apporter et, tu as certainement des idées. Le plus important pour toi c'est d'avoir bien compris l'ensemble des séquences. Pour continuer à te perfectionner, je te prépare une autre initiation dans un genre totalement différent, je t'en dirai plus la semaine pro-

Bien joystiquement vôtre
François LE GRIGUER

LA SEMAINE PROCHAINE

Toutes les améliorations possibles de JOYSTICK-BREAKER avec, en particulier, les sons.
A NE PAS MANQUER : le début d'un programme d'initiation d'un genre nouveau

PAR FRANÇOIS LE GRIGUER

JOYSTICK-BREAKER

UN PEU DE BRUIT DANS LE BREAKER

SALUT A TOUS LES AVENTURIERS DU "BREAKER" RETROUVE. TES 4Ø TABLEAUX S'ENCHAINENT SANS PROBLEMES ET, TU AS MEME REUSSI A DECROCHER UN BONUS, FORMIDABLE!!! LE PROGRAMME TOURNE MAIS UN PEU DE BRUIT NOUS FERAIT DU BIEN. ET SI ON AJOUTAIT UNE NUANCE D'ASSEMBLEUR, VOUS AVEZ DIT ASSEMBLEUR?? ALORS ACCROCHE-TOI, CA DEVIENT SERIEUX, LE RYTHME S'ACCELERE, IL S'AGIT DE BIEN TENIR LE JOYSTICK (HEBDO)!!!

J'APPRENDS

Les sons accompagnent l'action et, dans notre jeu, sans trop compliquer ni trop ralentir le traitement, tu vas prévoir un bruit à chaque collision. Donc pour chaque rebondissement tu programmes une instruction SOUND. Cet ordre Basic, développé dans JOYSTICK-Hebdo numéro 5, contient 7 paramètres:

- le numéro de canal
- la fréquence
- la durée du son
- le volume initial
- le numéro d'enveloppe de volume
- le numéro d'enveloppe de fréquence
- la génération de bruit

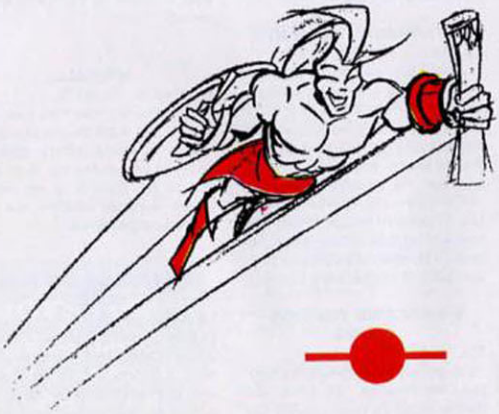
On aura l'occasion d'en reparler dans le détail, pour aujourd'hui, tu charges "BREAKER" et tu ajoutes la ligne 1485 ainsi que, tous les SOUND qui figurent dans le nouveau listing de la gestion de la balle. Pour obtenir des sons variés, la fréquence initiale est multipliée par la direction ou, par le code rencontré dans le fichier "tabl".

