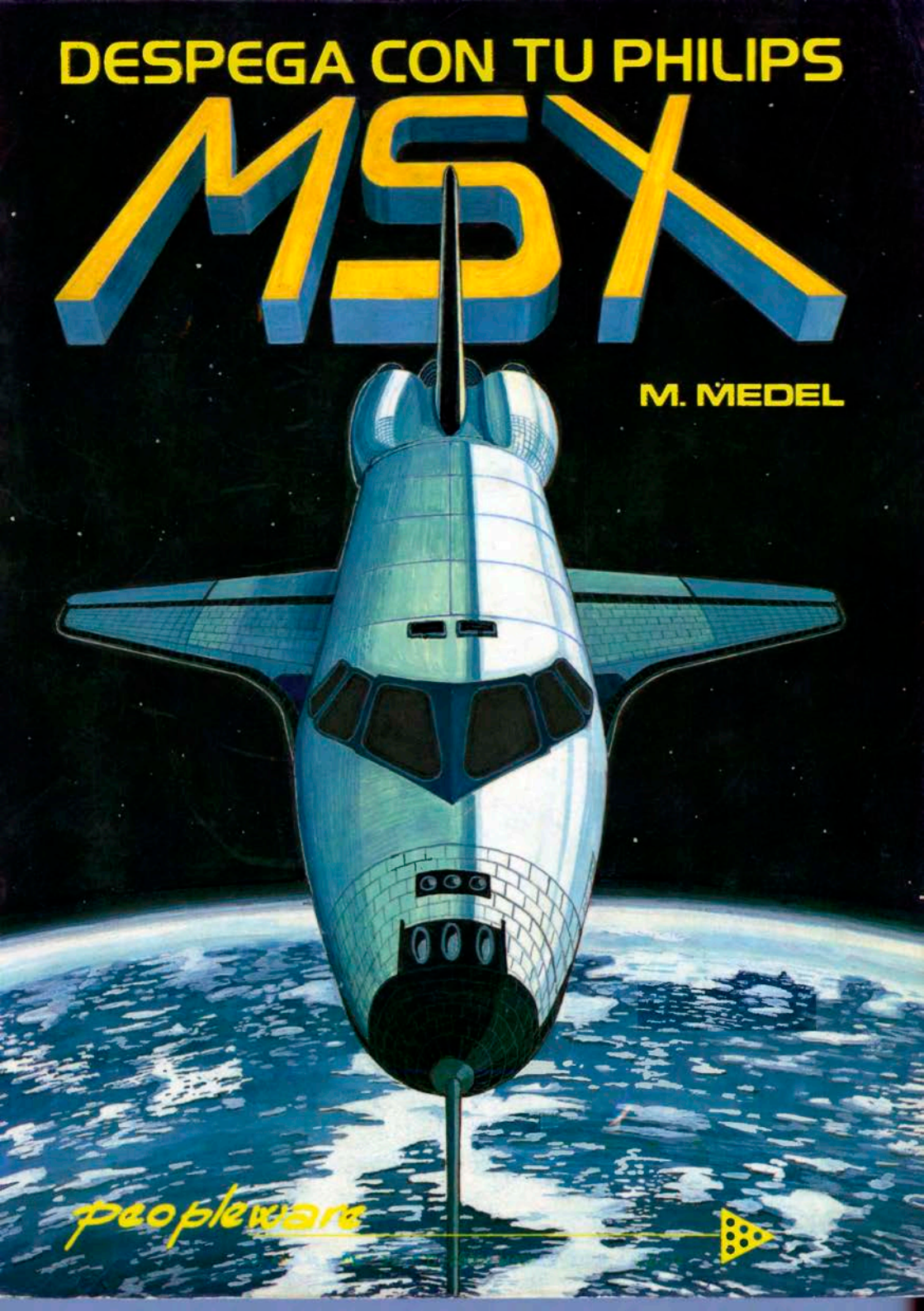
DESPEGA CON TU PHILIPS

MSX

M. MEDEL



peopleware

DESPEGA CON TU

MSX

DESPEGA CON TU

MSX

M. MEDEL

peopleware

CLARA DEL REY. 20 - 5°D - 28002 MADRID - TELF. 4.15.87.16

Despega con tu MSX

Producido en 1985 en España por:
PEOPLEWARE, S.A.

Consultas referentes al libro:

PEOPLEWARE, S.A. * Clara del Rey, 20 - 5º D
Teléfono: (91)415.87.16 * 28002 MADRID

Edita, compone e imprime:

CONORG, S.A. * Vía Lusitana, 62
28025 MADRID (ESPAÑA)

I.S.B.N.: 84-86406-01-3
Depósito Legal: M-33834-1985

Printed in Spain

Contenido

Introducción

La finalidad que el autor ha perseguido al escribir esta obra no ha sido la de realizar otro libro para copiar juegos, sino proporcionar a todos aquéllos poseedores de un ordenador MSX una forma completa de aprender las particularidades de este sistema a través de la confección de varios programas.

En ellos se hace uso de las características principales del BASIC MSX y están estructurados de manera que durante su realización nos iremos familiarizando con nuestro equipo de forma progresiva. Para conseguir este objetivo se ha procurado dividir los programas en bloques independientes, pudiendo ver los primeros resultados de nuestra programación prácticamente desde el principio. Así se consigue amenizar el proceso a la vez que se obtiene una idea más clara de la función que cada bloque cumple dentro del programa.

El Standard MSX

Al principio de la década de los setenta eran pocos los que pensaban que los ordenadores que conocemos como **home computers** u ordenadores caseros, fueran a tener una gran difusión, mucho menos que se pudiera llegar a los niveles actuales, donde hay millones de ellos pululando por nuestros hogares dedicados a los más variados fines.

Lo cierto es que debido al avance tecnológico y a la creciente demanda de un mercado, que en la mitad de los setenta no hizo sino crecer de forma vertiginosa, se han fabricado centenares de miles de estos ordenadores por firmas, que en muchos casos, hasta ese momento nada habían tenido que ver con la informática. Este es el caso del nacimiento de alguno de los más populares ordenadores personales de aquella época. Debido a esta rápida expansión y a la carencia de un estándar rígido al que ajustarse (la mayoría de los ordenadores de este tipo lo único que tenían en común era que empleaban el BASIC), cada firma comercializó sus equipos sin tener en cuenta la compatibilidad con el resto.

Esta situación ha conducido a un verdadero caos en la informática personal creando serios inconvenientes a aquellos usuarios que tratan de intercambiar, usar o adaptar programas de otros equipos al suyo. Estos problemas son igualmente graves cuando se trata de adquirir un periférico, disco, impresora, plotter, monitor, etc. donde uno queda sujeto a la oferta existente para aquel equipo específico, o bien arriesgarse a costosas y a veces poco seguras adaptaciones.

Debido a estos problemas, ya habían habido intentos de estándarización, pero hasta ahora ninguno que abarcara, como en el caso del **sistema MSX** una compatibilidad tan grande. La empresa norteamericana Microsoft fue la que efectuó los estudios para la realización de un sistema que sirviera como base a los ordenadores personales de 8 bits, contemplando una rigurosa estándarización en las áreas más importantes del ordenador con el fin de que la compatibilidad fuera total, no limitándose únicamente a la parte de programas, software, sino extendiéndose a los periféricos que se pueden conectar a estos equipos.

Efectivamente cualquier periférico para un ordenador MSX, impresoras, joystick, discos, etc., puede conectarse a cualquier equipo adscrito a dicho estándar. Esto significará a corto plazo un abaratamiento de periféricos para este sistema, ya que debido al número e importancia de los fabricantes implicados en esta norma, así como a las ventajas que el MSX tiene con respecto a otros BASIC existentes, se puede augurar un rápido crecimiento de los ordenadores que ostenten estas siglas, lo cual conllevará (de hecho está sucediendo) a que las empresas fabricantes de periféricos puedan hacer series mayores de éstos, sin los problemas de adaptación que hasta ahora tenían. Asimismo, los fabricantes de programas (software) contarán con un mercado compuesto por gran número de ordenadores de distintas marcas completamente compatibles entre sí, con lo que se podrá hacer más rentable el diseño de un programa para el vendedor y abaratarle, debido al mayor número de unidades vendidas.

