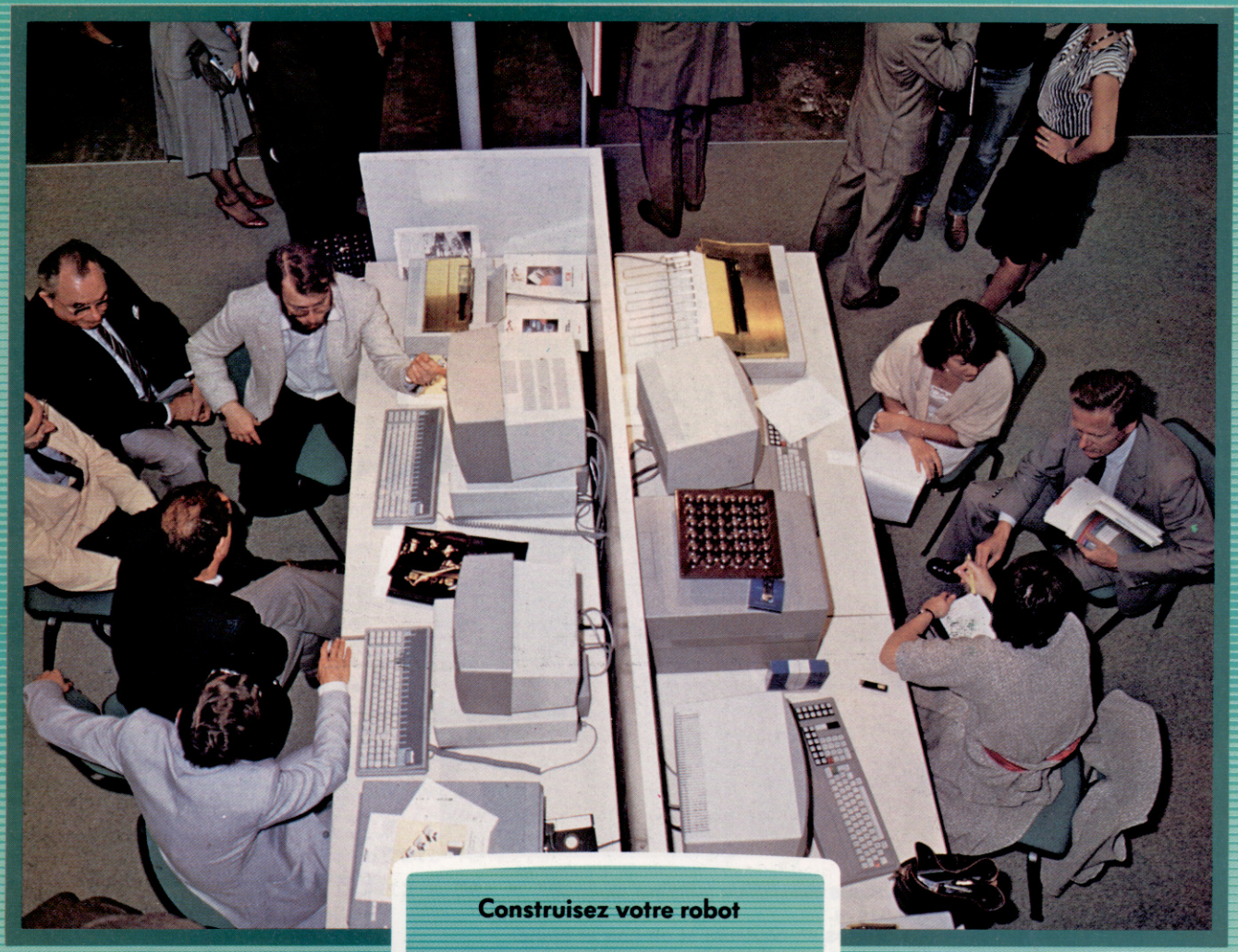
ABC INFORMATIQUE

N°65

COURS D'INFORMATIQUE PRATIQUE ET FAMILIALE



Construisez votre robot

Le tableur TK ! Solver

Le Spectrum +

Jouez avec le MO5

EDITIONS ATLAS

M 6062-65-12,50 F 90 FB - 3,80 FS - $ 1.95

Copie conforme

Nous avons étudié les méthodes à mettre en œuvre pour que les robots se comportent intelligemment. A présent, regardons comment simuler le comportement d'un robot par ordinateur.

Des « modèles » informatiques peuvent être élaborés pour reproduire fidèlement les événements du monde réel. Le grand public est déjà familier des simulateurs de vol, mécanismes extrêmement complexes permettant à des futurs pilotes d'acquérir de l'expérience. D'autres activités humaines peuvent utiliser la simulation par ordinateur : prévisions en matière de gestion, travaux d'ingénierie et processus physiques de toutes sortes, etc. Dans certains cas, le modèle informatique peut mener à bien des expériences qui seraient trop dangereuses dans la réalité. Il est essentiel, par exemple, de chercher à savoir ce qui se passerait en cas de fuite du liquide réfrigérant d'un réacteur d'une centrale nucléaire. Si le modèle mis au point à cette occasion est suffisamment détaillé, il est alors possible de savoir ce qui se produirait dans la réalité.

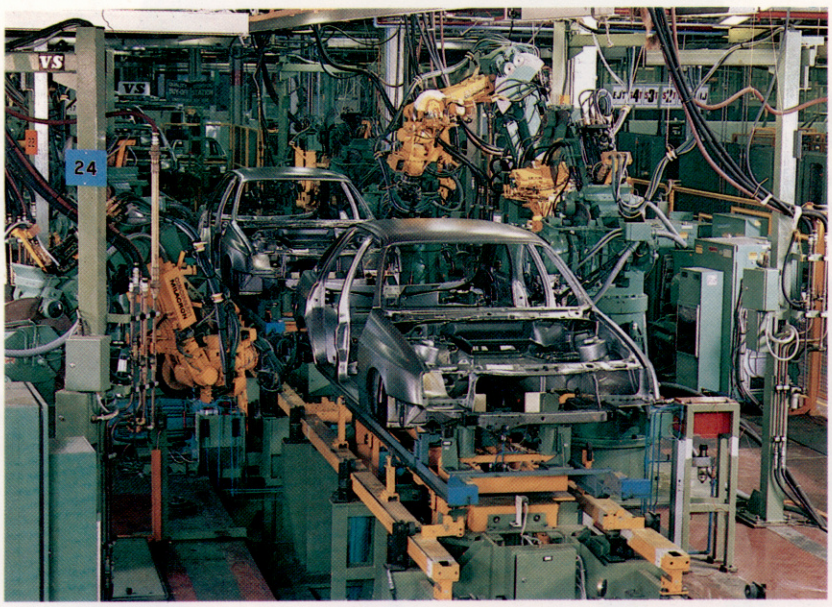
De manière similaire, la robotique peut utiliser la simulation informatique pour la conception de nouveaux robots. Il est bien sûr possible de procéder par essais en construisant un robot et en étudiant son comportement, puis en lui apportant les modifications nécessaires. Mais cela prend beaucoup de temps et revient très cher. La simulation permet de mettre au point un type de robot et de simuler les opérations sans avoir à construire de nombreux prototypes.

Prenons l'exemple d'une chaîne de montage automobile sur laquelle travaille une équipe de robots. Il vous faut programmer les robots afin qu'ils assemblent les véhicules de manière satis-

Vivre ensemble
Quand des robots sont attelés à une même tâche, il est nécessaire d'adapter leurs mouvements pour qu'ils ne se heurtent pas. Ici, un bras de robot doit prendre et tenir un jouet dans une position pendant que l'autre fixe un tambour. Puis le premier robot place le jouet entier dans son emballage. Si tout est bien réglé, le convoyeur peut être placé dans la position la plus efficace pour l'opération.
(Cl. Steve Cross.)

faisante. La programmation des robots prend néanmoins du temps et il faut savoir que chaque arrêt de la chaîne fait perdre de l'argent. Vous pouvez alors décider de mettre en place une chaîne fictive avec des robots tout neufs pour développer les nouveaux programmes. Cela s'avère également coûteux et peut susciter de nouveaux problèmes relatifs à la « chorégraphie » des robots, sujet que nous avons déjà abordé. Il est vital de s'assurer que les robots qui travaillent ensemble ne se gênent pas. Mais les détériorations éventuelles ne se limiteraient pas aux seuls robots. Imaginez le cas d'un robot déréglé dans sa programmation qui s'obstinerait à souder des portières de voitures en les verrouillant !

La réponse évidente est de concevoir la simulation pour chaque robot de sorte que l'utilisateur puisse vérifier l'interaction générale entre les



Sur la ligne
La complexité et la synchronisation des mouvements, demandées dans les applications concrètes de la robotique, apparaissent clairement sur cette photo montrant une chaîne d'assemblage automobile.
(Cl. Ford.)

robots. Le coût résultant devient ainsi faible et aucun dégât ne peut avoir lieu. Une fois la simulation terminée et satisfaisante sur l'ensemble, les programmes peuvent être transmis aux vrais robots en toute sécurité.

Nous exposons ici le principe de la simulation informatique par un programme, BRAS DE ROBOT, qui simule un bras de robot de type « prendre et positionner », doté de deux degrés de liberté. Il ne comporte pas de capteurs ; vous devez donc le guider vous-même en dirigeant les mouvements des articulations (épaule et coude), et en contrôlant le mécanisme de préhension terminal (la main) pour prendre et déplacer un objet. Reportez-vous également au programme du labyrinthe qui montre comment un robot peut être programmé pour chercher son chemin jusqu'au centre. Il s'agit en réalité d'une simulation par ordinateur visant à trouver une méthode pour guider un robot vers son but. Le programme reproduit le comportement d'un tel engin doté seulement d'un capteur tactile. Le chemin approprié est découvert en avançant là où il y a de la place jusqu'à ce que l'objet simulé (le robot) vienne heurter un obstacle ; le robot doit alors revenir au dernier embranchement rencontré pour essayer une autre route. Ce modèle est loin d'être très sophistiqué mais il illustre comment un programme peut traduire une simulation. Le « robot » du programme obéit à un ensemble déterminé de règles et procède par éliminations pour se repérer dans son environnement. Si le robot figuré dans le programme pouvait accéder directement et immédiatement à toutes les positions du labyrinthe, il pourrait atteindre directement son but ; mais il ne dispose pas de cette information et doit procéder par essais et erreurs.

De la même manière, le programme BRAS DE ROBOT simule le comportement d'un robot totalement dépourvu de capteurs. Le programme comporte un modèle de l'environnement du robot et un modèle du bras lui-même. Vous devez vous assurer que les deux modèles ont le même type d'interactions que dans la réalité. Ainsi, vous ne pouvez prendre un objet avec le bras si ce dernier n'est pas correctement positionné. De même, vous ne pouvez amener le bras simulé en dessous du niveau du sol, puisque le bras réel ne le pourrait pas ! Bien que nous utilisions le graphisme propre à l'informatique, qui autorise le croisement des deux lignes (celle représentant le bras et celle figurant le sol), une simulation fidèle l'interdit formellement. En outre, lorsque le robot lâche un objet, ce dernier ne doit pas rester dans sa position initiale, il doit tomber comme dans la réalité. Votre modèle doit permettre des effets gravitationnels. Faute de quoi, vous auriez des difficultés à développer un modèle informatique de simulation d'un robot de type « prendre et poser » destiné à manipuler des œufs !

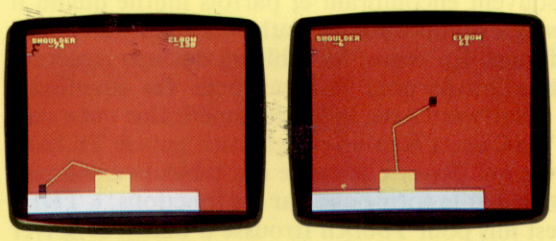
Ajouter du réalisme

La simulation ne connaît pratiquement aucune limite quant à son application. Plus cette dernière est complexe, plus la démarche d'écriture du programme est fascinante. L'intérêt peut être même plus grand que dans le cas d'une simple manipulation de robots réels. La simulation vous permet en effet de concevoir tous les robots que vous désirez. La programmation détaillée des robots et celle de leur environnement peuvent conduire à une meilleure compréhension du monde physique où ils évoluent. Prenons à nouveau l'exemple du programme BRAS DE ROBOT : vous vous apercevez qu'un objet lâché tombe au sol et cesse de bouger. Pour rendre le modèle encore plus réaliste, le programme pourrait être modifié de sorte que la chute de l'objet s'accélère en se rapprochant du sol, suivant ainsi les lois de la gravité. Il pourrait peut-être aussi rebondir en touchant le sol. Les possibilités sont très nombreuses et le programme que nous donnons ici est destiné à ce que vous l'adaptiez. En ajoutant des caractéristiques plus réalistes et mieux observées, la simulation obtenue se rapprochera davantage de la réalité.

La conception de modèles informatiques peut être très similaire à la création de jeux sur ordinateurs. La différence essentielle étant que la simulation doit représenter le monde réel.

Prendre, déplacer et poser

Ce programme simule un bras de robot capable d'atteindre, de prendre, de déplacer et de reposer des objets. Votre tâche consiste simplement à saisir une balle et à la lâcher.
Le bras doit utiliser des coordonnées de révolution à deux degrés de liberté : une articulation de type « épaule » et une autre de type « coude ».
L'articulation de l'épaule doit pouvoir tourner selon un angle de 180°, et celle du coude, de 360° c'est-à-dire faire un tour sur elle-même.
Les touches suivantes contrôlent le programme : E demande un mouvement pour l'épaule, S et O demandent respectivement de faire pivoter l'articulation du bras dans le même sens que les aiguilles d'une montre ou en sens inverse. Chaque frappe de ces touches entraîne une rotation de 6° pour l'épaule et de 12° pour le coude. P signifie que le bras doit prendre la balle. L est destinée à faire lâcher la balle par le bras.



