

# JUEGOS GANADORES: 2º EXPLOSS

Autor: Ismael Fernández

ismafb@hotmail.com

# Un programa explosivo

**En este documento podremos encontrar comentada la forma en la que se ha hecho este juego, los trucos utilizados para ciertas cosas y cada proceso, detallado paso a paso, para que uno se dé cuenta de todo y logre comprender perfectamente todo el código fuente.**

**La recomendación que hacemos desde aquí es tener delante no sólo este documento sino también el código fuente impreso para apreciar todos los detalles.**

Para explicar todo el código fuente comenzaremos explicar, primero, cada proceso por separado; una vez hecho se pasa a comentar el programa principal. Está claro que en el código fuente primero aparece el programa principal y luego los procesos, pero consideramos que será más fácil entenderlo de esta manera.

Para cada proceso se escribe su interfaz, es decir, algo del estilo de:

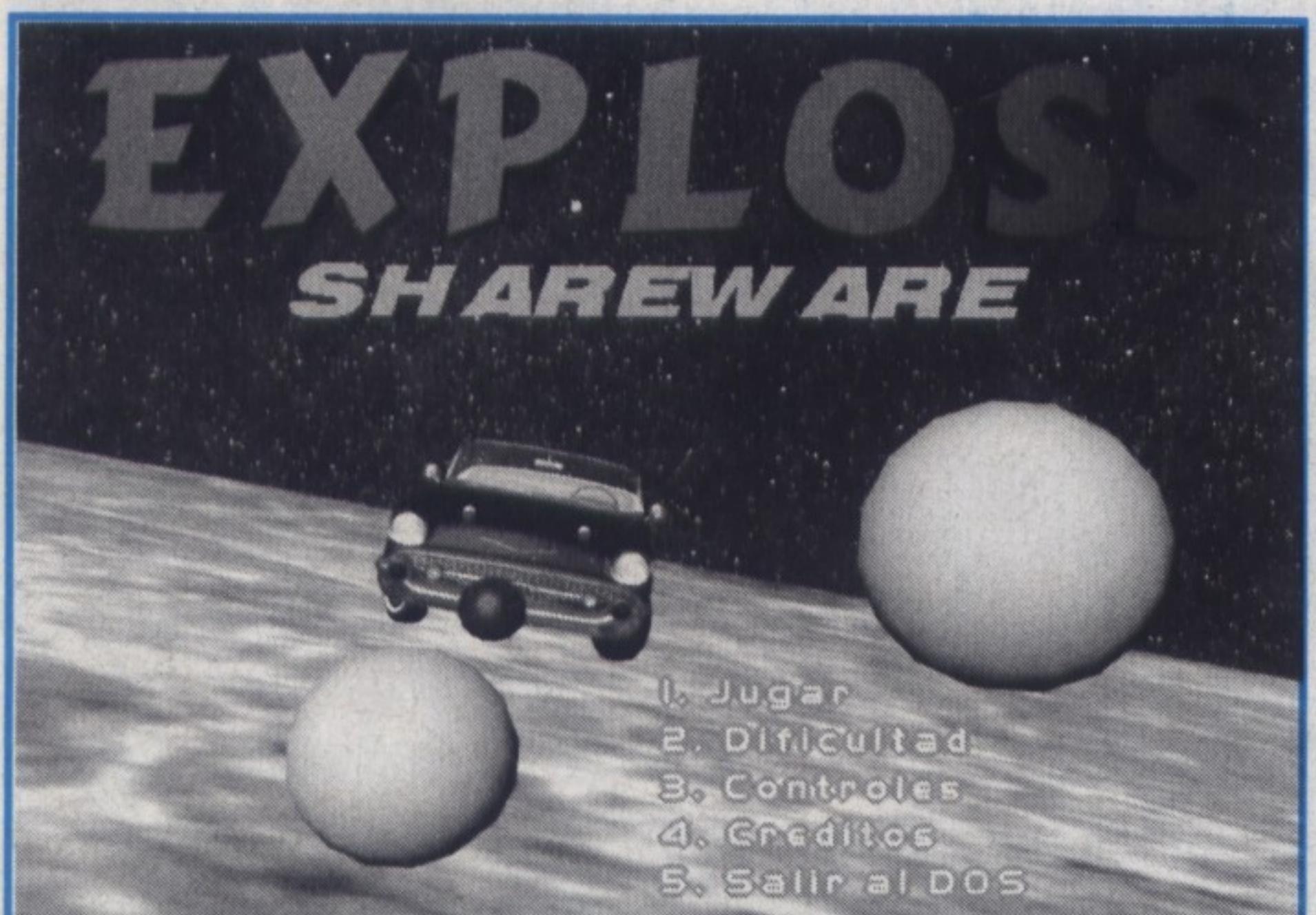
PROCESS Nombre\_Proceso (X, Y)