Pero el MSX no es un estándar que únicamente se haya preocupado de la compatibilidad, con ser ésta una de sus principales ventajas, también incorpora todos aquellos elementos que se consideran necesarios hoy en día en un ordenador de estas características, dedicado tanto a la pequeña y mediana gestión como al entretenimiento.

Así, en su aspecto gráfico, cuenta con un elemento tan importante como es la **VDP** (Video Display Processor) o **procesador de vídeo.**

Este circuito nos permite el control de los gráficos y texto en la pantalla, librando a la unidad central de proceso de ese trabajo y permitiendo unas posibilidades gráficas extraordinarias entre las que podemos contar la generación de sprites o dibujos móviles.

Este procesador de video cuenta con 16K de memoria RAM para su uso exclusivo. Es de resaltar que esta memoria no se resta de la RAM utilizable por el usuario, con lo que se puede hacer uso de los gráficos de alta resolución sin que esto acorte la memoria disponible para programación.

Posteriormente, durante la realización de los programas, tendremos oportunidad de comprobar el gran número de posibilidades gráficas que la VDP nos ofrece.

Este procesador de video está complementado con otro procesador independiente para generación de sonido, un circuito que es capaz de generar tres voces independientes permitiendo la programación de tonos, octavas, volúmenes, pausas, longitudes de las notas, y todo un repertorio de posibles combinaciones. Su rendimiento está prácticamente más limitado por nuestra capacidad de programarlo, que por sus posibilidades.

El sonido puede ser generado de forma independiente en el transcurso del programa, de manera que éste no tenga que ser interrumpido mientras se ejecuta una nota, lo cual, sobre todo en el caso de juegos, permite una sincronización total entre el efecto gráfico y sonoro.

También cuenta este sistema con otro procesador muy importante por su futura expansión, se trata de la PPI (Peripheral Programable Interface) o procesador programable para el control de periféricos. Este chip es el encargado de controlar, entre otras cosas, los cartuchos de memoria o juegos que conectamos en los zócalos que a este fin posee nuestro equipo. También permite, mediante un expansor, ampliar nuestra memoria hasta 1 Mbyte, es decir **1.000.000** de bytes, lo cual no está nada mal para un ordenador personal.

Este circuito, asimismo, libera a la unidad central de una serie de labores que de otra manera se vería obligada a realizar, permitiendo una flexibilidad y posibilidades difíciles de conseguir de otra forma.

Hay que reseñar también que estos equipos cuentan con entradas para dos joysticks, salidas para impresora, monitor de video, televisión, conexión para cassette con control de motor y un adaptador para conectarles un sistema de discos, el **MSX-DOS.**

Este sistema operativo para el disco (Disc Operating System) es compatible con el MS-DOS, usado por muchos equipos más potentes de 16 bits, entre ellos el PC de IBM, lo cual abre grandes perspectivas.

Estas son, a grandes rasgos, las características más destacadas y diferenciadoras del MSX, el lenguaje **BASIC** que utiliza lo iremos conociendo a medida que lo utilicemos en los programas que siguen; únicamente resaltar que Microsoft, la empresa que lanzó este sistema en Junio del 83, ha seguido trabajando en su perfeccionamiento y en la incorporación de mejoras y posibilidades a los equipos existentes.

Así, destacaremos que actualmente, según una entrevista realizada a David Fraser, General Manager de Microsoft, esta empresa se encuentra trabajando en varios proyectos MSX entre los que se citan compiladores para FORTRAN, BASIC, COBOL y PASCAL, y una versión de hoja electrónica. También hay rumores sobre una segunda generación de equipos MSX de 16 bits, compatible con los actuales en cuanto que serán capaces de aceptar sus programas y hacer uso de sus periféricos. Por lo tanto, y teniendo en cuenta el número e importancia de las firmas comprometidas con este sistema, sus posibilidades presentes y futuras, podemos decir que el MSX no ha hecho sino que comenzar su andadura con un amplio horizonte abierto ante él.

Programación Estructurada

No vamos a dar aquí una descripción exhaustiva de lo que es un programa ni de las técnicas empleadas en su confección, esto es algo que iremos viendo de una forma práctica y más amena a medida que realicemos los que hay en este libro. Sin embargo, es necesario dar una idea general de cómo se puede realizar dentro de la gran flexibilidad que la programación permite, pues a diferencias de otras técnicas, la de programar es completamente abierta, siendo la impronta del programador la que consigue que su realización sea una obra de arte o simplemente algo corriente.

Salvando las naturales distancias, podemos decir que la programación es como la pintura, no es tan importante lo que se ha pintado, sino cómo se ha hecho; es por eso que únicamente unos pocos merecen el calificativo de artistas, en programación sucede otro tanto, únicamente un grupo reducido merece este título, pues éste es verdaderamente un arte, y como tal, difícil de enseñar.

Podemos describir de una forma simple un programa diciendo que son un número de instrucciones ordenadas en una secuencia prefijada que al ser leídas por el ordenador determinan, mediante la función o comando que cada una contiene, la ejecución de una tarea específica. Estas tareas pueden ser de distinta índole y corresponderán a las posibilidades de interconexión que nuestro ordenador posea. Así podemos desde presentar en la pantalla de nuestra TV la solución de un complejo problema matemático de una forma gráfica, hasta comandar el brazo articulado de un robot, pasando por la activación de una impresora a la adquisición de datos de un disco o un cassette, siempre que contemos con el **interface** adecuado para esta función.

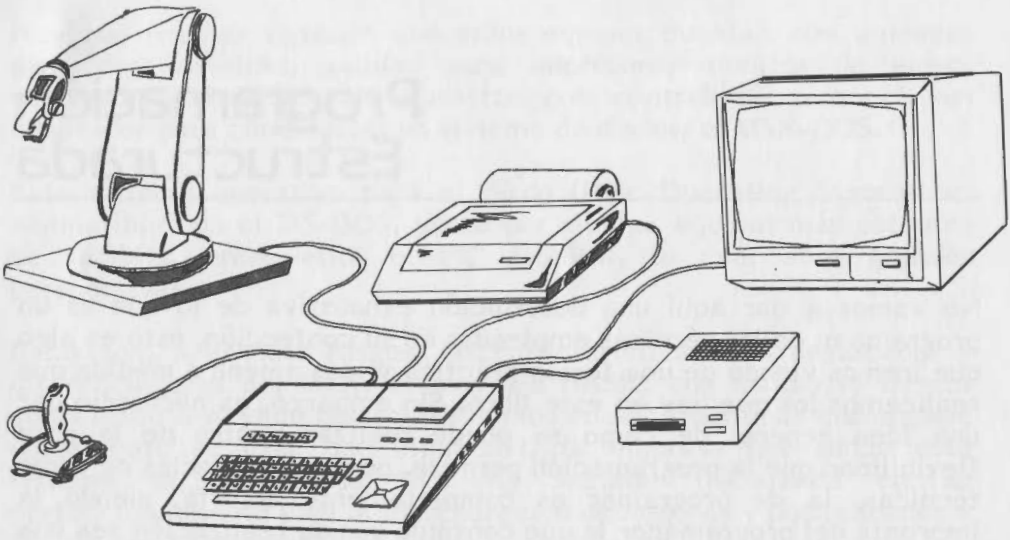


Fig. 1 Algunas posibilidades de conexión de nuestro ordenador

Es en la ordenación de estas instrucciones en donde podemos distinguir dos tipos de programación: aquélla que se realiza de una forma secuencial, repitiendo tantas veces como sea necesario en el transcurso del programa una instrucción o conjunto de éstas, o por el contrario, la que se realiza mediante la confección de bloques definidos para cada una de las tareas que van a realizarse con más frecuencia, y a los que se recurre de forma aleatoria siempre que es necesario, siguiendo únicamente las necesidades que el programa nos vaya planteando y no siguiendo un orden prefijado.

Este tipo de **programación** recibe el nombre de **estructurada** y los bloques o subprogramas que realizan una función determinada se les denomina **subrutinas.** Este es el tipo de programación que vamos a utilizar para hacer nuestro programa y que nos va a permitir subdividirlo en distintos bloques de acuerdo con nuestras necesidades.

