Tetris4px
por @ignacobo

****

1- INSTRUCCIONES

Tributo al clásico juego Tetris. Está realizado en BASIC puro para ZXSpectrum 48K, presentado al concurso de Radastan Bytemaniacos 2023.

Ejecutar en ZXSpectrum 48kb (excepto issue2)

Misión

Tienes que guiar las fichas que van cayendo, y encajarlas de forma que consigas hacer lineas horizontales. Cuando completas una, se elimina del tablero y baja el bloque que estuviera por encima de esa fila. Puedes mover la ficha a la izquierda, derecha, abajo, y rotar la posición.

Tipos de fichas

Hay 7 tipos de fichas diferentes, todas las que se pueden formar usando 4 cuadradillos básicos.

Menú principal.

Estas son las opciones seleccionadles en el menú:

Pueden jugar 1 o 2 jugadores simultáneamente. Las teclas son:

	Izq	Drch	Abajo	Rotar
Player 1 (azul)	o	p	m	j
Player 2 (rojo)	e	r	z	a

Music in-game: El juego tiene Musica In-Game, que se puede desactivar. Toca una nota Beep cada 25 frames. A partir del nivel 9, a medida que aumenta la velocidad de las fichas también aumenta la velocidad de la música. Para 2 jugadores, la música se desactiva automáticamente por motivos de rendimiento.

Grid: es una matriz de brillo encendido y apagado que puede servirnos de guía a la hora de encajar las fichas.

Texture: hay 9 tipos de gráficos diferentes para decorar las fichas. Se pueden seleccionar 3 conjuntos gráficos distintos.

Speed: Podemos elegir el Nivel de velocidad con la que empieza la partida. Hay unos 16 niveles diferentes. Si elegimos MIN se selecciona la velocidad mas lenta, nivel 1. MED establece velocidad media, nivel 9, y MAX es la velocidad más rápida que da el engine, nivel 16.

Play: empieza la partida.

****

2- DATOS TÉCNICOS

Lo más destacable del diseño es el movimiento de las fichas de 4 en 4 pixels, en lugar de hacerlo de carácter a carácter, que es lo habitual en juegos en Basic. Tanto horizontal como verticalmente, las fichas se mueven a medio carácter. Para lograrlo, cada grupo de texturas necesita dos juegos de 16 caracteres, que se imprimen usando OVER 1 pars crear hasta 3 tipos

de gráficos diferentes, y que al dibujar en una posición, no borre las posibles fichas que ya hubiera en ese carácter.

También es reseñable la música in-game a base de Beep, una nota cada 25 frames. Es poco habitual en un juego Basic. A medida que aumenta la velocidad convierto alguna linea en REM, para que el interprete no la ejecute, y el engine vaya algo mas rápido. Esto se hace en run-time, y las lineas modificadas son de la 24 a la 29. Después de hacer BREAK hay que ejecutar GOTO 9500 para restablecer estas lineas a su código original, y poner el juego de caracteres estándar, para poder ver el código. Para arrancar de nuevo, RUN 9000. Importante: como pokeo directamente en posiciones de memoria donde está el programa Basic, no se pueden editar las lineas desde la primera hasta la 29, o si no, el programa ya no se ejecutaría.

El engine principal del juego va desde la primera linea hasta la 29, para el jugador 1. Como puede verse en el código, se repite el mismo engine a partir de la linea 200 hasta la 300 replicando el código para el jugador 2.

Para el jugador 2 utilizo DefAdd para copiar ñas 11 variables que necesita y escribirlas sobre la linea REM 9993 al final del programa Basic (unos 100bytes). Cuando el engine tiene que trabajar con las fichas del jugador 2, primero pokea la variable VARS a esa zona REM. Al volver al jugador 1, restablece VARS al valor original de la zona de variables Basic.

Es cierto que la respuesta al teclado y la velocidad en 2 jugadores no es todo lo buena que desearía.

Para intentar acelerar la ejecución, he utilizado estas técnicas:
*El bucle principal empieza en la primera línea del código.
*Todos los PRINT terminan con un ; que hace que no se envíe RetornoDeCarro, ahorrando algunos frames.
*La mayoría de las condiciones de los IFs son Var o NOT Var, para que el interprete tenga que leer menos caracteres.
*Incluir los códigos de color dentro de las cadenas a imprimir, ya que INK, PAPER, etc son algo más lentos.
*La máquina de estados del engine se puede implementar con unas 5 líneas y 2 prints, pero en cada iteración realiza muchos cálculos y el print estaría lleno de AND, OR, y otras condiciones que ralentizan mucho. En el fichero Fichas, en el .TAP esta la información necesaria para imprimir cada posición, cada movimiento, cada rotación, incluyendo ya los cálculos necesarios, desplegando cada uno de los posibles estados y hay un print para cada estado, sin necesidad de cálculos redundantes, ganando mucha velocidad, pero a costa de gastar mucha memoria(el fichero ocupa 6kb!), unos 27 prints para el mismo engine (lineas 53 a 75 y 156 a 169), dependiendo de cuantos caracteres hay que dibujar sugar la ficha, y la posición actual.
*Repito mucho código en diferentes líneas, gastando mucha memoria, para evitar saltos GoSub que ahorrarían bytes, a costa de velocidad.
*Declarar las primeras las variables a las que se acceda con mayor frecuencia.
*A partir de la variable numero 13 o 14, la búsqueda de la variable es más lento que el acceso a una posición de memoria con POKE y PEEK, por lo que he sustituido las variables Byte a partir de la 13 por POKEs. El gran inconveniente de esto es que la casi imposibilita la legibilidad y depuración del código, así que solo recomiendo hacer esto al final del desarrollo del proyecto. Además, este cambio ocupa mucha más memoria.

*Para desplazar el bloque hacia abajo cuando has hecho una fila, utilizo DefAdd para mover los bytes del tablero de manera rápida a la zona de memoria necesaria.

****

3-CONTENIDO DEL FICHERO TETRIS4PX.TAP

tetris4px.bas
Código BASIC con la lógica del juego, y desde el que se cargan los ficheros auxiliares

tetris4px.bin

Pantalla de presentacion

udgNums2 y udgAZ

Redefinicion parcial del juego de caracteres, letras y números de 5 bits de alto

defadd1, defadd2 y defadd3

Estructura de datos para usar el truco de defadd, para mover gran cantidad de bytes en los tableros de juego, o en la zona de variables de inicialización y del jugador 2

TablasC

Conjunto de 7 tablas auxiliares unidas en un solo fichero:

- Q: tabla que devuelve operación lógica A AND B de dos numero de 4 bits. Hacer esa operación en tiempo de ejecución seria muy lento.
- P: tabla que devuelve operación lógica A OR B de dos numero de 4 bits.
- V: tabla que devuelve los dos bits menos significativos de un numero de 4 bits.
- W: tabla que devuelve los dos bits más significativos de un numero de 4 bits.
- Y: tabla ascii carácter textura 1
- Z: tabla ascii carácter textura 2
- EvenOdd: tabla que nos dice si un número es par o impar

music

Notas para que suene con Beep cada 25 frames. La melodía es la del juego original

initData

Inicialización de variables y tableros de trabajo. Después de un GameOver copiamos de nuevo este fichero en la zona de variables.