Depuis l'épaule
Notre programme de simulation de bras de robot vous permet de déplacer un bras à deux articulations selon deux directions, et de lui faire prendre un objet. Lorsque l'objet est lâché, le programme le repositionne de manière aléatoire. La photo d'écran montre les angles verticaux dessinés par les bras supérieurs et inférieurs.
(Cl. Ian McKinnell.)

```
   4 REM ******SPECTRUM***************
   5 REM * SIMULATION BRAS DE ROBOT *
   6 REM ******SPECTRUM***************
  10 CLS : PRINT BRAS DE ROBOT : PRINT : PRINT "LES
COMMANDES DU BRAS DE ROBOT SONT :
  15 E - SELECTIONNER ROTATION EPAULE ": PRINT
C - SELECTIONNER ROTATION COUPE": PRINT
S - FAIRE TOURNER L'ARTICULATION DANS LE SENS DES AIGUILLES
D'UNE MONTRE
  20 PRINT "O - FAIRE TOURNER L'ARTICULATION DANS LE
SENS OPPOSE A CELUI DES AIGUILLES ": PRINT
I P - PRENDRE LA BALLE": PRINT "L - LACHER LA BALLE":
  25 PRINT AT 20,11: FLASH 1:"PRESSER UNE TOUCHE": PAUSE
0:RANDOMIZE
1000 GOSUB 9500:    REM init
1100 GOSUB 5000:    REM saisie
1200 GOSUB 6000:    REM écrasement
1400 STOP
1500 REM *MISE A JOUR ARTICULATION xy ***************
1550 LET ex=11*SIN hd1: LET ey=11*COS hd1
1560 LET hx=sx+ex: LET hy=sy+ey
1650 LET wx=12*SIN hd2: LET wy=12*COS hd2
1660 LET hx=hx+wx: LET hy=hy+wy
1690 RETURN
2000 REM *   dessinebras *************
2020 INK coulbras
2050 PLOT  INVERSE effacer!sx,sy
2100 DRAW  INVERSE effacer!ex!ey: DRAW  INVERSE
effacer!wx,wy: IF balleievée THEN  LET br=FN r(hy): LET
bc=FN c(hx): GOSUB 2500
2490 RETURN
2500 REM *  dessineballe *************
2600 INK coulballe
2650 PRINT AT br,bc!"  "(effacement+1)
2740 RETURN
2750 REM *  balleiachée *************
2800 LET effacement=1: GOSUB 2000: LET effacement=0
2820 LET k=INT (xh*RND): IF k)=xs THEN  IF k<=xs
+wd THEN  GOTO 2820
2850 LET br=FN r(y0+4): LET bc=FN c(K): LET balleievée
=0: GOSUB 2000: GOSUB 2500
2990 RETURN
3000 REM *  rotation ***************
3100 LET effacement=1: GOSUB 2000
3120 LET t1=dirn*sr*a1: LET t2=t1+dirn*era2
3150 LET hd1=hd1+t1: LET hd2=hd2+t2
3200 GOSUB 1500
3300 IF ABS hd1)p2 THEN  LET ok=0
3320 LET pt=POINT (hx,hy)
3340 IF pt<)0 THEN  LET ok=0: IF br=FN r(hy) AND
 bc=FN c(hx) THEN  LET ok=2
3400 LET effacement=0: GOSUB 2000
3450 INK coulbase: PRINT AT 21,0!s$!AT 21,21!FN d(hd
1)!AT 21,26!FN d(hd2)
3490 RETURN
5000 REM *  saisie ************
5100 IF INKEY$<)"" THEN GOTO 5100
5120 FOR i=0 TO 1 STEP 0
5150 LET a$=INKEY$: IF A$)="A" AND a$<= "Z" THEN
  LET a$=CHR$ (CODE a$+32)
5200 IF a$="e" THEN  LET sr=1: LET er=0
5220 IF a$="c" THEN  LET er=1: LET sr=0
5250 IF a$="s" THEN  LET dirn=1: GOSUB 3000
5270 IF a$="o" THEN  LET dirn=-1: GOSUB 3000
5300 IF a$="p" THEN  IF ok=2 THEN  LET biup=1
5320 IF a$="f" THEN IF balleievée THEN GOSUB 2750
5400 IF NOT ok THEN  LET i=2
5450 NEXT i
5490 RETURN
6000 REM *  écrasement ************
6100 PRINT AT 8,12: FLASH 1:"!!écrasement!": BEEP .5
,-5: BEEP 1,-16: RETURN
9000 REM *  dessinesol ************
9050 PAPER coulpap: CLS
9100 INK coulverte
9120 FOR k=0 TO y0: PLOT 0,k: DRAW xh,0: NEXT k
9200 INK baco!: LET xs=(xh-wd)/2
9220 FOR k=y0+1 TO y0+ht
9240 PLOT xs,k: DRAW wd,0
9260 NEXT k
9300 INK coulbal: GOSUB 2500: INK coulbras
9400 PRINT AT 20,1:"EPAULE"!AT 20,26:"COUDE"
9490 RETURN
9500 REM *  init ********************
9550 DEF FN d(x)=INT (x*180/PI)
9560 DEF FN r(x)=21-INT (x/8)
9570 DEF FN c(x)=INT (x/8)
9600 DIM s$(32): LET x1=0: LET y1=0: LET xh=254:
 LET yh=174
9620 LET y0=23: LET larg=60: LET haut=23
9630 LET balleievée=0: LET bc=2: LET br=FN r(y0+4)
9640 LET coulverte=3: LET coulbase=2: LET coulbras=1 : LET b
col=6: LET paco!=7
9650 LET sx=xh/2: LET sy=y0+ht+2: LET 11=(yh-ht-
y0-2)/2: IF 11)xh/4 THEN  LET 11=xh/4
9660 LET 12=11: LET hx=0: LET hy=0
9670 LET p2=PI/2: LET a1=PI/32: LET a2=2*a1
9680 LET hd1=0: LET hd2=p2
9690 LET sr=1: LET er=0: LET dirn=1: LET effacement=
0: LET ok=1
9750 GOSUB 9000: GOSUB 1500: GOSUB 2000
9790 RETURN
```

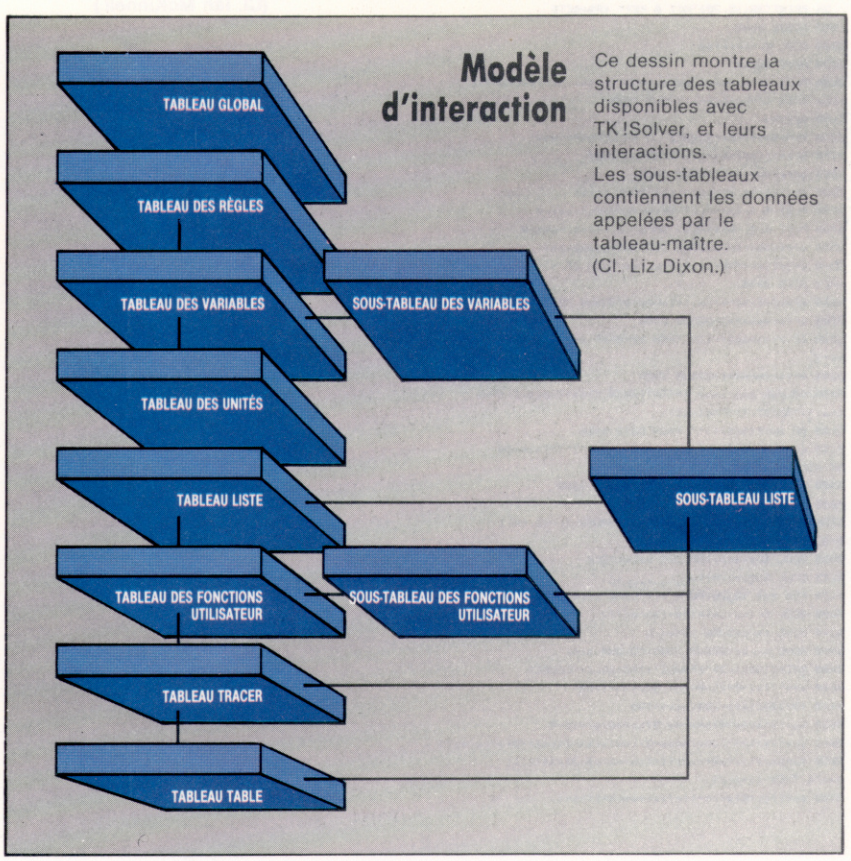
```
   4 REM*************BBC*************
   5 REM* SIMULATION BRAS DE ROBOT *
   6 REM*************BBC*************
   7 MODE 1:COLOUR 130:COLOUR 1:CLS
  25 PRINTTAB(15,20)"HIT A KEY" :A$=GET$
1000 GOSUB 9600
2020 GCOL 0,coulbras
2050 MOVE sx,sy
2100 PLOT effacement,ex,ey:PLOT effacement,wx,wy:IF
biup THEN br=hy!bc=hx!GOSUB 2500
2190 RETURN
2200 REM*********PRENDRE LA BALLE*********
2250 biup=1!effacement=3!GOSUB 2500
2300 effacement=1!GOSUB 2000
2600 coulverte 0,coulbleue!MOVE bc,br
2650 PLOT 0,0,bsz!PLOT 80+effacement,bsz,0
2700 PLOT 0,0,-bsz!PLOT 80+effacement,-bsz,0
2800 effacement=3!GOSUB 2000!effacement=1
2820 k=INT(xh*RND(1))!IF)k =xs THEN IF k<=xs+wd
THEN GOTO 2820
2850 br=y0+5!bc=k!balleievée=0!GOSUB 2000!GOSUB 2500
3100 effacement=3!GOSUB 2000
3340 IF pt<)paco!-128 THEN ok=0!IF pt=coulblanc THEN
ok=2
3400 effacement=1!GOSUB 2000
3450 COLOUR coulbase :PRINTTAB(0,3)ss!TAB(4,3)!FNd
(hd1)!TAB(27,3)!FNd(hd2)
5100 IF INKEY$(0)<)"" THEN GOTO 5100
5150 as=INKEY$(0)!IF a$)="A"ANDa$<="Z"THENa$=CH
R$(ASC(a$)+32)
5300 IF a$="p" AND ok=2 THEN GOSUB 2200
5400 IF ok=0 THEN i=2
6100 PRINTTAB(12,3)"!!CRASH!!"!SOUND 1,-15,48,1
0!SOUND 1,-15,4,20!RETURN
9050 GCOL 0,paco!!COLOUR paco!!CLS
9100 GCOL 0,coulverte
9120 FOR k=0 TO y0!MOVE 0,k!DRAW xh,k!NEXT k
9200 GCOL 0,coulbase!xs=(xh-wd)/2
9240 MOVE xs,k!DRAW xs+wd,k
9300 MOVE bc,br!GOSUB 2500!COLOUR acol
9400 PRINTTAB(1,2)"EPAULE"!TAB(26,2)"COUDE"
9600 s$=""!x1=0!y1=0!xh=1000!yh=1000
9620 y0=100!larg=200!haut=100
9630 balleievée=0!bsz=wd/5!bc=40!br=y0+5
9640 coulverte=3!coulbase=2!coulbras=2!bcol=0!p-col=129
9650 sx=xh/2!sy=y0+ht2111=(yh-ht-y0-2)/2!IF 11
)xh/4 THEN 11=xh/4
9690 sr=1!er=0!dirn=1!effacement=1!ok=1
```

La clé du problème

Nos articles sur les tableurs se poursuivent par une étude sur TK!Solver, programme de gestion de modèles mis au point par les créateurs de VisiCalc.

Nous avons vu que les tableurs sur micro-ordinateurs peuvent être très utiles pour toutes sortes de tâches mathématiques. Pour ceux qui sont habitués à travailler sur des documents très grands avec simplement un crayon et une calculatrice, le tableur électronique représente un gain de temps et d'énergie considérable. Cependant, les tableurs possèdent eux aussi des limites notables. Le format ligne/colonne, idéal pour les applications de comptabilité ou financières, est souvent encombrant et parfois inutile quand il s'agit d'applications mathématiques ou scientifiques. En outre, les tableurs ont une structure très lourde pour le maniement des équations.

Software Arts, la société américaine qui a créé VisiCalc, a écrit un programme appelé TK!Solver qui dépasse le cadre des tableurs classiques tant par la forme que par ses fonctions. « TK! » est mis pour *ToolKit* (trousse d'outils) et Solver fait référence à la partie du code qui traite les équations. Non seulement TK! diffère par son format d'écran mais il comporte les caractéristiques exclusives suivantes :



Résolution — Les formules mathématiques des tableurs habituels ne peuvent traiter que les équations à une seule variable.

Itération — S'il manque une valeur nécessaire à la résolution d'une équation, vous pouvez être sûr que TK! en fera son point de départ pour la résolution de l'équation. Cette dernière est alors résolue par une suite d'approximations.

Conversion d'unités — TK! peut convertir des *feet* en mètres; des dollars en francs, etc., de manière instantanée à partir de tables de conversion.

Fonctions mathématiques — TK! comporte de nombreuses fonctions mathématiques incorporées.

Les tableaux de TK!Solver

TK! fonctionne sur la base de trois tableaux interconnectés ayant chacun une fonction spécifique. Le Tableau des variables contient le nom de toutes les variables définies; des colonnes pour les valeurs entrées par l'utilisateur et pour les valeurs produites par le programme; un emplacement pour mentionner les unités retenues et de la place pour annoter chaque variable. Le Tableau des variables figure en haut du premier écran à s'afficher. Les variables sont également décrites en détail dans un sous-tableau spécifique. Le Tableau des règles est destiné à recevoir les équations à résoudre. Celles-ci doivent avoir au maximum 200 caractères de long et doivent être en conformité avec les conventions mathématiques standard de notation et d'opérations. Il remplit le bas de l'écran initial. Le Tableau des unités contient l'information nécessaire à la conversion des unités de mesure attribuées aux variables d'un modèle.

TK! utilise ces trois tableaux pour la plupart des opérations. Il existe d'autres tableaux dont le Tableau global qui permet à l'opérateur de personnaliser certaines procédures d'opérations; le Tableau liste qui sauvegarde tout tableau de valeurs pour des variables; le Tableau des fonctions utilisateur et d'autres fonctions encore, destinées à tracer ou à afficher des points graphiques ou des tables de valeurs.

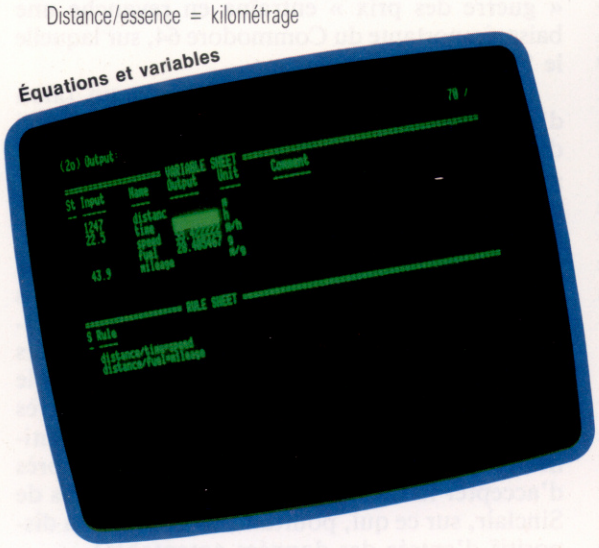
Nous commencerons par créer un modèle très simple adapté au manuel utilisateur de TK!. Il permet le calcul de la distance parcourue et de la vitesse moyenne lors d'un trajet en voiture. Les unités sont converties en unités du système métrique. Commençons par définir les variables d'équations. Nous tapons :

Distance/temps = vitesse

suivi de RC. TK! est initialisé pour transcrire les noms de variables en provenance d'équations, directement dans le Tableau des variables. Le programme évalue l'équation et affiche les variables dans la colonne Nom du Tableau des variables, dans l'ordre où elles figurent dans l'équation. Un astérisque est alors affiché dans la colonne Statut suivant l'équation. Il signifie que l'équation n'est pas résolue puisqu'il n'y a pas eu de valeurs entrées pour elle dans le Tableau des variables. Nous saisissons ensuite la deuxième équation de la même manière :

Distance/essence = kilométrage

Équations et variables



Les cinq variables sont alors listées dans la colonne Nom du Tableau des variables. Frappez la touche (;) afin de déplacer le curseur du Tableau des règles au Tableau des variables, dans lequel nous entrons les valeurs. Le curseur apparaît dans la colonne des entrées située après la première variable, Distance. Les valeurs suivantes sont saisies aux places libres et validées par un RC ou par la touche Flèche-vers-le-bas.