Con ésto podremos llegar a una mejor comprensión de cómo funciona el conjunto a través del conocimiento de cada uno de sus bloques, ya que de esta manera, al estar independizados dentro del programa, nos permitirán saber en cada momento dónde y cómo se realiza una determinada tarea, así como estudiar ésta aislándola del resto de las subrutinas. Este tipo de programación permite una mayor flexibilidad y un mejor aprovechamiento de la memoria de nuestro ordenador, así como facilita enormemente la depuración y seguimiento de nuestros programas. Tiene en su contra el alargar, en determinados casos, el tiempo de ejecución de un programa cuando debido a tener que acceder de forma repetida a una o varias subrutinas, hay que sumar los tiempos empleados en estos desplazamientos o **saltos** al tiempo total.

La diferencia entre los dos tipos de programación pueden apreciarse en el dibujo de la Figura 2. En este dibujo se han representado los **diagramas de flujo** pertenecientes a dos programas que realizan la misma función, poner un cohete y un meteorito en la pantalla, proporcionarles movimientos independientes y comprobar si ha habido colisión entre ambos o si el cohete ha alcanzado su destino. En el caso de la **programación secuencial** vemos cómo se repiten a lo largo del diagrama elementos que son iguales, mientras que en el caso de la **programación estructurada** hay una parte principal que hace uso de los bloques o subrutinas según sea necesario, dirigiéndose a éstos desde distintos puntos mediante saltos de programa.

Esto se puede apreciar en la subrutina de dibujo del cohete y meteorito, la cual es accedida desde el comienzo y desde la subrutina de posicionamiento indistintamente.

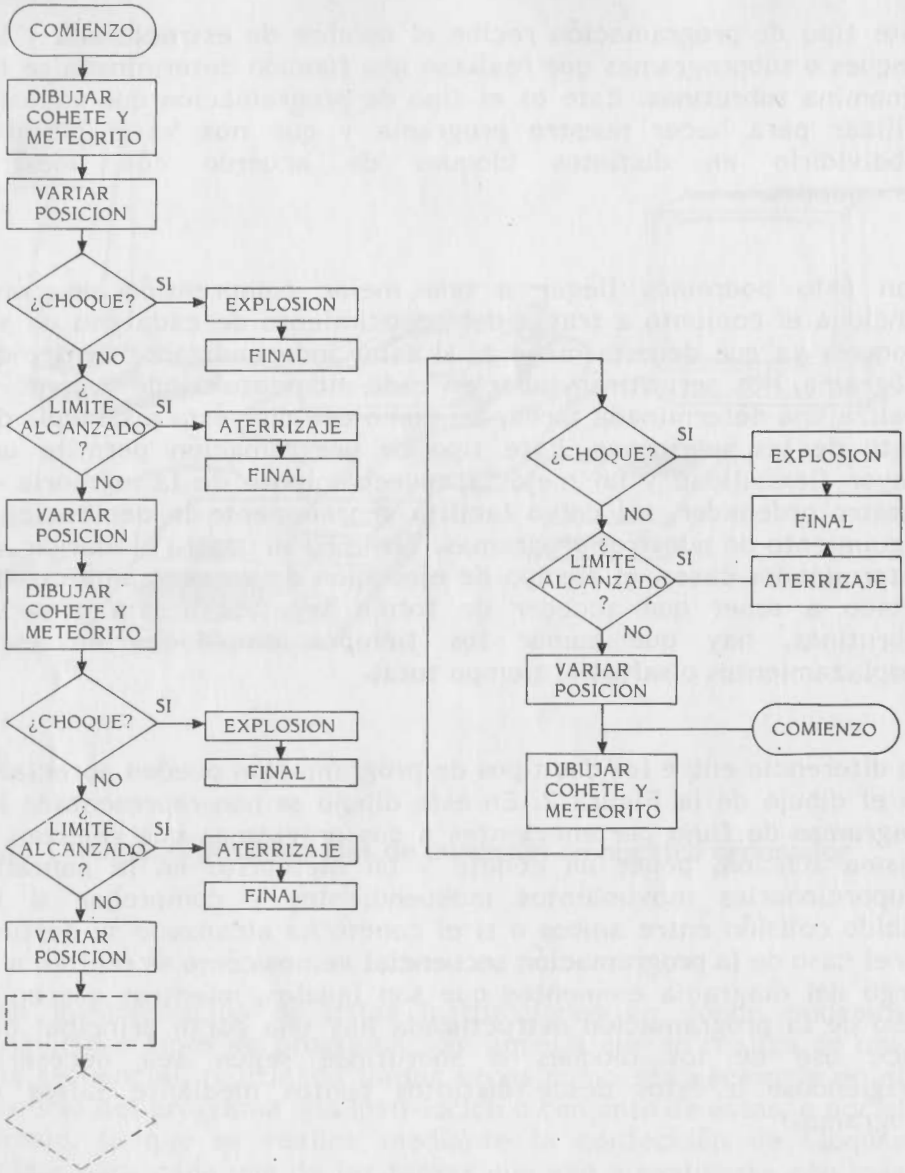


Fig. 2 Ejemplos de programación secuencial y estructurada

Estos ejemplos serán ampliados en los siguientes capítulos realizando de forma directa programas de mayor complejidad que los aquí expuestos.

Descripción de los Programas

Al conjunto de programas que vamos a realizar, podíamos titularlos "LA BASE MSX", y con ellos vamos a tener la oportunidad de emular las distintas fases por las que pasa la nave espacial COLUMBIA desde su lanzamiento hasta su aterrizaje.

Realizaremos una base de lanzamiento que estará construida de una forma sencilla pero que contará con todos aquellos elementos que caracterizan una instalación de ese tipo, es decir, contaremos con una torre de control, una plataforma de lanzamiento donde desplazaremos nuestra nave con sus cohetes impulsores hasta la torre desde donde será lanzada y todo esto lo situaremos en un paisaje solitario donde está amaneciendo y hay varias nubes en el horizonte. Algo similar a lo que puedes ver en la figura número 3.

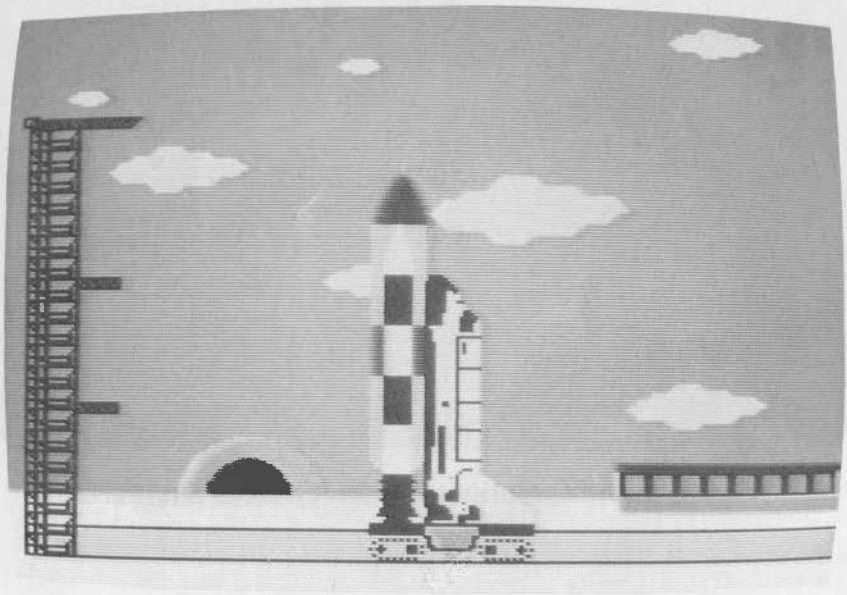


Fig. 3 Base de lanzamiento de la nave COLUMBIA

Nuestro programa nos permitirá desplazar el conjunto de la plataforma y el COLUMBIA hacia la derecha o la izquierda mediante las teclas del cursor (o el joystick), y una vez situado este conjunto en el punto de lanzamiento veremos cómo responde a nuestra orden de disparo, perdiéndose entre las nubes camino del cielo seguido por un penacho de llamas procedentes de sus toberas.

Todo esto después de haber levantado una gran humareda en la base de lanzamiento y provocar un ruido ensordecedor. Veremos también cómo una vez alcanzada la altura adecuada se produce el desacople de la nave y el conjunto de impulsores con el gigantesco tanque de combustible tal y como está representado en la figura número 4.

