CALCULATRICES

TI-82
le "top" des jeux !

Vincent Bastid et Emmanuel Neuville

Transformez votre calculatrice TI-82 en console de jeux !

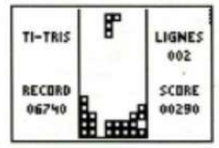
DUNOD TECH

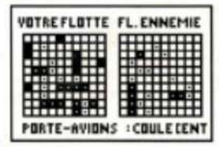
TEXAS INSTRUMENTS

CALCULATRICES

TI-82
le "top" des jeux !

Vincent Bastid et Emmanuel Neuville





La compilation des meilleurs jeux !

Exploitez toutes les possibilités de votre TI-82 grâce aux programmes de ce livre qui la transformeront en véritable console de jeux !

TI-TRIS, BATAILLE NAVALE LINK, CASSE-BRIQUES, GOB-MAN, DEMINEUR, etc..., autant de grands classiques que nous avons sélectionnés pour faire partie de ce "top" des jeux d'action et de réflexion sur TI-82.

Et au-delà de "l'enfer du jeu", découvrez également le plaisir de la programmation, avec les astuces qui vous permettront de repousser les limites de votre calculatrice dans vos propres programmes !

9 782100 019854

ISBN 2 10 001985 6 Code 041985

Environnement : ***Calculatrice Texas Instruments TI-82***

Thème : ***Jeux et programmation***

Utilisateur : ***Lycéen, Etudiant***

DUNOD EDITEUR

CALCULATRICES

TI-82
le "top" des jeux !

Vincent Bastid et Emmanuel Neuville

DUNOD TECH

TEXAS INSTRUMENTS

Illustration de couverture : Rachid Maraï

ISBN 2 10 001985 6

SOMMAIRE

CONTENU DE CE LIVRE

Dans ce livre vous trouverez des programmes de jeu et de détente qui feront appel à votre adresse, votre astuce, votre jugement et aussi à votre chance... Mais comment s'y retrouver au milieu de cette foule de programmes ? Pour vous aider à faire votre choix, en début de chapitre est indiqué pour chaque jeu :

Photos d'écran

Une ou plusieurs photos d'écran qui montrent des scènes caractéristiques des jeux.

Descriptif

Le descriptif de la première page vous expliquera la situation et vous mettra dans l'ambiance du jeu !

Nombre de joueurs

Indique combien de joueurs peuvent jouer *simultanément* au jeu. Cela permet de différencier deux catégories de jeux : ceux à 1 joueur où vous tenterez d'améliorer votre score, et où vous aurez éventuellement la machine comme adversaire, et les jeux à 2 joueurs, où vous affronterez un ami dans des compétitions acharnées !

Mémoire nécessaire :

Programme : x
Données : y

Indispensable avant de commencer à entrer un programme, vérifiez que vous disposez bien de suffisamment de mémoire libre dans votre machine pour le faire tenir (c'est la mémoire "programme"), et qu'ensuite celui-ci trouvera suffisamment de mémoire pour créer ses variables (c'est la mémoire "données").

But du jeu :

Sans doute le minimum à lire pour pouvoir jouer ! Mais connaître les règles et avoir lu le mode d'emploi est souvent appréciable.

Règles :

Les règles à connaître pour jouer. Pour les jeux où l'action et les réflexes priment, cette partie n'est pas présente. Pour les jeux plutôt orientés vers la réflexion, elle est plus ou moins longue suivant la complexité du jeu.

Mode d'emploi :

Le mode d'emploi décrit comment fonctionne le programme que vous utiliserez, et vous indique quelles touches presser, ce qui est représenté à l'écran, etc...
Eventuellement quelques conseils de jeu sont présents.

Le programme :

Cette partie commence par la liste des modules qui composent le jeu, que vous devez entrer dans votre machine. Les modules dont le nom commence par 'X' sont réutilisés dans de nombreux programmes ; si vous avez déjà en mémoire, inutile de les retaper.
Les modules qui commencent par 'Y' sont spécifiques d'un jeu. Ils ne donnent sauf exception aucun résultat par eux-mêmes, et ne sont jamais exécutés directement. Comme la TI-82 classe dans sa liste les programmes par ordre alphabétique, ces sous-programmes dont le nom commence par Y se retrouvent en fin de liste, après les programmes exécutables.

Quand vous rentrez des programmes :
- respectez les espaces blancs
- ne remplacez pas les majuscules par des minuscules ou inversement
- le signe (-) est la touche (-) sur votre clavier (signe de négation)
- commencez de préférence par entrer les sous-programmes avant le programme principal

Les commentaires en français sur la droite des pages ne sont pas à entrer, ils sont présents juste pour la compréhension des programmes.

Commentaires

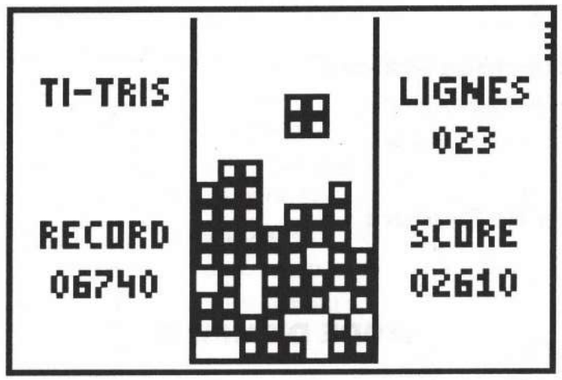
Après le programme, les commentaires viendront éclaircir pour ceux qui le souhaitent la manière dont certains problèmes posés par la réalisation du programme ont été résolus. Il s'agit parfois d'astuces de programmation qui ont été utilisées à l'occasion, et surtout de méthodes générales sur l'optimisation des matrices ou les résolutions d'équations de physique, que vous pourrez réutiliser dans vos propres programmes.

Sur la droite des pages de programme, les commentaires décrivent point par point son fonctionnement.

Utilisez ces commentaires pour :

- rechercher vos erreurs de frappe si le programme ne fonctionne pas ou mal
- comprendre le fonctionnement du programme
- faire des modifications ou des ajouts en toute connaissance de cause.

Ces commentaires ont pour but d'éclaircir la programmation, et de présenter des méthodes pour résoudre les problèmes courants posés par la réalisation des jeux. Nous espérons que ces lignes enlèveront un peu de mystère à la programmation et vous donneront envie de faire vos propres réalisations !



TI-TRIS

Avec TI-TRIS, vous allez transformer votre TI-82 en console de jeu et retrouver un des plus gros succès de jeux ! Dans TITRIS, des formes géométriques tombent du haut de l'écran ; à vous de les déplacer pendant leur chute afin qu'elles s'encastrent les une dans les autres et forment des lignes. Restez vigilant et ne vous laissez pas déborder par la succession de pièces, car alors vous risqueriez de perdre la tête et voir bientôt la fin de la partie !

But du jeu :

Emboiter les formes reçues afin de faire des lignes.

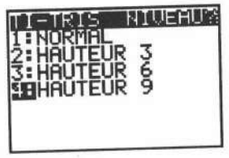
Mémoire nécessaire :

Programme : 2000 octets
Données : 800 octets

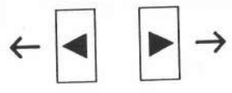
Nombre de joueurs : 1

MODE D'EMPLOI

Exécutez TITRIS. Le menu des options s'affiche. Choisissez "NORMAL" pour le niveau le plus facile. Si vous vous sentez plus en confiance, prenez l'un des autres choix. Suivant la hauteur que vous choisirez, un certain nombre de blocs seront déjà présents sur le terrain de jeu, corsant la difficulté...

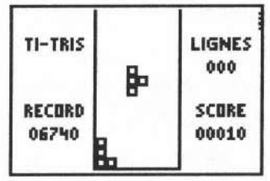


L'écran de jeu apparait. Une pièce tirée au hasard surgit en haut de l'écran, utilisez les flèches gauche et droite pour la déplacer, et la touche [2nd] pour la faire pivoter d'un quart de tour dans le sens inverse des aiguilles d'une montre.



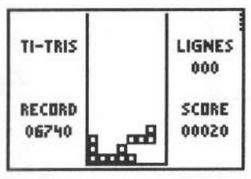
Vous pouvez également presser la flèche vers le bas pour accélérer sa chute. Arrivée en bas, la pièce se pose sur le fond ou les autres pièces existantes. Puis une autre pièce arrive, que vous devez également disposer en bas de l'écran.

Le but du jeu est de former des lignes horizontales de blocs. Après avoir posé une pièce, si une ou plusieurs lignes sont réalisées, elles sont supprimées, et rapportent des points en fonction du nombre de lignes faites simultanément :



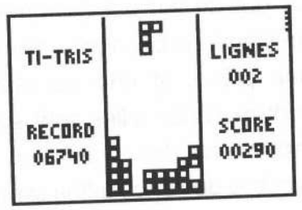
1 ligne	100 points
2 lignes	200 points
3 lignes	400 points
4 lignes	800 points

De plus, les lignes sont supprimées, et les blocs qui se trouvaient au-dessus descendent d'un ou plusieurs étages vers le bas. Par ailleurs, chaque pièce posée rapporte 10 points.



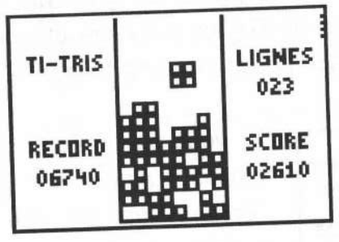
Si vous ne réalisez aucune ligne, les pièces s'accumulent dans le bas de l'écran et forment une pyramide qui va monter jusqu'en haut. Si elle atteint le sommet de l'écran, vous avez perdu !

Sur la droite de l'écran, votre score est affiché. Au-dessus, le nombre de lignes que vous avez réalisées. Vous pouvez également consulter le record, sur la gauche de l'écran, afin d'évaluer vos progrès.



Quand vous serez familiarisé avec le jeu, vous pourrez choisir au démarrage un niveau de difficulté plus élevé, en demandant au menu de départ une hauteur de blocs de 3, 6 ou 9 étages. Des blocs seront disposés en bas de l'écran de jeu, de façon aléatoire, ce qui complique sérieusement la réalisation de lignes !

Réussirez-vous à ramener la hauteur de ces blocs à 1 seul étage seulement ?



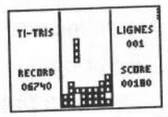
LE PROGRAMME

TI-TRIS est composé des modules suivants :

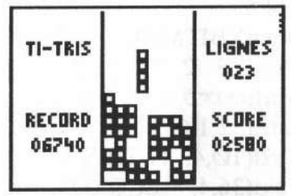
TITRIS	commence une nouvelle partie
YTIAFFIC	affiche la pièce
YTIEFFAC	efface la pièce
YTIINIT	initialise une nouvelle partie
YTILIGNE	teste les lignes de blocs
YTIPIECE	tire une nouvelle pièce
YTITEST	teste si la position de la pièce

```
Programme TITRIS              commence une nouvelle partie
 :prgmYTIINIT
 :Lbl Ø
 :prgmYTIPIECE                tire une nouvelle pièce
 :If N
 :Text(19,83-4int log N,N)    affiche le nombre de lignes
 :[B]→[D]
 :Lbl 1
 :prgmYTIAFFIC                affiche la pièce
 :I→E                         sauvegarde sa position
 :J→F
 :Ø→A
 :Ø→T
 :Repeat K+(T>2Ø)             attend 20 tours la pression
 :getKey→K                    d'une touche
 :T+1→T
 :End
 :If K=24                     flèche à gauche
 :Then
 :I-1→I                       déplace vers la gauche
```

```
:1→A
:Else
:If K=26                    flèche à droite
:Then
:I+1→I                      déplace vers la droite
:1→A
:Else
:If K=21                    touche [2nd]
:Then
:[C][B]→[B]                 fait pivoter la pièce
:1→A
:End
:End
:End                        on a pressé une touche ?
:If A
:Then
:prgmYTITEST                teste la nouvelle position
:If O                       elle n'est pas possible
:Then
:E→I                        on restaure les anciens paramètres
:[D]→[B]
:Else                       sinon la position est valide
:prgmYTIEFFAC               on efface la pièce
:I→E                        on mémorise les nouvelles valeurs
:[B]→[D]
:prgmYTIAFFIC               on la réaffiche
:End
:End
:J-1→J                      la pièce descend d'un étage
:prgmYTITEST                on teste sa nouvelle position
:If O                       on a touché quelquechose ?
:Goto 2                     fin pour cette pièce
:prgmYTIEFFAC               sinon on efface la pièce
:Goto 1                     et retour au début
:Lbl 2
```



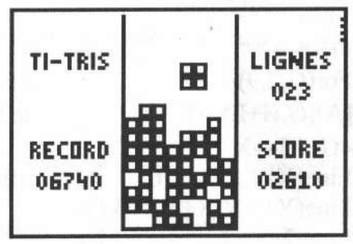

```
:If J=R-1                   la pièce n'a pas bougé ?
:Goto 3                     c'est la fin de la partie
:prgmYTILIGNE               sinon teste les lignes réalisées
:Text(45,87-4int log S,S)   affiche le score
:Goto Ø                     pièce suivante
:Lbl 3                      fin du jeu
:Text(Ø,39,"GAME")
:Text(7,41,"OVER")
:If S>W                     record battu
:S→W                        on sauve le record

Programme YTIAFFIC          affiche la pièce
 :1→C
 :For(B,1,4)
 :4(I+[B](1,B))+25→X
 :4(J+[B](2,B))-6→Y
 :Line(X,Y,X+3,Y)
 :Line(X+3,Y,X+3,Y+3)
 :Line(X,Y+3,X+3,Y+3)
 :Line(X,Y,X,Y+3)
 :End

Programme YTIEFFAC          efface la pièce
 :Ø→C
 :For(B,1,4)
 :4(E+[D](1,B))+25→X
 :4(F+[D](2,B))-6→Y
 :Line(X,Y,X+3,Y,C)
 :Line(X+3,Y,X+3,Y+3,C)
 :Line(X,Y+3,X+3,Y+3,C)
 :Line(X,Y,X,Y+3,C)
 :End

Programme YTIINIT           initialise une nouvelle partie
 :ClrHome
```



```
:Menu("TI-TRIS  NIVEAU?","NORMAL",Ø,"HAUTEUR
3",1,"HAUTEUR 6",2,"HAUTEUR 9",3)
:Lbl Ø
:Ø→H                          hauteur Ø
:Goto 5
:Lbl 1
:5→H                          hauteur 3
:Goto 5
:Lbl 2
:8→H                          hauteur 6
:Goto 5
:Lbl 3
:11→H                         hauteur 9
:Lbl 5
:prgmXBITMAP                  initialise l'écran
:Vertical 32
:Vertical 65
:Line(32,1,65,1)
:Text(1Ø,4,"TI-TRIS")
:Text(36,4,"RECORD")
:Text(45,6,"ØØØØØ")
:If W<1                       "log" n'accepte pas les
:1→W                          valeurs nulles
:Text(45,22-4int log W,W)     affiche le record
:Text(1Ø,69,"LIGNES")
:Text(19,75,"ØØØ")
:Text(36,71,"SCORE")
:Text(45,71,"ØØØØØ")
:{11,2Ø}→dim [A]              matrice contenant le jeu
:Fill(Ø,[A])
:For(I,1,2Ø)
:1→[A](1,I)                   des "1" sur les bords...
:1→[A](1Ø,I)
:If I<1Ø
:1→[A](I,1)                   ...et sur le fond du terrain de jeu
```

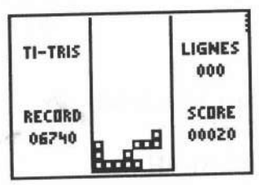


```
:End
:Ø→S
:Ø→N
:[[Ø,(-)1][1,Ø]]→[C]          matrice de rotation des pièces
:Ø→P
:15→R
:2→J
:While J<H                    remplit la matrice de blocs
:4J-6→Y
:For(I,2,9)
:int (2rand)→C
:C→[A](I,J)
:If C                         on affiche le bloc
:Then
:4I+25→X
:Line(X,Y,X+3,Y)
:Line(X+3,Y,X+3,Y+3)
:Line(X,Y+3,X+3,Y+3)
:Line(X,Y,X,Y+3)
:End
:End
:J+1→J
:End
```

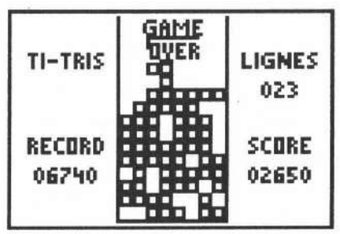


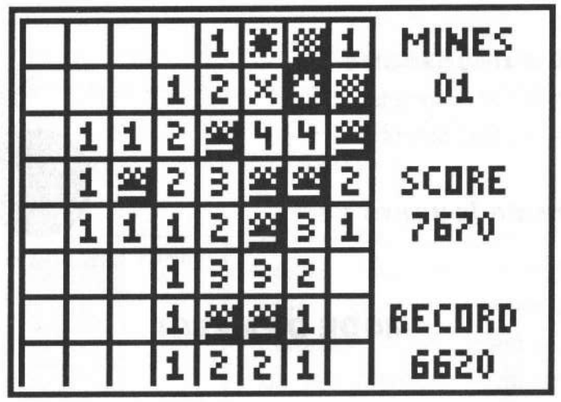

```
Programme YTILIGNE            teste les lignes de blocs
  :For(B,1,4)                 mémorise la nouvelle pièce
  :F+[D](2,B)→H               dans la matrice de jeu
  :If H>P
  :H→P                        hauteur max de la pièce posée
  :1→[A](E+[D](1,B),H)
  :End
  :F+2→M
  :Ø→L                        nombre de lignes trouvées
  :F-2→D                      hauteur minimum à tester
```

```
:If D<2
:2→D
:For(H,D,P)                    pour chaque étage de l'écran
:Repeat G≠11
:2→G
:If H<M                        étage susceptible d'avoir des lignes
:Then
:While [A](G,H+L)*(G<11)       il y a un bloc ?
:G+1→G
:End
:If G=11                       on a trouvé 8 blocs
:L+1→L                         une ligne de plus
:End
:End
:If L                          on a trouvé des lignes ?
:Then
:4H-6→Y
:For(G,2,9)                    décale les blocs d'un étage vers
:[A](G,H+L)→C                  le bas
:4G+25→X
:Line(X,Y,X+3,Y,C)             affiche les nouveaux blocs
:Line(X+3,Y,X+3,Y+3,C)
:Line(X,Y+3,X+3,Y+3,C)
:Line(X,Y,X,Y+3,C)
:C→[A](G,H)
:End
:End
:End
:S+1Ø→S                        10 points par pièce
:If L
:S+5Ø*2^L→S                    et un bonus pour la ligne
:N+L→N                         nombre de lignes réalisées
:P-L→P                         hauteur de la pile
```




```
Programme YTIPIECE             tire une nouvelle pièce
:int (5rand)→I                 5 pièces possibles
:If I=Ø
:[[Ø,Ø,Ø,Ø][Ø,(-)1,1,2]]→[B]
:If I=1
:[[Ø,Ø,Ø,1][Ø,(-)1,1,1]]→[B]
:If I=2
:[[Ø,Ø,Ø,(-)1][Ø,(-)1,1,1]]→[B]
:If I=3
:[[Ø,(-)1,1,Ø][Ø,Ø,Ø,1]]→[B]
:If I=4
:[[Ø,1,Ø,1][Ø,Ø,1,1]]→[B]
:Ø→I
:int (4rand)→J                 rotation initiale
:While I≤J                     J = nombre de quarts de tour
:[C][B]→[B]                    rotation de la pièce
:I+1→I
:End
:5→I                           position X et...
:R→J                           ...position Y initiales de la pièce

Programme YTITEST              teste si la position de la pièce
:1→O                           est valide
:For(B,1,4)
:If [A](I+[B](1,B),J+[B](2,B))
:Return
:End
:Ø→O
```





DEMINEUR

Attention ! Vous êtes au milieu du champ de mines ! Vous avez été envoyé en éclaireur pour nettoyer cette zone de toutes les mines qui y ont été laissées. Avec votre détecteur, vous pouvez déterminer combien de mines sont proches de l'endroit où vous vous trouvez, mais il ne vous dit pas où elles se trouvent ! A vous de corréler les résultats de vos mesures afin de déterminer par déduction où sont cachées les mines. Mais attention aux fausses manoeuvres : ne plantez pas votre détecteur juste sur une mine, car alors c'est l'explosion !!!

But du jeu :

Déterminer les emplacements des mines dissimulées.