DimO

Array numérico que almacena “punteros” al inicio de las estructuras de datos de cada ficha, según su orientación, y según su cuadrante dentro del carácter. Su tamaño es O(7,4,4) que son 7 fichas, con hasta 4 orientaciones, y 4 cuadrantes cada una.

editorTetris.bas

utilidad auxiliar que hice para crear cada uno de los registros de manera más automatizada y evitar errores manuales.

tetrisDATA.bas

Código BASIC utilizado para generar algunas de las tablas y ficheros anteriores. Es necesario hacerlo en archivo .bas separado del principal para ahorrar muuuchos bytes (DATAs) y velocidad de carga de los datos en memoria (READ)

Fichas

Información de los huecos que necesito para poder mover la ficha o rotarla, así como datos de qué cuadrantes hay que pintar o borrar para ese movimiento.
Este registro es necesario para cada ficha (7), para cada posible rotación (hay fichas con 4 rotaciones, con 2, y con 1), y esta información para cada uno de los 4 cuadrantes de un carácter de 8x8.
En total 76 registros, unos 6k bytes! Creé una utilidad en Basic (editorFichas.bas) para crear cada uno de los registros de manera más automatizada y evitar errores manuales.

Para cada ficha, datos comenzando en cada uno de los 4 cuadrantes de un carácter, y para cada posición rotada de esa ficha.

Para las fichas L 7 T:
son 4 posibles rotaciones, y 4 anclajes cada uno, en total 16 registros. Como son 3 tipos de fichas, 48 registros.

Para las fichas Z S I: son 2 posibles rotaciones, y 4 anclajes cada una, en total 8

registros. Como son 3 tipos de fichas, 24 registros.

Para la ficha cuadrado: son 1 sola rotación, y 4 anclajes cada una, en total 4 registros.

Sumando todos los registros, 76 registros contienen los datos de todas las fichas, sus anclajes y posibles rotaciones.

Información de cada RegistroFicha:

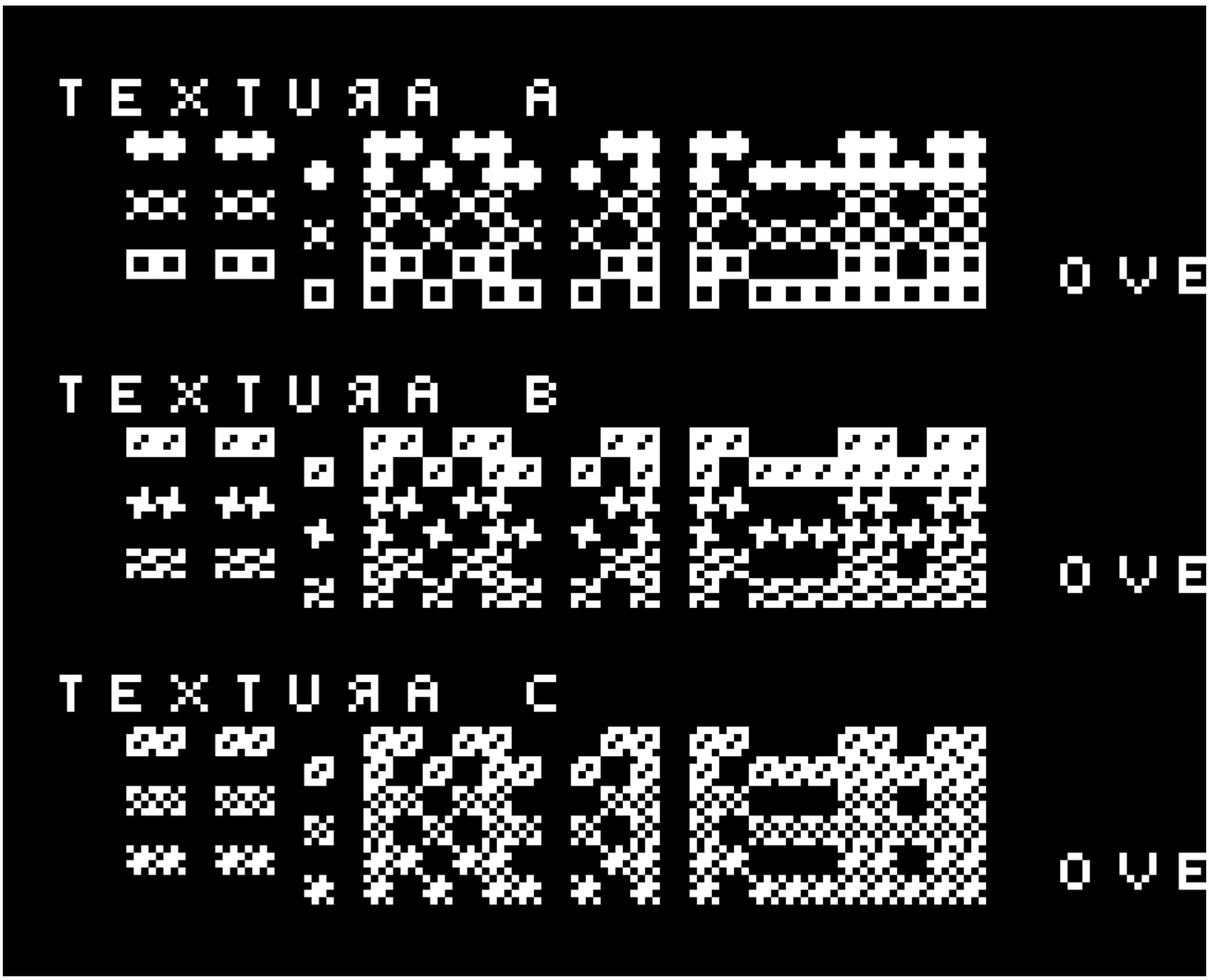
Registro(TipoFicha, Orientacion, Anclaje):
Nueva Orientacion y Anclaje tras este movimiento
Puntero a MascaraIzq
Puntero a MascaraDrc
Puntero a MascaraAbajo
Puntero a MascaraRotar

Puntero a DibujoIzq
Puntero a DibujoDrc
Puntero a DibujoAbajo
Puntero a DibujoRotar

MASCARA Izq:
    PosX,PosY del hueco que hace falta para permitir mover a la Izq
    …
    Asi hasta un máximo de 3 veces
…..

Dibujo Izq:
    Carácter (0a15) que hay que pintar con OVER 1 en la celda de posición B,C. Al ser OVER 1, mantiene las otras fichas que ya hubiera en esa celda, y además permite el borrado de las partes de esta ficha que ya no sean necesarias.

    textures
6 bancos de 16 gráficos (128bytes), que usamos de dos en dos según la textura elegida en el menu. Como combinamos los bancos con Over 1, con dos bancos podemos usar 3 tramas diferentes: las del banco A, la del banco B y la del A over B.