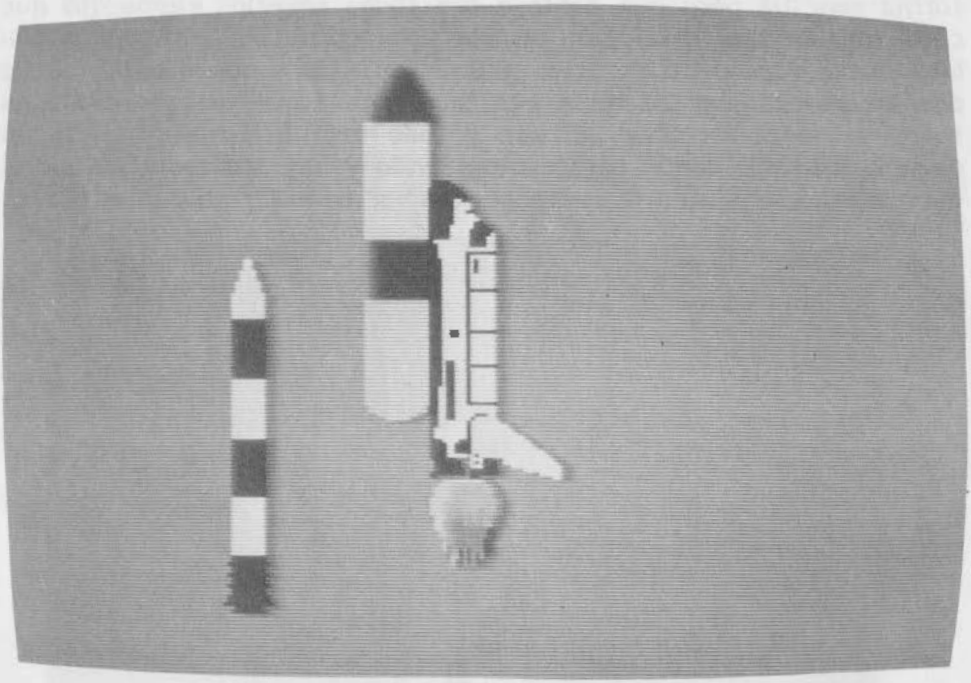


Fig. 4 Desacoplamiento de los cohetes auxiliares

Una vez que nuestra nave ha consumido los 705.800 Kg. de combustible que contenía el depósito que llevaba adosado a su panza, se encuentra en órbita a una altura comprendida entre los 240 y los 300 Km. con velocidades que pueden alcanzar hasta los 26.700 Km. por hora.

Desde esa altura podremos ver cómo se desplaza la tierra debajo de nosotros con una masa nubosa que se superpone a los continentes y mares que aparecen en nuestra imagen. En esta órbita permaneceremos rodeados del silencio del espacio, sin que ya el ruido de los motores nos dañe nuestros oidos y escuchando únicamente el murmullo de los instrumentos de a bordo.

La imagen que ofrecerá nuestra nave será como la que está representada en la figura número 5.

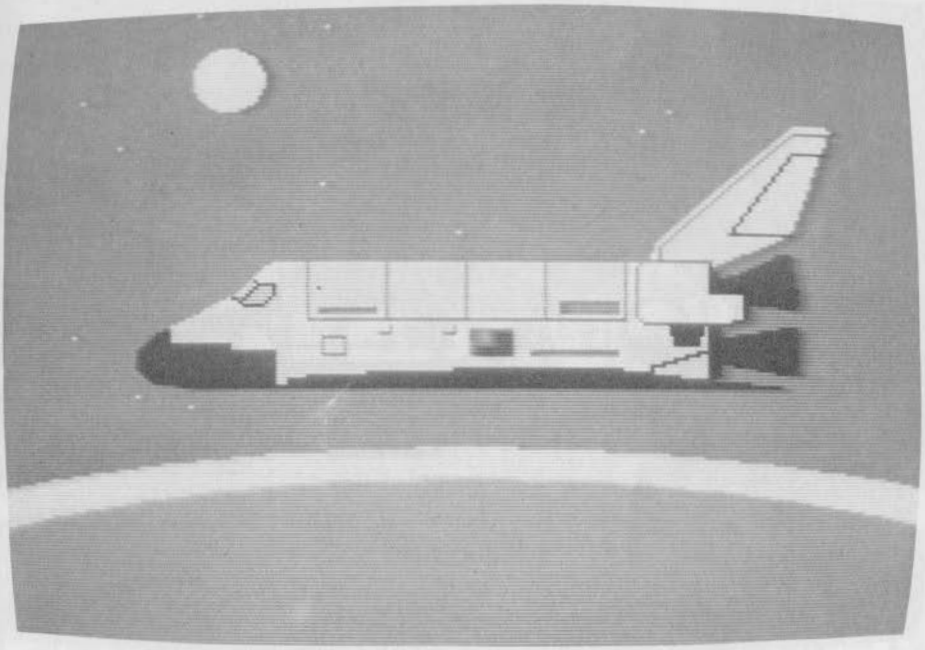


Fig. 5 El COLUMBIA en órbita terrestre

Ya que estamos en órbita, será el momento de que ejecutemos una de las funciones que este tipo de naves tienen asignadas, la recuperación de satélites artificiales. Hasta la aparición del COLUMBIA los satélites artificiales tenían una vida limitada y en caso de avería no había posibilidad de repararlos, ahora es posible recuperarlos por medio de un **brazo telemandado** que tiene en su interior, bajarlos a la tierra para su reacondicionamiento y devolverlos de nuevo al espacio.

Nosotros trataremos también de capturar nuestro satélite en nuestro programa, en la figura 5-bis está representada la imagen de esta parte.

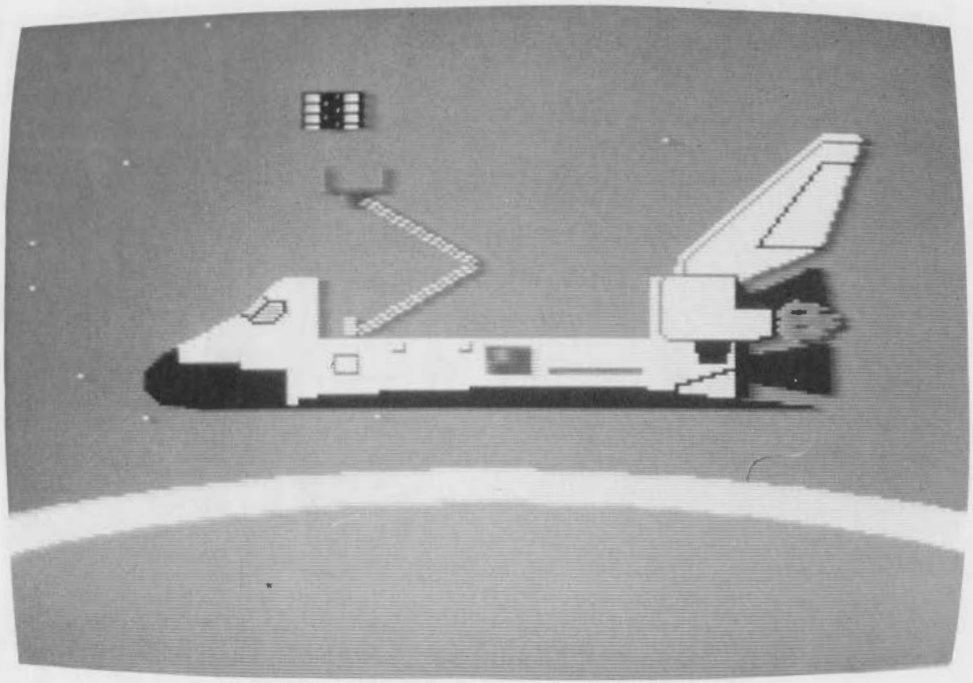


Fig. 5-bis Captura de Satélites

Nuestro segundo libro comenzará cuando hayamos decidido regresar, éste será el momento crucial de la misión (y el programa más complejo de los hasta ese momento realizados), el aterrizaje. Para este fin contaremos con un simulador de vuelo que será una reproducción de la cabina del COLUMBIA, en la parte correspondiente a los controles que tiene el comandante de vuelo según está representado en la figura número 6.