La ligne 1485 initialise les enveloppes de fréquence et de volume.

```
1485 ENT 1,3Ø,1Ø,2:ENV 1,1Ø,15,6


226Ø '----------------------------
227Ø 'gestion balle avec les instructions SOUND
228Ø '----------------------------

229Ø ' effacer balle
23ØØ LOCATE xb,yb:PEN Ø:PRINT bal$
231Ø ON db GOTO 234Ø,248Ø,263Ø,278Ø
232Ø '
233Ø '   direct 1
234Ø l=tabl(xb,yb-1)
235Ø IF l=Ø THEN 239Ø
236Ø IF l=1 THEN db=2:SOUND 129,5Ø*db,Ø,15,1,1:
GOTO 248Ø
237Ø 'brique
238Ø tabl(xb,yb-1)=Ø:LOCATE xb,yb-1:GOSUB 3Ø1Ø
2385 db=2:SOUND 129,5Ø*l*2,Ø,15,1,1:GOTO 248Ø
239Ø c=tabl(xb+1,yb-1)
24ØØ IF c=Ø THEN 244Ø
241Ø IF c=2 THEN db=4:SOUND 129,5Ø*db,Ø,15,1,1:
GOTO 278Ø
242Ø 'brique
243Ø tabl(xb+1,yb-1)=Ø:LOCATE xb+1,yb-1:GOSUB 3Ø1Ø
2435 db=4:SOUND 129,5Ø*c*2,Ø,15,1,1:GOTO 278Ø
244Ø xb=xb+1:yb=yb-1
245Ø LOCATE xb,yb:PEN 2:PRINT bal$:RETURN
246Ø '
247Ø '   direct 2
248Ø l=tabl(xb,yb+1)
249Ø IF l=Ø THEN 254Ø
25ØØ IF l=3 THEN db=1:SOUND 129,5Ø*db,Ø,15,1,1:
GOTO 234Ø
251Ø IF l=6 THEN SOUND 129,15ØØ,Ø,15,1,1:GOTO 292Ø
252Ø 'brique
253Ø tabl(xb,yb+1)=Ø:LOCATE xb,yb+1:GOSUB 3Ø1Ø
2535 db=1:SOUND 129,5Ø*l*2,Ø,15,1,1:GOTO 234Ø
254Ø c=tabl(xb+1,yb+1)
255Ø IF c=Ø THEN 259Ø
256Ø IF c=2 THEN db=3:SOUND 129,5Ø*db,Ø,15,1,1:
GOTO 263Ø
257Ø 'brique
258Ø tabl(xb+1,yb+1)=Ø:LOCATE xb+1,yb+1:GOSUB 3Ø1Ø
2585 db=3:SOUND 129,5Ø*db,Ø,15,1,1:GOTO 263Ø
259Ø xb=xb+1:yb=yb+1
26ØØ LOCATE xb,yb:PEN 2:PRINT bal$:RETURN
261Ø '
262Ø '   direct 3
263Ø l=tabl(xb,yb+1)
264Ø IF l=Ø THEN 269Ø
265Ø IF l=3 THEN db=4:SOUND 129,5Ø*db,Ø,15,1,1:
```



```
GOTO 278Ø
266Ø IF l=6 THEN SOUND 129,15ØØ,Ø,15,1,1:GOTO 292Ø
267Ø 'brique
268Ø tabl(xb,yb+1)=Ø:LOCATE xb,yb+1:GOSUB 3Ø1Ø
2685 db=4:SOUND 129,5Ø*l*2,Ø,15,1,1:GOTO 278Ø
269Ø c=tabl(xb-1,yb+1)
27ØØ IF c=Ø THEN 274Ø
271Ø IF c=4 THEN db=2:SOUND 129,5Ø*db,Ø,15,1,1:
GOTO 248Ø
272Ø 'brique
273Ø tabl(xb-1,yb+1)=Ø:LOCATE xb-1,yb+1:GOSUB 3Ø1Ø
2735 db=2:SOUND 129,5Ø*c*2,Ø,15,1,1:GOTO 248Ø
274Ø xb=xb-1:yb=yb+1
275Ø LOCATE xb,yb:PEN 2:PRINT bal$:RETURN
276Ø '
277Ø '  direct 4
278Ø l=tabl(xb,yb-1)
279Ø IF l=Ø THEN 283Ø
28ØØ IF l=1 THEN db=3:SOUND 129,5Ø*db,Ø,15,1,1:
GOTO 263Ø
281Ø 'brique
282Ø tabl(xb,yb-1)=Ø:LOCATE xb,yb-1:GOSUB 3Ø1Ø
2825 db=3:SOUND 129,5Ø*l*2,Ø,15,1,1:GOTO 263Ø
283Ø c=tabl(xb-1,yb-1)
284Ø IF c=Ø THEN 288Ø
285Ø IF c=4 THEN db=1:SOUND 129,5Ø*db,Ø,15,1,1:
GOTO 234Ø
286Ø 'brique
287Ø tabl(xb-1,yb-1)=Ø:LOCATE xb-1,yb-1:GOSUB 3Ø1Ø
2875 db=1:SOUND 129,5Ø*c*2,Ø,15,1,1:GOTO 234Ø
288Ø xb=xb-1:yb=yb-1
289Ø LOCATE xb,yb:PEN 2:PRINT bal$:RETURN
29ØØ '-----------------------------------
```