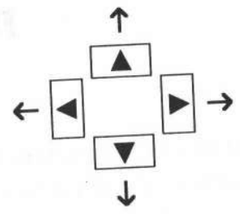
Mémoire nécessaire :

Programme : 2000 octets
Données : 800 octets

Nombre de joueurs : 1

MODE D'EMPLOI

Exécutez le programme DEMINEUR. La zone à déminer s'affiche, chaque case représentée par un bouton en relief étant susceptible de contenir une mine. Déplacez le curseur clignotant avec les flèches sur la case que vous désirez explorer, puis pressez [Enter].

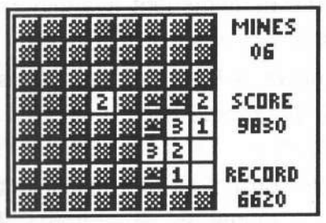


2nd poser un marqueur

3 cas peuvent se produire :

- vous trouvez une mine... BOOOMMMM !!! c'est l'explosion ! Hélas c'est déjà la fin du jeu, car votre corps a explosé en mille morceaux... Dans vos derniers instants, vous avez le temps de voir la position de toutes les mines qu'il restait à découvrir, ainsi que les mines que vous aviez crues trouver mais qui n'existaient pas... Vous ferez mieux la prochaine fois !

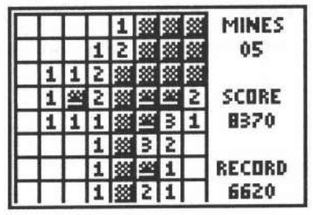
- il n'y a pas de mines, mais un chiffre s'affiche : il s'agit du nombre de mines dans les cases adjacentes à celle explorée, horizontalement, verticalement et en diagonale : c'est un chiffre entre 1 et 8
- il n'y a pas de mines, et aucun chiffre ne s'affiche, car il n'y a aucune mine dans les cases adjacentes : on peut alors explorer sans risque ces 8 cases, ce que fait le programme automatiquement pour vous. Tout va bien !!



Après avoir exploré quelques cases, vous avez peut-être une idée de l'emplacement de certaines mines : c'est le moment de les marquer d'un signe, qui vous empêchera par la suite d'explorer ces cases piégées.

Déplacez votre curseur sur la case à marquer, pressez la touche [2nd] pour y déposer un marqueur. Pressez de nouveau [2nd] pour le retirer.

Au fur et à mesure que vous déposez vos marqueurs, le nombre en haut à droite de l'écran diminue : il compte le nombre de mines qu'il vous reste à trouver, *selon vous !*Arrivé à zéro, vous avez trouvé toutes les mines, et toutes les cases inexplorées restantes sont logiques vides de toute mine... *à condition que vous n'ayez pas fait d'erreur !*



Ainsi vous n'avez pas gagné pour autant : votre victoire sera totale uniquement lorsque toutes les cases sauf les 10 contenant les mines auront été explorées.

Pendant que vous réfléchissez, le temps passe, et votre score diminue. Plus vous êtes rapide, plus vous marquerez de points, mais attention à ne pas faire de fausse manoeuvre qui pourrait vous en coûter...

Votre score est au milieu à droite de l'écran, il diminue en permanence, mais il n'est réaffiché que lorsque c'est un multiple de 100. En dessous, le record est affiché. C'est le score à battre ! Pour le moment il est sans doute à 0, mais certainement il atteindra rapidement des hauteurs raisonnables !...

Si le score arrive à 0, toutes les mines explosent ! Soyez rapide !

Besoin de faire un calcul urgent ou tout simplement de quitter le programme ? Pressez la touche [^], la sortie est instantanée.

Fin du jeu :

Si vous gagnez, vous sortez du programme, et un dernier écran vous propose deux choix :

- presser la touche [GRAPH] pour revoir l'écran de jeu; pressez ensuite [CLEAR] pour effacer la ligne de menu
- presser la touche [ENTER] pour commencer une nouvelle partie. La commande qui avait lancé le programme sera réexécutée.

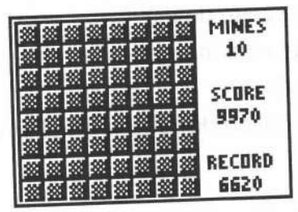
LE PROGRAMME :

DEMINEUR	programme principal
YDEMINIT	préparation d'une nouvelle partie
YDEMCOUP	saisie du coup du joueur
YDEMCURS	dessine le curseur
YDEMSCOR	affiche le score
YDEMTRIA	dessine un marqueur
YDEMMINE	dessine une mine
YDEMCROI	dessine une croix
YDEMEFFA	efface une case
YDEMNBMI	affiche le nombre de mines
YDEMLITC	lit une case du jeu
YDEMTEST	compte les mines adjacentes
YDEMFIN	fin du jeu
XBITMAP	initialisation de l'écran graphique

Programme DEMINEUR :

```
:1Ø→N                  nombre de mines
:64-N→Z                nombre de cases à explorer
:prgmXBITMAP           initialisation de l'écran
:prgmYDEMINIT          init jeu, affiche la grille
:prgmYDEMNBMI          affiche N
:Lbl Ø
:prgmYDEMCOUP          saisie du coup
:[A](I+1,J+1)→C        contenu de la case
:If C=2                case déjà explorée ?
:Goto Ø                → retour
:If M=2                on a posé un marqueur ?
:Goto 1
:If C≥3                il y a une marque ?
```

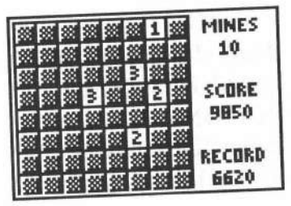
```
:Goto Ø                  → retour
:If C=1                  il y a une mine !
:prgmYDEMFIN             → fin du jeu...
:I→K
:J→L
:prgmYDEMTEST            affiche le nb de mines autour de K,L
:If Z=Ø                  tout est exploré ?
:prgmYDEMFIN             → gagné
:If H                    au moins une ?
:Goto Ø                  → retour
:K→F                     on va explorer maintenant les
:L→G                     cases autour de chacune des 8
:For(L,G-1,G+1)          cases vides adjacentes à F,G
:For(K,F-1,F+1)
:K→I
:L→J
```




```
:prgmYDEMLITC            lecture de la case K,L
:If C=Ø                  elle est vide ?
:prgmYDEMTEST            → affiche le nb de mines en K,L
:End
:End
:F→I                     I,J redevient la position que
:G→J                     l'on vient de jouer
:If Z                    il reste des cases inexplorées ?
:Goto Ø                  → retour
:prgmYDEMFIN             non : gagné !
:Lbl 1                   on a posé un marqueur
:1→H
:If C>1                  case déjà marquée
```



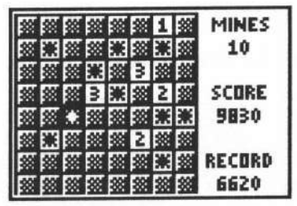
```
:(-)1→H                  → la marqueur sera retiré
:[A](I+1,J+1)+3H→[A](I+1,J+1)
:prgmYDEMTRIA            affichage du marqueur
:N-H→N
:prgmYDEMNBMI            affiche le nombre de mines
:Goto Ø                  retour

Programme YDEMINIT :     préparation d'une nouvelle partie
:For(I,Ø,64,8)           tracé de la grille 8x8
:Vertical I
:Line(1,I,65,I)
:Vertical I+1
:Line(1,I+1,65,I+1)
:End
:For(I,Ø,7)              affichage des boutons en relief
:For(J,Ø,7)              dans les cases
:8J+3→X
:8I+2→Y
:Pt-Off(X-1,Y-1)
:Line(X,Y+1,X+1,Y)
:Line(X,Y+3,X+3,Y)  :Line(X+1,Y+4,X+4,Y+1)
:Line(X+3,Y+4,X+4,Y+3)
:Pt-Off(X+5,Y+5)
:End
:End
:{8,8}→dim [A]           initialisation de la matrice
:Fill(Ø,[A])             contenant les valeurs des cases
:For(H,1,N)              tirage aléatoire des N mines
:Repeat [A](I,J)=Ø       jusqu'à trouver une case vide
:1+int (8rand)→I         position X
:1+int (8rand)→J         position Y
:End
:1→[A](I,J)              on mémorise la mine
:End
:Text(Ø,7Ø,"MINES")
```



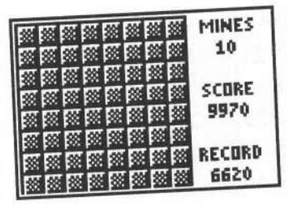

```
:Text(24,7Ø,"SCORE")
:Text(48,68,"RECORD")
:Text(56,72,W)
:1ØØØØ→T
:4→I
:4→J

Programme YDEMCOUP :          saisie du coup du joueur
 :Ø→M
 :Repeat M                    jusqu'à [Enter] ou [2nd]
 :Ø→P                         P=Ø : blanc  P=1 : noir
 :prgmYDEMCURS                affiche le curseur en I,J
 :Repeat K                    jusqu'à ce qu'on presse une touche
 :prgmYDEMCURS                le curseur clignote
 :1-P→P                       car P alterne entre Ø et 1
 :1→L
 :Repeat K+(L>2Ø)             on attend 2Ø tours de boucle max
 :1+L→L
 :getKey→K                    lit la touche pressée
 :End
 :T-1Ø→T                      le score diminue...
 :prgmYDEMSCOR                affiche le score
 :End
 :1→P                         restaure le cadre noir
 :prgmYDEMCURS
 :If (K=25)*(J<7)             flèche vers le haut
 :J+1→J
 :If (K=24)*(I>Ø)             flèche vers la gauche
 :I-1→I
 :If (K=26)*(I<7)             flèche vers la droite
 :I+1→I
 :If (K=34)*(J>Ø)             flèche vers le bas
 :J-1→J
 :If K=1Ø5                    touche [Enter]
```



```
 :1→M                         → exploration de la case
 :If K=21                     touche [2nd]
 :2→M                         → pose d'un marqueur
 :If K=55                     touche [^]
 :Then                        → sortie immédiate
 :ClrHome
 :Disp
 :Stop
 :End
 :End
```




```
Programme YDEMCURS :          dessine le curseur
 :8I+1→X
 :8J→Y
 :Line(X,Y,X+8,Y,P)           c'est un carré de couleur P
 :Line(X+8,Y,X+8,Y+8,P)       (blanc ou noir)
 :Line(X,Y+8,X+8,Y+8,P)
 :Line(X,Y,X,Y+8,P)

Programme YDEMSCOR :          affiche le score
 :If T>999                    score de 4 chiffres ?
 :Then
 :Text(32,72,T)
 :Else
 :If T>99                     score de 3 chiffres ?
 :Then
 :Text(32,72,Ø)               précédé d'un Ø
 :Text(32,76,T)
 :Else                        score de 2 chiffres
 :Text(32,76,Ø)               avec un Ø aux centaines
 :Text(32,8Ø,T)
 :If T=Ø                      temps à zéro ?
 :Then                        → c'est la fin...
```




```
:Text(4Ø,7Ø,"BOUM!")
:99→K
:prgmYDEMFIN
:End
:End
:End
```

Programme YDEMTRIA : dessine un marqueur

```
:8I+4→X
:8J+3→Y
:Pt-Change(X,Y)          c'est un triangle
:Pt-Change(X+2,Y)        affiché en surimpression
:Pt-Change(X+1,Y+1)      sur le motif en relief
:Pt-Change(X-1,Y-1)      dessinée sur la case
:Pt-Change(X+2,Y-1)      non explorée
```

Programme YDEMMINE : dessine une mine

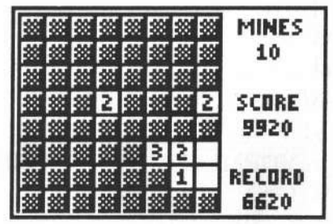
```
:8I+2→X
:8J+7→Y
:If 1ØI+J=K              c'est la mine qui a causé
:Then                    l'explosion ?
:Line(X,Y,X+6,Y)         elle est affichée en inverse
:Line(X,Y,X,Y-6)
:Line(X+3,Y-2,X+3,Y-4,Ø)
:Line(X+2,Y-3,X+4,Y-3,Ø)
:Else
:For(P,1,6)              on inverse les points déjà
:For(O,1,6)              affichés. attention : Y-P
:Pt-Change(X+O,Y-P)
:End
:End
:Line(X+2,Y-3,X+4,Y-3)   puis on finit le graphisme
:Line(X+3,Y-2,X+3,Y-4)
:End
```


Programme YDEMEFFA : efface une case

```
:8I+3→X
:8J+1→Y
:For(P,Ø,5)
:Line(X,Y+P,X+5,Y+P,Ø)   on efface ligne par ligne
:End
```

Programme YDEMCROI : dessine une croix

```
:prgmYDEMEFFA
:8I+3→X
:8J+2→Y
:Line(X,Y,X+4,Y+4)
:Line(X,Y+4,X+4,Y)
```



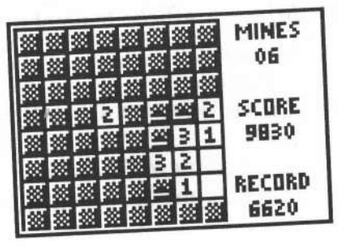
Programme YDEMNBMI : affiche le nombre de mines

```
:If N>9
:Then
:Text(8,76,N)            nombre à 2 chiffres
:Else
:Text(8,76,Ø)            un zéro
:Text(8,8Ø,N)            nombre à 1 chiffre
:End
```

Programme YDEMLITC lit une case du jeu

```
:(-)1→C                  on lit la case I,J
:If I(I-7)>Ø             I est hors-limite
:Return                  → retour
:If J(J-7)>Ø             J est hors-limite
:Return                  → retour
:[A](I+1,J+1)→C          sinon valeur de la case
```

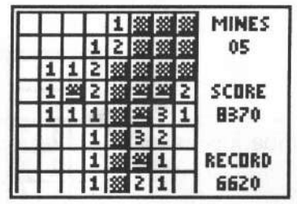
```
Programme YDEMTEST :        compte les mines adjacentes
  :2→[A](I+1,J+1)           la case est marquée "explorée"
  :Z-1→Z                    une case de moins à explorer
  :Ø→H                      H compte le nb de mines
  :For(J,L-1,L+1)           la case centrale est K,L
  :For(I,K-1,K+1)           la case I,J "tourne" autour de K,L
  :prgmYDEMLITC             on lit la case I,J
  :If (C=1)+(C=4)           elle contient une mine ?
  :H+1→H
  :End
  :End
  :K→I
  :L→J
  :prgmYDEMEFFA             efface la case K,L
  :If H=Ø
  :Return
  :Text(56-8J,8I+3,H)       affiche le nombre de mines
```




```
Programme YDEMFIN :         fin du jeu
  :If Z                     il restait des cases inexplorées ?
  :Then                     → perdu
  :1ØI+J→K                  n° de la mine qui a explosé
  :Text(32,72,"ØØØØ")       le score passe à Ø
  :For(J,7,Ø,(-)1)          affichage des mines restantes
  :For(I,Ø,7)               on balaye toute la matrice
```

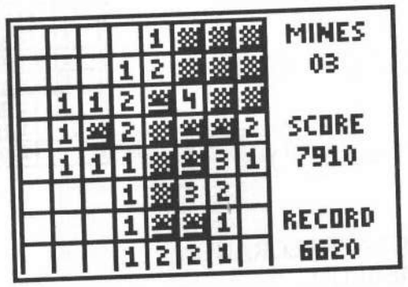


```
  :[A](I+1,J+1)→C           on lit la case I,J
  :If C=1                   c'est une mine sans marqueur
  :prgmYDEMMINE             → affiche une mine
  :If C=3                   c'est un marqueur sans mine
  :prgmYDEMCROI             → affiche une croix
  :End
  :End
  :Pause
  :Disp
  :ClrHome
  :Output(2,4,"BOOOUUMMM !")
  :Else                     sinon : gagné
  :Disp
  :ClrHome
  :If T>W
  :Then
  :Output(2,2,"RECORD BATTU !")
  :T→W
  :Else
  :Output(2,5,"GAGNE !!!")
  :End
  :End
  :Output(4,1,"[GRAPH] ECRAN")
  :Output(5,9,"DE JEU")
  :Output(7,1,"[ENTER] NOUVELLE")
  :Output(8,9,"PARTIE")
  :Stop                     fin (pas de retour)
```




```
Programme XBITMAP :         initialisation de l'écran graphique
  :FnOff
  :PlotsOff
  :GridOff
  :LabelOff
  :FullScreen
  :RectGC
```

```
:1→Xmin
:95→Xmax
:Ø→Xscl
:1→Ymin
:63→Ymax
:Ø→Yscl
:DispGraph
:ClrDraw
```



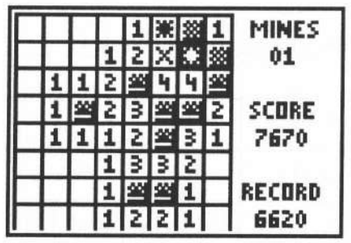
COMMENTAIRES

La surface de jeu est un carré de 8x8 cases, stockées dans une matrice 8x8 : [A].
Le contenu de chaque case est représenté par un code issu du tableau suivant :

codes des cases	sans mine	avec mine
case inconnue sans marqueur	0	1
case explorée	2	boum !
case inconnue avec marqueur	3	4

Au début du jeu, les cases sont toutes inconnues. Celles qui sont vides sont à 0, celles contenant une mine à 1.
Lorsqu'on pose un marqueur sur une case, on augmente son code de 3. Lorsqu'on retire le marqueur, on retranche 3 au code.
Lorsqu'on explore une case dans mine, son code passe à 2.

Il n'y a pas de code prévu pour une case explorée contenant une mine, car... cela ne correspond pas à un état durable du jeu !!!



Pour lire une case du jeu, on met son abscisse dans I, et son ordonnée dans J, puis on appelle le module YDEMLITC. Pourquoi appeler un sous-programme plutôt que de lire directement la matrice ? Tout simplement parce que ce sous-programme détecte si on demande une case qui est en dehors de la matrice, comme cela peut arriver lorsqu'on explore les cases autour de la case jouée. Cela permet d'avoir des boucles plus simples, en rejetant les tests de dépassement des limites de la matrice dans un sous-programme (lire en dehors d'une matrice provoque une erreur)

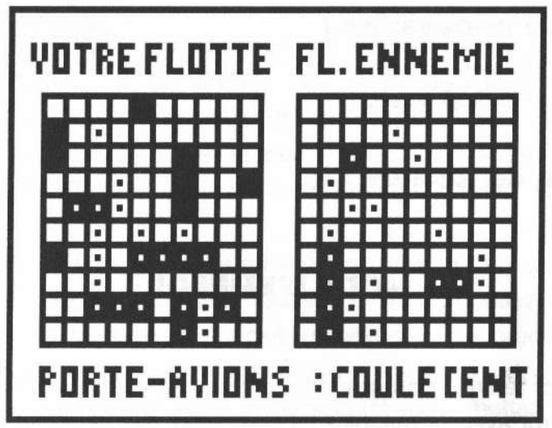
Ainsi dans YDEMTEST, pour calculer le nombre de mines à afficher en K,L, on écrit :

```
:0→H
:For(J,L-1,L+1)
:For(I,K-1,K+1)
:prgmYDEMLITC
:If (C=1)+(C=4)
:H+1→H
:End
:End
```

la case centrale est K,L
la case I,J "tourne" autour de K,L
on lit la case I,J
elle contient une mine ?

Les cases testées sont celles autour de K,L :

K-1,L+1	K,L+1	K+1,L+1
K-1,L	K,L	K+1,L
K-1,L-1	K,L-1	K+1,L-1



BATAILLE NAVALE-LINK

Reliez deux TI-82 avec le câble fourni en standard et jouez à la bataille navale avec un ami, chacun utilisant sa propre machine ! Vous découvrirez grâce à ce jeu les avantages que procure cette nouvelle calculatrice : jouer simultanément à deux au même jeu, avoir toute la surface de l'écran pour soi, et surtout ne pas risquer des coups d'oeil indiscrets de votre adversaire sur, par exemple, la position de vos bateaux...

Les jeux à deux calculatrices : une première ! Vous avez choisi la bonne machine !

But du jeu :

Trouver et couler les dix navires ennemis

Mémoire nécessaire :

Programme : 2100 octets
Données : 1500 octets

Nombre de joueurs : 2

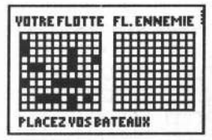
MODE D'EMPLOI

Après avoir entré le programme, connectez les deux TI-82 par le câble fourni avec la machine et trasnférez par le menu LINK les modules composant le jeu, c'est-à-dire NAVALE et les sous-programmes commençant par YNA, ainsi que XBITMAP (reportez-vous au manuel pour le transfert de programmes entre deux machines).

Exécutez sur chaque machine le programme NAVALE. Le jeu commence par le placement des bateaux, qui doit se faire au même moment sur les deux machines.