## PROCESS COGER\_BOMBAS (X, Y)

Este proceso es utilizado para visualizar en la pantalla el gráfico que si es cogido por el coche le proporciona cuatro bombas más; para ello, como parámetros recibe las coordenadas X e Y en donde el gráfico será visualizado, y como veremos en el programa principal, serán dos números aleatorios.

Asimismo, contiene una variable de tipo PRIVATE, es decir, cada vez que se llama a este proceso se genera para cada uno de ellos una variable que se llama CON\_1. Lo primero que hace es poner el flag EXISTE a 1 para que ya no se generen nuevos, o sea, no puedan

FIGURA 1.



aparecer dos a la vez en pantalla. Elegimos el gráfico adecuado (en este caso el 91), y ponemos la coordenada Z a -1 para que se visualice por encima de todos los procesos. Ahora, nos metemos en un bucle que irá de 0 a 100 a través de la variable CON\_1 que servirá como el límite máximo de tiempo que estará en pantalla este gráfico; dentro de este bucle lo que hacemos es comprobar si el coche está chocando con él. Si es así, incrementamos las bombas que teníamos en 4, comprobamos si nos hemos pasado del número máximo que puede tener el coche y con la sentencia BREAK salimos de bucle; justo antes de que acabe ponemos el flag EXISTE a 0 indicando que ya no hay nada en pantalla que proporcione cuatro bombas y así se pueda generar uno nuevo.

## PROCESS VIDAS( )

Este se utiliza para visualizar el gráfico de coche como símbolo del número de vidas que se tienen, es decir, lo único que hace es poner el gráfico del coche en una posición a la izquierda y abajo para representar el número de vidas que nos quedan.

Lo primero que se hace es seleccionar el gráfico idóneo (30) y colocarlo en unas coordenadas X, Y y Z adecuadas (20, 410, -1); posteriormente hay que introducirse en un bucle de tipo LOOP ... END para que no salga nunca de él.

## PROCESS BOMBAS( )

Este proceso se usa para visualizar el gráfico de la bomba como símbolo del número de bombas que se tienen, es decir, es similar al anterior ya que lo único que hace es poner el gráfico de una bomba en la posición a la izquierda y abajo para representar al número de bombas que nos quedan.

## PROCESS COCHE( )

Se utiliza para manejar el coche en función de las teclas que estén pulsadas o, en el caso de que se eligiera joystick, en función de lo que se apriete con él. En este proceso declaramos dos variables INC\_X e INC\_Y, inicializadas al valor 8 bien no van a ser utilizadas; se pueden poner como referencias para la distancia que se tiene que mover hacia los lados, hacia arriba y hacia abajo, y entonces bastará con cambiar el valor por otro simplemente en esta declaración para que se mueva más o menos en cada dirección. Lo primero que se hace es elegir el primer gráfico del coche a visualizar (30) y las coordenadas iniciales (65, 65, -1) y se mete en un bucle de tipo LOOP ... END, del que no saldrá nunca, a menos que destrocen al coche dentro de este bucle lo que se hace es comprobar los intentos de mover los coches. Para ello, lo primero es una sentencia IF (TECLADO) que se cumplirá si se eligieron las teclas como forma de controlar al coche; entonces, se harán comprobaciones de teclas de forma individual, es decir, se comprueba primero si se pulsa la tecla arriba y no se pulsa ni la de la izquierda ni la de la derecha y está en una coordenada Y > 65. En caso de que todo eso sea cierto, el gráfico del coche a visualizar es el coche mirando hacia arriba (31) se decrementa la coordenada Y en la distancia que se quiera mover al coche y se pone una nueva variable llamada dirección al valor de la dirección del coche; esta variable se utilizará por si acaso se dispara para saber en qué dirección tiene que ir el disparo. Las direcciones posibles son las siguientes:

- ARRIBA .....
- ABAJO .....
- DERECHA .....
- IZQUIERDA .....
- ARRIBA + IZQUIERDA .....
- ARRIBA + DERECHA .....
- ABAJO + IZQUIERDA .....
- ABAJO + DERECHA .....

A continuación se van comprobando todas las teclas para cada una de las direcciones, determinando, en cada caso, el gráfico adecuado para visualizar y moviéndolo la distancia adecuada en la dirección adecuada. Llegado al punto en el que se comprueba que ocurre al pulsar la tecla de la barra

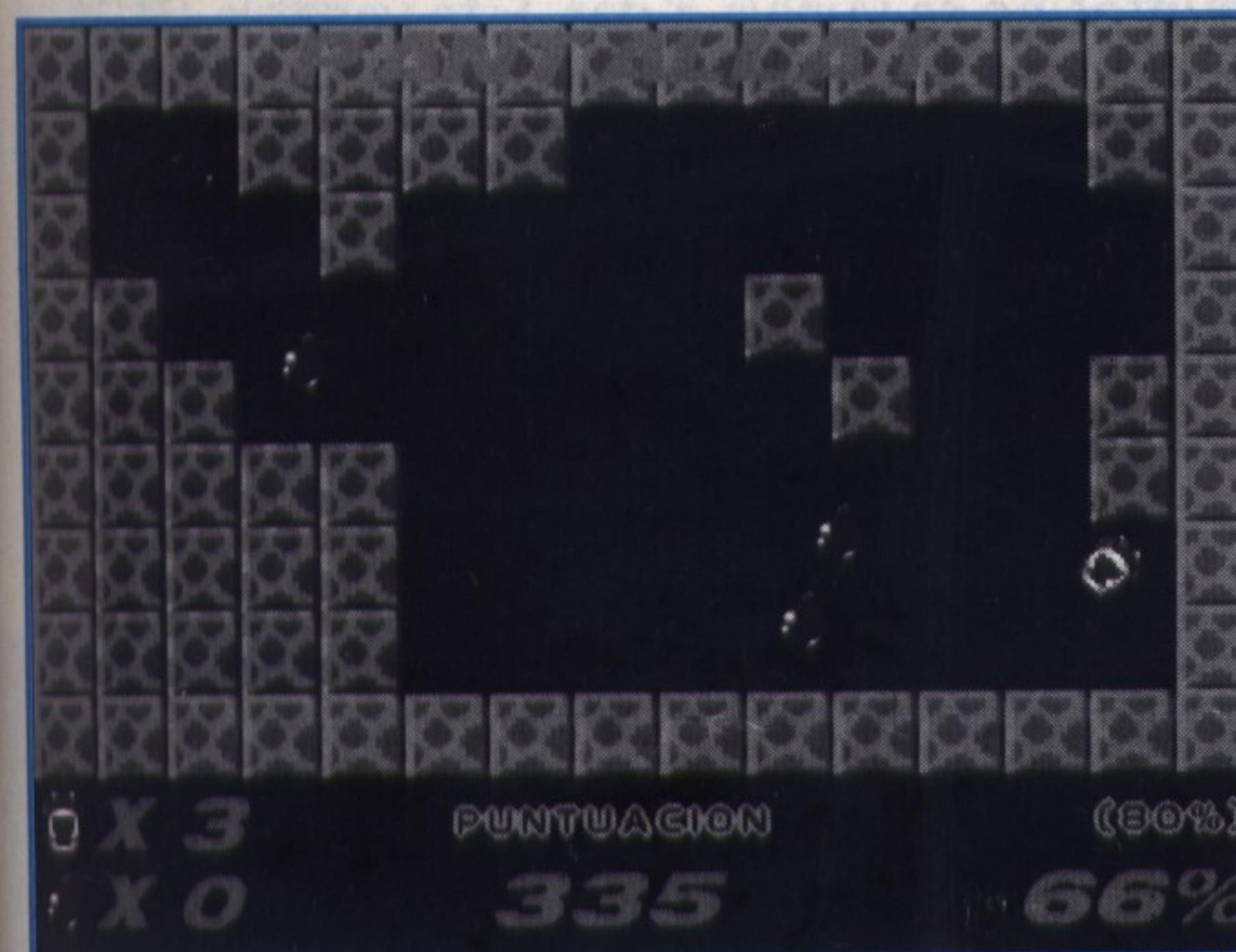


FIGURA 2.

espaciadora, lo que se hace es verificar si se está pulsando y se puede disparar, es decir, el coche no se encuentra subido en una baldosa marrón. En caso de que sea así, se comprueba otra variable, llamada *DISPARANDO*, para no poder dejar apretada la tecla para disparar. Si *DISPARANDO* vale 0, entonces crea un nuevo disparo a partir de las coordenadas X e Y del coche y en la *DIRECCION* que lleva el coche, después de crear el disparo pone la variable *DISPARANDO* a 1 para que así no se cree un nuevo disparo sin haber soltado la barra espaciadora; si no se tiene pulsada la tecla de la barra espaciadora, la variable *DISPARANDO* se pondrá a 0 para que se pueda disparar de nuevo cuando se apriete.

Después de comprobar si se intentaba disparar, se controla si se intenta poner una bomba con la tecla ALT; para ello, habrá que verificar si se pulsa esta tecla y hay bombas disponibles, es decir, *CUANTAS\_BOMBAS* > 0. Si eso es verdad, se comprueba una variable (*PONIENDO\_BOMBA*) que se utilizará para que no sitúe bombas sin haber soltado la tecla ALT, es decir, la variable *PONIENDO\_BOMBA* realiza la misma función para las bombas que *DISPARANDO* para el