Si tu veux changer la fréquence, modifie la valeur 5Ø à l'intérieur de l'instruction SOUND. Tu sauvegardes ton programme et tu lances l'exécution. Le volume sonore au maxi, ton casse-briques vient de se transformer, et tu vas, certainement, pouvoir battre tous les records.
Après les sons, tu peux encore ajouter plus de tableaux en changeant le test du numéro limite en redéfinition et, en augmentant la capacité prévue pour le fichier "Tableaux". Pour entrer d'autres types de briques tu dois ajouter des tests supplémentaires en redéfinition à la saisie d'une brique et, compléter les lignes ON GOSUB vers les lignes des nouveaux sous-programmes d'affichage.

LA RAQUETTE EN ASSEMBLEUR

Un programme écrit en Basic atteint très vite les limites du temps d'exécution, il devient impossible de jouer, l'animation rame littéralement. Une seule orientation possible: écrire le programme entièrement en Assembleur ou, déjà plus accessible pour les amateurs, remplacer certaines séquences Basic par de l'Assembleur. Pour une première approche j'ai choisi une partie très courte de JOYSTICK-BREAKER: la gestion de la raquette avec l'effacement, la mise à jour de xr et, l'affichage aux nouvelles coordonnées. Tu ne dois pas t'attendre à un changement des temps de traitement car la séquence est beaucoup trop limitée. La plus grande perte de temps, dans ce programme, se situe au niveau de la gestion de la balle, avec tous les tests de collisions, mais son écriture en assembleur reste un peu longue pour un début. Donc considère cette opération, non comme une amélioration de ton jeu, mais comme un exercice. Voici la nouvelle séquence de la gestion de la raquette, en Basic, avec tous les commentaires en italique, pour t'expliquer les différences:

```
2Ø4Ø xr=nombre:yr=25:POKE 39547,xr:
POKE 39549,yr
```

En initialisation de la raquette les valeurs d'origine des coodonnées, xr et yr, sont transmises au sous-programme assembleur.

```
3Ø8Ø '-----------------------------
3Ø9Ø '    gestion raquette
31ØØ '-----------------------------
311Ø 'joy=8   a droite
312Ø IF xr>16 THEN RETURN
313Ø CALL 395ØØ:xr=xr+2
```

L'instruction CALL fait appel au sous-programme assembleur commençant à l'adresse 395ØØ (en décimal) pour l'effacement de la raquette et l'affichage. Ensuite xr est mis à jour pour les tests en Basic. La ligne 314Ø n'existe plus.

```
315Ø tabl(xr-1,yr)=6:tabl(xr-2,yr)=6:
tabl(xr+1,yr)=3:tabl(xr+2,yr)=3:RETURN
316Ø '
317Ø 'joy=4   a gauche
318Ø IF xr<3 THEN RETURN
319Ø CALL 3951Ø:xr=xr-2
```

Même opération pour le déplacement à gauche avec l'adresse de début du sous-programme en 3951Ø. La ligne 32ØØ est supprimée.

```
321Ø tabl(xr+3,yr)=6:tabl(xr+4,yr)=6:
tabl(xr,yr)=3:tabl(xr+1,yr)=3:RETURN
322Ø '-----------------------------
```