TEXTURA A
OVE
TEXTURA B
OVE
TEXTURA C
OVE

****
4- CODIGO DEL PROGRAMA
tetris4px.bas

```
0 RETURN
Comprobar pulsacion ABAJO
2 IF IN 32766<>187 THEN GO TO IN 57342-171: REM salta a 20 si no pulso o(a 18) ni p(a 19).

Ver si la ficha puede mover abajo, si no, salta a Incrustar en el tablero
3 LET E=U(A): LET M=E+PEEK (E+1): LET J=50748+7*B+C: GO SUB PEEK M AND V<=B: IF M
THEN LET M=E+PEEK (E+5): GO SUB PEEK M: LET B=B+PEEK (E+15): LET A=PEEK (61140+A):
GO TO 5
4 GO TO 82: REM salta a Incrustar
Comprobar pulsacion Izq o Drc
7 IF IN 57342<>189 THEN GO TO 19

18 LET M=U(A)+20: LET J=50747+7*B+C: GO SUB PEEK M: IF M THEN LET E=U(A): LET C=C-1:
LET M=E+PEEK (E+3): GO SUB PEEK M: LET C=C+PEEK (E+10): LET A=PEEK (61143+A): GO
TO 20: REM izq

19 IF IN 57342=190 THEN LET E=U(A): LET M=E+PEEK E: LET J=50748+7*B+C: GO SUB PEEK
M: IF M THEN LET M=E+PEEK (E+4): GO SUB PEEK M: LET C=C+PEEK (E+13): LET A=PEEK
(61143+A): REM drch

Comprobar pulsacion Rotar. Si cabe, cargamos del vector los datos de la ficha rotada
20 IF IN 49150=183 THEN LET E=U(A): LET M=E+PEEK (E+2): LET J=50740+7*B+C: GO SUB
PEEK M: IF M THEN LET B=B-1: LET C=C-1: LET M=E+PEEK (E+6): GO SUB PEEK M: LET
B=B+PEEK (E+18): LET C=C+PEEK (E+19): LET M=PEEK (E+17): LET E=PEEK 60546: LET
U(1)=O(E,M,1): LET U(2)=O(E,M,2): LET U(3)=O(E,M,3): LET U(4)=O(E,M,4): LET O=M

Gestión de timing para tocar musica, una nota cada 25 frames
24 IF PEEK 23672-T>25 THEN LET i=62560+PEEK (i+110): LET T=PEEK 23672: IF PEEK i THEN
BEEP .02,PEEK i
25 IF PEEK 23673 THEN LET T=T-256: POKE 23673,0: LET Y=Y-256

Gestión de timing para bajar la ficha, una vez cada Y frames
27 IF PEEK 23672-Y>PEEK 51000 THEN LET Y=PEEK 23672: GO TO PEEK 60550
28 IF PEEK 23672-Y>PEEK 51000 THEN LET Y=PEEK 23672: LET i=62560+PEEK (i+110): BEEP
.015 AND PEEK i,PEEK i: GO TO PEEK 60550
29 LET i=62560+PEEK (i+110): BEEP .01 AND PEEK i,PEEK i: GO TO PEEK 60550

35 GO TO PEEK 60551: REM *** FIN ENGINE PPAL Player1 ***

Código para comprobar si la ficha cabe en la siguiente posición o están ya ocupadas
41 IF PEEK (60880+PEEK (M+6)+PEEK (61152+PEEK (J+PEEK (M+5)))) THEN LET M=0: RETURN
43 IF PEEK (60880+PEEK (M+4)+PEEK (61152+PEEK (J+PEEK (M+3)))) THEN LET M=0: RETURN
45 IF PEEK (60880+PEEK (M+2)+PEEK (61152+PEEK (J+PEEK (M+1)))) THEN LET M=0: RETURN
50 RETURN : REM *** 41 a 50 Comprobar si hay hueco para mover? ***

Imprimir la parte necesaria de la nueva posición de la ficha. Hay 26 Prints distintos, según el
número de filas y casillas que haya que pintar. Se les llama desde el engine ppal, en las
pulsaciones de teclas
53 PRINT AT B+PEEK (M+1),C+PEEK (M+2);CHR$ PEEK (M+3);CHR$ PEEK (M+4);AT B+PEEK
(M+5),C+PEEK (M+6);CHR$ PEEK (M+7);CHR$ PEEK (M+8);: RETURN
54 PRINT AT B+PEEK (M+1),C+PEEK (M+2);CHR$ PEEK (M+3);CHR$ PEEK (M+4);AT B+PEEK
(M+5),C+PEEK (M+6);CHR$ PEEK (M+7);: RETURN
55 PRINT AT B+PEEK (M+1),C+PEEK (M+2);CHR$ PEEK (M+3);AT B+PEEK (M+4),C+PEEK
(M+5);CHR$ PEEK (M+6);CHR$ PEEK (M+7);: RETURN
```

```
56 PRINT AT B+PEEK (M+1),C+PEEK (M+2);CHR$ PEEK (M+3);CHR$ PEEK (M+4);CHR$ PEEK
(M+5);: RETURN
57 PRINT AT B+PEEK (M+1),C+PEEK (M+2);CHR$ PEEK (M+3);AT B+PEEK (M+4),C+PEEK
(M+5);CHR$ PEEK (M+6);: RETURN
58 PRINT AT B+PEEK (M+1),C+PEEK (M+2);CHR$ PEEK (M+3);CHR$ PEEK (M+4);: RETURN
59 PRINT AT B+PEEK (M+1),C+PEEK (M+2);CHR$ PEEK (M+3);CHR$ PEEK
(M+5);"\#008\#008";CHR$ PEEK (M+4);CHR$ PEEK (M+6);AT B+PEEK (M+7),C+PEEK
(M+8);CHR$ PEEK (M+9);CHR$ PEEK (M+11);"\#008\#008";CHR$ PEEK (M+10);CHR$ PEEK
(M+12);: RETURN
60 PRINT AT B+PEEK (M+1),C+PEEK (M+2);CHR$ PEEK (M+3);AT B+PEEK (M+4),C+PEEK
(M+5);CHR$ PEEK (M+6);AT B+PEEK (M+7),C+PEEK (M+8);CHR$ PEEK (M+9);: RETURN
61 PRINT AT B+PEEK (M+1),C+PEEK (M+2);CHR$ PEEK (M+3);"\#008";CHR$ PEEK (M+4);AT
B+PEEK (M+5),C+PEEK (M+6);CHR$ PEEK (M+7);CHR$ PEEK (M+9);"\#008\#008";CHR$ PEEK
(M+8);CHR$ PEEK (M+10);: RETURN
62 PRINT AT B+PEEK (M+1),C+PEEK (M+2);CHR$ PEEK (M+3);CHR$ PEEK
(M+5);"\#008\#008";CHR$ PEEK (M+4);CHR$ PEEK (M+6);AT B+PEEK (M+7),C+PEEK
(M+8);CHR$ PEEK (M+9);"\#008";CHR$ PEEK (M+10);: RETURN
63 PRINT AT B+PEEK (M+1),C+PEEK (M+2);CHR$ PEEK (M+3);CHR$ PEEK
(M+5);"\#008\#008";CHR$ PEEK (M+4);CHR$ PEEK (M+6);: RETURN
64 PRINT AT B+PEEK (M+1),C+PEEK (M+2);CHR$ PEEK (M+3);CHR$ PEEK (M+4);AT B+PEEK
(M+5),C+PEEK (M+6);CHR$ PEEK (M+7);CHR$ PEEK (M+8);AT B+PEEK (M+9),C+PEEK
(M+10);CHR$ PEEK (M+11);: RETURN
65 PRINT AT B+PEEK (M+1),C+PEEK (M+2);CHR$ PEEK (M+3);CHR$ PEEK (M+4);AT B+PEEK
(M+5),C+PEEK (M+6);CHR$ PEEK (M+7);CHR$ PEEK (M+8);AT B+PEEK (M+9),C+PEEK
(M+10);CHR$ PEEK (M+11);CHR$ PEEK (M+12);: RETURN
66 PRINT AT B+PEEK (M+1),C+PEEK (M+2);CHR$ PEEK (M+3);CHR$ PEEK (M+4);CHR$ PEEK
(M+5);AT B+PEEK (M+6),C+PEEK (M+7);CHR$ PEEK (M+8);CHR$ PEEK (M+9);CHR$ PEEK
(M+10);: RETURN
67 PRINT AT B+PEEK (M+1),C+PEEK (M+2);CHR$ PEEK (M+3);AT B+PEEK (M+4),C+PEEK
(M+5);CHR$ PEEK (M+6);CHR$ PEEK (M+7);CHR$ PEEK (M+8);: RETURN
68 PRINT AT B+PEEK (M+1),C+PEEK (M+2);CHR$ PEEK (M+3);CHR$ PEEK (M+4);CHR$ PEEK
(M+5);AT B+PEEK (M+6),C+PEEK (M+7);CHR$ PEEK (M+8);: RETURN
69 PRINT AT B+PEEK (M+1),C+PEEK (M+2);CHR$ PEEK (M+3);AT B+PEEK (M+4),C+PEEK
(M+5);CHR$ PEEK (M+6);CHR$ PEEK (M+7);AT B+PEEK (M+8),C+PEEK (M+9);CHR$ PEEK
(M+10);: RETURN
70 PRINT AT B+PEEK (M+1),C+PEEK (M+2);CHR$ PEEK (M+3);AT B+PEEK (M+4),C+PEEK
(M+5);CHR$ PEEK (M+6);CHR$ PEEK (M+7);CHR$ PEEK (M+8);AT B+PEEK (M+9),C+PEEK
(M+10);CHR$ PEEK (M+11);: RETURN
71 PRINT AT B+PEEK (M+1),C+PEEK (M+2);CHR$ PEEK (M+3);AT B+PEEK (M+4),C+PEEK
(M+5);CHR$ PEEK (M+6);AT B+PEEK (M+7),C+PEEK (M+8);CHR$ PEEK (M+9);CHR$ PEEK
(M+10);: RETURN
72 PRINT AT B+PEEK (M+1),C+PEEK (M+2);CHR$ PEEK (M+3);CHR$ PEEK (M+4);AT B+PEEK
(M+5),C+PEEK (M+6);CHR$ PEEK (M+7);AT B+PEEK (M+8),C+PEEK (M+9);CHR$ PEEK (M+10);:
RETURN
73 PRINT AT B+PEEK (M+1),C+PEEK (M+2);CHR$ PEEK (M+3);CHR$ PEEK (M+4);CHR$ PEEK
(M+5);AT B+PEEK (M+6),C+PEEK (M+7);CHR$ PEEK (M+8);CHR$ PEEK (M+9);: RETURN
74 PRINT AT B+PEEK (M+1),C+PEEK (M+2);CHR$ PEEK (M+3);CHR$ PEEK (M+4);AT B+PEEK
(M+5),C+PEEK (M+6);CHR$ PEEK (M+7);CHR$ PEEK (M+8);CHR$ PEEK (M+9);: RETURN
75 PRINT AT B+PEEK (M+1),C+PEEK (M+2);CHR$ PEEK (M+3);: RETURN : REM *** Incrustar
ficha en tablero Player1 ***
```