disparo; en caso de que *PONIENDO\_BOMBA* fuera 0 decrementa el número de bombas disponible, crea una bomba en las coordenadas X e Y del coche y pone *PONIENDO\_BOMBA* a 1, si no se pulsa ALT esta variable es puesta a 0. Luego viene la sentencia *ELSE* que corresponde al *IF(TECLADO)*, es decir, a partir de ahora aparecen los controles del coche en caso de que el control elegido en lugar de teclado fuera el joystick, no merece la pena repetirlo porque es exactamente igual que con el teclado, pero con el joystick (mirar la estructura global joystick en el manual de referencia y dudas aclaradas); para el intento de disparo y de poner bombas con el joystick ocurre lo mismo que con el teclado.

Después, se hace una comprobación para ver si el coche está chocando con un obstáculo (baldosa marrón); en caso de que sea cierto coloca la variable *PUEDE* a 0 para que el coche no pueda disparar cuando está encima de ellas; además, con la sentencia *FOR* que hay se consigue un retardo que permite que el coche

## Process\_inicia\_fondo [STAGE]

Se utiliza para iniciar los gráficos de las baldosas de ambos colores (verdes y marrones) por toda la pantalla, es decir, al empezar cada pantalla rellena toda la pantalla de baldosas verdes y marrones. El parámetro **STAGE**, que recibe siempre, será el número de pantalla a la que vamos a acceder; entonces, con este parámetro se ejecuta una sentencia **SWITCH** para iniciar la pantalla adecuada, es decir, poner los cuadrados verdes por toda la pantalla y los marrones en ciertas posiciones. Como el número de pantallas es de 10 habrá desde **CASE 1** hasta **CASE 10**, aunque simplemente comentaremos **CASE 1** y el **CASE 2**.