Jusqu'ici tu me suis sans problème, il nous reste à découvrir de quoi est faite cette séquence en assembleur, pas de longues explications mais le listing avec, comme tu en as l'habitude, des commentaires toujours en italique. Ce listing s'appelle la SOURCE, c'est ce qu'écrit le programmeur. Les lignes très courtes, portent un numéro, comme dans ton Basic. Si tu vois de l'assembleur pour la première fois tu vas, sans doute, avoir un peu de mal à suivre, rassure-toi, le contraire serait surprenant.

```
1Ø ;SEQUENCE EFFACE/AFFICHE RAQUETTE
```

Cette ligne de commentaire dans le listing débute par ;

```
2Ø ;
3Ø ORG 395ØØ
```

Cette instruction déclare l'adresse mémoire de début pour l'assemblage du programme. Afin de simplifier la lecture toutes les adresses sont en décimal.

```
4Ø ;AFFICHER LA RAQUETTE
5Ø XP2:
```

Point d'entrée à partir du Basic (CALL 395ØØ) pour le déplacement à droite.

```
6Ø CALL EFFACE
```

branchement dans le sous-programme EFFACE

```
7Ø LD A,(XR)
```

L'ordre LD, abréviation de LOAD, charge dans le régistre A le contenu de l'emplacement mémoire XR.

```
8Ø ADD A,2
```

Mise à jour avec l'addition de 2 (comme en Basic).

```
9Ø JR SUITE
```

Branchement (ou saut, J comme JUMP) en SUITE.

```
1ØØ XM2:
```

Point d'entrée à partir du Basic (CALL 3951Ø) pour le déplacement à gauche.

```
11Ø CALL EFFACE
12Ø LD A,(XR)
13Ø SUB 2
```

Mêmes opérations que précédemment sauf la mise à jour (-2).

```
14Ø SUITE:
```

Point de branchement SUITE.

```
15Ø LD (XR),A
```

Transfert de XR, à jour, à son adresse.

```
16Ø LD A,1Ø
17Ø CALL #BB9Ø
```

Utilisation d'une routine système, à l'adresse hexadécimale (#)

BB9Ø, qui correspond à PEN en Basic, le numéro d'INK est en A (1Ø).

```
18Ø LD A,2Ø8
19Ø LD (CARACT),A
```

Chargement en A du numéro de caractère à utiliser pour afficher la raquette et transfert en CARACT.

```
2ØØ JR PROG
```

Branchement en PROG.

```
21Ø ;EFFACER LA RAQUETTE
22Ø EFFACE:
```

Point de branchement EFFACE.

```
23Ø CALL #BD19
```

Routine système pour attendre la fin du signal vidéo.

```
24Ø LD A,Ø
25Ø CALL #BB9Ø
```

Déclarer PEN avec le numéro d'INK Ø.

```
26Ø LD A,143
27Ø LD (CARACT),A
```

Le caractère 143 pour effacer.

```
28Ø PROG:
29Ø LD H,Ø
```

Chargement en H de la valeur de XR.

```
3ØØ XR:EQU $-1
```

Déclaration de l'adresse de XR, correspond à 39547. Le programme Basic exécute un POKE 39547,xr en ligne 2Ø4Ø.

```
31Ø LD L,Ø
32Ø YR:EQU $-1
```

Chargement en L de la valeur de YR et déclaration de l'adresse qui correspond à 39549 (POKE 39549,yr en ligne 2Ø4Ø du programme Basic).

```
33Ø CALL #BB75
```

Routine système qui positionne le début de l'affichage des caractères, en fontion du contenu de H (x) et de L (y).

```
34Ø LD A,Ø
35Ø CARACT:EQU $-1
```

Chargement en A du numéro de caractère à afficher et déclaration de l'adressse CARACT.

```
36Ø LD B,3
```

Chargement en B du nombre de caractères à afficher, nombre de passages à effectuer dans la boucle de programme.

```
37Ø RETOUR:
```

Point de branchement RETOUR.

```
38Ø PUSH AF
39Ø PUSH BC
```

Sauvegarde du contenu des régistres AF et BC (pour les initiés: empiler AF et BC).

```
4ØØ CALL #BB5D
```