Si no cabe la ficha, Incrustarla en el tablero Player1 (sumando a lo que ya haya en ese carácter)

```
82 LET M=E+PEEK (E+7): FOR M=M+1 TO M+PEEK M STEP 2: LET E=J+PEEK (M+1): LET
A=PEEK M: POKE E,A+PEEK E: POKE E+256,PEEK (E+256)+PEEK (61408+A): POKE
E+263,PEEK (E+263)+PEEK (61664+A): NEXT M: LET E=B*2+PEEK M: LET N=M-50430-E: IF
PEEK 60554>E THEN POKE 60554,E: LET V=B-2
```

Si fila completada, saltar a mover bloques

```
83 LET B=0: FOR M=E+50432 TO E+PEEK (M+1)+50431: POKE M,PEEK M+PEEK (N+M): IF
10=PEEK M THEN LET B=1+B: GO SUB 101
```

Repintado necesario del tablero, y comprobar si hay que subir de nivel

84 NEXT M: IF B THEN POKE 23564,0: LET E=INT (.5*PEEK 60554): FOR M=50749+7*E TO 3.5*M-125764 STEP 7: PRINT AT E,1;"\{o0}";CHR$ PEEK (61920+PEEK M);CHR$ PEEK (61920+PEEK (M+1));CHR$ PEEK (61920+PEEK (M+2));CHR$ PEEK (61920+PEEK (M+3));CHR$ PEEK (61920+PEEK (M+4));"\#008\#008\#008\#008\#008\{o1}";CHR$ PEEK (62176+PEEK M);CHR$ PEEK (62176+PEEK (M+1));CHR$ PEEK (62176+PEEK (M+2));CHR$ PEEK (62176+PEEK (M+3));CHR$ PEEK (62176+PEEK (M+4));: LET E=E+1: NEXT M: POKE 60554,B+PEEK 60554: POKE 60556,B+PEEK 60556: PRINT OVER 0;AT 16,3;PEEK 60556;: LET W=W-1: IF NOT W THEN LET W=PEEK 62808: GO SUB 450+PEEK 51000

Comprobar llegamos a la fila superior para fin de partida

86 IF PEEK 50435 THEN POKE 60551,1: GO TO 701: REM fin P1!

Sacar nueva ficha del listado aleatorio

87 LET M=PEEK 60548: LET J=O(M,1,1): LET U(1)=J: LET U(2)=O(M,1,2): LET U(3)=O(M,1,3): LET U(4)=O(M,1,4): POKE 60546,M: LET M=PEEK (60402+PEEK 60552): POKE 60548,M: POKE 60552,1+PEEK 60552 AND PEEK 60552<141: LET E=O(M,1,1): LET E=E+PEEK (E+7)+1: LET A=1: LET B=1: LET C=3: IF M=6 THEN LET M=J+PEEK (J+7)+1: PRINT AT 3,9; OVER 0;CHR$ PEEK (61920+PEEK E);" \#008\#008";: PRINT CHR$ PEEK (62176+PEEK E);: GO TO 91

89 LET M=J+PEEK (J+7)+1: PRINT AT 3,9; OVER 0;CHR$ PEEK (61920+PEEK E);CHR$ PEEK (61920+PEEK (E+2));"\#008\#008";: PRINT CHR$ PEEK (62176+PEEK E);CHR$ PEEK (62176+PEEK (E+2));

91 IF 6<>PEEK 60546 THEN PRINT AT 1,3; OVER 0;CHR$ PEEK (61920+PEEK M);CHR$ PEEK (61920+PEEK (M+2));"\#008\#008";: PRINT CHR$ PEEK (62176+PEEK M);CHR$ PEEK (62176+PEEK (M+2)): GO TO 5

92 PRINT AT 1,3;CHR$ PEEK (61920+PEEK M);" \#008\#008";: PRINT CHR$ PEEK (62176+PEEK M);: GO TO 5

Mover bloques al eliminar fila (usamos tablero ppel, secundario y otro pasa swap)

101 POKE 23564,198: LET A=M-50432: LET J=7+7*INT (A*.5): POKE 50692,57: LET b$=f$: POKE 50703,J: LET f$=s$: POKE 50692,50: POKE 50712,J: LET s$=b$: POKE 50730,A: LET d$=c$: LET E=50741+J: LET J=50997+J: IF PEEK (61100+A) THEN POKE J+8,PEEK (61664+PEEK (E+1)) +PEEK (61408+PEEK (E+8)): POKE J+9,PEEK (61664+PEEK (E+2))+PEEK (61408+PEEK (E+9)): POKE J+10,PEEK (61664+PEEK (E+3))+PEEK (61408+PEEK (E+10)): POKE J+11,PEEK (61664+PEEK (E+4))+PEEK (61408+PEEK (E+11)): POKE J+12,PEEK (61664+PEEK (E+5))+PEEK (61408+PEEK (E+12)): RETURN