En el **CASE 1** (primera pantalla), lo que se hace es llenar toda la pantalla con bloques verdes (**PROCESS FONDO (X,Y)** que explicaremos más adelante). Para ello, se utilizan dos bucles **FOR** anidados con las variables **CONT1** y **CONT2**, que variarán las coordenadas X e Y de las baldosas verdes para colocarlas por toda la pantalla; el bucle más externo variará la coordenada X de 43 en 43 unidades, que se corresponde con el ancho de cada baldosa verde (así quedarán unas al lado de otras), y el bucle interno variará la coordenada Y de 43 en 43 unidades, es decir, el ancho de cada baldosa verde (así quedarán unas debajo de otras). Como todas las pantallas tendrán mayoría de baldosas verdes, estos dos bucles, como se puede observar, se encontrarán en todos los **CASE** y así bastará solamente con iniciar luego los obstáculos que se quieran (baldosas marrones). Una vez que se ha iniciado la primera pantalla (toda con baldosas verdes) cuando se haya pasado se iniciará la pantalla 2 (**CASE 2**).

En el **CASE 2** (segunda pantalla), lo que se hace es colocar cuatro obstáculos (baldosas marrones) en ciertas coordenadas (**PROCESS OBSTACULO (X,Y)** que serán comentadas más adelante), y luego iniciar por completo todo de baldosas verdes (con los dos bucles anidados que utilicé en el **CASE 1**). Puede parecer que en algunos sitios haya dos procesos superpuestos (una baldosa verde y una marrón), pero como veremos más adelante esto se evita por una comprobación de colisión en el proceso **FONDO (X,Y)**. El resto de sentencias **CASE** funcionan igual que estas primeras: la diferencia es que se inician procesos de tipo obstáculo en distintas coordenadas. Una vez iniciada la pantalla adecuada, este proceso finaliza ya que el hecho de que se sigan visualizándose en pantalla los gráficos se debe a los procesos **FONDO** y **OBSTACULO** que a continuación pasamos a detallar.

vaya más lento. En caso negativo, coloca **PUEDE\_DISPARAR** a 1 para que se pueda disparar (ya que no pisa la baldosa marrón). Posteriormente, realiza una comprobación para ver si colisiona con alguna bola (grande, mediana o pequeña); en tal caso sitúa los gráficos de la explosión (80-85), pone la variable **MUERTO** a 1, decremente el número de vidas y ejecuta la sentencia **BREAK** para que se salga del bucle **LOOP END** del inicio.

## PROCESS\_INICIA\_BORDE ()

Lo que hace es visualizar el gráfico que forma el borde de la pantalla, que está formado por un grupo de bloques grises. Selecciona el gráfico (52) y las coordenadas en las que debe colocarse (321,241) y se entra en un bucle de tipo **LOOP ... END** en el que sólo ejecuta la sentencia **FRAME**. Así, cuando sea llamado por única vez el gráfico se estará visualizando siempre en esas coordenadas.

## PROCESS\_OBSTACULO (X,Y)

Se usa para visualizar el gráfico de las baldosas marrones en las coordenadas X e Y que se le indican como parámetros; selecciona el gráfico adecuado (50) y se entra en un bucle **LOOP ... END** en el que sólo se ejecuta la sentencia **FRAME**, es decir, no finaliza nunca a no ser porque otro proceso le mande la señal de morir.

## PROCESS\_BOMBA (X,Y)

Se utiliza para visualizar el gráfico de una bomba en las coordenadas X e Y que se le indican como parámetros; lo que se hace es

seleccionar el gráfico adecuado (90) y ejecutar un bucle **LOOP ... END** en el que sólo se ejecuta la sentencia **FRAME**, es decir, no finaliza nunca a no ser porque otro proceso le mande la señal de morir. Las coordenadas X e Y que recibirá serán siempre las coordenadas del coche en el momento en que se mandó poner la bomba.

## PROCESS\_INICIO\_BOLA\_GRANDE (X,Y)

Éste es utilizado para visualizar los gráficos que indican que se va a iniciar una nueva bola; se trata de los gráficos del 70 al 74. Lo que se hace es ejecutar un bucle **FOR**, desde 1 hasta 3, y dentro de este un bucle **FROM** desde 70 hasta 74 para que los cuatro gráficos se visualicen uno detrás de otro durante tres veces. Una vez que finalizan estos bucles lo que se hace es iniciar una bola grande en sus mismas coordenadas.