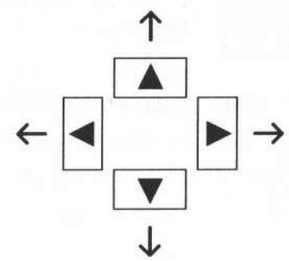
Vous disposez de :

- 1 porte-avions (4 cases)
- 2 croiseurs (3 cases)
- 3 destroyers (2 cases)
- 4 sous-marins (1 case)



Le programme va vous proposer successivement chaque bateau, en commençant par les plus longs. Sur la grille de gauche, un rectangle clignotant de la longueur du bateau s'affiche, déplacez-le avec les flèches, et pressez la touche VARS pour le déposer. Avec la touche [2nd], vous pouvez le faire pivoter d'un quart de tour.

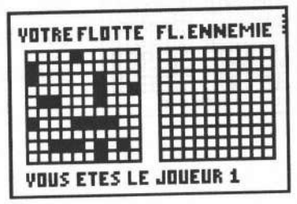
Placez vos 10 bateaux sur la grille. Vous devez laisser au minimum une case vide entre chaque bateau ; si vous déposez une pièce trop près d'un navire déjà posé, le navire placé précédemment sera retiré. Cela vous permet de corriger vos erreurs de placement simplement en superposant vos bateaux et en les mettant aux bons emplacements sans vous poser de question.



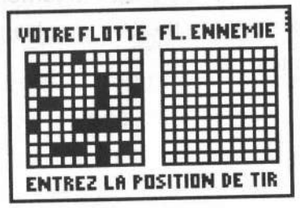
2nd rotation

VARS placer le bateau

Une fois vos bateaux posés, la machine se mettra en attente de synchronisation avec l'autre machine. Vous verrez un message du type :
VOUS ETES LE JOUEUR n
Si vous êtes le premier à avoir terminé le placement des bateaux, vous serez le joueur 1, sinon, vous serez le joueur 2.



Si vous ne voyez pas un des messages ou si les deux machines affichent le même numéro de joueur, vous avez un problème de connexion entre les machines. Voyez plus bas la rubrique "Problèmes de connexion".



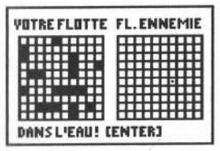
La partie commence. C'est au joueur 1 d'ouvrir le feu : déplacez le curseur sur la grille de droite, et pressez [VARS] pour désigner la case sous le curseur.

Le temps que votre obus atteigne sa cible, le résultat du tir s'affiche en bas de l'écran. De plus la case visée reçoit un petit point si le coup a atterri dans l'eau, ou se noircit si vous avez touché quelque chose.

Tant que le joueur 1 n'a pas tiré, le joueur 2 est bloqué. Une fois votre tir effectué, il peut presser [ENTER].
Note : l'autre joueur n'a aucun moyen de savoir à quel moment se presser [ENTER], aussi vous devez le prévenir ; ou bien l'autre

joueur peut presser régulièrement [ENTER] jusqu'à ce qu'il soit débloqué (le jeu se déroulera normalement).

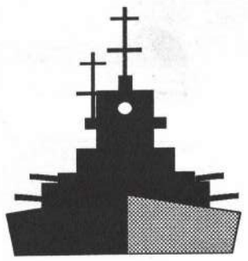
Quand le joueur 2 a été débloqué, il passe automatiquement à son tour en mode de tir. Sur la grille de gauche, une croix clignotante indique où l'autre joueur a tiré. Le tir fonctionne comme décrit ci-dessus pour le joueur 1.



Quant au joueur 1, il est bloqué à son tour. Il devra presser [ENTER] dès que son adversaire aura tiré.

Les deux joueurs tirent ainsi en alternance, jusqu'à ce que tous les navires d'un des joueurs aient été coulés. Un message de victoire apparaît alors sur l'écran du joueur qui vient de tirer. L'autre joueur peut presser [ENTER] et voir un message de fin sur son écran.

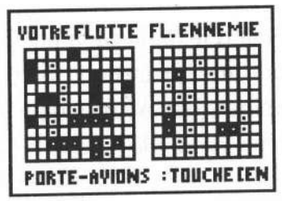
Problèmes de connexion



Si vous rencontrez des problèmes de connexion entre les machines, essayez les solutions suivantes :
- débranchez et rebranchez les prises
- enfoncez les prises bien à fond
- testez la connexion avec les commandes du menu LINK.
- vérifiez que tous les programmes ont bien été transmis

Si la connexion avec LINK fonctionne, et que vous venez juste de placer vos bateaux, plutôt que de relancer NAVALE, exécutez le programme YNACONT afin de passer la phase de placement. Bien

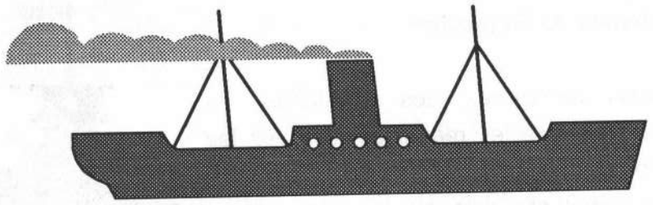
sur aucune variable utilisée par le programme ne devra avoir été modifiée, ni l'écran graphique altéré.



Si un problème de connexion survient en cours de partie, essayez simplement de rebrancher les prises. Si le problème persiste, tapez la ligne :

:Ø→Z

et pressez [ENTER] à partir de l'écran de calcul, puis relancez NAVALE.



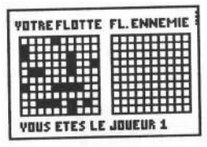
LE PROGRAMME

Bataille Navale - Link est composé des modules suivants :

NAVALE	commence une nouvelle partie
YNACARRE	dessine un bloc de bateau
YNACOMM	synchronisation des machines
YNACOUP	gère le déplacement du curseur
YNACURS	affiche le curseur
YNAEFFAC	efface la ligne des messages
YNAJOUEU	affiche le numéro du joueur
YNAPLACE	placement des bateaux
YNARECEP	reçoit un tir de l'autre joueur
YNATIR	tir du joueur
YNATOUCH	affiche le bateau touché
YNACONT	continue une partie

```
Programme NAVALE          commence une nouvelle partie
 :Ø→P                     variables pour la synchronisation
 :Ø→R
 :Ø→S
 :If Z≠123                code utilisé par YNACONT
 :prgmYNAPLACE            placement des bateaux
 :6→I                     position du curseur
 :7→J
 :[A]→[B]                 recopie la position des bateaux
 :Get(P)                  lit le code de synchro A
 :If P=123                c'est le bon code ?
 :Then
 :1→M                     nous sommes la machine 1
```

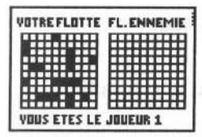
```
:Get([B])                lit les données de l'autre machine
:246→R                   stocke le code de synchro B
:prgmYNAJOUEU            affiche le numéro de joueur : 1
:Else
:123→P                   stocke le numéro de synchro A
:Ø→M                     nous sommes la machine Ø
:prgmYNAJOUEU            affiche le numéro de joueur : 2
:Repeat R=246            attend le code de synchro B
:Pause
:Get(R)                  lit le code de synchro B
:End
:[A]→[C]
:Get([A])                lit les données de l'autre machine
:[A]→[B]
:[C]→[A]
:End
```




```
:Ø→T                     à partir d'ici les deux machines sont
:Ø→C                     synchronisées et portent un n° M
:1→D                     différent : Ø ou 1
:If M                    machine 1 ?
:Goto 1                  va en attente
:Lbl Ø
:prgmYNATIR              le joueur tire
:Lbl 1
:prgmYNACOMM             attend le tir de l'autre joueur
:Get(C)                  reçoit les coordonées de tir
:Get(D)
:prgmYNARECEP            affiche le tir de l'autre joueur
```



```
:Goto Ø                  et on recommence...

Programme YNACARRE       dessine un bloc de bateau
:4E-2+47H→U              à la position E,F,
:4F+4→V                  du coté H;
:Line(U,V,U+2,V,G)       et de la couleur G
:Line(U,V+1,U+2,V+1,G)   G=1 : noir
:Line(U,V+2,U+2,V+2,G)   G=Ø : blanc
```




```
Programme YNACOMM        synchronisation des machines
:If M                    machine 1
:Then
:321+T→P                 envoie le "top" T à l'autre machine
:Repeat S=321+T          se met en attente du top T
:Pause
:Get(S)                  lit la valeur du top
:End
:T+1→T                   T augmente
:Else                    machine Ø
:321+T→S                 envoie le "top" T à l'autre machine
:T+1→T                   T augmente
:Repeat P=321+T          se met en attente du top T
:Pause
:Get(P)                  lit la valeur du top
:End
:End
```

```
Programme YNACOUP          gère le déplacement du curseur
 :Repeat K=44
 :If I+L(1-O)>11           vérifie que le curseur ne sort
 :11-L(1-O)→I              pas de la grille
 :If J+LO>11
 :11-LO→J
 :Ø→P
 :prgmYNACURS              efface le curseur
 :Repeat K
 :1-P→P
 :prgmYNACURS              fait clignoter le curseur
 :1→V
 :Repeat K+(V>15)          attend la pression d'une touche
 :1+V→V
 :getKey→K                 lit une touche
 :End
 :End
 :1-P→P
 :If P
 :prgmYNACURS              réaffiche le curseur
 :If (K=25)                flèche vers la haut
 :J+1→J
 :If (K=24)*(I>2)          vers la gauche
 :I-1→I
 :If (K=26)                vers la droite
 :I+1→I
 :If (K=34)*(J>2)          vers le bas
 :J-1→J
 :If K=21
 :1-O→O
 :End
```



```
Programme YNACURS          affiche le curseur
 :4I-3+47H→X
 :4J+3→Y
 :Line(X,Y,X+4+4L(1-O),Y,P) un carré
 :Line(X+4+4L(1-O),Y,X+4+4L(1-O),Y+4+4LO,P)
 :Line(X,Y+4+4LO,X+4+4L(1-O),Y+4+4LO,P)
 :Line(X,Y,X,Y+4+4LO,P)
 :Pt-Change(4C-1,4D+5)     et une croix (utilisée pour
 :Pt-Change(4C-2,4D+4)     afficher la position de tir
 :Pt-Change(4C-2,4D+6)     de l'autre joueur)
 :Pt-Change(4C,4D+4)
 :Pt-Change(4C,4D+6)
```

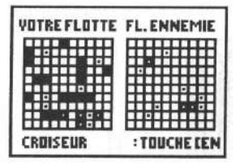

```
Programme YNAEFFAC         efface la ligne des messages
 :For(Y,3,7)
 :Line(5,Y,95,Y,Ø)
 :End

Programme YNAJOUEU         affiche le numéro du joueur
 :prgmYNAEFFAC
 :Text(55,4,"VOUS ETES LE JOUEUR")
 :Text(55,76,M+1)

Programme YNAPLACE         placement des bateaux
 :prgmXBITMAP
 :For(I,Ø,1Ø)              dessine les grilles
 :Line(5,4I+11,45,4I+11)
 :Line(4I+5,11,4I+5,51)
 :Line(52,4I+11,92,4I+11)
 :Line(4I+52,11,4I+52,51)
```

```
:End
:Text(3,2,"VOTRE FLOTTE")
:Text(3,51,"FL. ENNEMIE")
:Text(55,4,"PLACEZ VOS BATEAUX")
:1Ø→N                          nombre de bateaux à placer
:Ø→O
:{12,12}→dim [A]               matrice contenant les bateaux
:Fill(Ø,[A])
:1Ø→dim Li
:Fill(Ø,Li)                    liste contenant l'état des bateaux
:5→I
:6→J
:Ø→H
:(-)2Ø→D
```

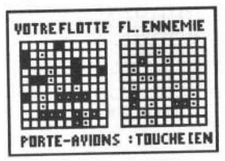
```
:While N                       tant qu'il reste de bateaux à placer
:1Ø→B
:While Li(B)                   recherche le plus long bateau
:B-1→B                         non encore placé
:End                           c'est B en sortie de boucle
:(B>4)+(B>7)+(B>9)→L            longueur de bateau B
:prgmYNACOUP                   saisie d'une position
:For(Y,J-1,J+1+OL)
:For(X,I-1,I+1+L(1-O))
:[A](X,Y)→C                    lit la case I,J
:If C                          elle est occupée par le bateau C
:Then
:Ø→G
```



```
:For(F,2,11)                   lit la matrice
:For(E,2,11)
:If [A](E,F)=C                 en E,F il y a le bateau C
:Then
:Ø→[A](E,F)                    on le supprime !
:prgmYNACARRE                  on l'efface de l'écran
:End
:End
:End
:Ø→Li(C)                       on note qu'il n'est plus placé
:N+1→N
:End
:End
:End
:I→E
:J→F
:1→G
:L+1→Li(B)
:While L≥Ø                     place les éléments du bateau B
:B→[A](E,F)
:prgmYNACARRE                  et l'affiche à l'écran
:E+1-O→E
:F+O→F
:L-1→L
:End
:N-1→N
:End                           tous les bateaux sont placés
:prgmYNAEFFAC                  efface la ligne de message
:1Ø→Q                          nombre de bateaux
:2Ø→N                          nombre de cases occupées

Programme YNARECEP             reçoit un tir de l'autre joueur
  :prgmYNAEFFAC                efface la ligne de message
  :If [A](C,D)                 il y avait un bateau ici !
```



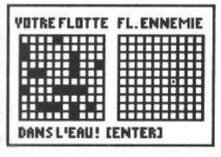
```
:Then
:Ø→[A](C,D)                    il n'y en a plus...
:N-1→N                         une case de moins occupée
:If N=Ø                        il n'en reste aucune
:Then
:Text(55,4,"TOUTE LA FLOTTE A COULE !")
:Ø→Z
:Stop                          c'est la fin du jeu : on a perdu
:End
:End
```




```
Programme YNATIR               tir du joueur
  :Text(55,4,"ENTREZ LA POSITION DE TIR")
  :1→H
  :Ø→L
  :prgmYNACOUP                 saisit une position
  :prgmYNAEFFAC                efface la ligne de message
  :Pt-Change(4C-1,4D+5)
  :I→C
  :J→D
  :[B](I,J)→B                  y a-t-il un bateau ?
  :If B                        oui : le bateau B
  :Then
  :Ø→[B](I,J)                  il est effacé
  :1→G
  :I→E
```



```
:J→F
:prgmYNACARRE
:Li(B)-1→A                     sa longueur diminue
:prgmYNATOUCH                  affiche le message correspondant
:A→Li(B)
:If A=Ø                        il est coulé ?
:Then
:Q-1→Q                         un bateau de moins
:If Q=Ø                        plus aucun bateau restant ?
:Then
:prgmYNAEFFAC                  on a gagné !
:Text(55,33,"VICTOIRE !")
:321+T→S                       envoie le code à l'autre machine
:S→P
:Ø→Z
:Stop
:End
:End
:Else                          sinon le coup est dans l'eau
:Text(55,4,"DANS L'EAU !  [ENTER]")
:End
:Pt-Change(4I-1+47H,4J+5)
```

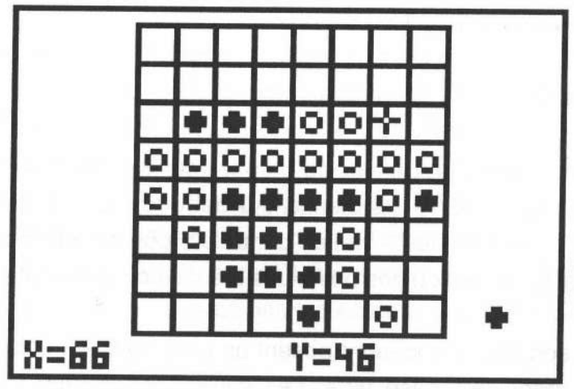



```
Programme YNATOUCH             affiche le bateau touché
  :If B<5
  :Then
  :Text(55,4,"SOUS-MARIN")
```

```
:Else
:If B<8
:Then
:Text(55,4,"DESTROYER")
:Else
:If B<1Ø
:Then
:Text(55,4,"CROISEUR")
:Else
:Text(55,4,"PORTE-AVIONS")
:End
:End
:End
:If A                          il reste au moins une case
:Then                          au bateau ?
:Text(55,55,": TOUCHE [ENTER]")
:Else                          sinon il est coulé
:Text(55,55,": COULE [ENTER]")
:End
```

Programme YNACONT — continue une partie

```
:123→Z
:prgmNAVALE
```



OTHELLIC

Le jeu d'Othellic, aussi connu sous d'autres noms, fait maintenant partie des grands classiques des jeux de pions, au même titre que les échecs ou les dames, bien qu'il soit beaucoup plus récent. Dans ce jeu, lorsqu'on capture des pions adverses, on les retourne, et ils deviennent des pions supplémentaires pour le joueur qui les a pris ! Les règles sont originales et très simples, mais donnent lieu à des parties acharnées et il faut rester vigilant car les retournements de situation ne sont pas rares.

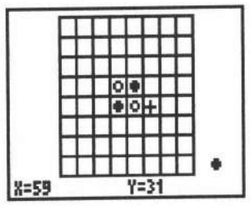
Dans cette version, vous pourrez affronter un autre joueur humain, la calculatrice se chargeant de retourner les pions et de faire

respecter les règles. Grâce à ce programme, vous aurez un jeu de poche utilisable partout, avec la possibilité d'interrompre la partie, pour la reprendre plus tard.

Règles :

Le jeu se joue sur un damier de 8 cases sur 8. L'un des joueurs a les pions noirs, l'autre les blancs. Au début du jeu, le damier est vide, à l'exception des 4 cases centrales, où sont déjà placés 2 pions blancs et deux pions noirs. Les blancs commencent.

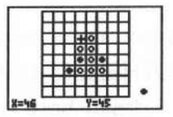
Chacun son tour, les joueurs posent un pion de leur couleur sur le plateau, dans une case vide. Les seuls endroits autorisés pour poser un pion sont ceux **permettant de faire une prise**. Un pion adverse, par exemple noir, est pris lorsqu'il est entouré par deux cotés opposés par un pion blanc, l'un de ces pions étant celui que vient de poser le joueur blanc. Comment se manifeste la prise ? Contrairement à la plupart des jeux, où le pion pris est retiré du jeu, ici il est "retourné", et il change de couleur ; il passe ainsi à l'adversaire, et lui permettra aux tours suivants de faire d'autres prises !



On peut prendre plusieurs pions d'un coup : si plusieurs pions noirs sont alignés, qu'à une extrémité se trouve un pion blanc, et que l'on pose un autre pion blanc à l'autre extrémité, l'ensemble de la ligne est retournée. Encore plus fort, si le pion qu'on vient de

poser permet d'encadrer plusieurs lignes de pions adverses, l'ensemble de tous ces pions est retourné !!!

On voit que l'on peut gagner rapidement beaucoup de pions, pour peu que l'adversaire ne fasse rien contre. Ceci dit, avoir beaucoup de pions en milieu de partie ne signifie pas forcément la victoire en fin de partie, car tous les pions n'ont pas la même valeur tactique : les pions sur les bords du plateau sont plus difficiles à prendre, puisqu'ils ne peuvent être pris que sur 3 cotés ; les pions des 4 coins sont eux imprenables, car on ne peut pas les entourer par deux cotés opposés.



A son tour de jouer, chaque joueur est **obligé de prendre**, s'il le peut. S'il n'y a pas de prise possible, il doit passer, et son adversaire peut donc rejouer. Si aucun des joueurs ne peut jouer, la partie s'arrête.

La partie s'arrête donc dans ce cas où aucun joueur ne peut jouer, et dans le cas où tous les pions ont été posés et qu'il ne reste plus de case vide. Le gagnant est alors celui qui a le plus de pions. Il y a match nul si les deux joueurs ont le même nombre de pions.

Mémoire nécessaire :

Programme : 1300 octets
Données : 600 octets

Nombre de joueurs : 2

But du jeu :
le gagnant est celui qui a le plus de pions lorsque la partie se termine.

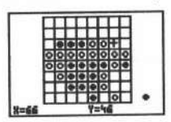
MODE D'EMPLOI

Pour commencer une nouvelle partie, exécuter le programme OTHELLIC. L'écran de jeu s'affiche, un pion blanc ou noir en base à droite hors du damier indique le joueur auquel c'est le tour de jouer. Le joueur blanc commence toujours.

Une croix apparaît au centre de l'écran. Pour jouer déplacez la croix avec les flèches sur la case désirée, et appuyez sur [Enter]. Le pion est posé, après avoir retourné un ou plusieurs pions adverses.

Si une croix clignotante apparaît sur la case, cela signifie que ce coup ne permet pas de faire de prise et n'est donc pas valide. La croix revient, afin de rentrer un nouveau coup.