ENTRÉES	NOM	SORTIES
500	distance	
0,5	temps	
	vitesse	
14	essence	
	kilométrage	

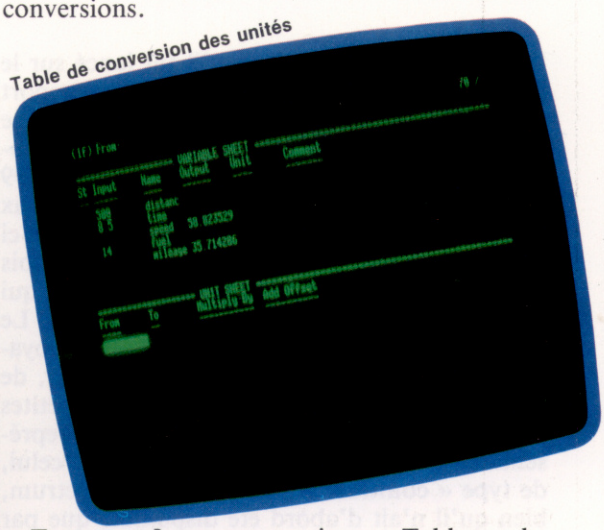
Les valeurs pour vitesse et kilométrage sont laissées en blanc, TK! devant les calculer. Les valeurs obtenues sont affichées dans la colonne SORTIES. Pour demander la résolution des équations, appuyez sur !, et TK! appelle la touche Action. La phrase « Résolution directe » s'affiche au-dessus du Tableau des Variables, toutes les données étant fournies pour trouver directement la solution. Immédiatement après, les valeurs pour vitesse et kilométrage s'affichent, au fur et à mesure.

Les valeurs données à notre modèle jusqu'ici n'ont pas d'unités. Nous ne pouvons taper simplement « miles » ou « litres » dans la colonne Unité du Tableau des variables, les unités ne pouvant être utilisées avant d'avoir été définies. Nous transférons le curseur dans le Tableau des règles par la touche (;), et en tapant =U. TK! nous remplaçons le Tableau des règles par celui des unités dans la fenêtre inférieure. Le Tableau des unités comporte quatre colonnes :

DE EN Multipliez par Ajoutez Décalage

Le curseur apparaît en dessous du mot « DE ». Nous pouvons alors entrer les unités de notre choix ainsi que les valeurs à utiliser pour les conversions.

Table de conversion des unités



Tapons =R pour revenir au Tableau des règles, et accédons au Tableau des variables. Nous pouvons maintenant taper les noms pour les variables dans la colonne Unité, « m » pour la distance (miles); « h » pour l'heure; « m/h » pour la vitesse; « g » pour l'essence (gallons); et « m/g » pour le kilométrage. Effaçons les valeurs courantes pour les remplacer par les valeurs suivantes : 1,247 pour la distance, 22,5

Conversion système anglais/système métrique

Ian McKinnell

pour le temps, et 43,9 pour le kilométrage. En tapant sur ! pour demander la résolution, on obtient les valeurs métriques équivalentes.

Positionnons le curseur sur m dans la colonne des unités. Tapons alors km pour kilomètres. Appuyons sur RC, et TK! convertit automatiquement la valeur de 1,247 correspondant à la distance, en une valeur équivalente en kilomètres : 2006,423.

Ravalement de façade

Le Sinclair Spectrum a connu un grand succès. Mais des nouveaux venus ont semblé surclasser cette vénérable machine. Son constructeur a donc décidé d'offrir une livrée toute neuve.

Le Sinclair Spectrum, lorsqu'il fut lancé sur le marché au printemps 1982, offrait un rapport qualité/prix extraordinaire. La concurrence se réduisait dans les faits au Vic-20, qui ne proposait qu'une maigre mémoire de 3,5 K et au TI99 4A de Texas Instruments, dont le prix était deux fois supérieur à celui du Spectrum. Celui-ci connut donc un succès foudroyant, à la fois auprès des débutants et des enthousiastes qui avaient déjà acheté un ZX80 ou un ZX81. Le nouvel appareil disposait d'une mémoire incroyable (48 K!), d'une bonne version de BASIC, de huit couleurs à l'affichage, et même de petites possibilités sonores. Le clavier lui-même représentait un très gros progrès par rapport à celui, de type « confiture », du ZX81. Le Spectrum, bien qu'il n'ait d'abord été disponible que par correspondance, devint aussitôt l'appareil le plus vendu de Grande-Bretagne.

Toutefois, plus de deux ans et demi se sont écoulés depuis, et de nouveaux compétiteurs ont fait leur apparition. Le plus sérieux est sans doute le Commodore 64; il est affligé d'un BASIC préhistorique mais il a davantage d'espace mémoire (encore que pour en tirer pleinement parti, il faille programmer en langage machine), des possibilités sonores extraordinaires, et un véritable clavier type machine à écrire. D'autres matériels, tant américains que français, ont, eux aussi, des caractéristiques très supérieures; ils auraient pu devenir une menace très inquiétante s'il n'y avait eu leur prix, bien trop élevé. Mais les constructeurs n'ont jamais voulu les modifier. La fameuse « guerre des prix » entraîna en revanche une baisse importante du Commodore 64, sur laquelle le Spectrum dut s'aligner.

L'appareil de Sir Clive Sinclair a un talon d'Achille : son clavier. On a produit pour lui d'innombrables logiciels, dont de nombreux programmes de gestion remarquables — mais faire du traitement de texte avec un Spectrum donne irrésistiblement l'impression de porter des mitaines. De nombreux usagers n'hésitèrent donc pas à faire l'acquisition d'un « vrai » clavier, tandis que Sinclair Research mettait en vente l'Interface 1 et le lecteur de microdrives. Les capacités du Spectrum s'en trouvaient accrues — mais le problème de la frappe restait. Aussi, il est très vite apparu, au cours de l'année 1984, que les utilisateurs de micro n'étaient toujours pas près d'accepter la thèse ou plutôt les conceptions de Sinclair, sur ce qui, pourtant, constituait un dispositif d'entrée des données acceptable!

Six d'un coup
Comme bien d'autres constructeurs, Sinclair Research a décidé d'offrir avec le Spectrum + (et le Spectrum 48 K) six logiciels d'un coup : un traitement de texte, un tableur, deux jeux et deux programmes graphiques. Tous sont de très bonne qualité.

Chirurgie esthétique

Le constructeur a donc choisi de procéder à une opération de chirurgie esthétique, qui a donné naissance au Spectrum +. Il s'agit en fait de la même machine, présentée dans un boîtier analogue à celui du QL, tout comme le clavier, qui comporte cependant quelques touches supplémentaires, un bouton de remise à zéro, et deux pattes de soutien rétractables. Tous les périphériques destinés à l'ancienne version devraient fonctionner sans difficultés avec la nouvelle. Mais Sinclair aurait-il mieux fait de procéder à de réelles modifications : amélioration du son, mise en place d'une prise moniteur ou intégration de l'Interface 1.

Les pathétiques possibilités sonores de l'appareil constituent désormais son plus lourd handicap. Il est vrai que, grâce aux pattes rétractables, le son est moins étouffé qu'auparavant — les bruits poétiques qui accompagnent chargements et sauvegardes le sont aussi... La commande BEEP reste toujours aussi insuffisante.

Le Spectrum + mesure 319 × 149 × 38 mm. La programmation est, certes, facilitée : le clavier comporte des touches supplémentaires pour les modes graphique ou « étendu », la vidéo inversée, certains signes de ponctuation (point-virgule, guillemets, virgule, point). On a ajouté une touche BREAK, une touche DELETE et une SYMBOL SHIFT, tandis que les touches curseur ont

Quelle différence?

A l'intérieur du boîtier, très séduisant, on trouve la version 4.5 du Spectrum, qui est en fait la version 3 de l'appareil (mise en vente en août 1983), simplement dotée d'un bouton de remise à zéro installé tant bien que mal sur le circuit imprimé. (Cl. Chris Stevens.)

SPECTRUM +

PRIX

★ ★

DIMENSIONS

319 × 149 × 38 mm.

MÉMOIRE ET INTERFACES

Semblables à celles du Spectrum 48 K. BASIC résident. Logiciels pleinement compatibles.

CLAVIER

Membrane, 58 touches sculptées (dont une véritable barre d'espace).

DOCUMENTATION

Manuel d'initiation en couleurs avec cassette de présentation.

FORCES

Énorme bibliothèque de programmes, de nombreux clubs d'utilisateurs, beaucoup de livres disponibles.

FAIBLESSES

En dépit d'une nouvelle présentation, le clavier reste peu sensible et peu maniable; les défauts n'ont pas été vraiment corrigés.

été déplacées et installées à côté de la barre d'espace.

Toutes les combinaisons fonctionnent comme autrefois, mais les vieux habitués auront peut-être parfois des problèmes. C'est ainsi qu'une touche EDIT est placée à côté de la touche A; si on appuie dessus par erreur alors même que l'on entre une longue ligne de programme, cette ligne sera perdue! Et l'on sait qu'à la frappe personne n'est parfait...

Les acquéreurs du Spectrum + se voient offrir six logiciels avec l'appareil : Psion Chess, Chequered Flag, Make-a-Chip, Scrabble, Vu-3D et surtout Tasword Two, un traitement de texte tout à fait remarquable. Tous ces programmes sont de grande qualité; mais on n'en dira pas autant de la documentation qui accompagne l'ordinateur. Elle est très bien présentée mais reste superficielle. Les éditeurs précisent, il est vrai, que des manuels plus approfondis sont disponibles pour un prix raisonnable...

Écrire les jeux

Les amateurs de jeux seront à la fête, mais il est vrai que beaucoup d'entre eux ont d'ores et déjà investi dans l'achat d'un manche à balai et d'une interface appropriée. Les modèles Kempston et Fuller ne posent aucun problème de fonctionnement; il en va de même pour l'interface Centronics de Kempston et pour les Wafadrives (un système de mémoire de masse mis au point pour suppléer aux insuffisances des microdrives).

Le Spectrum + représente très certainement un progrès par rapport au modèle original, mais l'idée que Sinclair se fait d'un « bon » clavier reste sujette à discussion. Vu l'augmentation de prix, il convient d'abord de voir s'il est réellement compétitif face aux autres périphériques déjà disponibles; la réponse est, malheureusement, non. La nouvelle version a plus de classe, mais les touches sont peu sensibles et trop agglutinées...

Les débutants pourront être séduits; reste à savoir si les nouvelles possibilités qu'il offre justifient l'augmentation du prix. Les cyniques diront que, en fait, c'est dans ce seul but que le nouveau modèle a été lancé; il ne serait pas étonnant que le modèle original disparaisse bientôt. Sinclair Research aurait sans doute mieux fait de consentir des réductions (ou de vendre moins cher l'Interface 1); la firme aurait pu aussi, pour justifier la montée des prix, doter le Spectrum de tout ce qui lui manque — de réelles possibilités sonores, un clavier authentique, une prise moniteur, ou au besoin un Microdrive intégré.

Mais jamais elle n'aurait été prête à temps pour les fêtes de Noël qui s'annonçaient...

Le facteur clé

Vous jugerez différemment le nouveau clavier suivant la façon dont vous tapez.
Un avantage toutefois : l'affectation de touches séparées aux fonctions les plus importantes. On peut ainsi reproduire les combinaisons de touches d'origine (ainsi SYM SHIFT + O pour obtenir le point-virgule).
(Cl. Ian McKinnell.)

Fin de partie

Notre projet de créer un jeu d'aventures avec LOGO arrive ici à sa fin. Après avoir défini les diverses positions et écrit les procédures pour se déplacer de l'une à l'autre, attardons-nous sur les détails.

Notre jeu « Les Reliques de Zoltoth » ne comporte que deux périls. Dans la PIÈCE 4, le joueur doit faire face à un monstrueux serpent, et le programme embraye sur une procédure spéciale, « péril » :

```
POUR ATTAQUE.DU.SERPENT
 AFFICHEL [[UN SERPENT MONSTRUEUX]
 [DOUCEMENT GLISSE VERS VOUS!]]
FIN
```

L'autre « péril » ne met pas le joueur dans une situation immédiatement périlleuse, mais lui prépare certainement bien des ennuis :

```
POUR PORTAIL
 AFFICHEL [[UN GIGANTESQUE PORTAIL
 DORÉ SE REFERME SUR VOUS][VOUS
 BARRANT L'ISSUE PAR LE SUD]]
 FAIT «PÉRILS[]
 FAIT «SORTIE.LISTE [[N7][E8]]
FIN
```

D'autres considérations doivent être prises en compte à certains moments du programme. La procédure PREND doit être modifiée de sorte que vous ne puissiez pas prendre l'anneau lorsque vous portez l'épée.

```
POUR LAPRENDRE :ÉLÉMENT
 SI :ÉLÉMENT = «ANNEAU ALORS
 PREND.ANNEAU STOP
 AJOUTE.A.L'INV :ÉLÉMENT
 RETIRE.DE.LA.PIÈCE :ÉLÉMENT
FIN

POUR PREND.ANNEAU
 SI APPARTIENT? «ÉPÉE :INVENTAIRE ALORS
 AFFICHE [VOUS NE POUVEZ PRENDRE
 L'ANNEAU] STOP
 AJOUTE.A.L'INV :ÉLÉMENT
 RETIRE.DE.LA.PIÈCE :ÉLÉMENT
STOP
```

C'est là la seule restriction que rencontre le joueur pour prendre un objet. Les routines suivantes vous permettent de savoir ce que vous portez.

```
POUR EXAMINE :OBJ
 SI :OBJ = «ANNEAU ALORS DESC.ANNEAU STOP
 SI :OBJ = «CHÂSSE ALORS DESC.CHÂSSE STOP
 SI :OBJ = «ÉPÉE ALORS DESC.ÉPÉE STOP
 AFFICHE [VOUS NE PORTEZ RIEN DE
 SPÉCIAL]
FIN
```

```
POUR DESC.ANNEAU
 SI ICI? «ANNEAU ALORS AFFICHEL
 [[L'ANNEAU PORTE UNE INSCRIPTION
 EFFACÉE:] [R– – –E]] SINON AFFICHE
 [JE NE VOIS PAS D'ANNEAU]
FIN

POUR ICI? :OBJ
 SI APPARTIENT? :OBJ :CONTENU ALORS
 RÉSULTAT «VRAI SI APPARTIENT? :OBJ :INVENTAIRE ALORS
 RÉSULTAT «VRAI
 RÉSULTAT «FAUX
FIN

POUR DESC.CHÂSSE
 AFFICHEL [[D'UNE TRÈS BELLE FACTURE]
 [ET MANIFESTEMENT UNE VRAIE FORTUNE]
 [SUR UN ANGLE DU COUVERCLE EST GRAVÉ
 UN MINUSCULE CRÂNE]]
 FIN

 POUR DESC.ÉPÉE
 SI ICI? «ÉPÉE ALORS AFFICHE [FAITE D'ACIER]
 SINON AFFICHE [JE NE VOIS PAS D'ÉPÉE]
FIN
```