Routine système pour l'affichage du caractère dont le numéro se trouve en A.

```
41Ø POP BC
42Ø POP AF
```

Reprendre les précédents contenus de BC et AF. Ils sont modifiés par l'exécution de la routine (pour les initiés: dépiler BC et AF)

```
43Ø DJNZ RETOUR
```

Cette instruction de fin de boucle ressemble au NEXT en Basic. Elle soustrait 1 au contenu du régistre B. Si B > Ø branchement en RETOUR, sinon suite du traitement à la ligne suivante en 44Ø.

```
44Ø RET
```

Instruction de fin de sous-programme. Abréviation du RETURN que tu connais en Basic.

La fumée sort par les oreilles, un nouveau grand coup d'éponge sur le front, tu trembles un peu et, tu hésites entre appeler le SAMU ou hurler par la fenêtre, alors tu respires lentement à fond et, je t'explique les A, H, L et autres symboles éparpillés dans ce listing. Le microprocesseur Z8Ø de ton AMSTRAD CPC gère les régistres AF, BC, HL et DE (il y en a d'autres) et, ces mémoires sont utilisées directement par leur symbole. Un régistre simple, comme A, est un régistre 8 bits, tandis qu'un régistre double, comme BC, est un régistre 16 bits. Sans aller plus loin dans les explications, le listing SOURCE, que tu viens de voir, est assemblé pour générer le programme OBJET que l'ordinateur va pouvoir exécuter. Il s'agit alors d'une suite de codes numériques à partir de l'adresse mémoire 395ØØ, pour notre routine. Chaque instruction LD, ADD, SUB etc... est remplacée par son code opération et, les étiquettes comme EFFACE, PROG ou RETOUR sont transformées en adresse. Les lignes de DATA qui suivent contiennent tous les codes (en décimal pour plus de clarté) générés par l'assemblage du listing SOURCE que l'on vient de voir.

```
1481 DATA 2Ø5,1Ø9,154,58,123,154,198,2,24,8,2Ø5,
1Ø9,154,58,123,154,214,2,5Ø,123
1482 DATA 154,62,1Ø,2Ø5,144,187,62,2Ø8,5Ø,13Ø,154,
24,13,2Ø5,25,189,62,Ø,2Ø5,144
1483 DATA 187,62,143,5Ø,13Ø,154,38,Ø,46,Ø,2Ø5,117,
187,62,Ø,6,3,245,197,2Ø5
1484 DATA 93,187,193,241,16,247,2Ø1
1486 RESTORE 1481:FOR j=395ØØ TO 39566:
READ a:POKE j,a:NEXT
```

Tu ajoutes dans ton programme "BREAKER" ces lignes, 1481, 1482, 1483, 1484 et 1486 en faisant attention aux coupures nécessaires à notre mise en page. Toutes les DATA, transférées à partir de l'adresse mémoire 395ØØ par la boucle de programme de la ligne 1486, forment le programme OBJET de notre listing assembleur. C'est la seule manière possible sauf si tu disposes d'un programme Assembleur qui te permet, alors, de saisir le listing SOURCE, de l'assembler et, d'enregistrer les codes OBJET , il te suffit, dans ce cas, de mettre en Basic: LOAD"nom de l'enregistrement",395ØØ.

THE END

Le vaisseau spatial passe une dernière fois en gros plan, entouré des flammes de ses formidables propulseurs, poussé par une force extraordinaire, il disparait au fond de l'écran, comme dans les meilleurs films de S.F. Il vient de vaincre tous les obstacles, la mission sur la planète BREAKER s'achève. Sur les moniteurs vidéo du bord, le casse-briques fonctionne parfaitement bien, la première routine en assembleur ne pose aucun problème majeur et, tout est en ordre pour la prochaine étape du voyage. Si tu rencontres la moindre difficulté, dans les jours qui viennent, tu m'envoies immédiatement un message.