102 POKE E+1,PEEK (61664+PEEK (J+1))+PEEK (61408+PEEK (J+8)): POKE E+2,PEEK (61664+PEEK (J+2))+PEEK (61408+PEEK (J+9)): POKE E+3,PEEK (61664+PEEK (J+3))+PEEK (61408+PEEK (J+10)): POKE E+4,PEEK (61664+PEEK (J+4))+PEEK (61408+PEEK (J+11)): POKE E+5,PEEK (61664+PEEK (J+5))+PEEK (61408+PEEK (J+12)): RETURN : REM Borrar fila completa y mover bloques

Prints de fichas menos probables de salir

156 PRINT AT B+PEEK (M+1),C+PEEK (M+2);CHR$ PEEK (M+3);"\#008";CHR$ PEEK (M+4);AT B+PEEK (M+5),C+PEEK (M+6);CHR$ PEEK (M+7);"\#008";CHR$ PEEK (M+8);: RETURN

161 PRINT AT B+PEEK (M+1),C+PEEK (M+2);CHR$ PEEK (M+3);CHR$ PEEK (M+5);CHR$ PEEK (M+7);"\#008\#008\#008";CHR$ PEEK (M+4);CHR$ PEEK (M+6);CHR$ PEEK (M+8);: RETURN

166 PRINT AT B+PEEK (M+1),C+PEEK (M+2);CHR$ PEEK (M+3);"\#008";CHR$ PEEK (M+4);AT B+PEEK (M+5),C+PEEK (M+6);CHR$ PEEK (M+7);"\#008";CHR$ PEEK (M+8);AT B+PEEK (M+9),C+PEEK (M+10);CHR$ PEEK (M+11);"\#008";CHR$ PEEK (M+12);: RETURN

169 PRINT AT B+PEEK (M+1),C+PEEK (M+2);CHR$ PEEK (M+3);"\#008";CHR$ PEEK (M+4);AT B+PEEK (M+5),C+PEEK (M+6);CHR$ PEEK (M+7);CHR$ PEEK (M+9);"\#008\#008";CHR$ PEEK (M+8);CHR$ PEEK (M+10);AT B+PEEK (M+11),C+PEEK (M+12);CHR$ PEEK (M+13);"\#008";CHR$ PEEK (M+14);: RETURN

Repetir todo (excepto los Prints) para Engine jugador 2

200 POKE 23627,PEEK 51204: IF IN 65278=189 THEN GO TO 251: REM **** Repetimos ENGINE para Player 2 ****

210 IF IN 65022=190 THEN LET E=U(A): LET M=E+PEEK (E+2): LET J=50840+7*B+C: GO SUB PEEK M: IF M THEN LET B=B-1: LET C=C+15: LET M=E+PEEK (E+6): GO SUB PEEK M: LET B=B+PEEK (E+18): LET C=C+PEEK (E+19)-16: LET M=PEEK (E+17): LET E=PEEK 60547: LET

U(1)=O(E,M,1): LET U(2)=O(E,M,2): LET U(3)=O(E,M,3): LET U(4)=O(E,M,4): LET O=M
212 IF IN 64510=187 THEN LET M=U(A)+20: LET J=50847+7*B+C: GO SUB PEEK M: IF M THEN LET E=U(A): LET C=C+15: LET M=E+PEEK (E+3): GO SUB PEEK M: LET C=C+PEEK (E+10)-16: LET A=PEEK (61143+A): POKE 23627,PEEK 51213: GO TO 1: REM izq
213 IF IN 64510=183 THEN LET E=U(A): LET M=E+PEEK E: LET J=50848+7*B+C: GO SUB PEEK M: IF M THEN LET M=E+PEEK (E+4): LET c=c+16: GO SUB PEEK M: LET C=C+PEEK (E+13)-16: LET A=PEEK (61143+A): POKE 23627,PEEK 51213: GO TO 1: REM drc
220 POKE 23627,PEEK 51213: GO TO 1: REM baja obligado por FR, GT230 y GT3
231 POKE 23627,PEEK 51204: LET E=U(A): LET M=E+PEEK (E+1): LET J=50848+7*B+C: GO SUB PEEK M AND V<=B: IF M THEN LET M=E+PEEK (E+5): LET c=c+16: GO SUB PEEK M: LET c=c-16: LET B=B+PEEK (E+15): LET A=PEEK (61140+A): POKE 23627,PEEK 51213: GO TO 3: REM * Hay suelo, Incrustar ficha en el tablero *
232 POKE 60558,3: GO TO 282: REM salta a incrustar, pero retorna a Bajar P1
251 POKE 60558,1: LET E=U(A): LET M=E+PEEK (E+1): LET J=50848+7*B+C: GO SUB PEEK M AND V<=B: IF M THEN LET M=E+PEEK (E+5): LET c=c+16: GO SUB PEEK M: LET c=c-16: LET B=B+PEEK (E+15): LET A=PEEK (61140+A): GO TO 210
282 LET M=E+PEEK (E+7): FOR M=M+1 TO M+PEEK M STEP 2: LET E=J+PEEK (M+1): LET A=PEEK M: POKE E,A+PEEK E: POKE E+256,PEEK (E+256)+PEEK (61408+A): POKE E+263,PEEK (E+263)+PEEK (61664+A): NEXT M: LET E=B*2+PEEK M: LET N=M-50430-E-25: IF PEEK 60555>E THEN POKE 60555,E: LET V=B-2: REM hay suelo, Incrustar!
283 LET B=0: FOR M=E+50432+25 TO E+PEEK (M+1)+50431+25: POKE M,PEEK M+PEEK (N+M): IF 10=PEEK M THEN LET B=1+B: GO SUB 301
284 NEXT M: IF B THEN POKE 23564,0: LET E=INT (.5*PEEK 60555): FOR M=50749+7*E+100 TO 3.5*M-125764+13 STEP 7: PRINT AT E,17;"\{o0}";CHR$ PEEK (61920+PEEK M);CHR$ PEEK (61920+PEEK (M+1));CHR$ PEEK (61920+PEEK (M+2));CHR$ PEEK (61920+PEEK (M+3));CHR$ PEEK (61920+PEEK (M+4));"\#008\#008\#008\#008\#008\{o1}";CHR$ PEEK (62176+PEEK M);CHR$ PEEK (62176+PEEK (M+1));CHR$ PEEK (62176+PEEK (M+2));CHR$ PEEK (62176+PEEK (M+3));CHR$ PEEK (62176+PEEK (M+4));: LET E=E+1: NEXT M: POKE 60555,B+PEEK 60555: POKE 60557,B+PEEK 60557: PRINT OVER 0;AT 16,19;PEEK 60557;: LET W=W-1: IF NOT W THEN LET W=PEEK 62808: GO SUB 450+PEEK 51000
286 IF PEEK 50460 THEN GO TO 701: REM fin P2!
287 LET M=PEEK 60549: LET J=O(M,1,1): LET U(1)=J: LET U(2)=O(M,1,2): LET U(3)=O(M,1,3): LET U(4)=O(M,1,4): POKE 60547,M: LET M=PEEK (60402+PEEK 60553): POKE 60549,M: POKE 60553,1+PEEK 60553 AND PEEK 60553<141: LET E=O(M,1,1): LET E=E+PEEK (E+7)+1: LET A=1: LET B=1: LET C=3: IF M=6 THEN LET M=J+PEEK (J+7)+1: PRINT AT 3,12; OVER 0;CHR$ PEEK (61920+PEEK E);" \#008\#008";: PRINT CHR$ PEEK (62176+PEEK E);: GO TO 291
288 LET M=J+PEEK (J+7)+1: PRINT AT 3,12; OVER 0;CHR$ PEEK (61920+PEEK E);CHR$ PEEK (61920+PEEK (E+2));"\#008\#008";: PRINT CHR$ PEEK (62176+PEEK E);CHR$ PEEK (62176+PEEK (E+2));
291 IF 6<>PEEK 60547 THEN PRINT AT 1,19; OVER 0;CHR$ PEEK (61920+PEEK M);CHR$ PEEK (61920+PEEK (M+2));"\#008\#008";: PRINT CHR$ PEEK (62176+PEEK M);CHR$ PEEK (62176+PEEK (M+2)): POKE 23627,PEEK 51213: GO TO PEEK 60558
292 PRINT AT 1,19; OVER 0;CHR$ PEEK (61920+PEEK M);" \#008\#008";: PRINT CHR$ PEEK (62176+PEEK M);: POKE 23627,PEEK 51213: GO TO PEEK 60558: REM * Borrar fila completa y mover bloques *