## PROCESS\_BOLA (X,Y)

Con este proceso se puede controlar a las bolas grandes. Se inicializa el gráfico adecuado (60) y se entra en un bucle **LOOP** en el que se comprueba si choca con un disparo; si es así destruye el disparo que chocó, incrementa los puntos y genera dos bolas medianas que se crean en sus mismas coordenadas y se mueven en distintas direcciones, a causa de *INC\_X* e *INC\_Y* ya que a la primera bola mediana se le pasan como parámetro con signo positivo y a la segunda con signo negativo. Además, ejecuta la sentencia **BREAK** para que se salga del bucle y desaparezca por tanto la bola grande. Se comprueba también si choca con una bomba,

## Juegos ganadores: 2º Explosos

### Process disparo (x,y, dire)

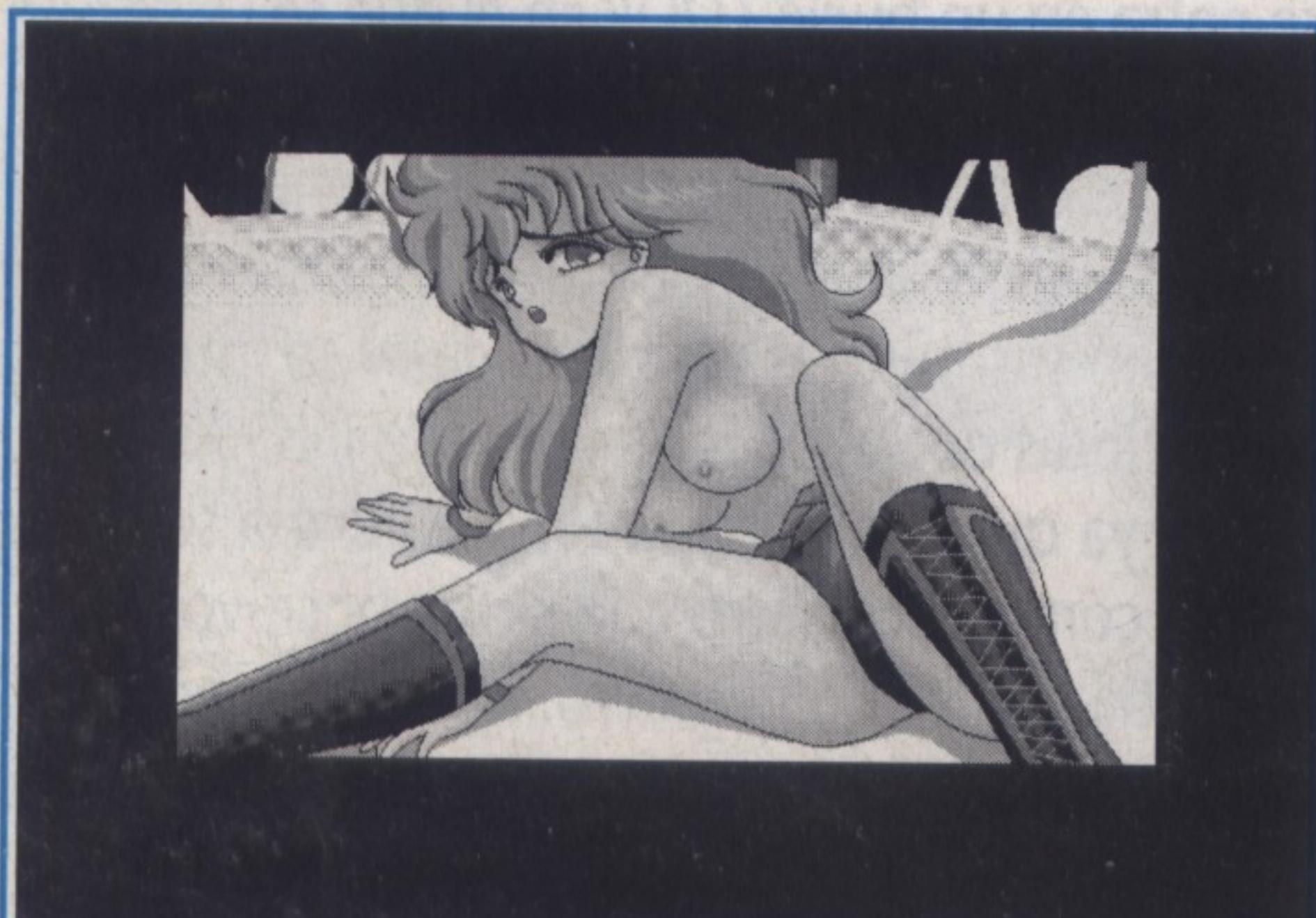
Este proceso se usa para visualizar el gráfico del disparo en ciertas coordenadas y que se moverá en una dirección determinada que recibirá como tercer parámetro. Las coordenadas en las que se inicia serán las del coche en el momento de disparar, por eso, lo que se hace es seleccionar el gráfico (38) y entrar en un bucle LOOP del que sólo se saldrá si se sale de la pantalla o choca con un obstáculo. Lo primero que hace dentro del bucle es, en función de DIRE a través de un SWITCH, mover al disparo en cierta dirección (que será la misma que tenía el coche cuando se disparó). Una vez que se ha movido el disparo, se comprueba si éste se sale de la pantalla en cuyo caso solamente se ejecuta la sentencia BREAK para que se salga del bucle y el disparo que se salió de pantalla desaparezca. A continuación si comprueba con un obstáculo; en caso de ser cierto se toma el identificador del obstáculo con el que choca y se manda matar a ese proceso obstáculo, aparte de incrementar los puntos y el porcentaje y comprobar si el porcentaje es mayor que 80; si es así, la pantalla estará pasada (se pone PASADA a 1) y luego se ejecuta la sentencia BREAK para hacer desaparecer también al disparo que chocó con el obstáculo.