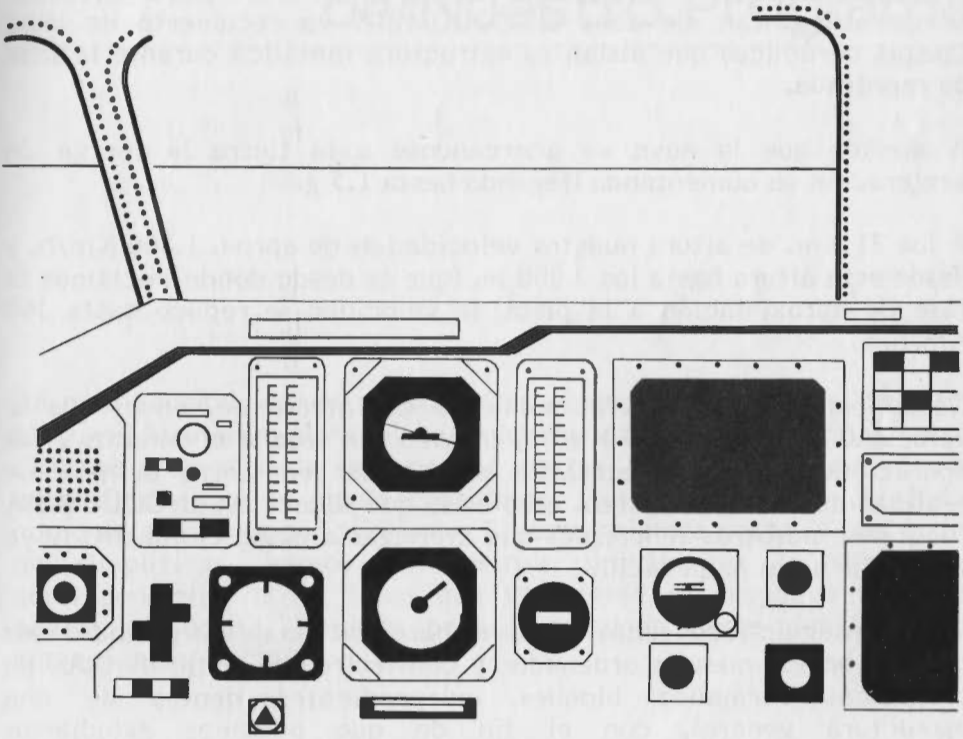


Fig. 6 Cabina de mando del COLUMBIA

Llegados a este punto hay que aclarar, para aquéllos que no lo sepan, que el COLUMBIA es un gigantesco **planeador** que pesa 68 toneladas sin carga y que puede transportar una carga máxima de 29.484 Kg. Este **planeador** aterriza a una velocidad de 350 Km/h. pero anteriormente ha tenido que pasar por una serie de situaciones extremas que únicamente una tecnología tan avanzada como la que te permite hacer uso de un ordenador como el que tienes, ha hecho posible superar.

Para empezar la **reentrada** en la atmósfera se produce aproximadamente a los 140 Km. de altura y a una velocidad de unos 28.000 Km/h., media hora más tarde la nave se encuentra a 130 Km. y el efecto de la fricción con la atmósfera empieza a sentirse calentando las superficies externas de la aeronave.

Estas temperaturas pueden llegar en algunos puntos, como por ejemplo el morro, a más de 1400º centígrados... Para poder aguantar temperaturas tan elevadas el COLUMBIA va recubierto de unas losetas cerámicas que aislan su estructura metálica durante la fase de reentrada.

A medida que la nave va acercándose a la tierra la fuerza de aceleración va aumentando llegando hasta 1.5 g.

A los 21 Km. de altura nuestra velocidad es de aprox. 1.800 Km/h. y desde esta altura hasta los 3.000 m. (que es desde donde iniciamos la fase de aproximación a la pista) la velocidad se reduce hasta 540 Km/h.

Desde este punto hasta el momento del aterrizaje, en el que la velocidad es de unos 350 Km/h., transcurren sólo tres minutos y las operaciones que son necesarias para poner en tierra la nave se realizan automáticamente... esto es lo que sucede en el COLUMBIA real, pero nosotros tendremos que aterrizar a **mano** el nuestro ¡y ya verás que no es nada fácil...!

Para conseguir todos estos objetivos haremos uso de las posibilidades que nos brinda nuestro ordenador y construiremos distintos tipos de programas formando bloques independientes dentro de una estructura general, con el fin de que podamos estudiarlos separadamente para una mejor comprensión de la función que cada uno debe cumplir.

Con todo lo hasta ahora descrito, tenemos ya una idea de cómo va a ser nuestro programa y como decíamos en la introducción de este libro, la finalidad que perseguimos es la de enseñar las particularidades del sistema MSX mediante la programación, así que una vez que ya sabemos de qué va el tema, vamos a poner manos a la obra y comenzar con nuestro programa.

Bloques Principales de Nuestro Programa

En esta sección vamos a establecer el **esqueleto** que los distintos bloques formarán dentro de nuestro programa principal, aquí diferenciaremos cada una de las partes que cumple una misión específica y que se puede considerar, vista de una forma aislada, como un programa por sí mismo. En esta primera parte, y con el fin de simplificar, vamos a construir únicamente la estructura correspondiente a las fases que van desde el despegue hasta la puesta en órbita, dejando para la segunda parte del libro las correspondientes al aterrizaje.

Lo que vamos a realizar es la fijación de los límites que van a ocupar cada uno de los bloques y dar la denominación a éstos con el fin de que podamos identificarlos por la misma. Para esto, lo primero que debemos hacer es una **carta de flujo** o 'flow chart' (como dicen los ingleses).

Esta **carta de flujo** es la que se representa de una forma simplificada en la figura número 7, veamos ahora el significado de cada uno de los bloques que conforman este programa.

INICIALIZACION

En todo programa que se precie -y éste queremos que sea uno de ellos- debe haber un primer apartado donde se realiza la **inicialización** de todas aquellas partes, como puede ser el espacio reservado en la memoria para el **almacenaje** de variables, cadenas, etc. También es en esta sección donde se elige el tipo de formato para la pantalla y se dan valores iniciales a ciertas variables aparte de otras cosas que veremos posteriormente.

Así que de momento vamos a poner el primer ladrillo de nuestro programa al cual llamaremos INICIALIZACION. No es mucho, pero ya hemos dado un paso hacia adelante.

INPUT

Después de este primer paso, vamos a seguir consolidando nuestra estructura, pues un ladrillo sólo no es suficiente, así que vamos otra vez a ver los objetivos que debíamos lograr. Bien, aquí tenemos uno que podemos poner después de la inicialización, le definiremos como el bloque de entrada al programa o sección de INPUT.

Esta parte nos permitirá comandar nuestra plataforma y lanzar el cohete desde lugar seguro, con éste ya son dos los ladrillos para el esqueleto de nuestro programa.

MOVIMIENTO

También debemos tener una sección que se encargue de direccionar el programa entre los distintos bloques, ésta es una subparte del programa principal que tiene una de las más importantes funciones, comparable a la de dirigir el tráfico en un cruce de muchas calles. En esta parte incorporaremos el sonido a nuestro cohete y haremos que éste se mueva.

A esta parte vamos a llamarla MOVIMIENTO.

GRAFICOS M-1

Siguiendo la secuencia por nosotros establecida, vamos a ver otro bloque, el que se refiere a la creación de las llamas y la nube de humo que generará el cohete en su despegue. A este bloque vamos a llamarle GRAFICOS M-1, la **m** es por lo de **móvil,** ya que éstos, a diferencia del paisaje, son gráficos que se desplazan. Como vemos, ya tenemos cuatro **ladrillos,** luego habrá que llenarlos de datos, pero esa será otra historia.

COHETE	PLATAFORMA

Bien, y ya para no romper el orden, vamos a continuar tomando el bloque en el cual se contempla la realización de los gráficos correspodientes a la plataforma y el COLUMBIA. Debido a que estos gráficos son los más laboriosos, a la vez que el alma de este programa, vamos a dividir este punto en dos, haciendo un bloque llamado COHETE y otro denominado PLATAFORMA. Bueno, ya vamos teniendo elementos suficientes para realizar una programación decente, pero debemos ejecutar los bloques que nos quedan pendientes.

GRAFICOS F-1

En este bloque es en el que se realizan los gráficos correspondientes a las nubes y la torre de lanzamiento, este bloque no tiene un valor funcional, sino meramente de adorno y realce de la presentación gráfica de este programa, dado su carácter de gráficos fijos (si bien, podríamos hacerlos móviles) denominaremos a este bloque GRAFICOS F-1, la f como habían adivinado es por el carácter **fijo** que tienen estas nubes, ya que en nuestras pantallas no sopla el viento... de momento.