Si aucun coup n'est possible, vous devez passer. Déplacez le curseur hors de la grille et pressez [Enter]. Une croix clignote sur le pion en bas à droite, puis c'est à votre adversaire de jouer.



Lorsque la croix est affichée, vous pouvez interrompre la partie en pressant [Enter] sans déplacer le curseur. Les scores de chaque

joueur sont alors affichés, et vous êtes sorti du programme, ce qui vous permet de faire des calculs, ou de tracer des courbes...
Pour continuer la partie exécutez OTHSUITE. L'écran sera redessiné, et la partie continuera exactement là où elle avait été laissée. Si vous n'avez pas tracé de courbes, ou modifié d'une autre manière l'écran graphique, le programme YOTH2 permet de continuer la partie sans tout redessiner.

LE PROGRAMME

OTHELLIC	initialise une nouvelle partie, puis appelle OTHSUITE
OTHSUITE	commande l'affichage de la grille, des pions, et appelle YOTH2
YOTH2	lit le coup de chaque joueur, et appelle les autres modules
XBITMAP	initialise l'écran graphique
YOTHCOOR	calcule les coordonnées graphiques
YOTHPION	affiche les pions posés sur le jeu
YOTHTEST	teste la validité d'un coup, et retourne les pions
YOTHGRIL	dessine la grille de jeu
YOTHBLAN	dessine un pion blanc
YOTHCENT	dessine un centre de pion noir
YOTHLITC	lit le contenu d'une case
YOTHCROI	dessine une croix clignotante

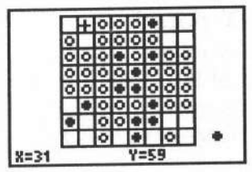
Programme OTHELLIC :

```
:{8,8}→dim [A]
:Fill(Ø,[A])
:1→[A](4,4)                positions initiales
:3→[A](4,5)
:3→[A](5,4)
:1→[A](5,5)
:Ø→V
:Ø→O
:1→R
:2.Ø2→S                    score : 2-2
:prgmOTHSUITE
```


Programme YOTH2 :

```
:DispGraph
:CoordOff
:Lbl D                     début d'un coup
:Input
:If (X=48)*(Y=32)          pas de déplacement
:Then
:ClrHome
:Output(2,4,"SCORES :")    affichage des scores
:Goto S
:End
:int ((X-21)/7)→I
:int ((Y-5)/7)→J
:prgmYOTHLITC
:If C>Ø                    case déjà occupée
:Then
:Lbl I
:prgmYOTHCOOR
:prgmYOTHCROI              affiche la croix
:Goto D                    on re-entre le coup
:End
:If C=(-)1                 case hors limites
:Goto F
:prgmYOTHTEST
:If O=Ø                    pas de prise
:Goto I
:prgmYOTHLITC
:2V+1→[A](I+1,J+1)         on mémorise le pion
:If iPart S*fPart S=Ø      plus de pions ?
:Goto J
:IS>(R,6Ø)                 6Ø coups max
:Goto F
:Lbl J                     fin du jeu
:ClrHome
:Output(2,3,"FIN DU JEU")
```

```
:Lbl S
:Output(4,3,"BLANCS :")
:iPart S→A
:Output(4,12,A)
:Output(6,3,"NOIRS  :")
:1ØØfPart S→A
:Output(6,12,A)
:Disp ""
:CoordOff
:Return
:Lbl F                      fin du tour
:85→X
:6→Y
:If C<Ø                     on a passé ?
:prgmYOTHCROI               croix clignotante
:prgmYOTHCENT               change la couleur du pion
:1-V→V                      qui indique à qui de jouer
:Goto D
```



Programme OTHSUITE :

```
:prgmYOTHGRIL               trace la grille de jeu
:prgmYOTHPION               pose les pions
:85→X
:6→Y
:prgmYOTHBLAN               au blanc de jouer
:If V                       non : au noir
:prgmYOTHCENT
:prgmYOTH2                  le jeu
```

Programme XBITMAP :

```
:FnOff                      initialisation de l'écran graphique
:PlotsOff
:GridOff
:LabelOff
:FullScreen
```



```
:RectGC
:1→Xmin
:95→Xmax
:Ø→Xscl
:1→Ymin
:63→Ymax
:Ø→Yscl
:DispGraph
:ClrDraw
```

Programme YOTHCOOR :

```
:23+7I→X                    calcule les coordonnées
:6+7J→Y                     graphiques
```

Programme YOTHPION :

```
:For(I,Ø,7)                 dessine les pions
:For(J,Ø,7)                 présents dans la matrice
:prgmYOTHLITC
:prgmYOTHCOOR
:If C≠Ø
:prgmYOTHBLAN
:If C=3
:prgmYOTHCENT
:End
:End
```

Programme YOTHTEST :

```
:Ø→O
:I→K
:J→L
:For(G,(-)1,1)
:For(F,(-)1,1)
:K+F→I
:L+G→J
```

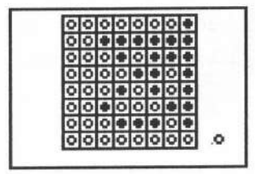
```
:prgmYOTHLITC
:If 2V+C≠3              couleur ≠ opposée ?
:Goto S
:Repeat 2V+C≠3
:I+F→I
:J+G→J
:prgmYOTHLITC
:End
:If 2V+1≠C              pion de même couleur ?
:Goto S
:Lbl R                  retourne les pions
:I-F→I
:J-G→J
:prgmYOTHCOOR
:If I+8J=K+8L
:Goto F
:prgmYOTHCENT
:prgmYOTHLITC
:2V+1→[A](I+1,J+1)      mémorise pion du joueur
:S+.99-1.98V→S          modifie scores
:Goto R
:Lbl F                  cette ligne est retournée
:If O=Ø
:Then
:prgmYOTHBLAN           pose le pion
:If V=1
:prgmYOTHCENT
:1→O
:S+1-Ø.99V→S            un point de plus
:End
:Lbl S
:End                    fin boucle F
:End                    fin boucle G
:K→I
:L→J
```

Programme YOTHGRIL :

```
:ClrHome
:prgmXBITMAP
:For(I,Ø,8)
:Line(21,4+7I,77,4+7I)
:Line(21+7I,4,21+7I,6Ø)
:End
```

Programme YOTHBLAN :

```
:Line(X+1,Y,X+2,Y)
:Line(X+3,Y+1,X+3,Y+2)
:Line(X+1,Y+3,X+2,Y+3)
:Line(X,Y+1,X,Y+2)
```



Programme YOTHCENT :

```
:For(M,1,2)
:Pt-Change(X+M,Y+1)
:Pt-Change(X+M,Y+2)
:End
```

Programme YOTHLITC :

```
:(-)1→C
:If I(I-7)>Ø
:Return
:If J(J-7)>Ø
:Return
:[A](I+1,J+1)→C
```

```
Programme YOTHCROI :
  :For(T,1,8)
  :For(U,(-)1,4)
  :Pt-Change(X+U,Y+U)
  :Pt-Change(X+3-U,Y+U)
  :End
  :sin sin sin 1→A
  :End
```

COMMENTAIRES

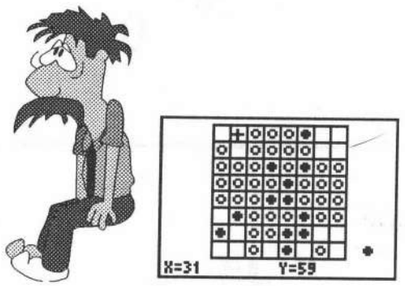
Dans ce jeu, les sous-programmes de contrôle des coup, de retournement des pions, etc... sont utilisés par les deux joueurs. Ils lisent le contenu de la variable V, qui contient 0 ou 1 suivant le joueur qui a joué, pour déterminer quelles variables modifier en fonction du coup du joueur.

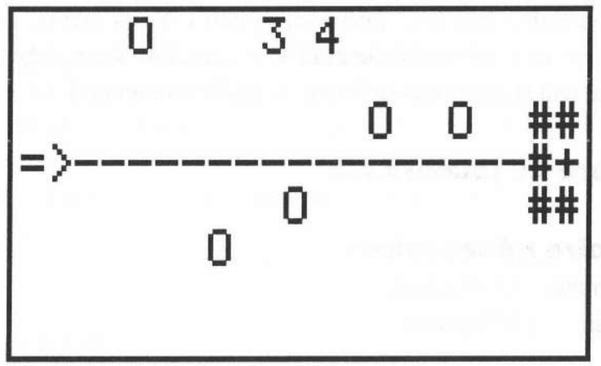
Les positions des pions sont affichées à l'écran, mais aussi mémorisées dans une matrice 8x8, afin d'être testée pour retourner les pions par exemple.

Le principe de mémorisation des cases est que chaque case du jeu peut avoir 3 états : elle peut être vide, contenir un pion blanc, ou contenir un pion noir. Comme on veut représenter son état par la valeur d'une case d'une matrice, il y a 3 valeurs possibles :

case vide	0
pion blanc	1
pion noir	3

Ainsi pour tester les alignements de pions identiques, on recherche dans la matrice des valeurs identiques, dans chacune des 8 directions autour de l'endroit où on vient de déposer un pion. C'est ce que fait le module YOTHTEST.





DUEL

Les sonneries retentissent, les regards se braquent vers l'arène, les respirations se bloquent... Voici les gladiateurs, pour ces nouveaux jeux du cirque dans la Rome du XXIIIe siècle ! Dans leurs armures de sub-latex recouvertes de plaques de zircon étincelantes, ces combattants du futur ressemblent à leurs célèbres prédécesseurs : mêmes regards hautains, même stature impressionnante, même démarche souple et puissante... à un détail près : leurs armes sont des décopants de poing, qui projettent un fin pinceau de particules instantanément sur la cible, et la détruisent généralement. Pour favoriser autant l'adresse que la réflexion, des obstacles mouvants en pierre de Grezon circulent entre les deux adversaires, qui

arrêtent les rayons des décopants et même parfois les renvoient vers le tireur !
Le responsable des jeux porte son energix à ses lèvres : un son retentit, un son qui va déclencher la haine chez les combattants ; ils dégainent leurs armes de mort : le duel commence !

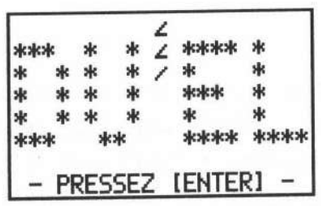
Nombre de joueurs : 2

Mémoire nécessaire :

Programme : 1200 octets
Données : 100 octets

But du jeu :

atteindre l'adversaire en évitant ses tirs.



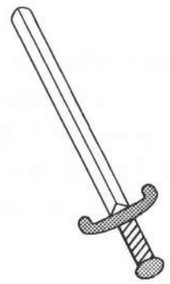
MODE D'EMPLOI

Dans ce jeu vous allez pouvoir affronter un ami dans un duel sans danger. Vous avez à votre disposition votre décopant, votre protection étant assurée par votre armure et surtout les obstacles au centre de l'écran. Vous pouvez monter et descendre, et tirer. Votre armure encaisse 9 coups, au dixième c'est la fin ! Soyez prudent !

Touches actives :

	Joueur de gauche	Joueur de droite
Monter	LOG	^
Descendre	x^2	×
Tirer	LN	÷

Evitez de tirer sur les obstacles, car souvent le tir est réfléchi vers son origine... De même, si vous manquez votre adversaire, votre tir peut être réfléchi par le mur qui se trouve derrière lui !
En haut de l'écran, les points de coup pouvant encore être absorbés par les armures des deux joueurs. Au début du jeu, les deux armures possèdent 9 points ; quand un joueur est touché, son armure perd un point. Lorsqu'un joueur est arrivé à 0, l'autre joueur est déclaré gagnant.



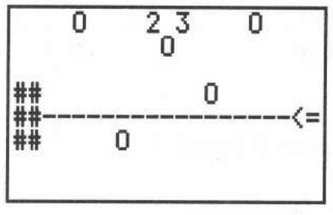
LE PROGRAMME

DUEL	initialisation du jeu
YDUELJEU	le jeu lui-même
YDUELTOU	lorsqu'un joueur est touché
YDUELEGA	affiche des signes "="
YDUELBLA	affiche des blancs

```
Programme DUEL :                initialisation du jeu
  :ClrHome
  :Disp ""
  :Disp "**   * * *** *"       affiche le titre
  :Disp "* * * * *    *"
  :Disp "* * * * **   *"
  :Disp "* * * * *    *"
  :Disp "**    *   *** ***"
  :Output(8,1,"PRESSEZ [ENTER]")
  :Pause
  :2→G                          positions initiales
```

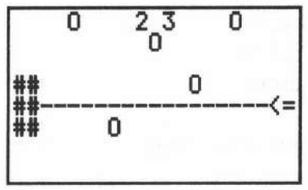


```
  :7→D
  :9→F                          points initiaux
  :9→H
  :prgmYDUELJEU

Programme YDUELJEU :            le jeu lui-même
  :Lbl 1
  :For(I,1,5)                   tirage des 5 obstacles :
  :int (8rand)+1→L1(I)          position de 1 à 8
  :2int (2rand)-1→L2(I)         sens de déplacement 1/-1
  :End
  :ClrHome
  :Output(G,1,"=>")             joueur de gauche
  :Output(D,15,"<=")            joueur de droite
  :Output(1,7,F)                score de gauche
  :Output(1,9,H)                score de droite
  :Lbl 2
  :For(I,1,5)                   pour chaque obstacle
  :L1(I)+L2(I)→A                nouvelle position
  :If  A=Ø                      trop haute
  :8→A
  :If  A=9                      trop basse
```




```
  :1→A
  :Output(L1(I),2+2I," ")       efface ancienne position
  :A→L1(I)
```

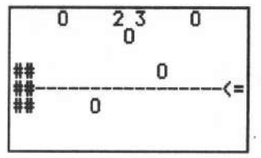
```
:Output(A,2+2I,"O")          affiche nouvelle position
:End
:.1getKey→K                  saisie d'une touche
:iPart  K-6→A                sens de déplacement
:If fPart  K=.1              touche du joueur de gauche ?
:Then
:If  A                       déplacement non nul ?
:Then
:Output(G,1," ")
:G+A/abs A→G                 nouvelle position
:If G>8
```



```
:1→G
:If G<1
:8→G
:Output(G,1,"=>")
:Else                        tir ?
:Output(G,3,"-")             le tir part...
:1→I
:Lbl 3
:If  G=L1(I)                 il rencontre l'obstacle I
:Goto 4
:Output(G,2+2I,"--")         il passe la position I
:IS>(I,5)
:Goto 3
:Output(G,14,"-")
:If  G≠D                     il n'a pas touché l'autre ?
```

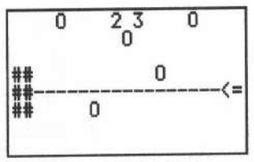


```
:Goto 4
:prgmYDUELTOU                il a touché !
:Goto 1
:Lbl 4                       le tir a été bloqué
:If  rand<.5                 il rebondit ?
:Then
:2I+1+3(I=6)→K
:1→L
:G→M
:prgmYDUELEGA                tracé du rayon de retour
:Ø→I
:prgmYDUELTOU                le joueur de gauche est touché !
:Goto 1
:Else                        le rayon n'a pas rebondi
```



```
:2I+1→K
:3→L
:G→M
:prgmYDUELBLA                on l'efface avec des blancs
:End
:End
:Else
:If fPart  K=.5              touche au joueur de droite ?
:Then
:If  A                       déplacement non nul ?
:Then
:Output(D,15," ")            nouvelle position
:D+A/abs A→D
:If D>8
```

```
:1→D
:If D<1
:8→D
:Output(D,15,"<=")
:Else                          tir ?
:Output(D,13,"--")             le tir part...
:5→I
:Lbl 5
:If  D=L1(I)                   il rencontre l'obstacle I
:Goto 6
:Output(D,2I+1,"--")           il passe la position I
:DS<(I,1)
:Goto 5
:If  D≠G                       il n'a pas touché l'autre ?
:Goto 6
:prgmYDUELTOU
:Goto 1
:Lbl 6                         le tir a été bloqué
:If  rand<.5                   il rebondit ?
```




```
:Then
:2I+3-2(I=Ø)→K
:16→L
:D→M
:prgmYDUELEGA                  tracé du rayon de retour
:6→I
:prgmYDUELTOU                  le joueur de droite est touché !
:Goto 1
:Else                          le rayon n'a pas rebondi
```



```
:2I+3→K
:14→L
:D→M
:prgmYDUELBLA                  on l'efface avec des blancs
:End
:End
:End  :End
:Goto 2

Programme YDUELTOU :           lorsqu'un joueur est touché
  :int (8rand)+1→A             position au tour suivant
  :If  I                       joueur de droite touché ?
  :Then
  :15→X                        précalculs pour l'explosion
  :D→Y
  :A→D
  :H-1→H                       un point de moins
  :Else                        joueur de gauche touché
  :1→X
  :G→Y
  :A→G
  :F-1→F
  :End
  :For(B,X,X+1)                colonne
  :For(C,Y-1,Y+1)              ligne
  :If (C>Ø)*(C<9)              si la case est sur l'écran
  :Output(C,B,"#")             affiche un #
  :End
  :End
  :For(B,X,X+1)                suite du dessin de l'explosion...
  :For(C,Y-1,Y+1)
  :If (C>Ø)*(C<9)
  :Output(C,B,"+")
  :End
  :End
```

```
:For(B,X,X+1)
:For(C,Y-1,Y+1)
:If (C>Ø)*(C<9)
:Output(C,B," ")
:End
:End
:Output(1,7,F)                affichage des points
:Output(1,9,H)
:While getKey                 on attend que l'on relache
:End                          toute touche
:If  min(F,H)>Ø               personne n'est à Ø ?
:Return
:If  H<1                      le joueur de droite est à Ø
:Output(4,3," <- GAGNE !")
:If  F<1                      le joueur de gauche est à Ø
:Output(4,4,"GAGNE ! -> ")
:Stop
```


Programme YDUELEGA : affiche des signes "="

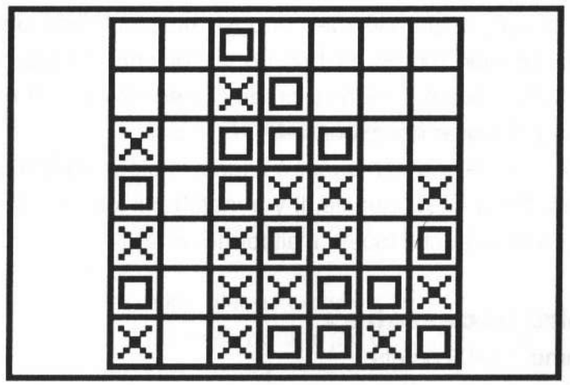
```
:1→E                          K : position de départ
:If K>L                       L : position d'arrivée
:(-)1→E                       E : sens de déplacement
:For(C,K,L,E)
:Output(M,C,"=")              M : ligne
:End
```

Programme YDUELBLA : affiche des blancs entre K et L

```
:1→E
:If K>L
:(-)1→E
```



```
:For(C,K,L,E)
:Output(M,C," ")
:End
```



PUISSANCE 7

Réussirez-vous à battre votre calculatrice à Puissance 7 ? C'est le défi qui vous est lancé dans ce jeu inspiré d'un jeu célèbre...
Tentez d'aligner 4 de vos pions que vous placez dans une grille de 7 x 7 cases. Facile direz-vous ? Et bien, pas tant que ça, quand vous saurez que vos pions sont soumis à la pesanteur, et "tomberont" ainsi au fond des colonnes de la grille. Ainsi vos pions se mêlent à ceux de votre adversaire, vous obligeant à tenir compte de ses prochains coups pour anticiper les vôtres !

Règles :

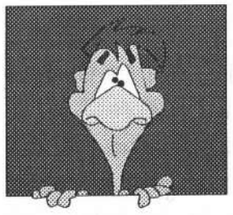
Chaque joueur, à tour de rôle, dépose un pion dans une des 7 colonnes. Le pion tombe en bas de la colonne. Chaque colonne peut contenir jusqu'à 7 pions ; lorsqu'elle est pleine, on n'est plus autorisé à y déposer des pions.
Le gagnant est le premier joueur à aligner 4 pions horizontalement, verticalement ou en diagonale. Il y a match nul lorsque la grille est saturée, sans qu'aucun joueur n'ait formé de ligne.

Mémoire nécessaire :

Programme : 1400 octets
Données : 700 octets

Nombre de joueurs : 1

MODE D'EMPLOI



Exécutez PUISS7. La grille s'affiche, et si la machine commence, elle dépose son pion ; sinon c'est à vous.
Déplacez le curseur avec les flèches gauche et droite afin qu'il pointe sur la colonne désirée, puis pressez [ENTER]. Vos pions sont les croix, ceux de la machine étant des carrés.
Vous jouez à tour de rôle jusqu'à ce qu'il y ait un gagnant, ou bien match nul. Le résultat du match est affiché.