301 POKE 23564,199: LET A=M-50432-25: LET J=7+7*INT (A*.5): POKE 50692+256,57: LET b$=f$: POKE 50703+256,J: LET f$=s$: POKE 50692+256,50: POKE 50712+256,J: LET s$=b$: POKE 50730+256,A: LET d$=c$: LET E=50741+100+J: LET J=50997+100+J: IF PEEK (61100+A) THEN POKE J+8,PEEK (61664+PEEK (E+1))+PEEK (61408+PEEK (E+8)): POKE J+9,PEEK (61664+PEEK (E+2))+PEEK (61408+PEEK (E+9)): POKE J+10,PEEK (61664+PEEK (E+3))+PEEK (61408+PEEK (E+10)): POKE J+11,PEEK (61664+PEEK (E+4))+PEEK (61408+PEEK (E+11)): POKE J+12,PEEK (61664+PEEK (E+5))+PEEK (61408+PEEK (E+12)): RETURN : REM Borrar fila completa y mover bloques
302 POKE E+1,PEEK (61664+PEEK (J+1))+PEEK (61408+PEEK (J+8)): POKE E+2,PEEK (61664+PEEK (J+2))+PEEK (61408+PEEK (J+9)): POKE E+3,PEEK (61664+PEEK (J+3))+PEEK (61408+PEEK (J+10)): POKE E+4,PEEK (61664+PEEK (J+4))+PEEK (61408+PEEK (J+11)): POKE E+5,PEEK (61664+PEEK (J+5))+PEEK (61408+PEEK (J+12)): RETURN : REM *** Aumentar velocidad cada W lineas ***

Aumentar velocidad cada W lineas
459 POKE 51000,50: POKE l24,234: POKE l25,234: POKE l27,234: POKE l28,234: POKE l29,l29U: LET W=255: PRINT OVER 0;AT 7,10;"max";: RETURN
460 POKE 51000,50: POKE l24,234: POKE l25,234: POKE l27,234: POKE l28,234: POKE l29,l29U: LET i=62560: LET W=255: PRINT OVER 0;AT 7,10;"max";: RETURN
472 POKE 51000,PEEK 51000-2: PRINT OVER 0;AT 7,11;PEEK (62761+PEEK 51000);: LET W=W+1: RETURN
474 POKE 51000,21: POKE l24,234: POKE l27,234: POKE l28,l28U: LET W=W+1: PRINT OVER 0;AT 7,11;PEEK (62761+PEEK 51000);: RETURN
499 POKE 51000,PEEK 51000-3: PRINT OVER 0;AT 7,11;PEEK (62761+PEEK 51000);: RETURN
500 RETURN : REM * Toca una nota de laMusica cada 20 frames *

En el menu principal, toca una nota de laMusica cada 20 frames
590 POKE 23635,PEEK 23637: POKE 23636,PEEK 23638: RETURN : REM Set PROG a linea 600 para que el menu cargue mas rapido
595 IF PEEK 23672-J>20 THEN LET i=62560+PEEK (i+110): LET J=PEEK 23672: IF PEEK i THEN BEEP .05,PEEK i: REM BEEP cada 20fr
596 IF PEEK 23673 THEN LET J=J-256: POKE 23673,0
597 RETURN

GameOver, reinicializar variables y volver al menu de inicio.
600 REM GAME OVER. RE-INIT. PLAY AGAIN
601 POKE 23564,200: LET x$=y$: POKE 23564,0:: LET W=1: LET Y=0: LET lvl=1: POKE 60546,1+RND*6: POKE 60547,PEEK 60546: POKE 60553,PEEK 60552: POKE 60554,20: POKE 60555,20: POKE 60556,0: POKE 60557,0: POKE 60558,1: POKE 60559,0:: POKE 62670,1: POKE l24,234: POKE l25,l25U: POKE l27,l27U: POKE l28,234: POKE l29,234:: GO SUB 7002: GO SUB 820: POKE 51000,l(lvl)-mu: GO SUB 450+PEEK 51000: POKE 51001,0: POKE 60550,3: POKE 60551,1: LET T=0: LET W=2: IF Z THEN LET W=4: POKE VAL "23564",200: LET w$=v$: POKE VAL "23564",NOT PI: LET Z=0: GO SUB 820: LET Z=1: POKE 60550,231: POKE 60551,200
605 POKE 62808,W: POKE VAL "60559",NOT PI: POKE VAL "23672",NOT PI: POKE VAL "23673",NOT PI: POKE VAL "23674",NOT PI: POKE VAL "23672",NOT PI: BEEP .5,12: INK 7: OVER 1: GO TO 5

700 REM * Imprimir GAME OVER *
701 GO SUB 590
702 BRIGHT 0: FOR m=0 TO 9: LET e=PEEK (62659+m): PRINT OVER 1; PAPER 0;AT m,0;"\{i1} \{i5} \{i1} ";AT m,16;"\{i2} \{i3} \{i2} ";: BEEP 0.1 AND e,e: NEXT m: PRINT OVER 1; PAPER 0;AT 10,0; INK 1;" ";AT 10,16; INK 2;" ";
703 OVER 0: FOR m=7 TO 0 STEP -1: POKE 23606,m: PRINT AT 11,7; INK 6;"game over";: PAUSE 5: NEXT m: FOR m=7 TO 0 STEP -1: POKE 23606,m: PRINT AT 12,7; INK 2 OR PEEK 60551=1;" player";2 OR PEEK 60551=1;: PAUSE 5: NEXT m
704 FOR m=7 TO 0 STEP -1: POKE 23606,m: IF PEEK 60556>PEEK 62558 THEN PRINT AT 14,10; INK 6;"new";AT 16,11; INK 7;PEEK 60556;
705 IF PEEK 60557>PEEK 62558 THEN PRINT AT 14,10; INK 6;"new";AT 16,11; INK 7;PEEK 60557;
706 PAUSE 5: NEXT m
707 FOR m=1 TO 12: PRINT OVER 1;AT 11,6+m; INK 7;" ";AT 11,5+m; BRIGHT 1;" ";AT 11,4+m; INK 6;" ";: PRINT OVER 1;AT 12,7+m; INK 7-2*(2 OR PEEK 60551=1);" ";AT 12,6+m; BRIGHT 1;" ";AT 12,5+m; INK 2 OR PEEK 60551=1;" ";: IF PEEK 60556>PEEK 62558 OR PEEK 60557>PEEK 62558 THEN PRINT OVER 1;AT 14,7+m; INK 7;" ";AT 14,6+m; BRIGHT 1;" ";AT 14,5+m; INK 6;" ";
708 LET m=(m AND m<12)+(25 AND INKEY$<>""): NEXT m: IF PEEK 60556>PEEK 62558 THEN POKE 62558,PEEK 60556
709 IF PEEK 60557>PEEK 62558 THEN POKE 62558,PEEK 60557
710 LET i=62560: POKE 23635,203: POKE 23636,92: POKE 23627,PEEK 51204: GO TO 600