PAISAJE

Este es el paisaje que nos servirá de marco para nuestros gráficos móviles, esta parte tiene la importancia de realzar y dar un toque de realismo a la imagen general, es un fondo sencillo con un programa corto que nos servirá como práctica para acometer los siguientes de más envergadura.

Creo que no hay duda para buscar una denominación a este bloque, la de PAISAJE le va como anillo al dedo.

SONIDO

En este bloque incorporaremos el sonido adecuado a las distintas fases por las que atravesará nuestra nave, desde la explosión de la salida hasta el rugido de los cohetes durante su ascensión, ésta es otra sección que contribuye de forma decisiva a aumentar el realismo que hayamos podido conseguir con nuestros gráficos y que nos permitirá hacer uso de las excelentes posibilidades de generación de sonidos que tiene nuestro ordenador.

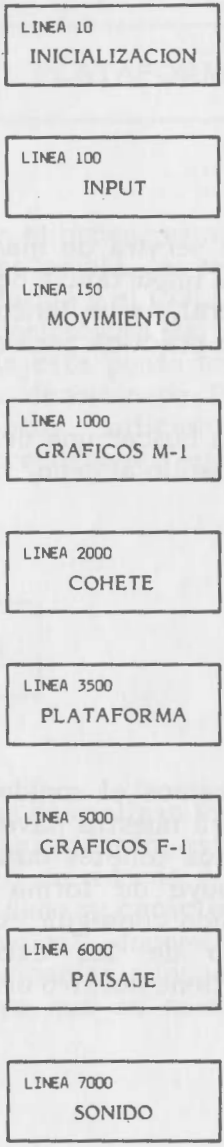


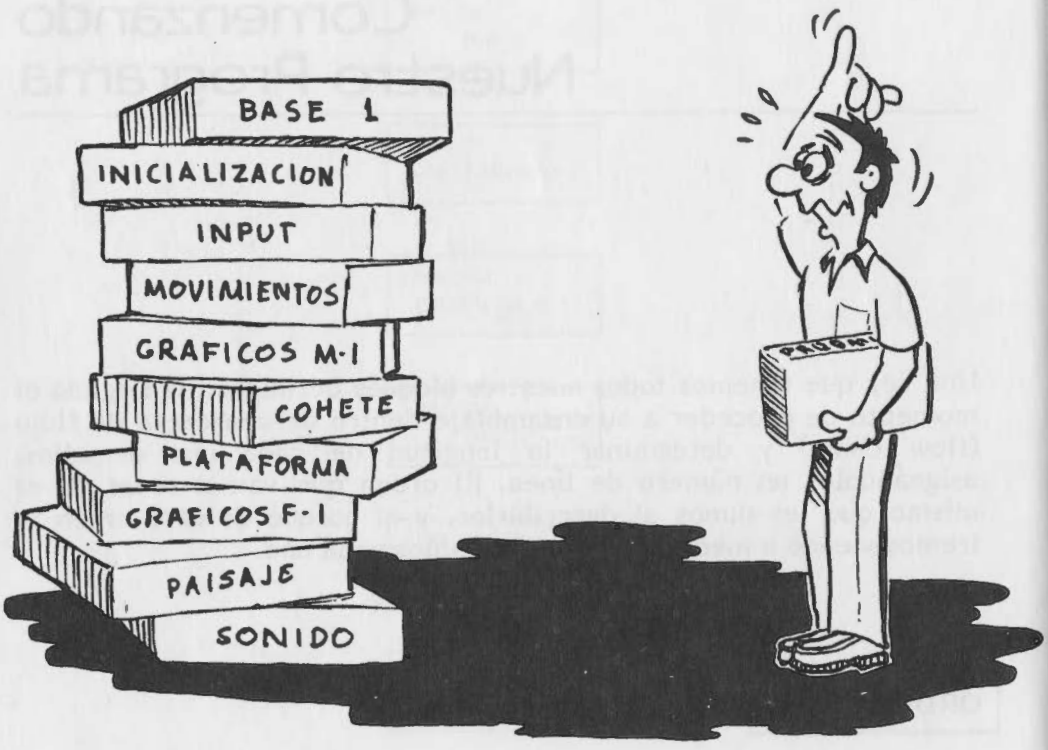
Fig. 7 Dibujo simplificado por bloques de nuestro programa

Comenzando Nuestro Programa

Una vez que tenemos todos nuestros **bloques** definidos, ha llegado el momento de proceder a su **ensamblaje** dentro de una carta de flujo (flow chart) y determinar la longitud de cada uno de ellos, asignándoles un número de línea. El orden que van a tener es el mismo que les dimos al describirlos, y el porqué de este orden lo iremos viendo a medida que construyamos cada uno.

ORDENACION

Ahora vamos a **construir** nuestro esqueleto para que quede como se ve en la figura nº 7. En esta figura podemos apreciar la numeración inicial de cada bloque, así como podemos deducir cuáles van a ser los de mayor número de líneas. La ventaja que representa el tener una configuración como ésta es la de que los bloques se pueden ordenar a nuestra conveniencia, pueden ser utilizados para otros programas, sabemos en cada momento dónde se ejecuta una función y sobre todo, nos permite construir nuestros programas de una forma más fácil y segura.


BASE 1
INICIALIZACION
INPUT
MOVIMIENTOS
GRAFICOS M-1
COHETE
PLATAFORMA
GRAFICOS F-1
PAISAJE
SONIDO

INFORMACIONES EN EL PROGRAMA: SENTENCIA REM

Es frecuente que después de un tiempo, y a veces incluso cuando lo estamos haciendo, nos perdamos en el laberinto de instrucciones que un programa significa. Esto es todavía más grave cuando es otra persona distinta a su realizador la que trata de interpretarlo. Para evitar en lo posible que esto suceda, es para lo que se emplea la sentencia **REM.** Esta sentencia no se **ejecuta** por el ordenador y su unica misión es añadir a nuestro programa comentarios que nos sirvan de aclaración en determinados puntos. Nosotros vamos a utilizar ahora esta sentencia **REM** para señalar cada uno de nuestros bloques. Prepara tu ordenador y escribe el siguiente programa:

```
1     REM BASE1
10    REM INICIALIZACION
100   REM INPUT
150   REM MOVIMIENTOS
1000  REM GRAFICOS M-1
2000  REM COHETE
3500  REM PLATAFORMA
5000  REM GRAFICOS F-1
6000  REM PAISAJE
7000  REM SONIDO
```

GRABACION DE PROGRAMAS

Hay algo importante que vamos a realizar como norma a partir de este momento. Cada vez que grabes una parte del programa **no rebobines la cinta hasta el inicio, únicamente rebobina la última grabación.**

El motivo de seguir este sistema es el de no perder todo nuestro trabajo por un error, creo que a todos nos ha pasado alguna vez el creer que el magnetofón estaba reproduciendo y lo que hacía era todo lo contrario...

Ese despiste no se subsana empleando el sistema de rebobinar parcialmente, pero sí que hace que la **catástrofe** sea igualmente parcial. Para aquéllos que tengan contador, lo que deben hacer es ponerle a cero al inicio de la grabación, grabar y dejar en la posición que quede el contador. Cuando haya que **cargar** el programa (que todavía está en fase de construcción) se retrocederá hasta el mismo punto donde empezamos, es decir, el cero o mejor un poco más adelante procediendo a la carga. Una vez realizada ésta, pondremos de nuevo a cero el contador, ya que éste será el nuevo punto de partida para nuestra próxima grabación. Así hasta que acabemos totalmente el programa, que será el momento en que rebobinaremos para grabarlo completo al inicio de la cinta.

VERIFICACION

Otra advertencia importante es la de no olvidarse de comprobar que la grabación fue correcta, utilizando la instrucción CLOAD? "nombre programa".

Otro truco que se puede emplear para la identificación de cada uno de los **subprogramas** que grabemos, es el de ponerles un número distinto al final del nombre, por ejemplo, y en el caso presente BASE1, BASE2, BASE3, etc.

Con este sistema podemos encontrar una sección determinada si el contador se nos va de vueltas y puede servir para aquéllos que no poseen éste en su magnetofón.