Pressez alors [ENTER] pour revoir le l'écran de jeu, puis [ENTER] de nouveau pour une nouvelle partie. Pressez [ON] et [F5] pour quitter le programme.

LE PROGRAMME :

Puissance 7 est composé des modules suivants :

PUISS7	le jeu
YPUISINI	initialisation d'une nouvelle partie
YPUISTST	test d'une case pour le jeu de la machine
YPUISCUR	dessine le curseur
XBITMAP	initialisation de l'écran
XPAUSE	pause d'environ 1 seconde

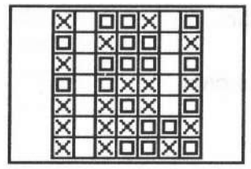
Programme PUISS7 :

```
:Lbl Ø
:prgmXBITMAP          initialise l'écran
:prgmYPUISINI         initialise le jeu
:If rand<.5           qui commence ?
:Goto 1               c'est le joueur
:Lbl 3                début du tour de jeu de
:H+1→H                la machine
:If H=5Ø              la grille est pleine : fin
:Goto 2
:Ø→V
:Ø→X
:Ø→Y
:Ø→Z
:For(K,1,7)           test des 7 colonnes
:C+1→C                on teste la colonne C
```

```
:If C>7                    C hors-limites
:1→C
:If L2(C)>Ø                la colonne n'est pas pleine
:prgmYPUISTST              teste un coup en colonne C
:If J                      coup gagnant !
:Goto 9                    on joue tout de suite
:End                       colonne suivante C+1
:X→C
:If C                      le joueur a aligné 3 pions :
:Goto 9                    on le bloque
:Y→C
:If C                      pas de danger immédiat :
:Goto 6                    on cherche encore
:Z→C
:If C
:Goto 6
:Lbl 5                     on ne sait pas quoi jouer :
:int (rand*7+1)→C          on tire une colonne au hasard
:If L2(C)<1                colonne pleine ?
:Goto 5                    on retire
:V+1→V
:If V≥5                    5 tirages max.
:Goto 6
:If C≠4
:Goto 5
:Lbl 6
:If L2(C)<8                il reste de la place ?
:Goto 9
:If V≥21                   aucun coup ne permet d'éviter
:E=1                       la défaite : abandon
:If V≥21
:Goto 2
:L2(C)-7→L2(C)             mise à jour du jeu
:Ø→Z
:prgmYPUISTST
```

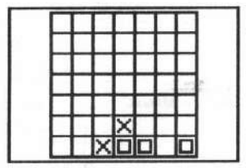


```
:L2(C)+7→L2(C)
:If J=Ø
:Goto 7
:Ø→J
:If V<9
:Goto 5                    on retire un n° de colonne
:Lbl 7
:If Z=Ø
:Goto 8
```



```
:Ø→Z
:If V<6
:Goto 5                    on retire un n° de colonne
:Lbl 8
:If X=Ø
:Goto 9
:Ø→X
:Goto 5
:Lbl 9                     la machine joue !
:61-(8((L2(C)-C)/7))→W     position Y du pion
:14+8C→X                   position X
:Line(X,W,X+4,W)           dessine le pion de la machine
:Line(X+4,W,X+4,W-4)       (un carré)
:Line(X+4,W-4,X,W-4)
:Line(X,W-4,X,W)
:5→L1(L2(C))               mémorisation du pion
:L2(C)-7→L2(C)             prochaine case libre
:If J>Ø                    la machine a gagné
:Goto 2
:Lbl 1                     début du tour du joueur
```

```
:H+1→H
:If H=5Ø                    grille pleine : fin du jeu
:Goto 2
:Lbl 4
:Input                      saisie du coup du joueur
:int ((X-12)/8)→C           colonne demandée
:If (C<1)+(C>7)             elle est hors limites
:Goto 4
:(L2(C)-C)/7→W
:If W<Ø                     colonne pleine
:Goto 4
:61-8W→W                    position Y du pion
:14+8C→X                    position X
```



```
:Line(X,W,X+4,W-4)          dessine le pion du joueur
:Line(X+4,W,X,W-4)          (une croix)
:1→L1(L2(C))                mémorisation du pion
:Ø→E
:prgmYPUISTST               teste les alignements
:L2(C)-7→L2(C)
:If E=Ø                     aligné 4 pions ?
:Goto 3                     non : fin du tour
:Lbl 2                      fin de la partie
:prgmXPAUSE                 une petite pause...
:ClrHome
:Disp "","
:If J                       la machine a gagné
:Then
:Disp "  J'AI GAGNE !"      apostrophe dans le menu ANGLE
```



```
:Else
:If H=5Ø                    49 pions ont été posés
:Then
:Disp "  MATCH NUL !"
:Else                       le joueur a gagné
:Disp "    BRAVO !","VOUS AVEZ GAGNE"
:End
:End
:Disp "","[ENTER] POUR","REVOIR LE JEU"
:Pause
:DispGraph                  réaffichage du jeu
:Pause                      ENTER pour continuer
:Goto Ø
```

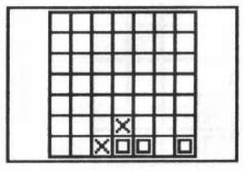
Programme XBITMAP : initialisation de l'écran graphique

```
:FnOff
:PlotsOff
:GridOff
:LabelOff
:FullScreen
:RectGC
:1→Xmin
:95→Xmax
:Ø→Xscl
:1→Ymin
:63→Ymax
:Ø→Yscl
:DispGraph
:ClrDraw
```


Programme YPUISINI : initialisation du jeu

```
:Ø→E
:Ø→J
:Ø→G
```

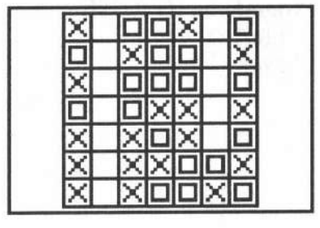
```
:Ø→H
:ClrHome
:For(A,1,5Ø)              initialise les cases du jeu
:Ø→L1(A)
:End
:2Ø→X
:63→Y
:For(C,1,8)               trace la grille vide
:Line(X,7,X,63)           traits verticaux
:Line(2Ø,Y,76,Y)          traits horizontaux
:42+C→L2(C)
:X+8→X
:Y-8→Y
:End
:4→C
:CoordOff
```



```
Programme YPUISTST :      test des alignements
  :16+8C→P                position X du curseur
  :prgmYPUISCUR           affiche le curseur
  :For(A,Ø,13)            initialise la matrice de test
  :Ø→L2(A+8)              qui commence à la position 8
  :End
  :Ø→I
  :L2(C)→M                haut de la colonne
  :For(U,M,M+21,7)
  :If U≤49
  :L2(I+8)+L1(U)→L2(I+8)
```



```
:End
:I+1→I
:C-3→A                    compte le nombre de pions
:If A<1                   alignés de la colonne C-3
:1→A                      à la colonne C+3
:Lbl 1
:If (A>4)+(A>C)
:Goto 2
:For(B,A,A+3)
:L2(I+8)+L1(M-C+B)→L2(I+8)
:End
:I+1→I
:M-8(C-A)→N
:If N<1
:Goto 2
:If N>25
:Goto 2
:For(D,Ø,3)
:L2(I+8)+L1(N)→L2(I+8)
:N+8→N
:End
:I+1→I
:Lbl 2
:If A>7
:Goto 3
:If (A<4)+(A<C)
:Goto 3
:M+6(C-A)→N
:If N<4
:Goto 3
:If N>28
:Goto 3
:For(D,Ø,3)
:L2(I+8)+L1(N)→L2(I+8)
:N+6→N
```



```
:End
:I+1→I
:Lbl 3
:IS>(A,C+3)                colonne suivante
:Goto 1
:For(T,Ø,I)
:L2(T+8)→D
:If D=4                    4 pions du joueur alignés :
:1→E                       il a gagné !
:If D=15                   4 pions de la machine
:C→J                       alignés : coup gagnant !
:If D=1Ø                   3 pions alignés : correct
:C→Z
:If D=3                    3 pions du joueur alignés :
:C→X                       danger !
:If T(D=2)                 2 pions alignés : à bloquer
:C→Y                       faute de mieux
:End
:prgmYPUISCUR              efface le curseur

Programme YPUISCUR         dessine le curseur
  :Pt-Change(P,3)
  :Pt-Change(P,4)

Programme XPAUSE           pause d'environ 1 seconde
  :For(I,Ø,2ØØ)
  :End
```

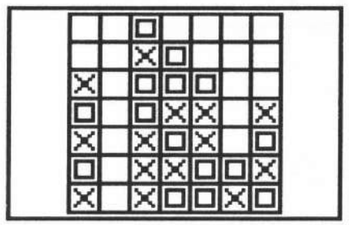

COMMENTAIRES

Mémorisation du jeu :

La grille de jeu de 7 x 7 cases est mémorisée dans la liste L_1 de 49 cases, chaque élément de L_1 correspondant à un élément de la matrice simulée M, avec l'équivalence suivante :

$M(I,J) = L_1(I+(J-1)*7)$

où I est le numéro de colonne, de 1 (gauche de la grille) à 7 (droite), et J le numéro de ligne, de 1 (haut de la grille) à 7 (bas).

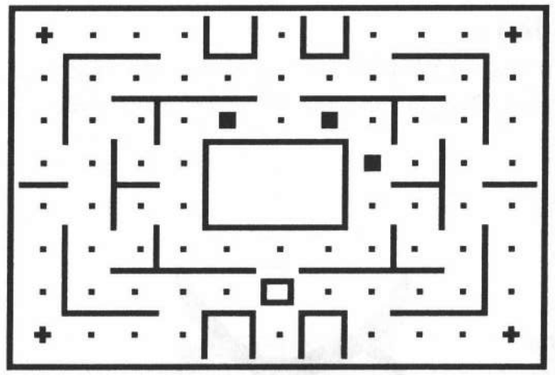


L'autre liste, L_2, est utilisée pour mémoriser pour chaque colonne la coordonnée de la première case libre, afin d'accélérer les traitements. A partir de PH(8) elle est également utilisée par YPUISTST comme variables intermédiaires pour la recherche des alignements de pions.

Stratégie résumée de la machine :

La machine examine chaque colonne, et regarde quelles seraient les conséquences si elle y jouait. Si le coup :

- permet de faire une ligne de 4 pions, il est joué immédiatement
- s'il permet d'empêcher le joueur d'aligner 4 pions, il est joué.
- s'il permet d'empêcher le joueur d'aligner 3 pions, il est aussi joué.
- s'il permet à la machine d'aligner 3 pions, elle le joue.
- sinon, elle joue un coup au hasard



GOB-MAN

Mangez les pastilles déposées dans les couloirs du labyrinthe, et évitez les fantômes. Si ce scénario vous rappelle quelque chose, ne vous inquiétez pas, c'est voulu ! Sauf que maintenant, c'est sur votre TI-82, et vous pourrez y jouer partout !

Mémoire nécessaire :

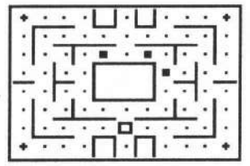
Programme : 2500 octets
Données : 6100

Nombre de joueurs : 1

But du jeu :

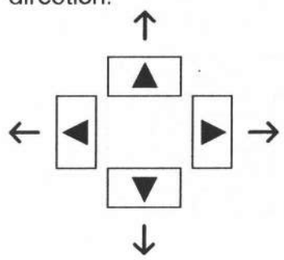
manger toutes les pastilles du tableau et réaliser le score le plus haut possible. Manger quelques fantômes après avoir ingurgité les vitamines !

MODE D'EMPLOI



Exécutez d'abord le programme GOBINI afin de créer l'image du labyrinthe et les matrices utilisées dans le programme GOB. Une fois ceci fait, vous pourrez faire autant de parties que désiré, tant que les matrices P, F et D ne seront pas effacées.

Exécutez GOB. Vous êtes représenté par le carré en bas de l'écran. Evitez les fantômes et mangez les pastilles dans les couloirs. Le déplacement se fait avec les quatre flèches de direction.



Mangez les vitamines dans les coins du tableau, et les rôles sont inversés ! C'est maintenant les fantômes qui vous fuient, car vous pouvez les manger ! Observez la jauge à droite de l'écran, elle vous indique le temps restant pour la chasse aux fantômes !

Chaque pastille rapporte 10 points. Une vitamine rapporte 50 points. Le premier fantôme mangé rapporte 100 points, le

deuxième, 200 points, le troisième 400 points, etc... Les points se trouvent dans les fantômes, alors... pas de pitié !!!

LE PROGRAMME

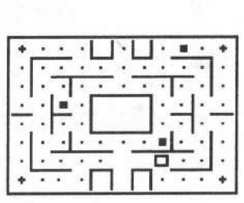
GOB-MAN est composé de :

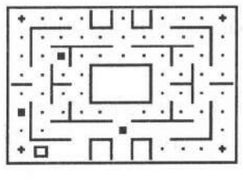
GOBINI	initialisation du labyrinthe
GOB	le jeu
YGOBGRA	initialisation de l'écran graphique

Programme GOBINI :

```
:prgmYGOBGRA
:Line(1Ø,56,3Ø,56)
:Line(4Ø,56,5Ø,56)
:Line(6Ø,56,7Ø,56)
:Line(8Ø,56,1ØØ,56)
:Line(6Ø,56,7Ø,56)
:Line(8Ø,56,1ØØ,56)
:Line(2Ø,48,5Ø,48)
:Line(6Ø,48,9Ø,48)
:Line(4Ø,4Ø,7Ø,4Ø)
:Line(1,32,1Ø,32)
:Line(2Ø,32,3Ø,32)
:Line(8Ø,32,9Ø,32)
:Line(1ØØ,32,11Ø,32)
:Line(4Ø,24,7Ø,24)
:Line(2Ø,16,5Ø,16)
:Line(6Ø,16,9Ø,16)
:Line(1Ø,8,3Ø,8)
:Line(4Ø,8,5Ø,8)
:Line(6Ø,8,7Ø,8)
:Line(8Ø,8,1ØØ,8)
:Line(1Ø,56,1Ø,4Ø)
:Line(1Ø,24,1Ø,8)
```

dessin des murs du labyrinthe






```
:Line(2Ø,4Ø,2Ø,24)
:Line(3Ø,48,3Ø,4Ø)
:Line(3Ø,24,3Ø,16)
:Line(4Ø,63,4Ø,56)
:Line(4Ø,4Ø,4Ø,24)
:Line(4Ø,8,4Ø,1)
:Line(5Ø,63,5Ø,56)
:Line(5Ø,8,5Ø,1)
:Line(6Ø,63,6Ø,56)
:Line(6Ø,8,6Ø,1)
:Line(7Ø,63,7Ø,56)
:Line(7Ø,4Ø,7Ø,24)
:Line(7Ø,8,7Ø,1)
:Line(8Ø,48,8Ø,4Ø)
:Line(8Ø,24,8Ø,16)
:Line(9Ø,4Ø,9Ø,24)
:Line(1ØØ,56,1ØØ,4Ø)
:Line(1ØØ,24,1ØØ,8)
:Line(11Ø,63,11Ø,1)
:StorePic Pic6
:{23,17}→dim [E]
:Fill(1,[E])
:For(I,1,12)
:8→[E](I,1)
:End
:For(I,1,9)
:8→[E](1,I)
:End
:For(I,1,7)
:8→[E](I+4,5)
:End
:For(I,1,5)
:8→[E](I+2,3)
:8→[E](I+8,7)
:8→[E](3,I+2)
```

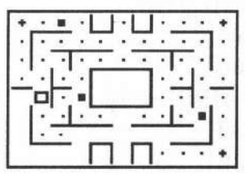
Matrice du terrain

Mémorisation des 'murs' (valeur = 8)

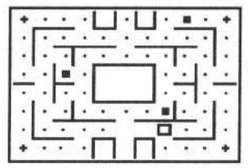
```
:End
:For(I,1,3)
:8→[E](I+8,3)
:8→[E](I,9)
:8→[E](I+4,9)
:8→[E](5,I+6)
:8→[E](7,I+4)
:8→[E](9,I+6)
:End
:8→[E](9,2)
```

```
:8→[E](11,2)
 :5→[E](2,2)
 :4→[E](8,2)
 :4→[E](12,2)
 :4→[E](4,4)
 :4→[E](8,4)
 :4→[E](12,4)
 :4→[E](4,6)
 :4→[E](6,6)
 :4→[E](8,6)
 :4→[E](12,6)
 :4→[E](2,8)
 :4→[E](4,8)
 :4→[E](6,8)
```

Mémorisation des 'carrefours'



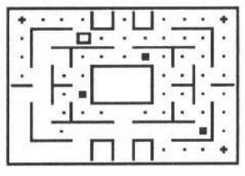

```
:4→[E](8,8)
:For(J,1,9)                    Remplit les 3/4 restants
:For(I,1,12)                   du labyrinthe en utilisant
:[E](I,J)→[E](24-I,J)          les symétries
:[E](I,J)→[E](I,18-J)
:[E](I,J)→[E](24-I,18-J)
:End
:End
:ClrHome
```



Programme GOB :

```
 :prgmYGOBGRA
 :Radian
 :{3,3}→dim [A]                Matrice des fantomes
 :[[(-)1,Ø,1,Ø][Ø,(-)1,Ø,1]]→[B]   Table de déplacement en
 :[B]ᵀ→[B]                     fonction de la direction
 :Ø→S                          Score
 :3→G                          Nombre de vies
 :Lbl Ø
 :162→N                        Nombre de pastilles
 :RecallPic Pic6
 :For(J,4,6Ø,4)                Dessine les pastilles
 :For(I,5,1Ø5,5)
 :Pt-On(I,J)
 :End
 :End
 :For(J,3,59,56)               Dessine les vitamines
 :For(I,4,1Ø4,1ØØ)
```

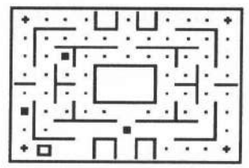
```
:Line(I,J+1,I+2,J+1)
:Line(I+1,J,I+1,J+2)
:End
:End
:StorePic Pic5                    Mémorise l'image
:For(J,1,17)                      Réinitialisation du terrain
:For(I,1,23)                      (pastilles)
:If ([E](I,J)=Ø)+([E](I,J)=3)
:[E](I,J)+1→[E](I,J)
:End
:End
:5→[E](2,2)                       Réinitialisation du terrain
:5→[E](22,2)                      (vitamines)
:5→[E](2,16)
:5→[E](22,16)
:Lbl 1
:ClrHome
:Output(4,4,"SCORE :")
:Output(4,1Ø,S)
:For(I,1,1ØØØ)
:End
:12→A                             Coordonnées de GOB
:6→B
:Ø→C                              Direction de GOB
:12→[A](1,1)                      Coordonnées du fantôme 1
:13→[A](1,2)
```



```
:4→[A](1,3)                       Direction du fantôme 1
:13→[A](2,1)                      Coordonnées du fantôme 2
```

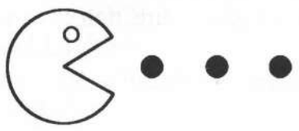


```
:12→[A](2,2)
:3→[A](2,3)                       Direction du fantôme 2
:11→[A](3,1)                      Coordonnées du fantôme 3
:12→[A](3,2)
:1→[A](3,3)                       Direction du fantôme 3
:Ø→H                              H=1 : manger les fantômes
:Ø→T                              Comteur de temps
:Ø→P                              numéro du fantôme mangé
:Lbl 2                            Début de la boucle d'animation
```



```
:getKey→K                         Saisie d'une touche
 :If K=24                         Flèche gauche = direction 1
 :1→C
 :If K=26                         Flèche droite = direction 3
 :3→C
 :If K=34                         Flèche bas = direction 2
 :2→C
 :If K=25                         Flèche haut = direction 4
 :4→C
 :If C≠Ø
 :Then
 :A+[B](C,1)→D                    Nouvelles coordonnées de GOB
 :B+[B](C,2)→E
 :If [E](D,E)≤6                   Teste si c'est un mur ou pas
 :Then
 :D→A                             nouvelles coordonnées
 :E→B
```