820 REM * FICHA NUEVA *
821 POKE 60548+Z,PEEK (60402+PEEK 60552): POKE 60400+Z,1: LET M=O(PEEK

(60548+Z),1,1)+PEEK (O(PEEK (60548+Z),1,1)+7)+1: PRINT OVER 0;AT 3,9+(3 AND Z);CHR$ PEEK (61920+PEEK M);" \#008\#008"; OVER 1;CHR$ PEEK (62176+PEEK M);CHR$ PEEK (61920+PEEK (M+2)) AND 4=PEEK (M-1);"\#008";CHR$ PEEK (62176+PEEK (M+2)) AND 4=PEEK (M-1);: POKE 60546+Z,PEEK (60546+Z)-(2 AND PEEK (60546+Z)>5): LET U(1)=O(PEEK (60546+Z),1,1): LET U(2)=O(PEEK (60546+Z),1,2): LET U(3)=O(PEEK (60546+Z),1,3): LET U(4)=O(PEEK (60546+Z),1,4): LET M=U(1)+PEEK (U(1)+7)+1: PRINT OVER 0;AT 1,3+(16 AND Z);CHR$ PEEK (61920+PEEK M);" \#008\#008"; OVER 1;CHR$ PEEK (62176+PEEK M);CHR$ PEEK (61920+PEEK (M+2)) AND 4=PEEK (M-1);"\#008";CHR$ PEEK (62176+PEEK (M+2)) AND 4=PEEK (M-1);: LET A=1: LET B=1: LET C=3: RETURN : REM ficha nueva Player 1 o 2

Cambiar texturas y mostrar fichas con las nuevas tramas en el menu

900 POKE 51250,245+tex: POKE 23564,200: LET a$=b$: POKE 23564,0: BRIGHT 1: PRINT AT 4,7;
901 FOR N=62809 TO 62815: LET M=PEEK N: GO SUB 595: IF M<>6 THEN INK M: LET M=O(M,1,1)+PEEK (O(M,1,1)+7)+1: PRINT OVER 0;CHR$ PEEK (61920+PEEK M);CHR$ PEEK (61920+PEEK (M+2));"\#008\#008"; OVER 1;CHR$ PEEK (62176+PEEK M);CHR$ PEEK (62176+PEEK (M+2));" ";: NEXT N: RETURN : REM cambiar texturas
902 INK 6: LET M=O(6,1,1)+PEEK (O(6,1,1)+7)+1: PRINT OVER 0;CHR$ PEEK (61920+PEEK M);"\#008"; OVER 1;CHR$ PEEK (62176+PEEK M);" ";: NEXT N: RETURN

6989 REM * PANTALLA INICIO *
6990 POKE 60559,NOT PI: LET i=62560: POKE 23607,232: PAPER NOT PI: INK 6: CLS : OVER NOT PI: PRINT AT 12,6;" t e t r i s \{i5}4 px ";AT 19,18;"\{i3}pure basic by";: FOR m=56 TO 0 STEP -1: GO SUB 595: POKE 23606,m: PRINT AT 20,25;"c";: IF m>7 THEN POKE 23606,m-8: PRINT AT 20,22;"g";AT 20,28;"o";: IF m>15 THEN POKE 23606,m-16: PRINT AT 20,26;"o";: IF m>23 THEN POKE 23606,m-24: PRINT AT 20,23;"n";: IF m>31 THEN POKE 23606,m-32: PRINT AT 20,21;"i";AT 20,27;"b";: IF m>39 THEN POKE 23606,m-40: PRINT AT 20,24;"a";
6996 NEXT m: OVER 1: FOR m=0 TO 9: PRINT AT 20,21+m; INK 7;" ";AT 20,20+m; BRIGHT 1;" ";AT 20,19+m; INK 6;" ";: GO SUB 595: PAUSE 5: NEXT m:: POKE 51250,245+tex: POKE 23564,200: LET x$=y$: LET a$=b$: POKE 23564,0:: LET m=0: LET e=0: LET b=0
7000 GO SUB 595: PRINT AT 20,21+m; INK 7;" ";AT 20,20+m; BRIGHT 1;" ";AT 20,19+m; INK 6;" ";: LET m=m+1 AND m<9: LET e=e+1 OR e=19: PRINT AT 20,21+b;" ";: PRINT AT 12,5+e; INK 8;" t e t r i s 4 px"(e);: GO SUB 595: IF INKEY$="" THEN PRINT AT 20,20+b;" ";: PRINT AT 12,5+e; INK 8;" t e t r i s 4 px"(e);: LET b=b+1 AND b<9: GO TO 7000
7001 OVER 0: RETURN : REM * MENU DE INICIO *