Le joueur a besoin de l'épée pour tuer le serpent ; s'il ne l'a pas, le serpent le tue.

```
POUR TUE :LA.CHOSE
 SI :LA.CHOSE = «SERPENT ALORS TUE.SERPENT STOP
 AFFICHE [VOUS NE POUVEZ PAS !]
FIN

POUR TUE.SERPENT
 SI NON APPARTIENT? «ATTAQUE.DU.SERPENT :PÉRILS
 ALORS AFFICHE [JE NE VOIS PAS DE SERPENT] STOP
 SI APPARTIENT? «ÉPÉE :INVENTAIRE ALORS
 MORT.DU.SERPENT SINON LE.SERPENT.TUE
FIN

POUR MORT.DU.SERPENT
 AFFICHE [LE SERPENT SE MEURT, S'ENTORTILLANT
 DE DOULEUR]
 FAIT «PÉRILS []
FIN

POUR LE.SERPENT.TUE
 AFFICHEL [[VOUS ÊTES DÉSARMÉ] [POUR
 LE TUER] [ET VOUS L'AVEZ RENDU FOU
 FURIEUX] [IL VOUS MORD ! VOTRE VISAGE
 DEVIENT NOIR] [VOUS TOMBEZ EN VOUS
 TORDANT DE DOULEUR]]
 MORT
FIN
```

La procédure MORT est une anticipation pour les joueurs qui croient à la réincarnation. Si vous tapez autre chose que DÉBUT après être mort, l'ordinateur vous rappelle que vous êtes mort !

```
POUR MORT
 AFFICHE [VOUS ÊTES MORT !]
 AFFICHE1 »?
 FAIT «SAISIE COMMANDE
 SI (:SAISIE = «DÉBUT) ALORS STOP
 AFFICHE [SORTEZ-VOUS-EN !]
 MORT
FIN

POUR COMMANDE
 FAIT «SAIS QUESTION
 SI :SAIS = [] ALORS AFFICHE1 »?
 RÉSULTAT
 COMMANDE
 RÉSULTAT PREMIER :SAIS
FIN
```

En frottant l'anneau, le génie apparaît :

```
POUR FROTTE :OBJ
 SI :OBJ = «ANNEAU ALORS FROTTE.ANNEAU
 STOP
 AFFICHE [IL EST NETTEMENT PLUS PROPRE]
FIN

POUR FROTTE.ANNEAU
 SI ICI! «ANNEAU ALORS GÉNIE SINON
 AFFICHE [JE NE VOIS PAS D'ANNEAU]
FIN
```

Le génie vous propose de vous ramener chez vous mais si vous refusez, un grand vent vous projette au hasard d'une pièce dans la partie est de la cave :

```
POUR GÉNIE
 AFFICHEL [UN GÉNIE APPARAÎT ET VOUS DEMANDE:]
 [«VOULEZ VOUS RETOURNER CHEZ VOUS?»]]
 AFFICHE 1»
 FAIT «REP PREMIER COMMANDE
 SI AU.MOINS.UN :REP = «OUI :REP = «O
 ALORS RETOUR SINON GRAND.VENT
FIN

POUR RETOUR
 AFFICHE [ENFIN DE RETOUR]
 SI APPARTIENT? «SCEPTRE :INVENTAIRE
 ALORS AFFICHE [BRAVO D'AVOIR TROUVÉ
 LE SCEPTRE!] SINON AFFICHEL [AU MOINS
 VOUS ÊTES] [EN VIE]]
FIN

POUR GRAND.VENT
 AFFICHE [UN GRAND ET TERRIBLE VENT SE LÈVE]
 AFFICHE»
 DÉPLACE1 (6 +(RANDOM 5) )
FIN
```

La seule chose que vous puissiez ouvrir est la trappe, et elle contient une araignée mortelle. Le crâne sur le couvercle est un avertissement, mais des gens ne comprennent pas...

```
POUR OUVRE :OBJ
 SI :OBJ = «TRAPPE ALORS OUVRE.TRAPPE
 SINON AFFICHE [VOUS NE POUVEZ PAS L'OUVRIR]
FIN

POUR OUVRE.TRAPPE
 AFFICHE [[IL Y A UNE ARAIGNÉE MORTELLE]
 [DANS LA TRAPPE] [ELLE VOUS PIQUE]]
 MORT
FIN
```

Voici en dernier lieu une liste de tous les noms du jeu :

```
POUR ÉPÉE
 RÉSULTAT «ÉPÉE
FIN

POUR TRAPPE
 RÉSULTAT «TRAPPE
FIN

POUR SCEPTRE
 RÉSULTAT «SCEPTRE
FIN

POUR ANNEAU
 RÉSULTAT «ANNEAU
FIN

POUR SERPENT
 RÉSULTAT «SERPENT
FIN
```

Si vous désirez sauvegarder le jeu au stade où vous en êtes, pour le reprendre ultérieurement, tapez simplement SAVE «AVENTURE. LOGO est par bien des aspects un langage idéal pour la programmation de jeux d'aventures. Un problème, le manque réel de place pour le langage.

Variantes de Logo

Certaines versions du LOGO MIT ne comportent pas VIDE?, ÉLÉMENT, COMPTE et APPARTIENT? Des définitions de remplacement ont déjà été données pour ces mots. Pour toutes les versions LCSI, utilisez :

```
VIDEP pour VIDE?
LISTEP pour LISTE?
APPARTIENTP pour APPARTIENT?
TAPE pour AFFICHE1
OU pour AU.MOINS.UN
```

La primitive EGALP teste si ses deux entrées sont égales. Vous l'utiliserez à la place du signe « égal » pour comparer des listes et des mots. Le signe « égal » est admis par certaines versions LCSI.
La syntaxe SI pour les versions LCSI a la forme suivante :

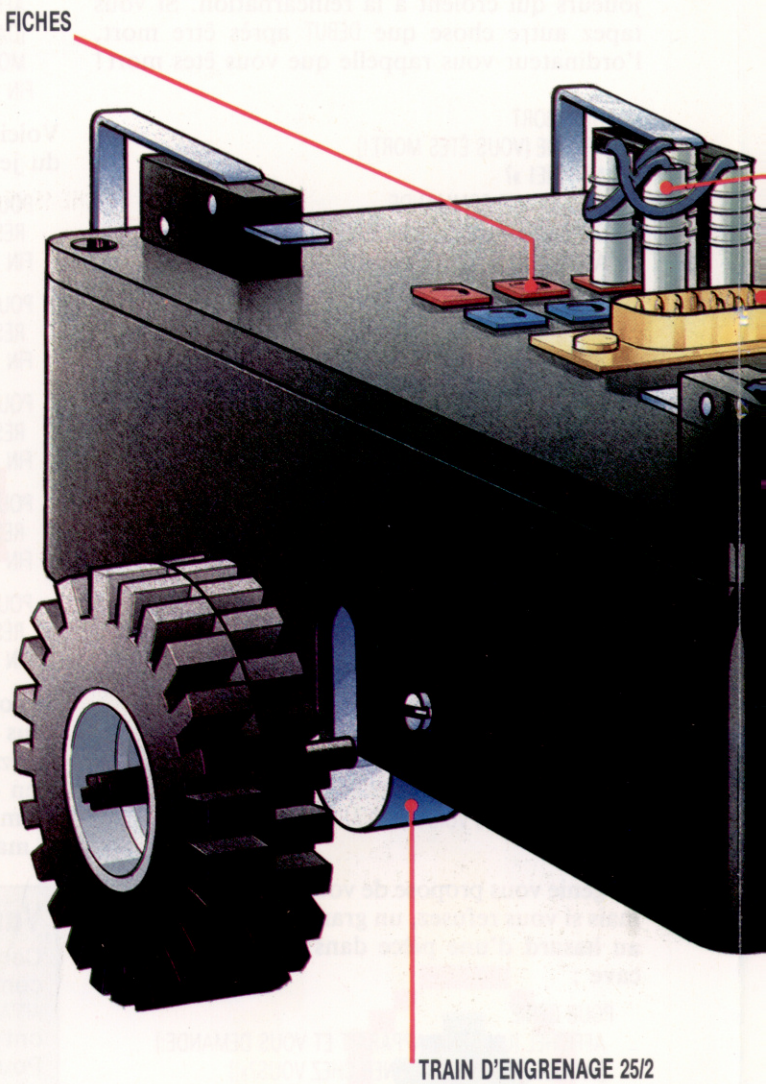
```
SI VIDEP :CONTENU [AFFICHE [RIEN DE
SPECIAL]] [AFFICHE :CONTENU]
```

La première liste après la condition est exécutée pour un résultat VRAI, la seconde, pour un résultat FAUX. Pour LOGO Atari, utilisez P pour Phrase, D pour Demande, et vous remarquez qu'ÉLÉMENT n'est pas implémenté. La version donnée ici concernait le Commodore 64. Certaines machines n'auront pas suffisamment de place mémoire pour le jeu tel qu'il se présente. Dans ce cas, vous devrez réduire l'importance du jeu en ne faisant pas figurer certains mots de description.

Liz Dixon

Premiers pas

Nous allons entreprendre la réalisation d'un nouveau projet : construire un robot d'appartement équipé de détecteurs de lumière et de palpeurs. Nous nous bornerons ici à présenter ce projet dans ses grandes lignes et à donner quelques détails sur l'assemblage de la carrosserie et du moteur.



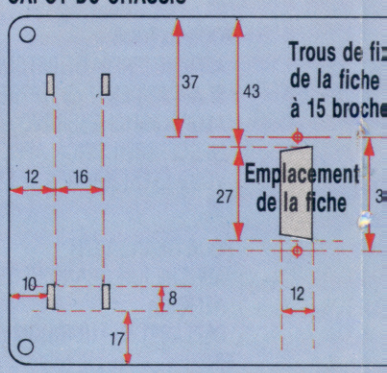
Commandé par logiciel, notre robot d'appartement sera mû par deux moteurs pas à pas guidant deux roues par l'intermédiaire d'un système d'engrenages. Les moteurs dont nous ferons usage peuvent être contrôlés de façon à tourner selon un angle de 7,5° à la fois. L'emploi d'un rapport d'engrenage de 25/2 permettra de commander les roues pour une rotation d'essieu de 0,6°. Par ailleurs, les moteurs pas à pas fonctionnent par impulsions ; ils sont donc parfaitement adaptés pour être contrôlés par un dispositif numérique. Le port utilisateur d'un ordinateur en tiendra lieu, et nous rédigerons un logiciel simplifié à utiliser en conjonction avec le robot. Celui-ci sera, de surcroît, équipé de divers capteurs — détecteurs de proximité et de lumière, de sorte que l'engin puisse suivre une ligne tracée sur le sol. Pour qu'il puisse être suivi par l'ordinateur, celui-ci doit affecter à cette tâche quatre canaux de données du port utilisateur, ce qui signifie que les informations transmises par les capteurs ne pourront emprunter que les quatre canaux restants. Un ensemble de fiches et de fils électriques assurera leur diffusion ; ce système, assez souple, permet de faire usage de capteurs différents : à telle occasion, nous aurons peut-être besoin de quatre détecteurs de proximité, à telle autre, de deux seulement, plus deux capteurs de lumière. Il suffira alors de procéder aux raccords nécessaires.

Nous entreprendrons également d'écrire un logiciel assez précis qui fournisse au robot une sorte de carte intérieure de son environnement immédiat. Nous pourrons alors nous adonner aux joies de la recherche d'algorithmes permettant à notre engin de se déplacer dans son espace. Pour le moment, nous en resterons à la construction de l'engin lui-même. Elle ne présente pas de problèmes insurmontables ; il s'agit pour l'essentiel de découper et de mettre en forme le plastique qui constituera le boîtier et le châssis. Vous devrez faire preuve de précision en perçant les trous destinés à accueillir les engrenages et la fiche à quinze broches, mais ce ne sera pas utile pour la mise en place des pieds de caoutchouc qui servent de contrepoids.

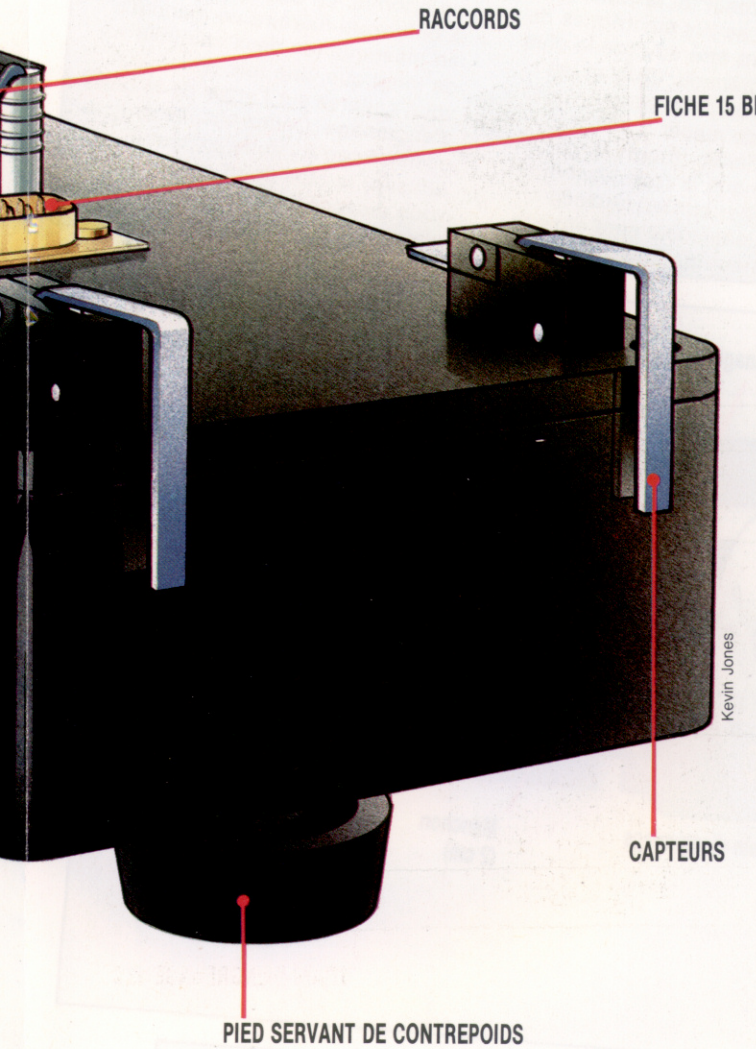
Premier point

Il faudra d'abord découper les trous dans le plastique qui formera le châssis. Nos schémas vous en donnent les positions et les dimensions. Ceux qui sont situés sur les côtés et le fond de la caisse accueilleront les essieux sur lesquels viendront s'emboîter les roues ; il est important qu'ils soient bien en ligne les uns par rapport aux autres. Les deux trous percés dans le plancher de l'engin sont destinés aux pieds de caoutchouc qui assureront l'équilibre du robot. Celui que l'on remarque sur le capot sert à la fixation d'une fiche de type D qui sera directement raccordée à l'ordinateur. Pour découper tous ces trous, la méthode générale consiste d'abord à ôter le plus gros du plastique avec un fer à souder ou un outil métallique chauffé, puis, à l'aide d'une petite lime, à procéder à la finition.