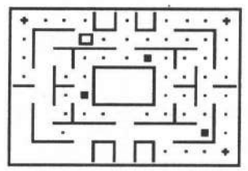
```
:End
:End
:For(I,1,3)                    Boucle pour le calcul du
:[A](I,1)→L                    mouvement des fantômes
:[A](I,2)→M
:[A](I,3)→O
:If (L=A)*(M=B)                GOB et fantôme I même position ?
:I→P                           Mémorise le numéro du fantôme
:[E](L,M)→V
:If V>2                        Est-ce un carrefour?
:Then                          (R▶Pθ est dans le menu ANGLE)
:int ((3.927+R▶Pθ(A-L,B-M))/1.57)+2*H+1→F
:If F>4
:F-4→F
:L+[B](F,1)→D                  Nouvelle direction
:M+[B](F,2)→E                  Nouvelles coordonées
```



```
:While [E](D,E)≥6              Tant que la nouvelle position
:int (4rand)+1→F               donne dans un mur : recherche
:L+[B](F,1)→D                  aléatoire d'une nouvelle
:M+[B](F,2)→E                  position
:End
:Else                          Si ce n'est pas un carrefour,
:L+[B](O,1)→D                  le fantôme continue dans
:M+[B](O,2)→E                  la même direction
:O→F
:End
:D→[A](I,1)                    Mémorise la nouvelle position
```

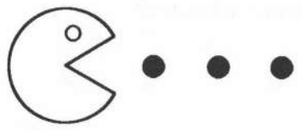


```
:E→[A](I,2)
:F→[A](I,3)                    Mémorise la nouvelle direction
:End
:ClrDraw
:RecallPic Pic5                Rappel de l'image
:5A-7→X                        Coordonnées graphiques de GOB
:4B-6→Y
:[E](A,B)→V
:If (V=1)+(V=4)                La case de GOB contient-elle
:Then                          une pastille ?
:V-1→[E](A,B)                  Efface pastille du tableau P
:Pt-Off(X+2,Y+2)               Efface la pastille
:S+1Ø→S                        1Ø points de gagné...
:N-1→N                         ...pour une pastille mangée !
:If N=Ø                        Plus de pastilles?
:Goto Ø                        Si oui, un nouveau tableau, un !
:End
:If (V=2)+(V=5)                Est-ce une vitamine?
:Then
```



```
:V-2→[E](A,B)                  Efface la vitamine du tableau P
:Pt-Off(X+1,Y+2)               Efface la vitamine du graphisme
:Pt-Off(X+2,Y+2)
:Pt-Off(X+3,Y+2)
:Pt-Off(X+2,Y+1)
:Pt-Off(X+2,Y+3)
:Line(114,1,114,3Ø)            Affiche la jauge
:Line(115,1,115,3Ø)
```

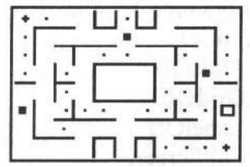
```
:S+5Ø→S          La vitamine rapporte 5Ø points
:1→H             GOB peut manger les fantômes
:31→T            Le compteur est initialisé
:Ø→Q             Nombre de fantômes mangés
:End
:T-H→T           Décrémente le compteur (si H=1)
:Pt-Off(114,T)   La jauge baisse...
:Pt-Off(115,T)
:StorePic Pic5   Mémorise l'image
:Line(X,Y,X+4,Y) Affiche GOB
:Line(X,Y+4,X+4,Y+4)
:Line(X,Y+1,X,Y+3)
:Line(X+4,Y+1,X+4,Y+3)
:For(I,1,3)      Boucle pour l'affichage
:[A](I,1)→L      des fantômes
:[A](I,2)→M
```



```
:5L-6→X          Coordonnées graphiques
:4M-5→Y          du fantôme n°I
:Line(X,Y,X+2,Y) Affiche le fantôme n°I
:Line(X,Y+2,X+2,Y+2)
:Pt-On(X,Y+1)
:Pt-On(X+2,Y+1)
:If (L=A)*(M=B)  GOB et fantôme I même position ?
:I→P             Mémorise le numéro du fantôme
:End
:If T=1          GOB ne peut plus manger
:Ø→H             de fantômes
:If P≠Ø          Est-ce que GOB se trouve
```



```
:Then            avec un fantôme ?
:If H=1          Si GOB a le droit de manger
:Then            les fantômes alors...
:Q+1→Q           1 fantôme mangé
:S+(1ØØ*Q²)→S    Calcul des points gagnés
:If A≤12         Repositionne le fantôme mangé
:Then            à l'opposé de GOB
:22→[A](P,1)     (pas idiot le fantôme !)
:Else
:2→[A](P,1)
:End
:If B≤9
:Then
:16→[A](P,2)
:Else
:2→[A](P,2)
:End
:Else            GOB ne peut manger les fantômes
:G-1→G           ...et hop, une vie en moins
```



```
:If G=Ø          Reste-t-il une vie?
:Then
:Goto 3          Non, et bien c'est la FIN
:Else
:Goto 1          Sinon on continue
:End
:End
```

```
:Ø→P                    Réinitialise à zéro le n°
:End                    du fantôme mangé
:Goto 2                 Fin de la boucle d'animation
:Lbl 3                  Fin de la partie
:ClrHome
:Output(3,3,"GAME OVER !")
:Output(6,4,"SCORE :")
:Output(6,1Ø,S)
:Stop

Programme YGOBGRA :     initialisation de l'écran graphique
:FnOff
:PlotsOff
:GridOff
:LabelOff
:FullScreen
:RectGC
:1→Xmin
:115→Xmax
:Ø→Xscl
:1→Ymin
:63→Ymax
:Ø→Yscl
:DispGraph
:ClrDraw
```

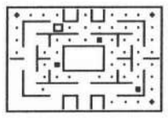

COMMENTAIRES

GOBINI crée le dessin du labyrinthe et le stocke dans Pic6. Le labyrinthe avec les pastilles est stocké dans une autre image : Pic5. Lorsque le GOB mange une pastille, celle-ci est effacée et l'on mémorise la nouvelle image dans Pic5. Le GOB et les fantômes sont redessinés après que l'on ait stocké la nouvelle image : cela évite de les effacer à chaque fois.

GOBINI crée également la matrice [E] (23x17) qui contient le terrain sous une forme codée. Chaque élément de la matrice code le contenu d'une case, de la manière suivante :

case	vide	avec pastille	avec vitamine
MUR	8	-	-
COULOIR	0	1	2
CARREFOUR	3	4	5

Les "carrefours" sont utilisés par les fantômes pour se déplacer. Lorsqu'un fantôme arrive à un carrefour, il choisit une nouvelle direction. Il avance ensuite en ligne droite jusqu'au prochain carrefour. Cette méthode est la plus économique en vitesse d'exécution.



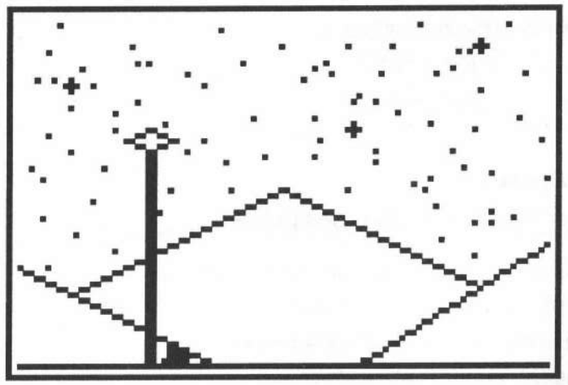
La direction est donnée par une valeur de 1 à 4 : 1 = gauche, 2 = bas, 3 = droite, 4 = haut. La table D permet donne le déplacement en X et en Y en

fonction de la direction.

Structure du programme :

Le programme contient une boucle principale (Lbl 2 ... Goto 2) qui :
- Gère le déplacement du GOB.
- Gère le déplacement des fantômes.
- Affiche le terrain : rappel de Pic5, efface éventuellement une pastille ou une vitamine, affiche la jauge, stocke Pic5, affiche le GOB et les fantômes.
- Gère les collisions entre GOB et un fantôme.

Au label 1, le GOB commence une de ses trois vies, en début de jeu et aussi après avoir rencontré un fantôme...
Au label 0, un nouveau tableau est initialisé, en début de jeu et quand toutes les pastilles ont été mangées.



TI-INVADERS

Alerte ! Les envahisseurs attaquent votre TI, puis une fois ceci fait, ils envahiront la Terre ! Pas une minute à perdre, vous sautez dans votre vaisseau spatial, afin de les pulvériser avant qu'ils n'atterrissent ! Ce jeu d'action vous demandera beaucoup de réflexes et de sang-froid, car les aliens ne sont pas venus les mains dans les poches : ils sont équipés de puissants laser gamma, qui peuvent vous foudroyer en un instant, et, si vous mourrez, qui les arrêtera ?

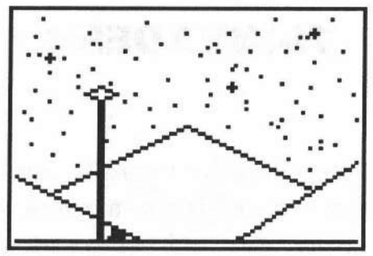
Nombre de joueurs : 1

Mémoire nécessaire :

Programme : 1300 octets
Données : 1200 octets

But du jeu :

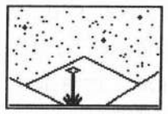
repousser les aliens et faire des points.



MODE D'EMPLOI

Avant la première utilisation du jeu, exécutez le programme INVINI.

Exécutez le programme INVADERS. Le record s'affiche, avec le nom de son détenteur. Si vous voulez passer à la postérité, il vous faudra battre ce score.



Le terrain de jeu s'affiche. Vous êtes le vaisseau en bas de l'écran, vous vous déplacez à gauche et à droite avec les flèches. Au-dessus de vous, un alien vindicatif survole la Terre, et tente d'atterrir. Pressez [ALPHA] pour tirer une salve de rayons laser. Soyez prudents, car les aliens sont armés eux aussi, et on les connaît, ils n'hésitent pas à tirer !

← ◀ ▶ →

ALPHA tir

Chaque alien rapporte 100 points, plus un bonus d'autant plus grand qu'il s'est rapproché du sol. Tous les 1000 points, vous passez au niveau suivant, plus difficile. Les aliens en effet ont repéré de la résistance, et ils envoient des vaisseaux plus puissants. Vous aurez à être encore plus vigilant !

Si vous battez le record, vous pourrez laisser votre nom parmi ceux des plus grands héros ! Entrez votre nom ou votre pseudonyme, de 1 à 6 lettres.

Que la force soit avec vous !

LE PROGRAMME

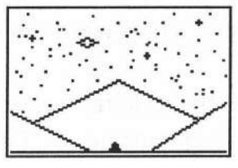
Avant de commencer à jouer, exécutez le programme YINVINI pour créer l'image de fond. YINVINI vous propose un fond assez simple, mais vous pouvez personnaliser cette image, en dessinant ce que vous voulez ! YINVINI doit avoir été exécuté au moins une fois. Ensuite, éditez l'image avec les fonctions du menu DRAW, puis mémorisez votre image avec StorePic, dans la variable Pic4 du menu VARS/Picture.

TI-INVADERS est composé des modules :

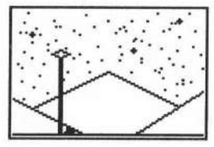
INVADERS	commence une nouvelle partie, gère le record
YINVJEU	le jeu lui-même
YINVTIR	gère le tir du joueur
YINVALI	gère le tir de l'alien
YINVEXP	dessine l'explosion du vaisseau du joueur
XBITMAP	initialise l'écran graphique
XPAUSE2	pause de 3 secondes
YINVINI	initialise l'image de fond et le record

Programme INVADERS :

```
:ClrHome
:Output(2,3,"TI-INVADERS")
:Output(5,3,"RECORD :")
:Output(7,4,W)
```




```
:prgmXPAUSE2                    une pause de 2 secondes
:ClrHome
:Ø→S                            score Ø
:prgmXBITMAP
:prgmYINVJEU                    appel du jeu
:prgmXPAUSE2
:ClrHome
:Output(2,1,"** GAME OVER **")  retour du jeu...
:Output(4,3,"SCORE")
:Output(4,1Ø,S)
:Disp "",""
:If S>W                         record battu
:Then
:Output(6,2,"RECORD BATTU !")
:S→W
:Else
:Output(6,5,"RECORD :")
:Output(7,6,W)
:End
```



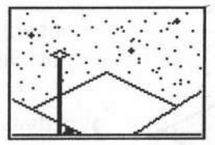
Programme YINVJEU :

```
:46→X                position du joueur
:11→J                vitesse de tir de l'alien
:1→N                 niveau 1
:Lbl D               début d'un round
:int (75rand)+1Ø→U   abscisse de l'alien
:int (3Ørand)+4Ø→V   ordonnée
:1→I
:Ø→B
```

```
:Lbl S                    boucle d'animation
:U+8 int (3rand-1)→U      déplacement alien x
:V+4 int (3rand-1)-2→V    idem en y
:If U<1                   il sort par la gauche ?
:1→U
:If U>88                  il sort par la droite ?
:88→U
:ClrDraw                  les calculs sont faits,
:RecallPic Pic4           on remet le décor
:Line(U,V,U+5,V+2)        dessin de l'alien
:Line(U+5,V+2,U+9,V)
:Line(U+5,V-2,U+9,V)
:Line(U,V,U+5,V-2)
:Line(X,2,X+4,2)          dessin du joueur
:Line(X+1,3,X+3,3)
:Line(X+1,4,X+3,4)
:Pt-On(X+2,5)
:If fPart(I/J)=Ø          l'alien tire
:prgmYINVALI
:If B                     il a touché ?
:Return
:If V<3                   il a atterri ?
:Then
:prgmYINVEXP              explosion du joueur
:Return
:End
:getKey→K                 saisie du mouvement du joueur
:If K=24                  à gauche ?
:Then
:X-D→X                    x diminue
:8→D
:Else
:If K=26                  à droite ?
:Then
:X+D→X                    x augmente
```

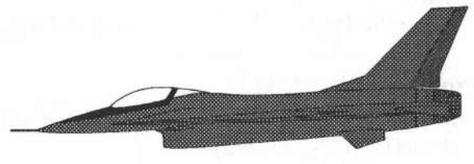


```
:8→D
:Else                     pas de déplacement ?
:4→D                      déplacement diminue
:End
:End
:If K=31                  joueur tire avec [ALPHA]
:prgmYINVTIR
:If B                     il a touché ?
:Then
:If S≥1ØØØN               il a fini le niveau ?
:Then
:1+N→N                    niveau suivant
:ClrHome
:Output(3,4,"NIVEAU")
:Output(3,11,N)
:Output(6,4,"SCORE")
:Output(6,1Ø,S)
:prgmXPAUSE2
:If J>4                   l'alien augmente sa
:J-2→J                    cadence de tir, jusqu'à 4
:DispGraph
:End
:Goto D                   round suivant
:End
:I+1→I
:Goto S                   boucle l'animation
```



Programme XBITMAP : initialisation de l'écran graphique

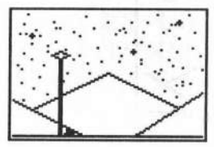
```
:FnOff
:PlotsOff
:GridOff
:LabelOff
:FullScreen
:RectGC
:1→Xmin
```



```
:95→Xmax
:Ø→Xscl
:1→Ymin
:63→Ymax
:Ø→Yscl
:DispGraph
:ClrDraw
```

Programme XPAUSE2

```
:For(P,1,8ØØ)          pause 2-3 secondes
:End
```



Programme YINVINI :

```
:1ØØØ→W                initialise le record
:prgmXBITMAP           initialise l'écran
:Horizontal 1          le sol
:Line(34,1,1,24)       les montagnes
:Line(64,1,95,32)
:Line(9,19,47,38)
:Line(47,38,82,21)
:For(I,1,3Ø)           les étoiles
:Pt-On(97 rand,4Ø+25 rand)
:End
:StorePic Pic4         sauve l'image
```

Programme YINVALI :

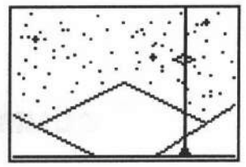
```
:Line(U+4,V-3,U+4,2)   tir de l'alien
:Line(U+5,V-3,U+5,2)
:If (U+4<X)+(U>X)      manqué le joueur ?
```



```
:Return
:prgmYINVEXP           explosion du joueur
:1→B
```

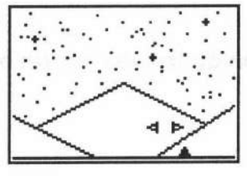
Programme YINVEXP :

```
:For(I,Ø,5)
:Line(X+1.5-I/2,1,X+2-I,I)
:Line(X+2,1,X+1.5-I/2,I+2)
:Line(X+2,1,X+2+I/2,I+2)
:Line(X+2+I/2,1,X+2+I,I)
:End
```



Programme YINVTIR :

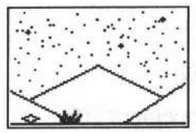
```
:Vertical X+2          tir du joueur
:If (X+2<U)+(X>U+7)    manqué l'alien ?
:Return
:S+1ØØ+1Øint .1(99-V)→S  score augmente
:1→B
:(-)1.5(X=U+3)-3(X>U+3)→A  précalcul du déplacement en x
:U+5→T
:1→I
:While V>(-)2Ø         boucle d'animation
:ClrDraw
:RecallPic Pic4
:Line(U,V,U+5,V+2)     dessin de l'alien
:Line(U+5,V+2,U+5,V-2)  il est en 2 morceaux
:Line(U,V,U+5,V-2)     maintenant !
:Line(T,V+2,T+5,V)     l'un à l'abscisse U
:Line(T,V-2,T+5,V)     l'autre à l'abscisse T
:Line(T,V+2,T,V-2)
:Line(X,2,X+4,2)
:Line(X+1,3,X+3,3)
:Line(X+1,4,X+3,4)
:Pt-On(X+2,5)
:U+1+A→U
```



```
:T+2+A→T
:V-I→V
:2I→I
:End                    fin boucle d'animation
```

COMMENTAIRES

Sauvegarde du record



Entre deux parties le record est conservé dans la variable W. Si le contenu de cette variable était perdu à cause d'un autre programme ou pour tout autre raison, exécutez en mode direct la ligne :

```
:Ø→W
```

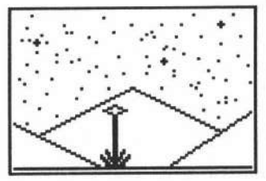
Sauvegarde du fond d'écran

Le fond d'écran avec les étoiles est sauvegardé dans la variable Pic4. Si celle-ci est effacée, il faudrait la recréer en exécutant le programme YINVINI.

Animation de formes sur un écran

L'animation consiste à afficher sur un décor fixe des formes (ou *"sprites"*) mobiles, un ou plusieurs étant déplacés par le joueur, les autres par la machine. L'animation se décompose en "cycles d'animation", un cycle étant composé des étapes suivantes :

1. calcul des nouvelles positions
2. saisie du mouvement du joueur - calcul de la nouvelle position du sprite du joueur
3. effacement des sprites aux anciennes positions
4. affichage aux nouvelles positions
5. retour en 1.



L'ordre dans lequel sont faites ces opérations peut varier, l'important étant qu'à la fin d'un cycle, l'affichage sur l'écran corresponde bien à ce qu'on est en droit d'attendre après un déplacement. En particulier, le sprite du joueur doit réagir fidèlement aux touches de déplacement dès qu'elles sont pressées.

Sur TI-82, on peut effacer les sprites aussi rapidement qu'on les a affiché, avec l'instruction Line() et un zéro en 5ème paramètre. Mais cet effacement est destructif également pour le décor de fond : on bout des quelques secondes on verrait des traits blancs un peu partout sur le décor. De plus avec cette méthode, il faut effacer chaque sprite.
On préfère effacer tout l'écran d'un coup avec l'instruction ClrDraw. On utilise ensuite l'instruction RecallPic pour rappeler le décor instantanément !

Un programme réalisant une animation graphique s'écrit alors :

- **dessin du décor**
- **StorePic PIC**

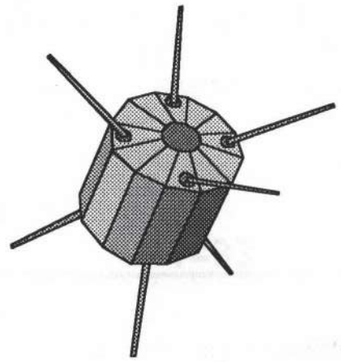
- **Lbl BOUCLE**
- **calculs des nouvelles positions des formes mobiles**
- **ClDrw (efface l'écran)**
- **RecallPic PIC (remet le décor)**
- **affichage des formes mobiles**
- **saisie du mouvement du joueur**
- **détection des collisions**
- **Goto BOUCLE**

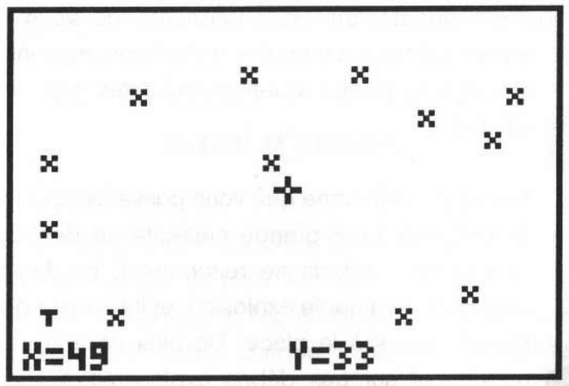
C'est la structure utilisée dans TI-Invaders.