en caso de ser así incrementa los puntos y se hace desaparecer a la bomba que la mató, ejecutando BREAK para que se salga del bucle y desaparezca (ahora no se generan dos bolas medianas). Después de estas comprobaciones se varían las coordenadas X e Y en INC\_X e INC\_Y, respectivamente, y se comprueba si la bola choca con una pared, en cuyo caso cambia de signo INC\_X si choca con una pared lateral, e INC\_Y si choca con la superior o inferior.

### PROGRAMA PRINCIPAL

Es la parte del programa encargada del control del juego; lo primero que hace es cargar los archivos necesarios, tanto los de sonido como los de gráficos y fuentes; además se inicia el modo gráfico a 640X480 y el número de imágenes por segundo a 200. Se inician dos variables del control de juego: NIVEL, que cada vez que se inicie el juego será puesto a 2 (nivel medio) y TECLADO como controlador del coche (es puesto a 1).

FIGURA 3.



Juegos ganadores: 2º

Posteriormente, se entra en un bucle de tipo LOOP del que sólo se saldrá cuando se pulse la tecla 5 (salir al DOS); éste será el bucle general (para iniciar una nueva partida).

### BUCLE GENERAL

Al entrar en él se inician variables al comienzo de cada partida, como los puntos, la pantalla actual, el número de vidas, de bombas, si está muerto, si se quiere acabar una partida, si se ha continuado alguna vez o si se ha pasado una pantalla; posteriormente, se pone el gráfico de presentación y se escriben las opciones, entrando en un nuevo bucle LOOP en el que se comprueba la tecla pulsada. Si se pulsa 1 se sale de este último bucle LOOP (se ejecuta BREAK) y se inicia una nueva partida, si se pulsa 2 se elige el nivel de dificultad, y en función de la dificultad que se seleccione se varía la variable NIVEL a 1, 2 o 3. Si se elige 3 se escoge la forma de controlar el coche (por teclado o joystick); si se elige teclado la variable TECLADO se pone a 1 y si se elige joystick se pone a 0. Si se elige 4 se pone la pantalla de créditos y si se pulsa el 5 se sale del juego con EXIT. A partir de aquí aparece todo el código acerca de cada partida en individual, es decir, se inicia un nuevo bucle, esta vez de tipo REPEAT ... UNTIL del que sólo se saldrá cuando no queden vidas (nueva partida).