Inicialización

Este va a ser nuestro primer bloque y como ya dijimos anteriormente, vamos a preparar o **inicializar** nuestro ordenador con todas aquellas instrucciones que sean necesarias para este programa. El que vamos a construir es de los denominados **gráficos**, en el cual hay una serie de dibujos fijos y otros que son **móviles** (SPRITES), generalmente tanto los unos como los otros deben construirse previamente y estar disponibles para ser utilizados por el programa.

SUBRUTINAS

La construcción de estos dibujos se hace mediante subrutinas que son **accedidas normalmente** al principio del programa, en estas subrutinas se realizan los dibujos de fondo y también se **definen** los dibujos móviles o SPRITES, a éstos se les asigna un número al que basta posteriormente **llamar** para tener el dibujo deseado. Esta llamada ya no se realiza a la subrutina, sino que es como otra variable más de las que normalmente usamos en el programa.

El acceso a esta subrutina para confección de gráficos es, por lo tanto, una de las labores que debemos hacer en esta sección. En nuestro ordenador disponemos de varias modalidades de pantalla que podemos controlar por medio de la instrucción SCREEN, dado el carácter gráfico de nuestro programa, debemos emplear el **modo 1** que es el que nos permite mayor definición (en el capítulo siguiente se explican los modos gráficos). Este modo se consigue con SCREEN 2 y ésta es otra de las inicializaciones que debemos hacer.

Otra de las órdenes que debemos dar previamente es la que se refiere al color, en el caso que nos concierne debemos definir qué color queremos para nuestro cielo y con cuál vamos a **enmarcar** nuestra pantalla. Esta instrucción puede ser usada en cualquier momento, pero es necesario darla al principio. Ya, por último (en este caso), debemos indicar qué subrutinas y en qué orden deben ser accedidas para contar con todos los elementos precisos para el funcionamiento del programa.

GOSUB

Esto lo conseguimos por medio de la instrucción GOSUB (GO = **ir** en inglés y SUB de **subrutina**). Con esto ya tenemos los datos necesarios para poder hacer nuestro primer ladrillo.

En el encabezamiento anterior habíamos puesto el nombre del programa en la línea número 1, vamos a eliminar esa línea y hacer que el nombre forme parte de la inicialización. Conecta tu ordenador y carga el programa que hicimos anteriormente, escribe **CSAVE** "BASE1" y pulsa **RETURN.**

Una vez que tienes el programa cargado, escribe el número '1' y pulsa RETURN; con esto ya hemos borrado la línea 1, como podrás apreciar si **listas** el programa con la tecla LIST. Verás que la línea 1 ha desaparecido.

Ya tenemos todo dispuesto para empezar nuestra programación, disponte a teclear las siguientes líneas:

```
10 REM BASE2
20 CLS
30 SCREEN 2
40 COLOR 1,5,1
50 GOSUB 6000
60 GOSUB 5000
70 GOSUB 3500
80 GOSUB 2000
90 GOSUB 1000
```

En la línea 10 ponemos el nombre del programa, en este caso BASE2 en lugar de 1 para distinguirle del anterior.

```
CLS
```

En la línea 20 utilizamos una nueva instrucción **CLS** que se emplea para **limpiar**, cuando se comienza un programa siempre tenemos en la pantalla instrucciones y comandos que conviene eliminar y esto lo conseguimos con esta instrucción que **borra** cualquier texto o imagen que tengamos (CLS viene de CLear Screen = Limpiar la Pantalla).

Posteriormente, en la línea 30, tenemos la instrucción SCREEN 2 con la que disponemos nuestro ordenador para gráficos y en las siguientes líneas (50-90), tenemos una serie de instrucciones GOSUB que nos envían a los distintos bloques o subrutinas.

Como vemos, el envío es en orden descendente, empezamos por el bloque 6000 para finalizar en 1000 (GRAFICOS M-1), a partir de este punto, una vez que todas las subrutinas han sido accedidas, el programa abandona el bloque de **INICIALIZACION** para ir al de **INPUT** (línea 100).

Con la parte que acabamos de realizar nuestro dibujo simplificado de la Fig. 7 ha quedado convertido en el representado en la Fig. 8, en el cual apreciamos cómo el **enlace** de los bloques ya no es correlativo y su acceso es independiente de su posición relativa en el programa.

Para finalizar esta sección prepara tu grabadora, NO REBOBINES de la posición en que quedó cuando cargaste el programa y teclea lo siguiente:

```
CSAVE "BASE2"
```

Una vez grabado, comprueba que la grabación es correcta, teclea:

```
CLOAD?"BASE2"
```

Si la respuesta es OK ¡perfecto!, en caso contrario revisa tu cassette.

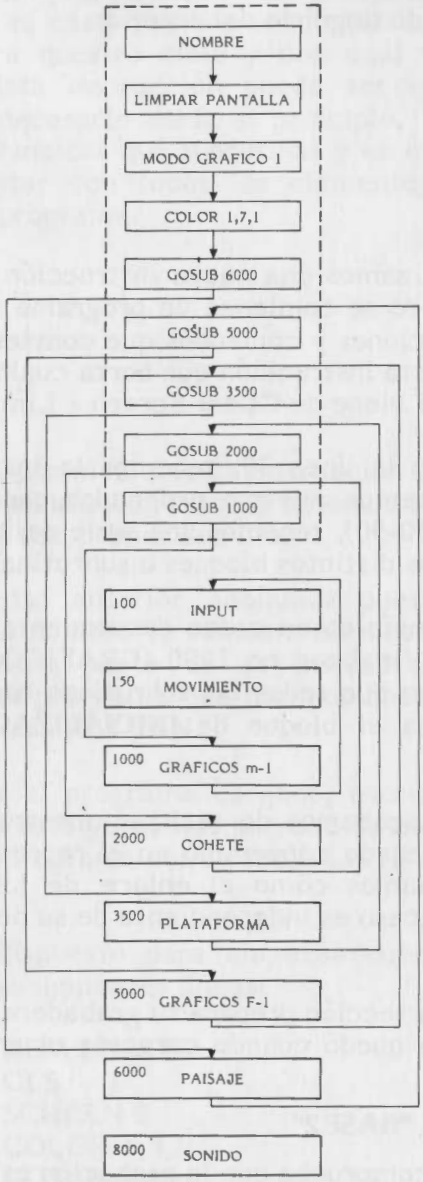


Fig. 8 Carta de flujo (flow chart) del programa

Gráficos Paisaje

Esta es la parte a la que normalmente no se le dedica una gran atención, teniendo sin embargo, mucha importancia en la impresión general que un programa puede causarnos.

RESOLUCION GRAFICA: MATRIZ DE PANTALLA

Para realizar esta sección es necesario que conozcamos antes que nada cómo está **organizada** nuestra pantalla. Una de las características que definen en cualquier ordenador su capacidad para representar gráficos, es el número de puntos de imagen o **pixels** que éste es capaz de producir vertical y horizontalmente. Así en nuestro caso, tenemos 256 puntos o pixels horizontales por 192 verticales.

Con esta matriz formada por las líneas de 256 pixels y las columnas de 192 obtenemos un total de 49152 puntos independientes en nuestra pantalla, dada la dificultad para representar gráficamente todos estos puntos vamos a dividirla en bloques de 8 x 8, es decir, la misma matriz empleada para generar cada uno de los caracteres que vemos al escribir en el ordenador. Con esto obtendremos un tablero como el representado en la figura 9.