Menu de inicio

7002 RANDOMIZE 256*PEEK 23673+PEEK 23672: OVER 0: INK 6: CLS : LET mu=1: LET grid=0: LET tex=1: LET j=0: LET Z=0: FOR m=7 TO 0 STEP -1: GO SUB 595: POKE 60400+m,1+RND*6: POKE 60448+m,1+RND*6: POKE 60496+m,1+RND*6: POKE 23606,m: PRINT AT 8,7;"\{i5vi}1\{vni6b0} player opjm";: NEXT m
7003 FOR m=7 TO 0 STEP -1: GO SUB 595: POKE 60408+m,1+RND*6: POKE 60456+m,1+RND*6: POKE 60504+m,1+RND*6: POKE 23606,m: PRINT AT 10,7;"\{i1}2\{b0i6} players opjm eraz";: NEXT m
7004 FOR m=7 TO 0 STEP -1: GO SUB 595: POKE 60416+m,1+RND*6: POKE 60464+m,1+RND*6: POKE 60512+m,1+RND*6: POKE 23606,m: PRINT AT 12,7;"\{vi}m\{vnb0}usic ingame \{b1i5}on";: NEXT m
7005 FOR m=7 TO 0 STEP -1: GO SUB 595: POKE 60424+m,1+RND*6: POKE 60472+m,1+RND*6: POKE 60520+m,1+RND*6: POKE 23606,m: PRINT AT 14,7;"\{vi}g\{vnb0}rid \{b1i1}off";: NEXT m
7006 FOR m=7 TO 0 STEP -1: GO SUB 595: POKE 60432+m,1+RND*6: POKE 60480+m,1+RND*6: POKE 60528+m,1+RND*6: POKE 23606,m: PRINT AT 16,7;"\{vi}t\{vnb0} exture \{i5}a";: NEXT m
7007 FOR m=7 TO 0 STEP -1: GO SUB 595: POKE 60440+m,1+RND*6: POKE 60488+m,1+RND*6: POKE 60536+m,1+RND*6: POKE 23606,m: PRINT AT 18,7;"\{vi}s\{vnb0}peed \{i1}min";: NEXT m
7008 FOR m=7 TO 0 STEP -1: GO SUB 595: POKE 23606,m: PRINT AT 20,7;"\{vi}p\{vnb0}lay game";: NEXT m: GO SUB 900: POKE 60400,1: POKE 60401,1
7100 IF PEEK 23672-T>20 THEN LET i=62560+PEEK (i+110): LET T=PEEK 23672: IF PEEK i

```
THEN BEEP .05,PEEK i: REM BEEP cada 25
7101 IF PEEK 23673 THEN LET T=T-256: POKE 23673,0
7104 LET n$=INKEY$: IF n$="g" THEN LET grid=NOT grid: BEEP 0.02,24: FOR n=7 TO 0 STEP
-1: GO SUB 595: POKE 23606,n: PRINT AT 14,12;"\{i1}off" AND NOT grid;"\{i5}on " AND grid;:
NEXT n
7105 IF n$="m" THEN LET mu=NOT mu: BEEP 0.02,24: FOR n=7 TO 0 STEP -1: GO SUB 595:
POKE 23606,n: PRINT AT 12,20;"\{i1}off" AND NOT mu;"\{i5}on " AND mu;: NEXT n: IF mu AND Z
THEN LET n$="1": GO TO 7106
7106 IF n$="1" THEN LET Z=0: FOR n=7 TO 0 STEP -1: GO SUB 595: POKE 23606,n: PRINT AT
8,7;"\{i5vi}1\{vn}";AT 10,7;"\{i1}2";: NEXT n
7107 IF n$="2" THEN LET Z=1: FOR n=7 TO 0 STEP -1: GO SUB 595: POKE 23606,n: PRINT AT
8,7;"\{i1}1";AT 10,7;"\{i5vi}2";: NEXT n: IF mu THEN LET n$="m": GO TO 7105
7108 IF n$="p" THEN GO TO 7115
7109 IF n$="t" THEN LET tex=(tex+1) OR tex=3: FOR n=7 TO 0 STEP -1: GO SUB 595: POKE
23606,n: PRINT AT 16,15;"\{i5}";"abc"(tex);: NEXT n: GO SUB 900
7110 IF n$="s" THEN LET lvl=(lvl+1) OR lvl=3: BEEP 0.02,24: FOR n=7 TO 0 STEP -1: GO SUB
595: POKE 23606,n: PRINT AT 18,13;l$(lvl);: NEXT n
7112 GO TO 7100
7115 BRIGHT 8: PAPER 8: INK 6: CLS : LET n$=" \{b1} \{b0} \{b1} \{b0} ": LET g$="\{b1} \{b0} \
{b1} \{b0} \{b1} \{b0}": LET e=grid: FOR m=0 TO 63 STEP 7: GO SUB 595: LET e=-1*e: PRINT AT
m/7+1,0;"\{p0b0}<"; PAPER 1;n$ AND e=1;g$ AND e=-1;"      " AND NOT e;"\{p0}=";: IF Z THEN
PRINT AT m/7+1,16;"\{p0b0}<"; PAPER 2;n$ AND e=1;g$ AND e=-1;"      " AND NOT e;"\{p0}=";
7116 NEXT m: POKE 62670,mu: IF mu THEN POKE l24,l24U
7117 PRINT BRIGHT 0; PAPER 0;AT 1,9;"next:";AT 10,0;" ;;;;; ";AT 10,16;" ;;;;; " AND Z;AT
15,1;"lines  hiscore";AT 15,17;"lines" AND Z;AT 6,9;"level"; INK 7;AT 16,3;"0       ";PEEK 62558;AT
16,19;"0" AND Z;AT 7,11;"1";: INK 7: OVER 1: BRIGHT 8 AND grid: POKE 23635,203: POKE
23636,92: RETURN
7120 RETURN : REM * CALCULAR ESPACIO PARA VARS PLAYER 2 *
8900 LET b=VAL "256"*PEEK VAL "23628"+PEEK VAL "23627"-VAL "100": POKE VAL
"51204",PEEK VAL "23627": POKE VAL "51205",PEEK VAL "23628": POKE VAL
"51213",b-256*INT (b/256): POKE VAL "51214",INT (b/256): POKE VAL "51206",100: POKE VAL
"51215",100: IF PEEK VAL "51205"<>PEEK VAL "51214" THEN PRINT AT 11,0;"vl:";PEEK VAL
"51213";"   ovl:";PEEK VAL "51204": POKE 23635,203: POKE 23636,92: STOP
8901 RETURN

8999 REM * INICIO PROG. DECLARA VARS *
9000 PAPER NOT PI: INK 7: CLEAR 197*256-1: FOR m=1 TO 0: NEXT m: LET E=0: LET J=0: LET
A=1: LET B=1: LET C=1: FOR N=SGN PI TO NOT PI: NEXT N: DIM U(4): LET V=7: LET i=VAL
"62560": LET T=0: LET Y=0: LET n$="w lineas para subir de level": LET W=2: LET Z=201: DIM
O(7,4,4): POKE 60552,0: POKE 60553,0: GO SUB 9200: LET vL=0: LET grid=0: LET tex=1: LET
l24=24687: LET l25=24780: LET l27=24849: LET l28=24912: LET l29=25023: LET l24U=250: LET
l25U=250: LET l27U=250: LET l28U=250: LET l29U=241: DIM l$(3,5): LET l$(1)="\{i1}min": LET
l$(2)="\{i5}med": LET l$(3)="\{i7}max": DIM l(3): LET l(1)=49: LET l(2)=25: LET l(3)=10: POKE VAL
"23672",0: POKE VAL "23673",0: POKE VAL "23674",0: GO SUB 6990: GO SUB 8900: GO SUB
590: GO TO 600

9204 REM * CARGA FICHEROS DE DATOS *
9207 BORDER NOT PI: LOAD "tetris4px.bin" CODE VAL "16384",VAL "6912": PRINT AT NOT
PI,29; OVER NOT PI;"\{i1}v2 ";AT 1,28;"2023": FOR m=NOT PI TO 9: PRINT #0;AT 1,21+m; OVER
1; INK 7;" ";AT 1,20+m; BRIGHT 1;" ";AT 1,19+m; INK 6;" ";: LET m=(m AND m<9)+(25 AND
INKEY$<>""): NEXT m: POKE VAL "23739",VAL "111": LOAD "udgNums2" CODE VAL
"59776",VAL "120": LOAD "udgAZ" CODE VAL "60168",VAL "208": LOAD "defadd1" CODE VAL
"50688",VAL "54": LOAD "defadd2" CODE VAL "50944",VAL "54": LOAD "defadd3a" CODE VAL
"51200",VAL "54": LOAD "Fichas" CODE VAL "51600",VAL "6426": LOAD "TablasC" CODE VAL
"60896",VAL "1536": LOAD "music" CODE VAL "62560",VAL "256": LOAD "textures" CODE VAL
"63000",VAL "768": LOAD "initData" CODE VAL "63768",VAL "768": LOAD "DimO" DATA O():
POKE VAL "23739",VAL "244": POKE VAL "23607",VAL "232": CLS : RETURN
```

Después de un BREAK, escribir GOTO 9500 para ver el listado basic

```
9500 POKE VAL "23607",VAL "60": POKE VAL "23606",NOT PI: POKE VAL "23635",VAL "203":
POKE VAL "23636",VAL "92": POKE I24,I24U: POKE I25,I25U: POKE I27,I27U: POKE I28,I28U:
POKE I29,I29U: STOP : REM Chrset,IniProg,UnRem24,25,27,28,29
```

Bytes reservados para copiar las VARS del jugador 2 sobre el REM 9993

```
9990 REM * VARS PLAYER 2 *
9992 REM
12345678901234567890123456789012345678901234567890123456789012345678901234567890123456789012345678901234567890123456789012345678901234567890123456789012345678901234567890
9993 REM
12345678901234567890123456789012345678901234567890123456789012345678901234567890123456789012345678901234567890123456789012345678901234567890
```