CAPOT DU CHÂSSIS



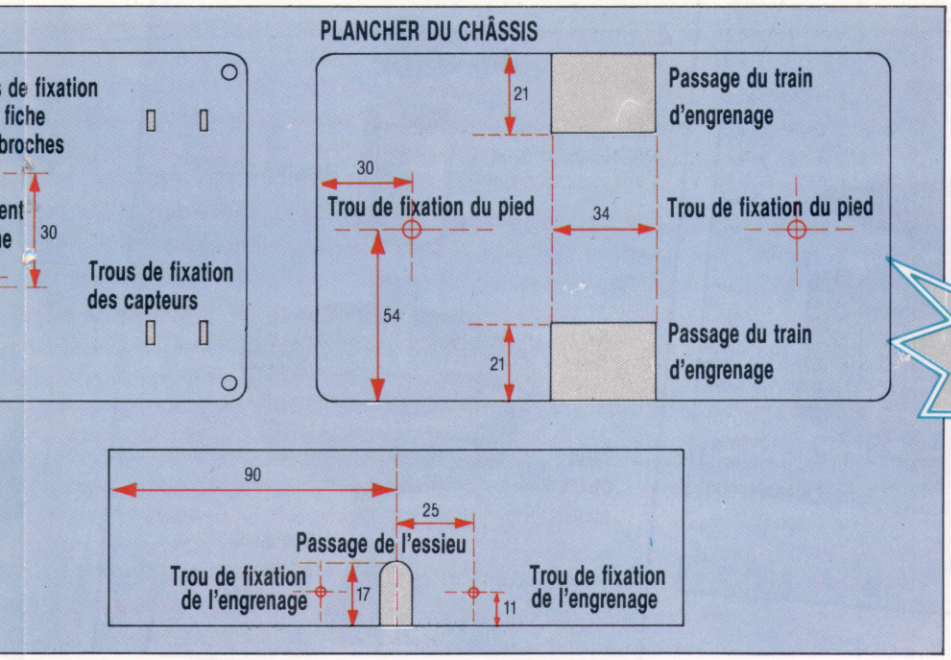
TOUTES LES MESURES SONT DONNÉES EN MILLIMÈTRES



Accessoires

Quantité	Article	Référence
Radio-amateurs		
2	Puces de contrôle SAA 1027	300-237
2	Moteurs pas à pas	332-947
2	Trains d'engrenage 25/2 synchrones	336-450
Cartes programmes		
1	Puce-tampon 40109	QW67X
3	Prises DIL à 16 broches	BL19V
2	Résistances de 100 Ω	M100R
2	Résistances 270 Ω 0,5 W	S270R
2	Capacités 0,1 µF	YR75S
1	Capacité 25 V 1 000 µF	FB83E
1	Plaquette d'essai de type « veroboard »	FL07H
1	Rouleau de fil électrique	BL13P
1	Fiche D à 15 broches mâle	BK58N
1	Fiche D à 15 broches femelle	BK59P
1	Bouchon pour fiche D 15 broches	BK60Q
1	Prise de courant 2,1 mm	RK37S
1	Prise IDC à 20 voies	FG87U (BBC)
1	Connecteur plat à 24 voies	BK74R (64)
1	Boîtier de 180 × 110 × 55 mm	LF51F
1	Rouleau de ruban isolant adhésif	HB22V
2	Patins de fauteuil en plastique	FW39N
Assortiment		
2	Roues Lego de 62 mm	Lego 1246
1	Ensemble essieux Lego	Lego 1233
4 m	Câble-ruban à 12 fils	
1 m	Câble-ruban à 20 fils	(BBC)
1	Prise d'alimentation 12 V 1 ampère	

Tous ces articles peuvent être achetés chez les détaillants spécialisés en composants électroniques. Vous aurez de plus besoin de boulons et d'écrous.



Attention!

Notre robot consomme beaucoup d'énergie; si la prise d'alimentation doit également fournir du courant à la puce-tampon, il n'en recevra pas assez pour lui-même, et ne fonctionnera pas. Il doit par ailleurs être connecté directement au port utilisateur de votre ordinateur. Celui-ci peut être endommagé s'il y a des erreurs de construction. Si vous n'êtes pas suffisamment sûr de vos talents de bricoleur, peut-être vaudrait-il mieux vous abstenir. Suivez très soigneusement toutes les instructions qui vous seront données.

Deuxième point

Les moteurs pas à pas et les engrenages sont vendus séparément. Le train d'engrenage est accompagné d'une petite roue dentée métallique et d'un manchon de plastique. La roue doit être fixée sur l'arbre du moteur, à l'aide de colle très puissante (de type « super Glue »), que l'on dépose sur l'alésage de la roue. Celle-ci est alors mise en place (le côté de la fraisure ne doit pas faire face au moteur). Servez-vous d'une extrémité du manchon pour positionner correctement le train d'engrenage par rapport au moteur. Prenez bien garde à laisser sécher complètement la colle (deux à trois heures minimales).

Troisième point

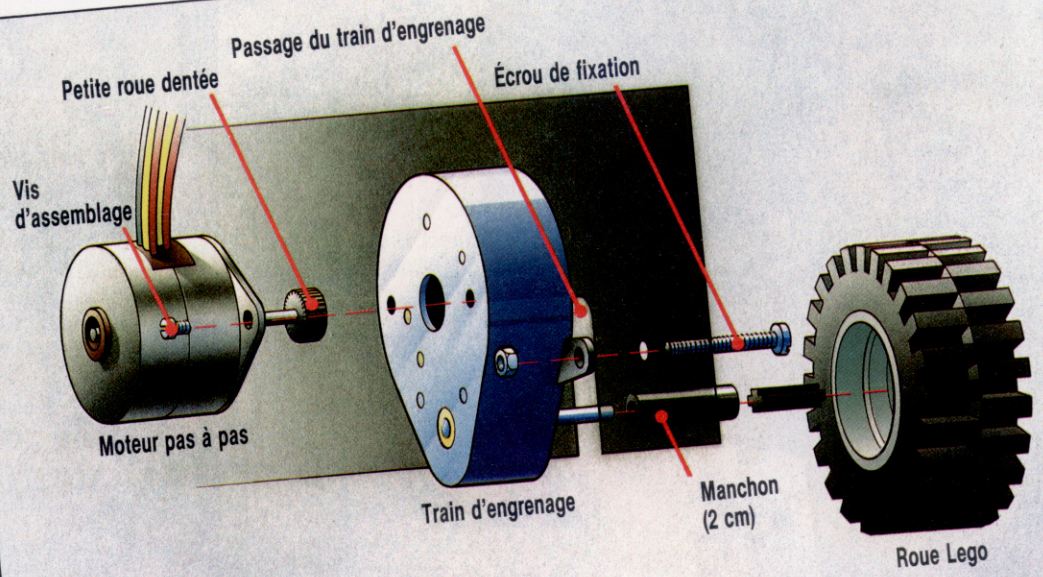
Installez le moteur (et la roue) sur l'engrenage. Les fils électriques du moteur doivent être du côté le plus large de l'engrenage. Celui-ci est accompagné de deux vis qui vous permettront de mener à bien l'opération. Faites attention, ce faisant, à ce que la roue vienne bien s'engrener sur les rouages intérieurs. L'ensemble sera ensuite fixé à l'intérieur du châssis. Celui-ci accueille des boulons sur lesquels les engrenages sont fixés, sans être en contact direct avec le boîtier; on pourra donc les ajuster, si besoin est, en même temps que les roues.

Ces dernières nécessitent un essieu spécifique à section cruciforme. La carcasse d'un stylo à bille vous fournira un manchon en plastique (de deux centimètres de long) que, une fois découpé, vous collerez sur l'arbre de la boîte d'engrenage. Collez ensuite un essieu Lego à l'autre extrémité, en utilisant le plus court de ceux dont vous disposez. La roue est mise en place sur lui par simple poussée.

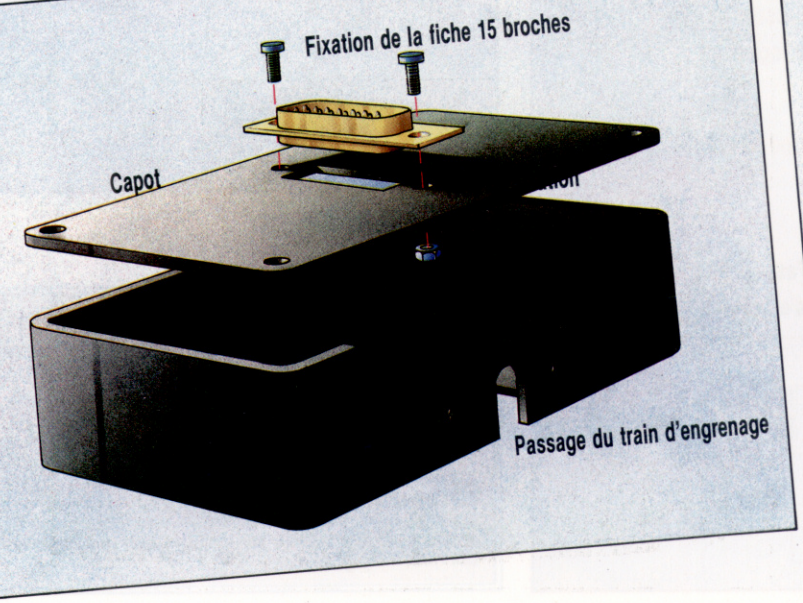
Quatrième point

A l'aide de boulons et d'écrous, mettez en place sur le capot du châssis la fiche D (mâle), les broches dirigées vers le haut. Après quoi, fixez les deux pieds servant de contrepoids, en procédant de la même façon. La base des pieds doit se trouver à environ trois centimètres du plancher du boîtier. Pour cela, vous pourrez utiliser un manchon en plastique de deux centimètres, ou recourir à un contre-écrou.

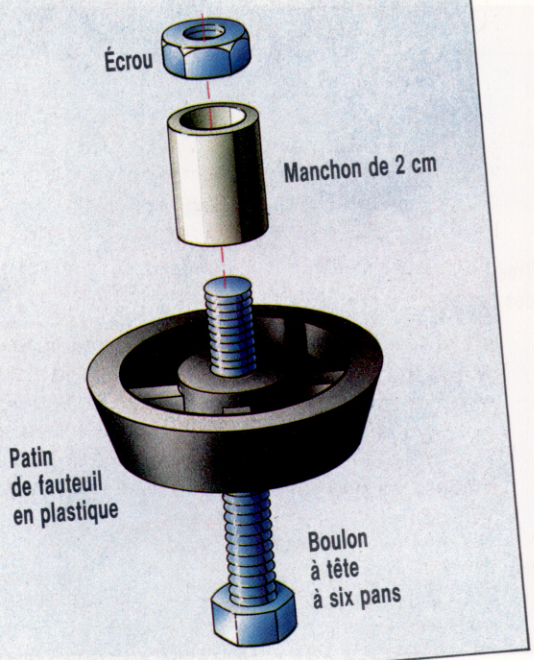
2 et 3



4



4



Kevin Jones

En position

Dans notre projet de programmation d'un jeu d'aventures, écrivons des programmes qui décrivent des positions à l'intérieur du jeu et permettent aux joueurs de se déplacer.

La description de base pour chaque position figure dans le tableau LN$() que nous avons déjà donné. On y accède en spécifiant le numéro de la position atteinte. Dans la Forêt hantée, la position occupée par un joueur à un instant donné est sauvegardée dans la variable P, et sa description est donc dans LN$(P). Lors de la conception des données de positions, le contexte grammatical final des descriptions a toujours été pris en compte ; la description est formulée de telle sorte qu'elle puisse être préfixée par « Vous êtes... ». Pour une position donnée, P, elle peut être formatée et sortie vers l'utilitaire développé précédemment, en combinant « Vous êtes » avec la description correspondant à cette position dans le tableau LN$(). La ligne 2010 du listage de la Forêt hantée le montre.

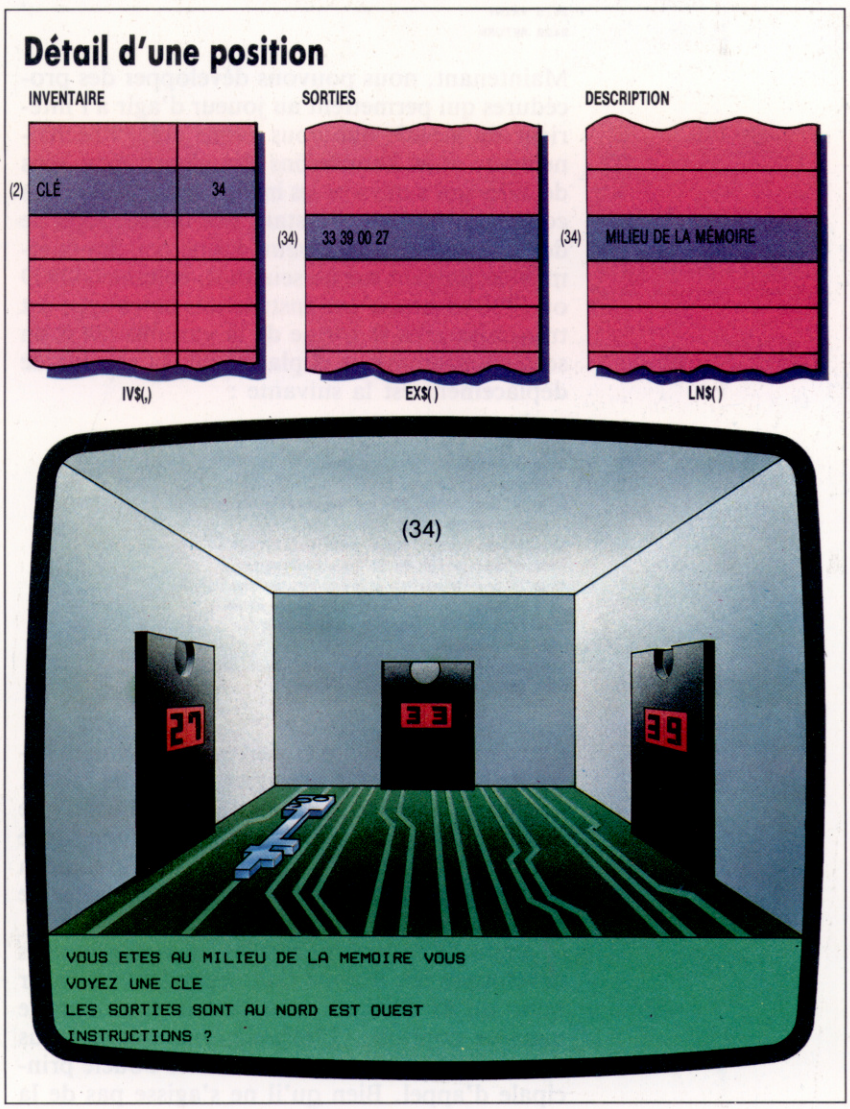
Outre la description sommaire de la position atteinte, le joueur voudra également savoir s'il y a d'autres objets présents. Les objets utilisés dans le jeu figurent — avec leur position initiale dans l'inventaire — dans un tableau à deux dimensions, IV$(,). Par exemple, IV$(N,1) contient la description du *nième* objet de l'inventaire, IV$(N,2) contient la position de ce même objet. Si nous voulons savoir s'il y a un objet à une position donnée, il nous faut passer en revue l'inventaire, et comparer la position de chaque objet avec le numéro de la position examinée. Comme il n'y a que trois objets pour la Forêt hantée, et huit pour Digitaya, il suffira d'effectuer une recherche linéaire au moyen d'une boucle FOR...NEXT.