Quelques commentaires sur cette boucle :

- avant la boucle, le décor est dessiné, puis stocké dans l'image PIC.
- dans la boucle, juste avant le RecallPic, l'écran est effacé, car RecallPic n'efface pas l'écran mais se superpose à l'affichage précédent.
- juste après RecallPic, il faut afficher très vite tous les sprites. Plus on les affiche vite, moins il y aura de clignotement. Pour cette raison, les calculs pour l'affichage doivent être finis avant l'effacement de l'écran.
- une fois que tout l'écran est dessiné, le mouvement du joueur est saisi, avec getKey pour ne pas interrompre le programme dans une attente de touche
- les nouvelles positions de tous les sprites sont alors calculées, en fonction du mouvement du joueur, et des logiques de mouvement des sprites contrôlés par la machine.

- la détection des collisions teste si les tirs touchent des cibles, etc...

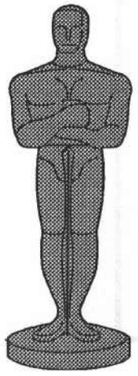




L'INVASION DES ROBOTS

Les robots ont envahi la station spatiale ! Jusqu'à présent vous n'en étiez pas très sûr, mais après avoir joué à ce jeu vous en serez persuadé !!! Alors que vous sortiez tranquillement sur la surface de Vega-1, station spatiale jusqu'à présent parfaitement tranquille, vous voilà encerclé par des robots ! Il y en a partout, et ils sont particulièrement agressifs... ils se dirigent droit sur vous, pour vous détruire. Et autant vous le dire tout de suite, ces robots sont des bombes ambulantes, et leur contact est explosif !

Comble de malchance, vous n'avez pas d'arme avec vous. Vous possédez seulement beaucoup de courage (enfin on espère), et votre téléporteur portatif, qui vous permettra de vous déplacer instantanément à un autre endroit des environs, et ainsi de vous sortir des situations délicates, pour un temps au moins.



En fait la seule arme que vous possédiez est les robots eux-mêmes. Leur grande capacité de destruction fait que si deux robots se rencontrent, ils disparaissent dans une formidable explosion, et il ne reste qu'un petit tas de débris à la place. De plus un autre robot qui marcherait sur ces débris exploserait à son tour. A vous donc de manoeuvrer pour que tous les robots se détruisent mutuellement ou par l'intermédiaire de débris.

Ces débris vous sont également utiles d'une autre manière. Votre téléporteur de poche, s'il est très pratique, a un défaut : si l'on s'en sert de façon répétée, il ne fonctionne plus aussi bien, et il a des ratés, pas mortels en eux-mêmes, mais qui peuvent le devenir si un robot se trouve à proximité... En passant sur un débris, vous pourrez récupérer son énergie pour repolariser votre appareil, et lui redonner un peu plus de capacités. Par contre le débris disparaîtra.

Mémoire nécessaire :

Programme : 1600 octets
Données : 800 octets et plus

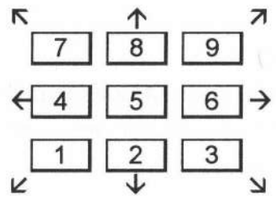
Nb de joueurs : 1

But du jeu :

survivre !

MODE D'EMPLOI

Exécutez le programme ROBOTS. Le record s'affiche, puis le niveau, et votre score (0 pour le moment), après avoir pressé [Enter]. [Enter] à nouveau, et l'écran de jeu s'affiche. Vous êtes le "T", quelque part sur l'écran, chaque "x" est un robot. Le terrain fait 24 x 16.
Une croix apparaît au centre de l'écran. Vous pouvez vous déplacer d'une case dans une des 8 directions, ou rester sur place.
Pour vous déplacer tapez simplement un des chiffres situés autour de la touche [5], chacun représentant une des 8 directions, ou pressez [5] pour rester sur place.



0 téléportation

Chaque robot avance d'une case, dans votre direction. S'ils sont juste à coté de vous et que vous ne pouvez plus fuir, une seule solution : la téléportation ! Pressez [0]. Si le téléporteur fonctionne, vous serez instantanément transporté ailleurs sur l'écran ; sinon, vous resterez sur place... aie aie aie !

Au début d'un niveau, votre téléporteur fonctionne à 100 %, mais chaque utilisation réussie lui enlèvera environ 20 % de fiabilité.

N'en abusez pas, à moins de passer régulièrement sur des débris, qui lui feront regagner 20 % également.

Une collision de deux robots rapportent 200 points, un robot contre un débris 50 points. Si vous finissez un niveau, vous aurez un bonus conséquent, dépendant du niveau. Vous aurez alors à faire face à une nouvelle vague de robots, encore plus nombreux, mais avec plus de points à la clé !

Bonne chance !

Sauvegarde du record :

Entre deux parties, le record est conservé dans la variable W. Si son contenu était modifié par erreur, vous pouvez la réinitialiser à 0 en tapant en mode direct :

:Ø→W

Tapez également cette ligne avant la toute première utilisation du programme ROBOTS.

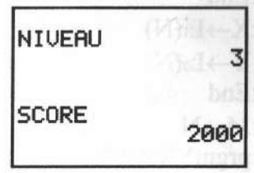
LE PROGRAMME

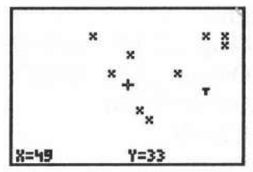
Il est composé des modules :

ROBOTS	initialisations et boucle principale
XBITMAP	initialise l'écran graphique
XENTDIR	saisit un déplacement
YRBJEU	jeu pour un niveau
YRBENERG	détecte l'arrivée du joueur sur un débris
YRBLITC	lit le contenu de la case X,YYYY
YRBJOU	dessine le joueur
YRBROB	dessine un robot
YRBDEB	dessine un débris
YRBEFFAC	efface une case

Programme ROBOTS :

```
:ClrHome
:Output(2,1,"L'INVASION DES")
:Output(3,5,"ROBOTS")
:Output(6,4,"RECORD :")
:Output(7,5,W)
:Ø→S
:1→L
:1Ø2→F
:Pause
:ClrHome
:prgmXBITMAP
:For(L,1,99)
:4+3L→M
:Output(3,4,"NIVEAU")
:Output(3,11,L)
:Output(6,4,"SCORE")
:Output(6,1Ø,S)
```

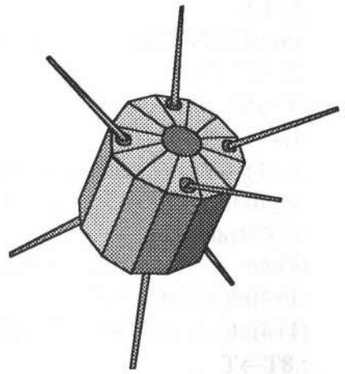




```
:1→T
:Pause
:ClrHome
:ClrDraw
:For(N,1,M)                initialisation des positions
:Ø→L1(N)
:Ø→L2(N)
:Repeat J=Ø
:1+4int (24rand)→X         position en X
:1+4int (16rand)→Y         position en Y
:prgmYRBLITC               lit cette case
:End                       elle est vide
:If N=1
:Then
:prgmYRBJOU                affiche le joueur (n° 1)
:Else
:prgmYRBROB                affiche un robot
:End
:X→L1(N)                   mémorise la position
:Y→L2(N)
:End
:M→N
:prgmYRBJEU                le jeu
:If R≠Ø                    il restait des robots ?
:Goto 1                    oui : c'est la fin...
:S+25ØL→S                  bonus de niveau
:End                       niveau suivant
:Lbl 1
:Output(1,1,"GAME OVER")
:Output(3,1,"SCORE")
:Output(3,8,S)
:If S>W                    record battu
:Then
:Output(5,1,"RECORD BATTU !")
:S→W
```



```
:Else
:Output(6,1,"RECORD :")
:Output(6,1Ø,W)
:End
:Stop
```



Programme XBITMAP :

initialisation de l'écran graphique

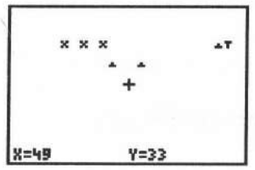
```
:FnOff
:PlotsOff
:GridOff
:LabelOff
:FullScreen
:RectGC
:1→Xmin
:95→Xmax
:Ø→Xscl
:1→Ymin
:63→Ymax
:Ø→Yscl
:DispGraph
:ClrDraw
```

Programme XENTDIR :

```
:Ø→Z                       répète jusqu'à avoir un X et
:Repeat (abs X≤1)*(abs (Y≤1)+(K=F)
:Repeat K                  un Y valide, ou bien que l'on
:getKey→K                  aie pressé la touche F
:End
:1ØfPart .1K-3→X           convertit la touche en un
:8-int .1K→Y               déplacement X Y
:End
:If K=F                    touche spéciale
:1→Z
```

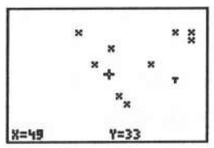
Programme YRBJEU :

```
:Lbl 1
:prgmXENTDIR
:X→U
:Y→V
:L1(1)→X
:L2(1)→Y
:prgmYRBEFFAC
:If Z*(rand≤T)                téléportation demandée et
:Then                         réussie
:1+4int(24rand)→X             nouvelle position
:1+4int(16rand)→Y
:.8T→T
:prgmYRBJOU                   clignotement du joueur
:prgmYRBLITC
:prgmYRBEFFAC
:sin cos sin cos sin 1→A      légère attente
:prgmYRBJOU
:prgmYRBEFFAC
:If J>1                       écrasé un robot ?
:Then
:.2→L1(J)                     robot détruit
:1ØØ+S→S                      1ØØ points
:End
:Else
:X+4U→X                       nouvelle position du joueur
:Y+4V→Y
:End
:X→L1(1)
:Y→L2(1)
:prgmYRBJOU                   ré-affiche le joueur
:prgmYRBENERG                 récupération d'énergie ?
:Ø→R
```




```
:2→I
:Lbl 2                        déplacement des robots
:L1(I)→X
:If fPart X=Ø                 robot actif ?
:Then
:L2(I)→Y
:prgmYRBEFFAC
:L1(1)-X→U                    le robot se déplace vers
:L2(1)-Y→V                    le joueur...
:If U
:X+4U/abs U→X                 X+4 sign U→X
:If V
:Y+4V/abs V→Y                 Y+4 sign V→Y
:prgmYRBLITC
:If J=1                       trouvé le joueur ?
:Then
:1→R
:Return
:End
:If J≥1                       trouvé le robot J
:Then
:.2→L1(J)                     J est supprimé
:X+.1→X                       l'autre devient un débris
:prgmYRBEFFAC
:prgmYRBDEB
:S+2ØØ→S
:Else
:If J=(-)2                    trouvé un débris
:Then
:.2→X
:S+5Ø→S
:Else                         rien trouvé
:prgmYRBROB
:1→R
```

```
:End
:End
:X→L1(I)                    mémorise les coordonnées
:Y→L2(I)
:End
:IS>(I,N)                   robot suivant
:Goto 2
:If R                       encore des robots ?
:Goto 1
```



Programme YRBENERG :

```
:2→J
:Lbl Ø
:If L1(J)≠X+.1              pas de débris ici
:Goto 1
:If L2(J)≠Y                 l'ordonnée n'est pas la bonne
:Goto 1
:Ø.2→L1(J)                  débris détruit
:Pt-Off(X,Y)
:Pt-Off(X+2,Y)
:If T<.9
:T/.8→T                     l'énergie augmente
:Return
:Lbl 1
:IS>(J,N)                   suite de la liste
:Goto Ø
```

Programme YRBLITC :

```
:1→J
:Lbl Ø
```



```
:If X≠int L1(J)             rien à cette abscisse
:Goto 1
:If Y≠L2(J)                 ordonnée différente
:Goto 1
:If fPart L1(J)=Ø           trouvé un robot ?
:Return
:(-)2→J                     non, c'est un débris
:Return
:Lbl 1
:IS>(J,N)                   suivant dans la liste
:Goto Ø
:Ø→J
```

Programme YRBJOU :

```
:Line(X,Y+2,X+2,Y+2)        affiche le joueur
:Line(X+1,Y,X+1,Y+2)
```

Programme YRBROB :

```
:Line(X,Y,X+2,Y+2)          affiche un robot
:Line(X,Y+2,X+2,Y)
```

Programme YRBDEB :

```
:Line(X,Y,X+2,Y)            affiche un débris
:Pt-On(X+1,Y+1)
```

Programme YRBEFFAC :

```
:For(A,Ø,2)                 efface cette case
:For(B,Ø,2)
:Pt-Off(X+A,Y+B)
:End
:End
```

COMMENTAIRES

La particularité de ce programme est de mémoriser le contenu d'une matrice 24x16 sans utiliser de matrice ! En effet, il serait pratique de mémoriser le contenu de chaque case du jeu, afin de lire directement le contenu d'une case lors des déplacements, pour voir s'il y a collision entre deux robots par exemple. Mais sur TI-82, une telle matrice occuperait plus de 3000 octets... Or, on peut faire autrement, et "simuler" une matrice de cette taille avec un peu d'astuce...

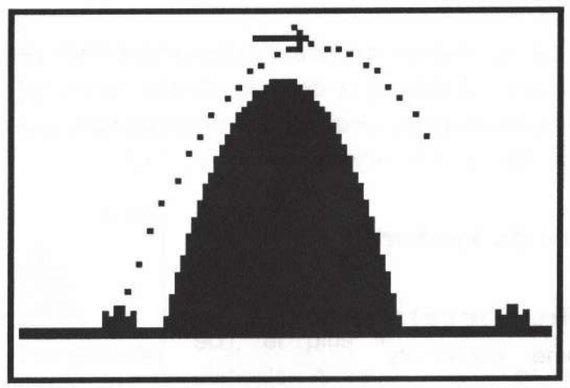
La méthode qui va être employée tient compte de deux particularités de cette matrice que l'on va simuler :
- peu de valeurs possibles : une case peut être soit vide, soit contenir le joueur, un robot ou un débris.
- la matrice est presque vide : au niveau 1 il n'y a que 6 robots, et 19 au niveau 5 par exemple, alors qu'il y a 384 cases (24x16)

La solution qui vient à l'esprit est de mémoriser uniquement les cases occupées, dans un tableau à une dimension. On va mémoriser les couples (X,Y) des coordonnées des robots dans deux listes L_1 et L_2.

Reste à différencier le joueur des robots et des débris. On décide de placer le joueur dans l'élément numéro 1, ce qui permettra de le différencier du reste. Les débris auront 0,1 ajouté à leur abscisse, les robots détruits ayant une abscisse de 0,2.
Les tests de retrouvent simplifiés. Après un déplacement, pour voir s'il y eu collision, on appelle le programme YRBLITC. Il renvoie dans J une valeur qui est :

- 1 si on a rencontré le joueur
- 0 si la case (X,Y) est vide
- -2 si la case contient un débris
- n si la case contient le robot numéro n

Pour comparer des positions, il faut utiliser les parties entières. Les robots morts ont une abscisse nulle, ils n'interfèrent donc pas (abscisse minimum est 1).
Pour tester si le robot I est actif, on regarde si la partie fractionnaire de son abscisse est nulle : fPart $L_1(I) = 0$.



CHATEAUX

C'est la guerre !

Le roi de Westerie a déclaré les hostilités au royaume d'Esterie, sous prétexte que des Esteriens ont été aperçus sur la montagne séparant les deux royaumes, considéré traditionnellement comme un territoire neutre interdit à quiconque. En fait les deux royaumes n'attendaient qu'une occasion pour reprendre les hostilités, mais cette fois ils disposent de canons très puissants capables de tirer bien au-dessus de la montagne, et chacun croit pouvoir

bombarder directement le château ennemi sans se montrer, et sans prendre de risques !...

Hélas ! Chaque château dispose d'un armement similaire, et en fait l'issue du combat dépendra de l'habilité des canonniers à régler leur tir le mieux possible, pour détruire le donjon central du château ennemi, et tous ceux qui s'y trouvent !

Nombre de joueurs : 2

Mémoire nécessaire :

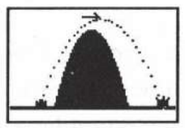
Programme : 800 octets

But du jeu :

régler le tir du canon de manière à envoyer un obus sur la tour centrale du château ennemi.

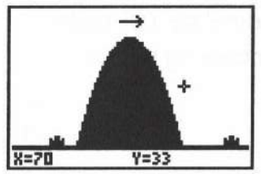
MODE D'EMPLOI

Après avoir entré en mémoire les différents éléments du programme, exécuter le programme "CHATEAUX". Une vue en coupe de la situation apparaît, le château de Westerie à gauche, et celui d'Esterie à droite. Entre les deux, une haute montagne, qui arrêtera tous les boulets qui viendraient à la toucher. En haut de l'écran, une flèche indique la direction du vent. La longueur de la flèche indique la force du vent.



Une croix apparaît au centre de l'écran, et ses coordonnées X et Y s'affichent en bas de l'écran. Déplacez la croix avec les flèches, en regardant les chiffres en bas de l'écran, de manière à donner à X et Y les valeurs désirées, sachant que :

- X représente l'angle de tir, de 1 à 95 degrés
- Y représente la quantité de poudre, de 1 à 63 onces de poudre.Pressez [Enter] une fois les valeurs réglées.



Plus Y est grand, plus le boulet partira avec une grande puissance, et plus il ira loin. Plus X, l'angle de tir, sera proche de 90□, et plus le boulet partira à la verticale. A vous de trouver les bons paramètres, pour que le boulet tombe juste au bon endroit !

Une fois [Enter] pressée, le boulet part, sa trajectoire est représentée par des petits points. S'il tombe ailleurs qu'au centre du château ennemi, il fera simplement un petit cratère. Par contre, s'il touche le donjon, vous aurez droit à une superbe explosion !

Bon amusement !

Note : les positions des châteaux, la hauteur et la largeur de la montagne, la force et la direction du vent sont tirées au hasard au début de la partie. Ne vous étonnez pas de devoir trouver les bons réglages à chaque fois... Ceci dit, l'expérience du tir balistique s'acquiert à chaque partie, et sans nul doute les bons canonniers se feront reconnaître !