### BUCLE GENERAL POR PARTIDAS

Aquí se comprueba si se ha pasado una pantalla. Como se puso la variable PASADA a 1 se inicia una nueva pantalla, la primera, y lo que se inicia el borde la pantalla, el fondo y escribir los textos que indican la puntuación, las vidas, etc. Luego se comprueba si le han matado; en caso afirmativo, no se inicia el fondo (no se llena de baldosas desde el principio sino que se deja tal y como estaba cuando le mataron), eliminando todas las bolas que existían y todos los disparos y se vuelven a escribir los textos de vidas, bombas, puntos... Ahora se entra en el bucle que regula las pantallas, que es de tipo LOOP.

### BUCLE GENERAL POR PANTALLAS

De este bucle sólo se sale si le matan, si se pasa de pantalla o si se pulsa ESC, además se puede comprobar si la puntuación es mayor o igual que PROXIMA\_VIDA que indica a cuántos puntos se conseguirá la próxima vida; para ello, se verifica la puntuación con un margen de 10 puntos y se comprueba que no se ha dado la vida, en cuyo caso se da (incrementando el número de vidas y PROXIMA\_VIDA), y se pone la variable DADA a 1 para que no se vuelva a dar; es decir, si se llevan 3005 puntos y se da por primera vez la vida, si por casualidad se está un rato sin hacer

puntos no se vuelve a dar. Esta variable DADA sólo se pone a 0 cuando ya no se esté en el rango en que se puede dar. En este bucle se generan números aleatorios y se comprueba que si es menor que cierto valor se inicia una bola grande o un objeto que da cuatro bombas más; el rango en el que se sacan números aleatorios cada vez es menor a causa de las variables PANTALLA y NIVEL, que son utilizadas dividiendo para que el rango disminuya a medida que ambas aumentan de valor. Por otro lado, se comprueba si se pasa de pantalla o le matan o se pulsa Escape para salir de este bucle por pantallas.

### FIN DEL BUCLE POR PANTALLAS

Si se salió del bucle porque nos habíamos pasado la pantalla, lo que hacemos es congelar los procesos coche, bolas y disparos y escribir en pantalla «CLEAR», para indicar que se la ha pasado y a continuación muestra la foto «manga» en pantalla, en función de la pantalla que nos acabemos de pasar (cuando se llama PUT se indica la pantalla en la que estamos, ya que las fotos están unas detrás de otras con los códigos del 1 al 10). Ésta se visualizará hasta que se pulse espacio o el botón 1 del joystick. Despues se comprueba si la pantalla que se acaba de pasar es la 10, en cuyo caso se coloca el final del juego: el movimiento de texto por pantalla. Para dicho final se sitúa el gráfico de presentación y se inicializan los textos que, posteriormente, se moverán; por medio de la sentencia MOVE\_TEXT se van moviendo hacia arriba hasta que se llega a cierta posición. Esto se hace con todos los textos hasta que acaban de ejecutar BREAK para volver a la pantalla de presentación; si no se ha ejecutado es porque la pantalla no era la 10, por lo que se tendrá que incrementar la pantalla (en el caso de que se pasen de pantalla). Ahora llega la comprobación del número de vidas. Se verifica si el número de vidas es cero y no se ha continuado. Si ocurre esto se da la opción de continuar consistente en dos bucles. El externo irá de 9 a 0 contando (simulando el tiempo que queda para poder pulsar) y el interno irá de 0 a 10, provocando un retardo (para que no baje de 9 a 0 rápidamente) y se comprueba si se pulsa S o N. Si se pulsa S se ponen las vidas a las bombas a 4, indicando que le han matado (MUERTO=1), para que no inicie la pantalla en la que estábamos desde el principio, y se pone CONTINUADO a 1 señalando que ya se ha continuado. Si se pulsa la N se ponen las vidas a -1, y si las vidas son mayores que 0 es porque se ha continuado, mientras que si son menores que cero se ponen a cero; entonces se matan los procesos bomba, vidas y bombas y se borran los textos, llegando al final del bucle por partidas del que sólo se sale si las vidas son 0.