Les lignes 2040 à 2080 montrent la boucle de recherche utilisée pour la Forêt hantée. La deuxième colonne du tableau de l'inventaire est passée en revue pour trouver la position courante, P. Lorsque cette dernière est trouvée, sa description est ajoutée à la phrase qui décrit les objets. Du fait qu'une position peut contenir plusieurs objets, la phrase devra permettre de mentionner une liste d'objets séparés par une virgule. En utilisant SP$, d'abord comme une chaîne vide, et ensuite comme virgule, nous pouvons insérer la ponctuation voulue. Un drapeau, F, initialisé à zéro est mis à un pour signaler qu'un objet a été trouvé. Si le drapeau reste à zéro, c'est qu'il n'y a rien. Cette information peut être transmise au joueur, comme dans la ligne 2090 de la Forêt hantée.

```
2000 REM **** DECRIRE POSITION ****
2010 SN$="VOUS ETES "+LN$(P):GOSUB5500
2020 SN$="VOUS VOYEZ "
2030 REM ** RECHERCHE DE L'OBJET DANS L'INVENTAIRE **
2040 F=0:SP$=""
2050 FOR I=1 TO 3
2060 IF VAL(IV$(I,2))<>P THEN 2080
2070 SN$=SN$+SP$+"A "+IV$(I,1):F=1:SP$=", "
2080 NEXT I
2090 IF F=0 THEN SN$=SN$+" PAS D'OBJET "
2100 GOSUB5500:REM FORMAT OUTPUT
2110 RETURN
```

Les données indiquant les détails des sorties possibles à partir de chaque position figurent au tableau EX$(). Chaque valeur de la chaîne peut comprendre jusqu'à huit chiffres. En divisant ces huit chiffres en groupes de deux, nous obtenons de gauche à droite les nombres pour les positions situées respectivement au nord, à l'est, au sud et à l'ouest de la position courante. Afin de déterminer les sorties possibles, le programme divise la chaîne de huit chiffres en quatre nombres

Chambre avec vue
Les détails des positions figurent dans trois tableaux-chaîne qui contiennent le nom des objets et d'autres informations qui leur sont relatives (V$). Il y a en outre les sorties des positions (EX$) et les descriptions (LN$). Par exemple, EX$(34), peut contenir le nombre à huit chiffres 33390027. Cela signifie alors que la position 34 est en liaison avec les positions 33, 39 et 27 respectivement par ses sorties nord, est et ouest. LN$(34) contient la position médiane de la mémoire et décrit la position 34. IV$(2,2) contient le nombre 34, indiquant que IV$(2,1), la clé, est en position 34. Selon le nombre courant, le programme assemble l'information sous forme descriptive.



Kevin Jones

correspondant aux positions pointées par les quatre directions.

```
2300 REM **** DECRIRE SORTIES S/R ****
2310 EX$=EX$(P)
2320 NR=VAL (GAUCHE EX$,2))
2330 EA=VAL (MID$ (EX$,3,2))
2340 SO=VAL (MID$ (EX$,5,2))
2350 WE=VAL (DROITE (EX$,2))
```

S'il n'y a pas de sortie dans une direction donnée, la valeur correspondante est de zéro, ce qui est d'une grande aide pour la description des sorties. Une vérification préliminaire doit avoir lieu pour vérifier s'il y a bien une sortie avant de construire la phrase « Il y a des sorties... ». On utilise pour cela un OU logique pour les quatre variables de directions, le résultat étant de zéro seulement lorsque les quatre variables de direction le sont. Si tel n'est pas le cas, la routine continue par le test de chacune des directions. Lorsque la variable est différente de zéro, la direction correspondante est ajoutée à la phrase.

```
2355 IF (NO OR ES OR SU OR OE)=0 THEN RETURN
2360 PRINT:SN$="IL Y A DES SORTIES "
2370 IF NO<>0 THEN SN$=SN$+" NORD "
2380 IF ES<>0 THEN SN$=SN$+" EST "
2390 IF SU<>0 THEN SN$=SN$+" SUD "
2400 IF OU<>0 THEN SN$=SN$+" OUEST "
2410 GOSUB 5500:REM FORMAT
2415 PRINT
2420 RETURN
```

Maintenant, nous pouvons développer des procédures qui permettent au joueur d'agir à l'intérieur du monde que nous avons créé. Prochainement, nous aborderons des algorithmes plus détaillés qui analysent les instructions. Nous nous contenterons pour l'instant des instructions de déplacement que le joueur donne par des commandes portant sur un seul mot, comme NORD ou SUD. Lorsqu'une instruction de ce type est transmise sous la forme de la variable NN$ à un sous-programme de déplacement, la routine de déplacement est la suivante :

```
3500 REM **** MOVE S/R ****
3510 MF=1:REM SET MOVE DRAPEAU
3520 DR$=GAUCHE$(NN$,1)
3530 IF DR$<>"N"ANDDR$<>"E"ANDDR$<>"S"ANDDR$<>"W"
  THEN GOTO3590
3540 IF DR$="N"AND NO<>0 THEN P=NO:RETURN
3550 IF DR$="E"AND EA<>0 THEN P=ES:RETURN
3560 IF DR$="S"AND SU<>0 THEN P=SU:RETURN
3570 IF DR$="OU"AND OU<>0 THEN P=WE:RETURN
3580 PRINT:PRINT" VOUS NE POUVEZ";IS$
3585 MF=0:RETURN
3590 REM ** CE NOM N'EST PAS UNE DIRECTION **
3600 PRINT "QU'EST ";NN$;" ?"
3610 MF=0:RETURN
```

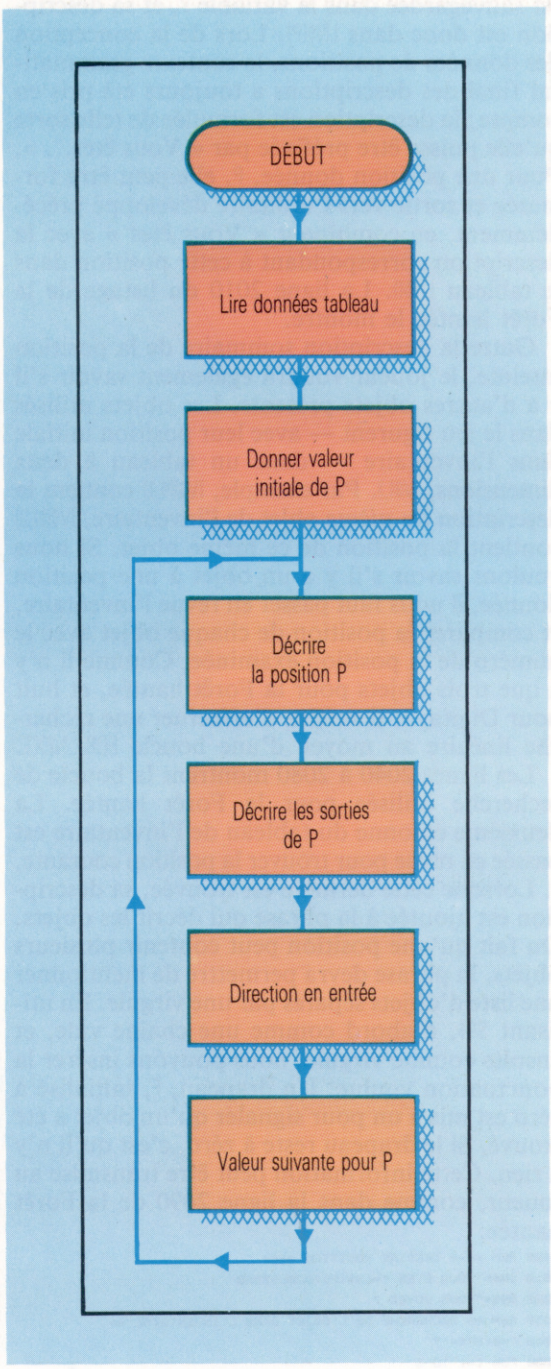
Cette routine n'utilise en fait que la première lettre de la commande de direction qu'elle reçoit. Elle commence par vérifier qu'il s'agit bien d'une direction. Après avoir vérifié qu'il y a une sortie dans cette direction, P, la variable qui indique la position du joueur, est changée pour la valeur de NO, ES, SU ou OU, selon le cas.

Avant de pouvoir utiliser les sous-programmes développés ici, il nous faut cependant les relier pour qu'ils forment une boucle tournant de manière continue. L'organigramme ci-dessous donne la structure logique de cette boucle principale d'appel. Bien qu'il ne s'agisse pas de la structure finale, cela illustre les différents aspects du programme que nous avons traités. Pour utiliser les sous-programmes donnés ici, insérez les lignes suivantes qui font partie de la boucle principale.

```
200 GOSUB6000:REM READ DONNEES TABLEAU
210 P=INT(RND(TI)*10(1):REM POINT DE DEPART
230 REM **** DEBUT DE LA BOUCLE PRINCIPALE ****
240 MF=0:REM DEPLACER DRAPEAU
245 PRINT
250 GOSUB2000:REM DECRIRE POSITION
255 GOSUB2300:REM DECRIRE SORTIES
260 PRINT:INPUT"INSTRUCTIONS";IS$
```

Vous insérerez également les lignes suivantes :

```
270 NN$=IS$:GOSUB 3500:REM DÉPLACEMENT
280 GOTO 230:REM RECOMMENCER LA BOUCLE PRINCIPALE
```



Variantes pour le Spectrum

Du fait que le Spectrum considère tous les tableaux-chaînes comme des chaînes de longueur fixe, un problème se pose quand il s'agit d'afficher un élément dans une grande phrase. Lors du dimensionnement d'un tableau sur le Spectrum, le dernier numéro de l'instruction définit la longueur de chaque élément du tableau. Par exemple, DIM a$(3,2,20) dimensionne un tableau de trois éléments sur deux, chaque élément ayant une longueur fixe de vingt caractères. Si nous assignons un élément du tableau à une chaîne de moins de vingt caractères, l'appoint se fait en ajoutant des espaces à la fin de la chaîne. Cela gaspille une place précieuse en mémoire. Aussi, pour insérer les variables tableau-chaîne dans des phrases, il nous faut d'abord supprimer tous les espaces terminaux. Les utilisateurs du Spectrum utiliseront, à cette fin, la routine suivante dans la Forêt hantée.

```
7000 REM****TRONCATURE SPECTRUM****
7010 FOR I=LEN(A$) TO 1 STEP-1
7020 IF A$(I TO I)<>«»THEN LET N=I:LET I=1
7030 NEXT I
7040 LET S$=S$+A$(TO N)
7050 RETURN
```

Pour le listage destiné au Digitaya, tapez les mêmes commandes, mais utilisez les numéros de lignes 8500 à 8550.

Cette routine tronque A$, et lui retire donc tous les espaces de queue avant de l'ajouter à S$. Souvenez-vous que S$ est la variable-chaîne utilisée pour assembler une phrase pour le formatage. Pour utiliser cette routine, il nous faut donner à la variable A$ l'élément du tableau-chaîne devant être ajouté à la phrase, et appeler ensuite le sous-programme. Il nous faut donc apporter les modifications suivantes aux versions Spectrum de la Forêt hantée et de Digitaya :

La Forêt hantée

```
2010 LET S$=«VOUS ÊTES»:A$=L$(P):GOSUB 7000:
  GOSUB 5500
2070 LET S$=S$+P$+«A»:A$=V$(I,1):
  GOSUB 7000:LET F=1:LET P$=«,»
```

Digitaya

```
1450 LET S$=«VOUS ÊTES»:A$=L$(P):
  GOSUB 8500:GOSUB 5880
1500 IF VAL(V$(I,2))=P THEN LET S$=S$+P$+«A»
  :A$=V$(I,1):GOSUB 8500:LET F=1:LET P$=«»
```

Listage Digitaya

La structure de Digitaya est similaire à celle de la Forêt hantée. Ajoutez les lignes suivantes aux listages donnés jusqu'ici :

```
1100 GOSUB6090:REM LIRE LES DONNEES DU TABLEAU
1210 PRINT:INPUT"INSTRUCTIONS";IS$
1120 P=47:REM POINT DE DEPART
1130 :
1140 REM **** LA BOUCLE PRINCIPALE COMMENCE ICI ****
1150 :
1160 MF=0:PRINT
1170 GOSUB1440:REM DECRIRE LA POSITION
1180 GOSUB1560: REM LISTER LES SORTIES
```

Ajouter aussi les lignes suivantes :

```
1220 NN$=IS$:GOSUB 2000:REM DÉPLACEMENT
1230 GOTO 1140:REM RECOMMENCER LA BOUCLE PRINCIPALE
```

Décrire la position et les sorties

```
1440 REM **** DECRIRE POSITION S/R ****
1450 SN$="VOUS ETES "+LN$(P):GOSUB5880
1460 SN$="VOUS VOYEZ "
1470 REM ** RECHERCHE D'OBJET **
1480 F=0:SP$=""
1490 FOR I=1TO8
1500 IF VAL(IV$(I,2))=P THEN SN$=SN$+SP$+"A
"+IV$(I,1):F=1:SP$="   "
1510 NEXTI
1520 IF F=0 THENSN$=SN$+ PAS D'OBJET"
1530 GOSUB5880:REM FORMAT
1540 RETURN
1550 :
1560 REM **** LISTER SORTIES S/R ****
1570 EX$=EX$(P)
1580 NR=VAL(GAUCHE$(EX$,2))
1590 EA=VAL(MID$(EX$,3,2))
1600 SO=VAL(MID$(EX$,5,2))
1610 WE=VAL(DROITE$(EX$,2))
1620 IF NO OR ES OR SU OR OU=0THEN RETURN
1630 PRINT:SN$="IL Y A DES SORTIES "
1640 IF NO<>0 THEN SN$=SN$+"NORD "
1650 IF ES<>0 THEN SN$=SN$+"EST "
1660 IF SU<>0 THEN SN$=SN$+"SUD "
1670 IF OU<>0 THEN SN$=SN$+"OUEST "
1675 GOSUB 5880:REM FORMAT
1680 PRINT:RETURN
```

Déplacement sur le sous-programme

```
2000 REM **** MOVE S/R ****
2010 MF=1:REM DEPLACER LE DRAPEAU ATTRIBUE
2020 DR$= GAUCHE$(NN$,1)
2030 IFOR$<>"N"ANDDR$<>"E"ANDDR$<>"S"ANDDR$
"W"TH EN2100
2040 IF DR$="N" AND NO<>0 THEN P=NO:RETURN
2050 IF DR$="S" AND SU<>0 THEN P=SU:RETURN
2060 IF DR$="E" AND ES<>0 THEN P=ES:RETURN
2070 IF DR$="O" AND OU<>0 THEN P=OU:RETURN
2080 PRINT"YOU CANT ";IS$
2090 MF=0:RETURN
2100 REM NOM INCORRECT
2110 PRINT"QU'EST ";NN$;" ?"
2120 MF=0:RETURN
```

Variantes de basic

Spectrum :

Dans les deux listages, remplacez EX$() par E$(), EX$ par X$, SN$ par S$, IS$ par T$, LN$() par L$(), NN$ par R$, SP$ par P$, DR$ par D$.