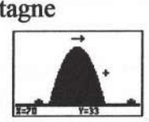
LE PROGRAMME

Le programme est composé des modules :

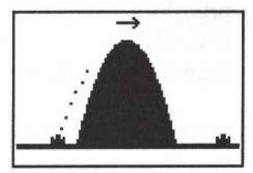
CHATEAUX	programme principal
XBITMAP	initialisation de l'écran graphique
YCHDESS	dessin d'un château en X
YCHJOUE	saisie du coup du joueur, et appel de YCHBOUL
YCHBOUL	calcul et affichage de la trajectoire du boulet
XPAUSE	pause de 1 seconde

Programme CHATEAUX :

```
:Func                          mode fonction Y=F(X)
:Radian                        calculs en radians
:ClrHome
:"H-(BX-L)²"→Y1                équation de la montagne
:prgmXBITMAP
:CoordOn
:3Ø+int (31rand)→H             hauteur
:1Ø+int (11rand)→L
:L/48→B                        largeur
:For(X,1,95)                   tracé de la montagne
:Line(X,7,X,8)
:If Y1>8
:Line(X,8,X,Y1)
:End
:(L-√(H-9))/B-5→G              espace libre
:1+int (Grand)→X               position du château de gauche
:prgmYCHDESS                   dessin des châteaux
:X→U
:91-int (Grand)→X              château de droite
:prgmYCHDESS
```



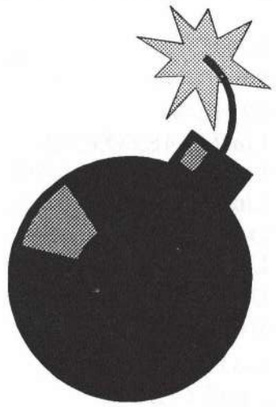

```
:X→V
:8rand-4→E                     calcul du vent
:49+4E→F
:Line(49-4E,6Ø,F,6Ø)           tracé de la flèche
:Line(F,6Ø,49+2E,6Ø+E)
:Line(F,6Ø,49+2E,6Ø-E)
:Ø→G
:Lbl 1                         début du tour
:U+3→D
:V+2→O
:1→J                           joueur de gauche
:prgmYCHJOUE
:If G
:Goto 2
:O→D
:U+1→O
:(-)1→J
:prgmYCHJOUE                   joueur de droite
:If G=Ø
:Goto 1
:Lbl 2
:For(I,Ø,5)                    explosion
:Line(X-1-I/2,Y+I,X,Y+I/3)
:Pt-On(X,Y+I+1)
:Line(X+.5+I/2,Y+I,X,Y+I/3)
:End
:prgmXPAUSE
:ClrHome
:Output(4,3,"VICTOIRE !")
```



Programme XBITMAP : initialisation de l'écran graphique

```
:FnOff
:PlotsOff
:GridOff
:LabelOff
```

```
:FullScreen
:RectGC
:1→Xmin
:95→Xmax
:Ø→Xscl
:1→Ymin
:63→Ymax
:Ø→Yscl
:DispGraph
:ClrDraw
```



dessine le château

Programme YCHDESS :

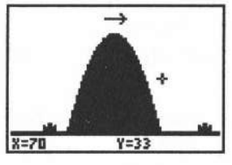
```
:Line(X,9,X+5,9)
:Line(X,1Ø,X+5,1Ø)
:Line(X,11,X+5,11)
:Pt-Off(X+1,11)
:Pt-Off(X+4,11)
:Line(X+2,12,X+3,12)
```

Programme YCHJOUE :

```
:Input                  saisie de x et y (en minuscules)
:X°→A                   angle converti en radians
:Y→F                    force
:prgmYCHBOUL            affiche la trajectoire
:If G=Ø                 et si on n'a pas touché...
:prgmYCHBOUL            ...l'efface
```

Programme YCHBOUL :

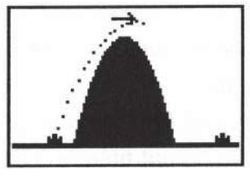
```
:Ø→T
:Ø→S
:E+JFcos A→C            précalcul
:Lbl A
:T+.1→T
:1-S→S
```



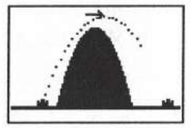

```
:CT+D→X                 calcul de la position
:FTsin A-5T²+12→Y       du boulet
:If S
:Pt-Change(X-.5,Y-.5)   affichée une fois sur 2
:If X<O                 avant le château ?
:Goto B
:If X>O+2               au-delà ?
:Goto B
:If Y>12                au-dessus ?
:Goto A
:1→G                    touché le château !
:Lbl C
:Pt-Off(X,Y)            un trou dans le sol
:Pt-Off(X-1,Y)
:Pt-Off(X+1,Y)
:Pt-Off(X,Y-1)
:Pt-Off(X,Y+1)
:Return
:Lbl B
:If Y<Y1                touché la montagne
:Goto C
:If Y>9                 au-dessus du sol ?
:Goto A
:Goto C                 touché le sol
```

Programme XPAUSE :

```
:For(I,Ø,2ØØ)           boucle d'attente
:End
```



COMMENTAIRES



Dans CHATEAUX, chaque joueur joue en alternance. Les sous-programmes de saisie du coup, de calcul du déplacement du boulet, etc... sont communs aux deux joueurs. Ils sont appelés une fois pour le joueur de gauche, et une fois pour le joueur de droite ; afin de faire la différence entre les deux joueurs, on place dans des variables des paramètres spécifiques du joueur en cours :

- D contient l'abscisse de départ du boulet
- O contient l'abscisse de la tour du château ennemi (l'endroit à atteindre)
- J contient le sens de déplacement du boulet (1 vers la droite, -1 vers la gauche)

Le sous-programme YCHBOUL utilisera ces paramètres, combinés avec l'angle de tir et la force choisis par le joueur, pour calculer la trajectoire du boulet.

Utilisation des fonctions :

La montagne au centre de l'écran est modélisée par l'équation d'une parabole, de centre L. Elle est mémorisée dans Y_1, ce qui permet de l'utiliser sans consommer de pas de programmes supplémentaires. YCHBOUL compare l'ordonnée du boulet avec la valeur de Y_1, pour savoir à un X donné si le boulet est au-dessus de la montagne ou non.

L'équation, $H-(BX-L)^2$, est fonction de X, mais dépend aussi des paramètres H, B et L qui sont tirés au hasard. Cela permet d'avoir une "nouvelle" montagne à chaque partie.

L'équation de la trajectoire du boulet, telle qu'elle est calculée dans le module YCHBOUL, est la résolution du système composé d'un boulet, projeté avec une vitesse initiale F, et un angle initial θ, à partir de l'origine O(0,0). En l'absence de frottements, il est soumis à une seule force, g, la gravité. Selon la relation fondamentale de la dynamique, son accélération vaut donc g.

Projetée sur les axes x, parallèle au sol et passant par O, et y, orthogonal à x, passant par O, orienté vers le haut, l'accélération est a(0,-g).

En prenant la primitive de a, on a le vecteur vitesse v, de valeurs initiales v(t=0)(F cos θ, F sin θ) :

v(F cos θ, F sin θ - gt)

En prenant à nouveau la primitive, on trouve les coordonnées B(x,y) du boulet, de valeurs initiales B(t=0)(0,0) :

$B((F \cos \theta)t, (F \sin \theta)t - (g/2)t^2))$

Pour tenir compte du vent, de vitesse e, on considère que l'équation B(x,y) obtenue est celle du boulet non pas par rapport à un repère fixe, mais par rapport à l'air où est plongé le boulet. Cet air se déplace par rapport au sol à la vitesse e. On fait donc un changement de repère, en ajoutant e à la composante x de la vitesse. Le vecteur vitesse par rapport au sol est donc :

v'(F cos θ + e, F sin θ - gt)

et le mouvement devient :

$B'((F \cos \theta + e)t, (F \sin \theta)t - (g/2)t^2))$

Dans YCHBOUL, pour gagner du temps, F cos θ + e est précalculé par la ligne (A représente θ) :

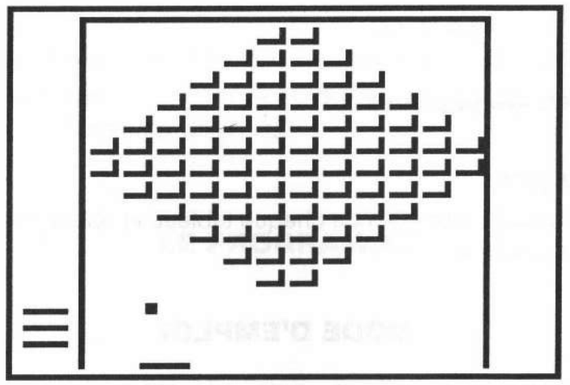
:E+JFcos A→C.

Les autres équations sont écrites telles quelles :

:CT+D→X

:FT*sin A-5T²+12→Y

D et 12 sont les valeurs initiales pour X et Y (le tir part du sommet du château). g est pris égal à 10 (valeur approchée de 9,81, suffisante par rapport à la résolution de l'écran), g/2 vaut donc 5. J est le sens de déplacement du boulet : 1 vers la droite, -1 vers la gauche.



CASSE-BRIQUES

Le célèbre jeu d'arcade sur votre calculatrice TI-82 ! Au moins aussi bien que l'original, avec des angles de rebonds variables, une animation rapide, des briques en relief, et 8 tableaux différents !!! De plus les tableaux proposés et le système de score encouragent vivement à faire un trou le plus vite possible au milieu des briques afin de passer derrière et les détruire, en laissant travailler la balle. Visez bien, et faites des compétitions avec ce superbe jeu !

Mémoire nécessaire :

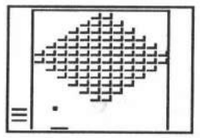
Programme : 1600 octets
Données : 1800 octets

Nombre de joueurs : 1

But du jeu :

Détruire toutes les briques de chaque tableau et réaliser le score le plus haut possible.

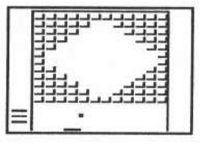
MODE D'EMPLOI



Utilisez les flèches droite et gauche pour déplacer la raquette. Frappez la balle avec les extrémités de la raquette pour contrôler sa direction après le rebond.

← ◀ ▶ →

Une brique détruite rapporte 10 points. Tant que la balle ne touche pas la raquette, la valeur de briques augmente, jusqu'à un maximum de 200 points par brique. Visez bien, de manière à détruire un maximum de briques avec un minimum de rebonds sur la raquette. Lorsque vous rattrapez la balle, la valeur des briques retombe à 10 points.



Vos raquettes restantes sont affichées sur la gauche de l'écran. Si vous ne rattrapez pas la balle, vous perdez une de vos trois raquettes.

Heureusement tous les 5000 points vous gagnez une nouvelle raquette.

Lorsque vous avez détruit toutes les briques, un autre tableau vous est proposé avec une nouvelle configuration des briques. Le jeu comprend huit tableaux différents.

LE PROGRAMME

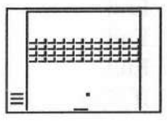
Il est composé de :

CASSE	programme principal
XBITMAP	initialisation de l'écran graphique.

Reportez-vous à la partie "Commentaires" après le listing afin de créer facilement vos propres tableaux !

Programme CASSE :

```
:prgmXBITMAP
:{12,16}→dim [A]                matrice contenant le mur
:{(-)2,(-)2,(-)1,0,1,2,2,2}→L1  rebond de la balle sur la raquette
:4→V                            nb de vies = nb de raquettes
:5ØØØ→U                         score pour gagner la 1ere vie
:Ø→S                            score
:Ø→T                            numéro du tableau
:Lbl 1                          initialisation du tableau
:Fill(Ø,[A])
:T+1→T
:If T=1                         crée le tableau 1
:Then
:For(I,1,12)
```



```
:For(J,1,4)
:1→[A](I,J+8)
:End
:End
:End
:If T=2
:Then
:For(I,1,12)
:For(J,1,3)
:1→[A](I,J+6)
:1→[A](I,J+11)
:End
:End
:End
:If T=3
:Then
:For(I,1,8)
:For(J,1,4)
:1→[A](I+2,J+8)
:1→[A](J+4,I+6)
:End
:End
:End
:If T=4
:Then
:For(I,1,1Ø)
:For(J,1,3)
:1→[A](I+1,J+6)
:1→[A](I+1,J+13)
:1→[A](J+1,I+6)
:1→[A](J+8,I+6)
:End
:End
:End
:If T=5
```

tableau 2

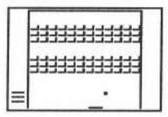


tableau 3

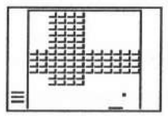


tableau 4

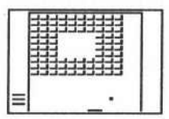


tableau 5

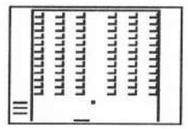

```
:Then
:For(I,1,6)
:For(J,1,I)
:1→[A](7-J,I+4)
:1→[A](6+J,I+4)
:1→[A](7-J,17-I)
:1→[A](6+J,17-I)
:End
:End
:End
:If T=6
:Then
:For(I,1,6)
:For(J,1,I)
:1→[A](J,I+1Ø)
:1→[A](13-J,I+1Ø)
:1→[A](J,11-I)
:1→[A](13-J,11-I)
:End
:End
:End
:If T=7
:Then
:For(I,1,12)
:1→[A](1,I+4)
:1→[A](3,I+4)
:1→[A](5,I+4)
:1→[A](8,I+4)
:1→[A](1Ø,I+4)
:1→[A](12,I+4)
:End
:End
:If T=8
:Then
```

tableau 6

tableau 7

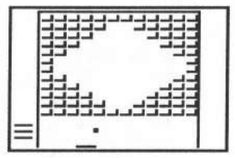
tableau 8

```
:For(I,1,6)
:For(J,1,5)
:1→[A](2*I-1,2*J+5)
:1→[A](2*I,2*J+6)
:End
:End
:End
:If T=9                    fin du jeu
:Then
:ClrHome
:Output(3,5,"GAGNE !")
:Goto 2
:End
:Lbl 3                     début : nouvelle balle
:ClrHome
:Output(3,4,"TABLEAU")
:Output(3,12,T)
:Output(6,4,"SCORE")
:Output(6,1Ø,S)
:For(I,1,1ØØØ)
:End
:ClrDraw                   affichage du terrain
:DispGraph
:Vertical 11
:Vertical 84
:Line(11,63,84,63)
:Ø→N
:For(J,1,16)               affichage des briques
:For(I,1,12)
:If [A](I,J)=1
:Then
:N+1→N                     totalise le nombre de briques
:7+6I→X                    dessin d'une brique
:(-)4+4J→Y
:Line(X+4,Y,X+4,Y+2)
```

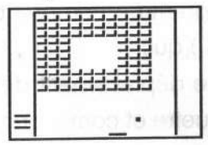



```
:Line(X,Y,X+3,Y)
:End
:End
:End
:If V>1                    affichage des raquettes
:Then
:For(I,1,V-1)
:Line(3,2I,8,2I)
:End
:End
:45→R                      abscisse de la raquette
:Line(45,3,50,3)           affichage de la raquette
:48→A                      abcisse de la balle
:4→B                       ordonnée de la balle
:1→C                       direction en abcisse de la balle
:(-)4→D                    "  en ordonnée de la balle
:Ø→P                       compteur
:Ø→L                       touche pressée au coup précédent
:Lbl 4                     début de la boucle d'animation
:If A≤12                   rebond sur le mur gauche
:abs C→C
:If A≥82                   rebond sur le mur droit
:(-)(abs C)→C
:If B≥6Ø                   rebond sur le mur du haut
:(-)4→D
:If B≤4                    balle au niveau de la raquette
:Then
:If (A≥R-1)*(A≤R+5)        raquette sous la balle ?
:Then
:(-)D→D                    rebond vertical
:Ø→P
:Lı(A-R+2)→C               direction de la balle après le
:End                       rebond sur la raquette
:If B≤2                    la balle a-t-elle dépassée la
```

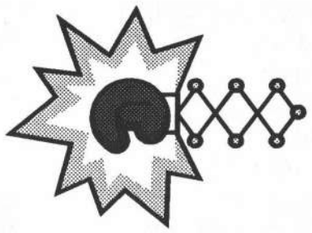
```
:Then                          raquette
:V-1→V                         perte d'une vie
:If V=Ø                        plus de vie ?
:Then
:ClrHome
:Output(3,4,"GAME OVER")
:Goto 2
:End
:Goto 3                        si il reste une vie retour au
:End                           label 3
:End
:Line(A,B,A+1,B,Ø)             efface la balle
:Line(A,B+1,A+1,B+1,Ø)
:A+C→A                         nouvelle position de la balle
:B+D→B
:Line(A,B,A+1,B)               affiche la balle
:Line(A,B+1,A+1,B+1)
:int ((A-6)/6)→E               calcul de la position de la
:min(E,12)→E                   balle dans la matrice [A]
:max(E,1)→E
:int ((B+4)/4)→F
:If [A](E,F)=1                 la balle est-elle sur
:Then                          une brique ?
:(-)D→D                        rebond de la balle sur la brique
:Ø→[A](E,F)                    efface la brique dans la matrice
:min(2ØØ,int (9+e^(√P)))+S→M   points gagnés
:If (int (M/U)-int (S/U))≠Ø    gain d'une vie ?
:Then
:V+1→V                         ajoute une vie
:Line(3,2V-2,8,2V-2)           dessine la nouvelle vie
:End
:M→S
:P+1→P
:7+6E→G                        efface la brique
```




```
:(-)4+4F→H
:Line(G+4,H,G+4,H+2,Ø)
:Line(G,H,G+3,H,Ø)
:N-1→N                         une brique de moins !
:If N≤Ø                        reste-t-il des briques ?
:Goto 1
:End
:Ø→Q
:getKey→K                      saisie d'une touche
:If (K=24)*(R>12)              flèche gauche
:(-)2→Q
:If (K=26)*(R<78)              flèche droite
:2→Q
:If K=L                        déplacement rapide
:2Q→Q
:If Q
:Then
:Line(R,3,R+5,3,Ø)
:R+Q→R
:Line(R,3,R+5,3)               dessine la raquette
:End
:K→L
:Goto 4                        fin de la boucle d'animation
:Lbl 2                         fin de partie
:S+5ØØ*V→S                     chaque raquette restante rapporte
:Output(6,4,"SCORE :")         5ØØ points
:Output(6,1Ø,S)
:Stop

Programme XBITMAP :            initialisation de l'écran graphique
  :FnOff
  :PlotsOff
  :GridOff
  :LabelOff
  :FullScreen
```



```
:RectGC
:1→Xmin
:95→Xmax
:Ø→Xscl
:1→Ymin
:63→Ymax
:Ø→Yscl
:DispGraph
:ClrDraw
```



COMMENTAIRES

Le programme comprend une boucle principale (entre "Lbl 4" et "Goto 4") qui:

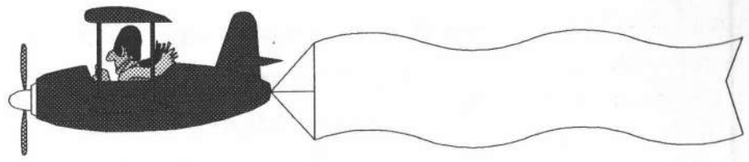
- gère le déplacement de la balle : rebond contre les murs, contre la raquette et contre une brique.
- gère le déplacement de la raquette : déplacement rapide ou non, vers la droite ou vers la gauche (le déplacement rapide se produit lorsqu'on garde une touche appuyée : le mouvement est alors deux fois plus rapide)

Au label 3 on lance une nouvelle balle sur le tableau en cours, à condition qu'il reste encore au moins une raquette.
Au label 1 un nouveau tableau est initialisé, en début de partie ou après avoir terminé un tableau.

Créer de nouveaux tableaux :

Vous pouvez créer vos propres tableaux dans la partie du programme qui se trouve après "Lbl 1". Les tableaux sont calculés

et stockés dans la matrice [A] ; regardez comment ils sont faits, et inspirez-vous en pour faire vos propres tableaux !
Vous pouvez simplement modifier les tableaux existants, ou en rajouter de nouveaux en plus des 8 existants auquel cas vous rajouterez des lignes sur le modèle de celles existantes (If T='no du tableau'... jusqu'au End). Modifiez également le test de fin de jeu (If T='no du dernier tableau'+1).



Collection

CALCULATRICES EFFICACES

Les grands atouts de la "petite informatique"

Extrait du catalogue de la collection pour les calculatrices

TEXAS INSTRUMENTS

- **TI-81 : programmez votre succès !**
- **300 programmes TI-81**
- **Jeux et graphisme sur TI-81**
- **TI-81 : le "top" des jeux !**
- **TI-82 : mathématiques au lycée**
- **TI-82 : programmes pour le lycée**
- **Ti-82 : le "top" des jeux !**
- **TI-85 par l'exemple**
- **TI-85 du lycée à la prépa**
- **TI-85 : le "top" des jeux !**

DUNOD TECH

Imprimé en France. - JOUVE, 18, rue Saint-Denis, 75001 PARIS
N° 212350J. - Dépôt légal : Septembre 1993

CATALOGUES ET "LIVRES MICRO"

Je désire recevoir gratuitement :

❒ les catalogues DUNOD/TECH suivants : ▫ Informatique ▫ Photo-Vidéo ▫ Electronique

❒ la revue "Livres Micro" sur les nouveautés DUNOD/TECH

❒ le(s) catalogue(s) DUNOD suivants :

▫ Management, Marketing ▫ Sciences
▫ Economie, Gestion ▫ Sciences humaines
▫ Lettres ▫ Enseignement technique

❒ le catalogue de la Librairie Technique Texas Instruments

QU'EN PENSEZ-VOUS ?

Pour nous permettre de vous proposer des livres répondant toujours davantage à vos besoins, merci de nous faire part de vos remarques et suggestions sur : ***TI-82 : le "top" des jeux !*** (0 41985)

Ce livre vous donne-t-il toute satisfaction ? .....
Avez-vous des commentaires à formuler ? .....
.....
.....
Avez-vous déjà acquis des livres ❒ Dunod/Tech ❒ Dunod ❒ Texas ?
Si oui lesquels ? .....
.....
Qu'en pensez-vous ? .....
.....
.....
Où les avez-vous acquis ? ▫ Librairie ▫ Par correspondance ▫ Boutique micro ▫ Stage de formation
Votre centre d'intérêt ? ▫ PC (ou compatibles) ▫ Macintosh ▫ Autre.....

❒ Mr ❒ Mme ❒ Mlle..... Prénom .....
Société ..... Profession .....
Adresse .....
.....
Code Postal ⌊_⌊_⌊_⌊_⌊_⌋ Ville .....

Merci de renvoyer à : **DUNOD/TECH**
Courrier des lecteurs
BP 20
92122 MONTROUGE Cedex