Pour le listage du Digitaya, substituez les lignes suivantes :

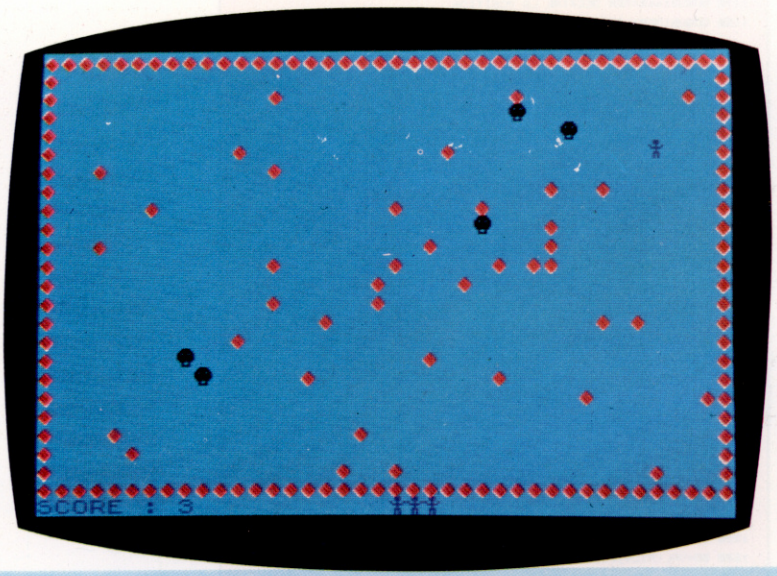
```
1580 LET NO=VAL(X$(TO 2))
1590 LET ES=VAL(X$(3 TO 4))
1600 LET SU=VAL(X$(5 TO 6))
1610 LET OU=VAL(X$(7 TO ))
2020 LET D$=R$(TO 1)
```

Pour le listage de la Forêt hantée, substituez les lignes suivantes :

```
210 RANDOMISE:P=INT(RND(1)*10+1)
2320 LET NO=VAL(X$(TO 2))
2330 LET ES=VAL(X$(3 TO 4))
2340 LET SU=VAL(X$(5 TO 6))
2350 LET OU=VAL(X$(7 TO))
3520 LET D$=R$(TO 1)
```

Robots sur MO5

Sur une planète défendue par des robots meurtriers, vous êtes seul, et le sol est truffé de mines... Vous connaissez le problème. Mais peut-être pas avec le MO5 de Thomson.



Les mines sont représentées sur l'écran par des losanges rouges. Au début du jeu, cinq robots sont présents sur le terrain. Sans perdre une seconde, ils se précipitent sur vous en suivant toujours le plus court chemin. Par chance, les robots sont aveugles et ne voient pas les mines qui se trouvent entre eux et vous, ce qui vous permet, en vous déplaçant judicieusement, de les éliminer. Utilisez pour cela le joystick ou les touches :

A, Z, E, Q, D, W, X, C,

suivant la direction que vous avez choisie. Lorsque tous les robots sont éliminés, le jeu reprend avec un robot supplémentaire. Si vous sautez sur une mine ou si un robot vous tue, tout n'est pas perdu. Vous disposez en effet de cinq vies. Si vous désirez changer le nombre de mines, modifiez la valeur de la variable NM à la ligne 80.

```
10 REM **********
20 REM * ROBOTS *
30 REM **********
40 DEFINT A-Z
50 CLEAR ,,3
60 NH=5
70 N1=5
80 NM=40
90 NR=N1
100 DIM R(30,1)
110 GOSUB 1580
120 GOSUB 1470
130 GOSUB 910
140 ON JS GOSUB 710,810
150 C=POINT(HX*8+4,HY*8+4)
160 IF C<>-13 AND C<>4 THEN 470
170 COLOR 4
180 LOCATE X,Y
190 PRINT N$;
200 LOCATE HX,HY
210 PRINT H$;
220 X=HX
230 Y=HY
240 T=0
250 FOR I=1 TO NR
260 IF R(I,0)=0 THEN 400
270 T=1
280 RX=R(I,0)+SGN(HX-R(I,0))
290 RY=R(I,1)+SGN(HY-R(I,1))
300 C=POINT(RX*8+4,RY*8+4)
310 IF C=1 OR C=0 THEN S=S+1:LOCATE R(I,
0),R(I,1):PRINT N$;:R(I,0)=0:GOTO 400
320 IF C=4 THEN 470
330 COLOR 0
340 LOCATE R(I,0),R(I,1)
350 PRINT N$;
360 LOCATE RX,RY
370 PRINT R$;
380 R(I,0)=RX
390 R(I,1)=RY
400 NEXT I
410 IF T=0 THEN 430
420 GOTO 140
430 S=S+10
440 IF INKEY$<>"" THEN 440
450 IF NR<30 THEN NR=NR+1
460 GOTO 130
470 NH=NH-1
480 COLOR 7
490 LOCATE X,Y
500 PRINT N$;
510 LOCATE HX,HY
520 PRINT H$;
530 PLAY "L96REL72REL24REL96REL72FAL24MI
L72MIL24REL72REL24DO#L96RE"
540 IF INKEY$<>"" THEN 540
550 IF NH>0 THEN NR=N1:GOTO 130
560 CLS
570 SCREEN 1,6,6
580 ATTRB 1,1
590 LOCATE 9,10
600 PRINT "SCORE :";S;
610 LOCATE 9,20
620 PRINT "UNE AUTRE ?";
630 COLOR 4
640 ATTRB 0,0
650 IF INKEY$<>"" THEN 650
660 D$=INKEY$
670 IF D$="" THEN 660
680 IF D$<>"N" THEN RUN
690 CLS
700 END
710 D$=INKEY$
720 IF D$="A" THEN HX=HX-1:HY=HY-1
730 IF D$="Z" THEN HY=HY-1
740 IF D$="E" THEN HY=HY-1:HX=HX+1
750 IF D$="Q" THEN HX=HX-1
760 IF D$="D" THEN HX=HX+1
770 IF D$="W" THEN HX=HX-1:HY=HY+1
780 IF D$="X" THEN HY=HY+1
790 IF D$="C" THEN HY=HY+1:HX=HX+1
800 RETURN
810 J=STICK(0)
820 IF J=1 THEN HY=HY-1
830 IF J=2 THEN HY=HY-1:HX=HX+1
840 IF J=3 THEN HX=HX+1
850 IF J=4 THEN HX=HX+1:HY=HY+1
860 IF J=5 THEN HY=HY+1
870 IF J=6 THEN HY=HY+1:HX=HX-1
880 IF J=7 THEN HX=HX-1
890 IF J=8 THEN HX=HX-1:HY=HY-1
900 RETURN
910 CLS
920 COLOR 4
930 LOCATE 0,24
940 PRINT "SCORE :";S;
950 IF NH=1 THEN 1000
960 FOR HX=1 TO NH-1
970 LOCATE 19+HX,24
980 PRINT H$;
990 NEXT HX
1000 COLOR 1
1010 FOR HX=0 TO 39
1020 LOCATE HX,0
1030 PRINT M$;
1040 LOCATE HX,23
1050 PRINT M$;
1060 NEXT HX
1070 FOR HY=1 TO 22
1080 LOCATE 0,HY
1090 PRINT M$;
1100 LOCATE 39,HY
1110 PRINT M$;
1120 NEXT HY
1130 FOR I=1 TO NM
1140 HX=INT(RND*38)+1
1150 HY=INT(RND*22)+1
1160 IF SCREEN(HX,HY)<>32 THEN 1140
1170 LOCATE HX,HY
1180 PRINT M$;
1190 NEXT I
1200 COLOR 0
1210 FOR I=1 TO NR
1220 R(I,0)=INT(RND*38)+1
1230 R(I,1)=INT(RND*22)+1
1240 IF SCREEN(R(I,0),R(I,1))<>32 THEN 1
220
1250 LOCATE R(I,0),R(I,1)
1260 PRINT R$;
1270 NEXT I
1280 HX=INT(RND*38)+1
1290 HY=INT(RND*22)+1
1300 IF SCREEN(HX,HY)<>32 THEN 1280
1310 X=HX
1320 Y=HY
1330 FOR I=1 TO 5
1340 LOCATE HX,HY
1350 COLOR 5
1360 PRINT CHR$(127);
1370 BEEP
1380 FOR J=1 TO 50
1390 NEXT J
1400 LOCATE HX,HY
1410 COLOR 4
1420 PRINT H$;
1430 FOR J=1 TO 50
1440 NEXT J
1450 NEXT I
1460 RETURN
1470 CLS
1480 SCREEN 4,12,0
1490 ATTRB 1,1
1500 LOCATE 10,10,0
1510 PRINT "JOYSTICK ?";
1520 ATTRB 0,0
1530 D$=INKEY$
1540 C=RND
1550 IF D$="" THEN 1530
1560 IF D$="O" THEN JS=2 ELSE JS=1
1570 RETURN
1580 DEFGR$(0)=28,28,73,62,8,28,20,20
1590 DEFGR$(1)=60,126,219,255,255,126,36
,60
1600 DEFGR$(2)=0,0,24,60,126,126,60,24
1610 H$=GR$(0)
1620 R$=GR$(1)
1630 M$=GR$(2)
1640 N$=CHR$(32)
1650 RETURN
```

Derniers ordres

Nous allons considérer le mécanisme d'interruption utilisé pour transférer le contrôle entre le programme de débogage et le programme à déboguer aux points d'interruption.

Le mécanisme d'interruption est utilisé aux points d'interruption dans le programme initial, où nous avons remplacé une instruction par SWI (SoftWare Interrupt). L'opc (code opération) SWI, comme les autres interruptions sur le 6809, est vectorisé par un emplacement mémoire spécifique, à savoir $FFFA. Cela signifie que lorsqu'un SWI est exécuté, les registres sont sauvegardés sur la pile et le processeur charge l'adresse 16 bits en $FFFA et $FFFB dans le compteur de programme (PC). L'exécution continue à partir de cette adresse. Notre tâche consiste à modifier ce vecteur afin qu'il indique l'entrée de notre programme débogueur. Le problème ici est que les vecteurs d'interruption sont presque toujours en ROM. Le fait que ces adresses soient fixées implique donc que le système d'exploitation ait d'autres moyens pour vectoriser des interruptions.

Normalement, on a une table de saut contenue dans une mémoire de travail RAM; celle-ci n'est normalement pas disponible pour les programmes, mais elle est réservée à l'usage du système d'exploitation. L'adresse indiquée par le vecteur contient une instruction JMP suivie d'une adresse, qui normalement revient au système d'exploitation.

Toutefois, nous pouvons changer cette adresse en celle que nous voulons, de sorte que la première instruction exécutée après SWI sera un saut (JMP) à l'adresse d'entrée du débogueur. Nous devons faire attention à remplacer le contenu original de la table de saut avant la fin de l'exécution de notre programme, parce qu'il est toujours possible que le système d'exploitation exécute ensuite un SWI. Il faut rappeler que le 6809 a trois SWI, et il n'y a pas de raison de ne pas utiliser SW12 (opc 10 3F et vecteur en $FFF4) ou SW13 (opc 11 3F et vecteur en $FFF2) — quoique, ceux-ci utilisant des opc sur 2 octets, il soit nécessaire de modifier le programme débogueur en conséquence.

Un autre problème vient de notre programme qui peut seulement occuper la partie de mémoire laissée libre par le programme à déboguer. Le débogueur doit donc être relogeable. On aura remarqué que toutes les références à des emplacements mémoire dans le programme ont été (ou auraient dû être) faites à l'aide d'adressage indexé de compteur de programme.

La question est qu'il faut savoir, à ce niveau, l'adresse absolue de l'entrée du programme afin de pouvoir la placer dans la table de saut d'interruption. Cette adresse doit être calculée au moment de l'exécution, puisque l'assembleur ne peut pas la manier.

Notre première tâche consiste alors à calculer cette adresse et à l'insérer dans la table de saut. Notez que l'adresse du point d'entrée pour SWI sera différente de celle du début du programme débogueur, car la routine, à l'adresse de début du programme, doit manier cette procédure d'initialisation, laquelle ne sera pas nécessaire lorsque nous rentrerons dans le progamme via SWI. Par conséquent, nous traiterons toute l'initialisation à l'intérieur d'une routine; le point d'entrée sera alors l'adresse contenant l'instruction après l'appel BSR à ce sous-programme. Heureusement, cette adresse est précisément celle qui est sauvegardée sur pile par BSR; nous pouvons ainsi la lire sur la pile afin de la placer correctement dans la table de saut.

L'autre étape de cette procédure d'initialisation est d'obtenir l'adresse de départ du programme à déboguer. Voici la conception d'ensemble :

Procédure d'initialisation

Données :
Adresse-vecteur est l'adresse à trouver en $FFFA dans X.
Opc-JMP est l'opc pour l'instruction JMP dans A.
Adresse-entrée est l'adresse du point d'entrée dans Y.
Adresse-départ du programme à déboguer dans D.

Traitement :
Entre Adresse-vecteur.
Stocke Opc-JMP à Adresse-vecteur.
Entre Adresse-entrée.
La stocke en (Adresse-vecteur + 1).
Entre Adresse-départ au clavier.
La sauvegarde.

Nous pouvons maintenant compléter le codage des trois commandes restantes. Il faut encore considérer la commande R qui affiche le contenu des registres. Bien sûr, nous ne voulons pas afficher le contenu actuel des registres pendant le débogage, mais au moment de l'interruption. C'est-à-dire que nous voulons connaître les valeurs qui étaient placées sur pile par l'instruction SWI. Cependant, il y aura d'autres valeurs au-dessus de celles-là sur la pile au moment où nous en aurons besoin. Nous pourrions certes calculer le nombre d'octets indésirables sur la pile et obtenir les valeurs des registres en écartant ces derniers. Mais il est plus simple de sauvegarder la valeur du pointeur de pile aussitôt

B	Insérer interruption
U	Supprimer interruption
D	Afficher interruptions en cours
S	Début exécution progr.
G	Go (reprendre depuis interruption programme)
R	Affiche contenu des registres
M	Inspecte et change emplacements mémoire
Q	Quitte

après l'interruption, afin de pouvoir s'y référer ensuite.

En codant la commande R, nous supposerons que cela a été fait, de sorte que nous pouvons récupérer ces contenus. La structure de la routine est évidente — nous prenons simplement chaque valeur, sans la sortir de la pile, et l'affichons avec le label approprié, la seule exception sera la valeur de S — ce devrait être la valeur précédant l'interruption.

Elle peut être obtenue en additionnant la valeur appropriée à la valeur sauvegardée de S que nous utilisons pour référencer les valeurs de registres sur pile.

Commande R

Données :

Pointeur-pile est la valeur du sommet de la pile après interruption dans X.
Valeur-1-octet contient les valeurs des registres à un octet dans B.
Valeur-2-octets contient les valeurs des registres 16 bits dans D.
Labels contient les labels pour les neuf registres.

Traitement :

Entre pointeur-pile.
Charge CC dans Valeur-1-octet.
Affiche label (1), Valeur-1-octet.
Répète ci-dessus pour A, B et DP.
Charge X dans Valeur-2-octets.
Affiche label (5), Valeur-2-octets.
Répète ci-dessus pour Y, U et PC.
Additionne 12 à la valeur initiale de Pointeur-pile.
Affiche label (9), Pointeur-pile.

Il reste deux commandes : Q, pour quitter le programme, ne nécessite pas de routine spéciale; et G, pour reprendre l'exécution du programme après une interruption. En ce point, il nous faut remplacer l'instruction SWI qui a causé l'interruption par l'instruction initiale et repasser le contrôle à cette instruction. Nous pouvons restaurer assez simplement les registres à leurs contenus initiaux, simplement à l'aide de RTI qui les désempile tous. Nous devons cependant faire attention car la valeur de PC qui est désempilée sera la valeur de l'instruction suivante; étant donné qu'elle est supérieure d'une unité à la valeur requise, il faut ajuster la valeur avant de retourner.

Commande G

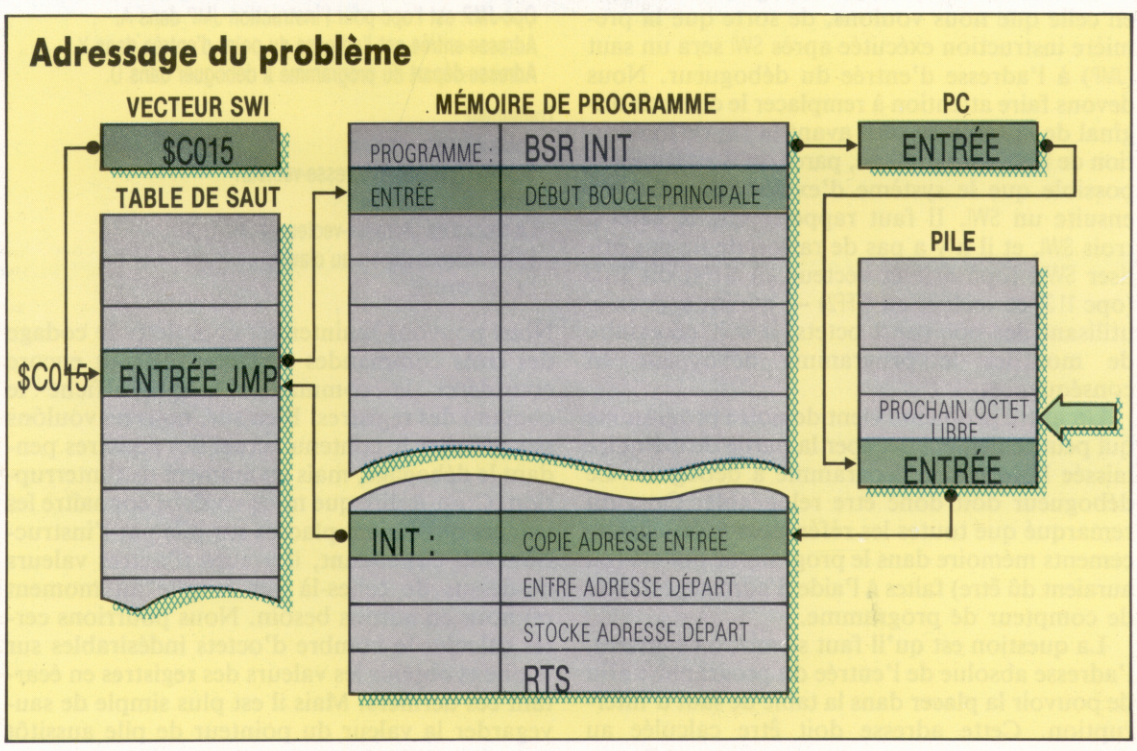
Données :

Table-d'interruptions est une table d'adresses 16 bits de points d'interruptions.
Valeurs-supprimées est une table d'opc remplacés par des SWI.
Interruption-suivante est un nombre compris entre 1 et 16.
Pointeur-pile est la valeur sauvegardée du pointeur de pile après SWI.

Traitement :

Si Interruption-suivante > 0 et < = 16 alors
Entre opc de Valeurs-supprimées (Interruption-suivante).
Stocke à l'adresse dans Table-d'interruptions (Interruption-suivante).
Met S à Pointeur-Pile.
Décrémente valeur de PC sur pile.
Incrémente Interruption-suivante.
Retour d'interruption.
Sinon
Retour de sous-programme.

Notre série de langage machine 6809 se termine dans le prochain numéro, où nous coderons le module principal de notre débogueur et considérerons l'ensemble du traitement effectué par le programme.



Une pile de temps
Le programme débogueur commence par un appel BSR à la routine d'initialisation, suivi par le début de la boucle du programme principal. L'une des tâches d'initialisation consiste à déterminer l'adresse absolue du début de cette boucle, et à la copier dans la table de saut d'interruption de sorte que, lorsqu'un SWI est exécuté, le contrôle passe par la table de saut et revienne au début de la boucle. Cette adresse ne peut être connue d'avance car le programme doit être entièrement relogeable; heureusement, l'adresse de retour empilée par BSR est précisément l'adresse en question, de sorte que la routine d'initialisation a simplement besoin de la copier de la pile à la table de saut.

Procédure d'initialisation

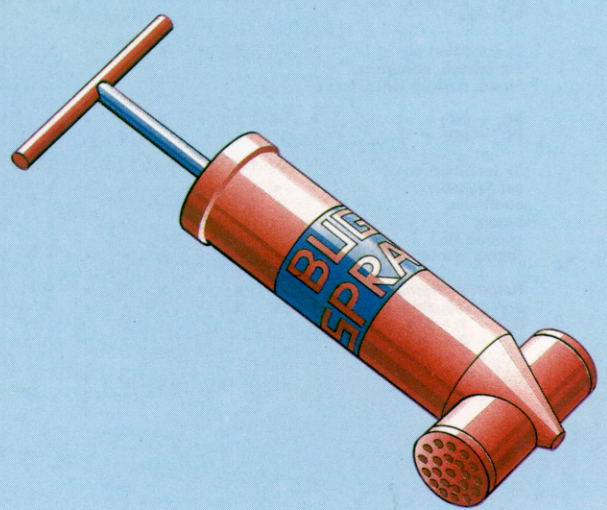
```
START   RMB   2          Pour sauvegarder l'adresse de départ
OPJMP   FCB   $OE        Opc-JMP
INIT    LDX   $FFFA      Entre Adresse-vecteur
        LDA   OPJMP,PCR  Entre Opc-JMP et
        STA   ,X+        Le sauvegarde à Adresse-vecteur
        LDY   1,S        Entre Adresse-entrée à partir de pile
        STY   ,X         Sauvegarde à Adresse-vecteur + 1
        BSR   GETADD     Entre Adresse-départ au clavier
        STD   START,PCR  Sauvegarde
        RTS              Retour
```

Commande R

```
STACKP  RMB   2          Pointeur-pile
LABELS  FCC   'CC A BDP XY
                 UPC S'
SPACE   FCB   32         Code ASCII pour espace
CMDR    PSHS  A,B,X,Y    Sauvegarde registres utilisés
        LDX   STACKP,PCR Entre Pointeur-pile
        LEAY  LABELS,PCR Utilise Y pour pointer label
        LDA   #4         Nombre de registres à 1 octet
FOR01   BSR   CMDR1      Affiche registre suivant
        DECA             Quatre fois
        BGT   FOR01
        LDA   #4         Nombre de registres 2 octets
FOR02   BSR   CMDR2      Affiche registre suivant
        DECA             Quatre fois
        BGT   FOR02
        LDA   ,Y+        1er caractère de label
        BSR   OUTCH      Affiche
        LDA   ,Y+        2e caractère de label
        BSR   OUTCH      Affiche
        LDA   SPACE,PCR  Affiche espace
        BSR   OUTCH
        TFR   X,D        X contient maintenant la valeur de S voulue
        BSR   DSPADD     Affiche S
        PULS  A,B,X,Y,PC Restaure et retour
* Sous-programme pour afficher un registre à un octet
CMDR1   PSHS  A          Sauvegarde A
        LDA   ,Y+        1er caractère de label
        BSR   OUTCH      Affiche
        LDA   ,Y+        2e caractère de label
        BSR   OUTCH      Affiche
        LDA   SPACE,PCR  Affiche espace
        BSR   OUTCH
        LDB   ,X+        Entre registre suivant dans Valeur-1-octet
        BSR   DSPVAL     Affiche Valeur-1-octet
        PULS  A,PC       Restaure A et retour
* Sous-programme pour afficher un registre à deux octets
CMDR2   PSHS  A          Sauvegarde A
        LDA   ,Y+        1er caractère de label
        BSR   OUTCH      Affiche
        LDA   ,Y+        2e caractère de label
        BSR   OUTCH      Affiche
        LDA   SPACE,PCR  Affiche espace
        BSR   OUTCH
        LDD   ,X++       Entre registre suivant dans Valeur-2-octets
        BSR   DSPADD     Affiche Valeur-2-octets
        PULS  A,PC       Restaure A et retour
```

Commande G

```
BPTAB   RMB   32         Table-d'interruptions
REMTAB  RMB   16         Valeurs-supprimées
NEXTBP  RMB   1          Interruption-suivante
CMDG    PSHS  A          Sauvegarde A dans le cas d'un retour normal
        LDA   NEXTBP,PCR Interruption-suivante
IF04    BLE   ENDF04     Si Interruption-suivante > 0
        CMPA  MAXBP,PCR  et < = 16
        BGT   ENDF04     (nombre max. d'interruptions)
        DECA             Convertit en décalé dans table
        LEAX  BPTAB,PCR  Adresse de Table-d'interruption
        LEAY  REMTAB,PCR Adresse de Valeurs-supprimées
        LDB   A,Y        Entre Valeurs-supprimées
        LSLA             Convertit A en décalé pour table de 16 bits
        STB   [A,X]      Stocke à l'adresse dans Table-d'interruptions
        LDS   STACKP,PCR Entre Pointeur-pile dans S
        DEC   10,S       Ajuste valeur de PC sur pile
        INC   NEXTBP,PCR Incrémente Interruption-suivante
        RTI              Retour d'interruption
ENDF04  PULS  A,PC       Restaure et retour
```



Carte Z80 (suite et fin)

Voici reproduit, avec l'aimable autorisation de Zilog Inc., la fin de la carte référence qui vient en complément du programmeur Z80, et de nos articles sur le langage machine.

Groupes d'Appel et de Retour et Reprise

Groupes d'Appel et de Retour

			CONDITION									
			UN COND.	RETENUE	PAS DE RETENUE	ZÉRO	PAS DE ZÉRO	PARITÉ PAIRE	PARITÉ IMPAIRE	SIGNE NÉG.	SIGNE POS.	REG. B = 0
Appel « CALL »	**Extension immédiate**	nn	CD n n	DC n n	D4 n n	CC n n	C4 n n	EC n n	E4 n n	FC n n	F4 n n	
Retour « RET »	**Registre indirect**	(SP) (SP + 1)	C9	D8	D0	C8	C0	E8	E0	F8	F0	
Retour de Int « RETI »	**Registre indirect**	(SP) (SP + 1)	ED 4D									
Retour de Int non masquable « RETN »	**Registre indirect**	(SP) (SP + 1)	ED 45									

Note : Certains drapeaux ont plus d'une fonction.
Se reporter au manuel technique Z80 pour plus de détails.

Groupe de reprise

	OPC	
Adresse d'appel	0000_H	**C7** **'RST 0'**
	0008_H	**CF** **'RST 8'**
	0010_H	**D7** **'RST 16'**
	0018_H	**DF** **'RST 24'**
	0020_H	**E7** **'RST 32'**
	0028_H	**EF** **'RST 40'**
	0030_H	**F7** **'RST 48'**
	0038_H	**FF** **'RST 56'**

Mnémonique	Opération symbolique	Drapeaux S	Z	H	P/V	N	C	Opc 76	543	210	Hex	Nombre d'octets	Nombre de cycles M	Nombre d'états T	Commentaires
CALL nn	(SP − 1) ← PC_H (SP − 2) ← PC_L PC ← nn	•	•	X	•	X	•	•	•	11 ← ←	001 n n	101 → →	CD	3	5	17	
CALL cc, nn	Si condition cc est fausse, continuer. Sinon, comme CALL nn	•	•	X	•	X	•	•	•	11 ← ←	cc n n	100 → →		3 3	3 5	10 17	Si cc est faux Si cc est vrai
RET	PC_L ← (SP) PC_H ← (SP + 1)	•	•	X	•	X	•	•	•	11	001	001	C9	1	3	10	
RET cc	Si condition cc est fausse, continuer. Sinon, comme RET	•	•	X	•	X	•	•	•	11	cc	000		1 1	1 3	5 11	Si cc est faux Si cc est vrai
RETI	Retour d'interruption	•	•	X	•	X	•	•	•	11 01	101 001	101 101	ED 4D	2	4	14	
RETN1	Retour d'interruption non masquable	•	•	X	•	X	•	•	•	11	101	101	ED	2	4	14	
RST p	(SP − 1) ← PC_H (SP − 2) ← PC_L PC_H ← 0 PC_L ← 0	•	•	X	•	X	•	•	•	11	1	111		1	3	11	

cc	Condition
000	NZ non nul
001	Z nul
010	NC pas de retenue
011	C retenue
100	PO parité impaire
101	PE parité paire
110	P signe positif
111	M signe négatif

t	p
000	00H
001	08H
010	10H
011	18H
100	20H
101	28H
110	30H
111	38H

Note : 'RETN charge $IFF_2 \rightarrow IFF_1$

Notation de drapeaux : • = drapeau non affecté, 0 = drapeau à zéro ;
1 = drapeau mis, X = drapeau inconnu ;
↕ = drapeau affecté suivant le résultat de l'opération.