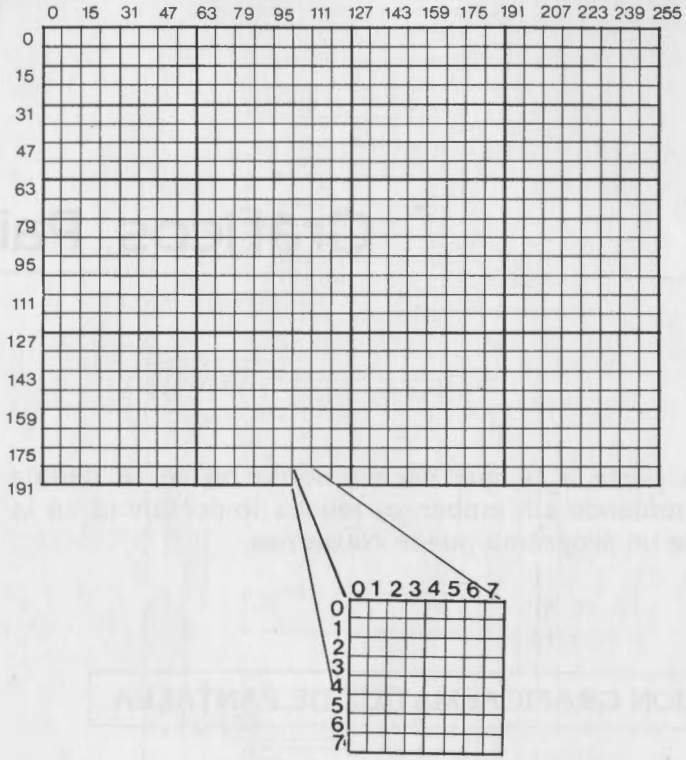


Fig. 9 Matriz de la pantalla y ampliación de uno de sus bloques

LINE

Una vez que tenemos nuestra cuadrícula, vamos a situar en esta línea del horizonte, si nos fijamos en el dibujo nº 3 veremos que está situada aproximadamente a la altura de la cuarta línea inferior.

Dado que la numeración de las líneas horizontales comienza por la parte superior (0), es la última línea inferior la que hace el nº 191 (desde 0 al 191=192 posiciones), teniendo en cuenta que nuestro tablero está compuesto de cuadrículas de 8 x 8 pixels para situar el horizonte en la cuarta línea debemos restar a 191 (líneas final) el producto de 4 x 8, lo que le situaría en la línea 159, pero para hacer el número redondo vamos a ponerle en la 160 de la coordenada vertical. La longitud que debe tener es la del ancho de la pantalla, es decir, desde el punto 0 al 255 (256 posiciones).

Con eso ya tenemos nuestras coordenadas (vertical y horizontal), ahora lo que nos falta es la instrucción adecuada para hacer el dibujo. Esta instrucción es LINE (línea en inglés) y con ella vamos a escribir lo siguiente:

```
6010 LINE (0,160)-(255,191),3,BF
```

Así hemos indicado al ordenador lo siguiente: que trace una línea desde la intersección X-0 (horizontal), Y-160 (vertical), hasta la intersección X-255, Y-191; que esta línea sea de color verde claro, el cual corresponde al nº 3 según la tabla de la figura 10.

0	Transparente	8	Rojo
1	Negro	9	Rojo claro
2	Verde	10	Amarillo oscuro
3	Verde claro	11	Amarillo claro
4	Azul oscuro	12	Verde oscuro
5	Azul claro	13	Magenta
6	Rojo oscuro	14	Gris
7	Ciano	15	Blanco

Fig. 10 Tabla de colores

Por medio de la letra B conseguimos que se trace un rectángulo cuyos vértices opuestos tienen la longitud de la línea indicada en la instrucción, y por último con la letra F logramos que este rectángulo sea **rellenado** (fill=llenar en inglés) con el color seleccionado. Fijándonos veremos que la línea no es horizontal, sino inclinada con una diferencia de altura de 191-160=31 puntos de imagen o pixels, ésta será la altura que tendrá el rectángulo y por lo tanto, nuestro campo.

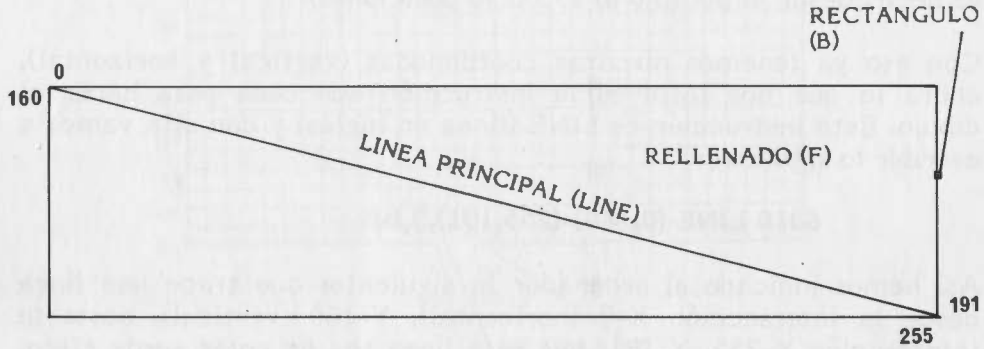


Fig. 11 Ejecución de la instrucción LINE

La realización gráfica de nuestra instrucción está representada en la figura 11. Una vez que el campo está realizado vamos a situar sobre éste el camino para la plataforma mediante las siguientes instrucciones:

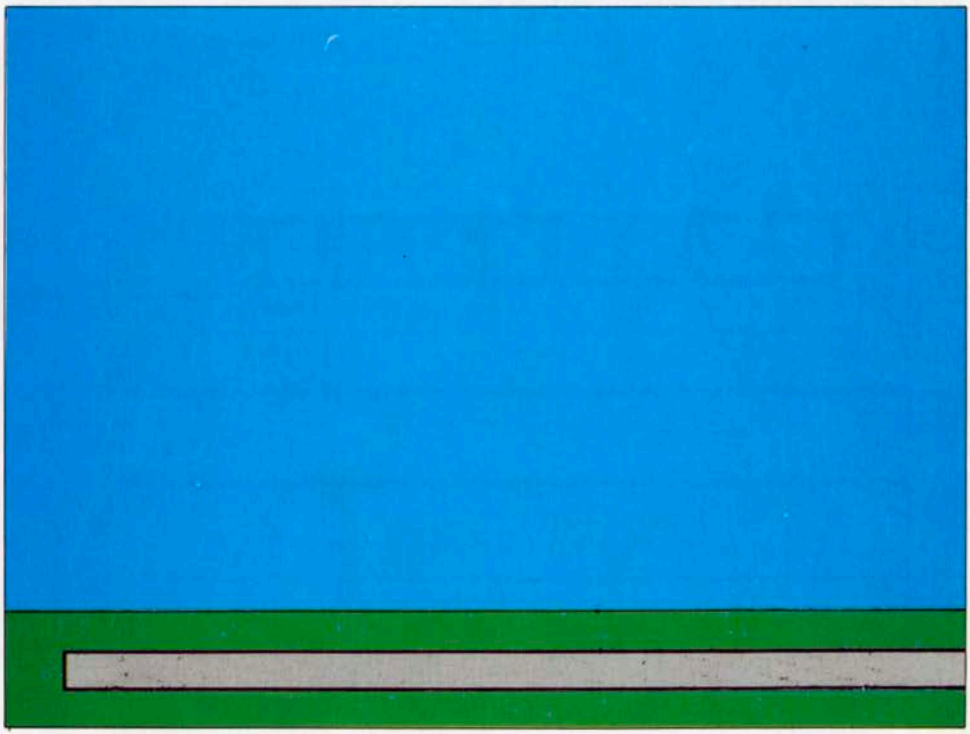
```
6020 LINE (15,171)-(255,181),1,B
6030 LINE (16,172)-(255,180),14,BF
```

Aquí vemos que se ha empleado también la letra B, ya que son rectángulos lo que queremos trazar, si bien de menor altura y comenzando separados del margen izquierdo 15 pixels. También vemos que en la línea 6020 la B no es seguida por la F, esto es debido a que esta instrucción es empleada para hacer el borde del camino empleando el color negro (1), mientras que en la línea 6030 sí que es empleada la F, ya que se trata de **rellenar** ese borde, por eso también el rectángulo que se dibuja en esta instrucción es un pixel más estrecho en cada uno de sus lados con el fin de que quede dentro del margen marcado por el borde.

El color empleado para el camino es el gris (14). Si quieres ver cómo queda el campo con el camino, teclea la siguiente línea y pulsa luego RUN.

```
6040 GOTO 6040
```

Para parar el programa pulsa CTRL y STOP simultáneamente.



Edificio de Control

Ahora le toca el turno al edificio de control que aparecía al fondo del campo, para construirle utilizaremos también la instrucción LINE pero primero vamos a fijarnos en la figura 12. En ella hemos ampliado la parte de la cuadrícula donde está situado el edificio, como vemos, esta sección comprende los puntos X-160 hasta el X-255 en el eje horizontal, y el Y-136 al Y-191 en el vertical.

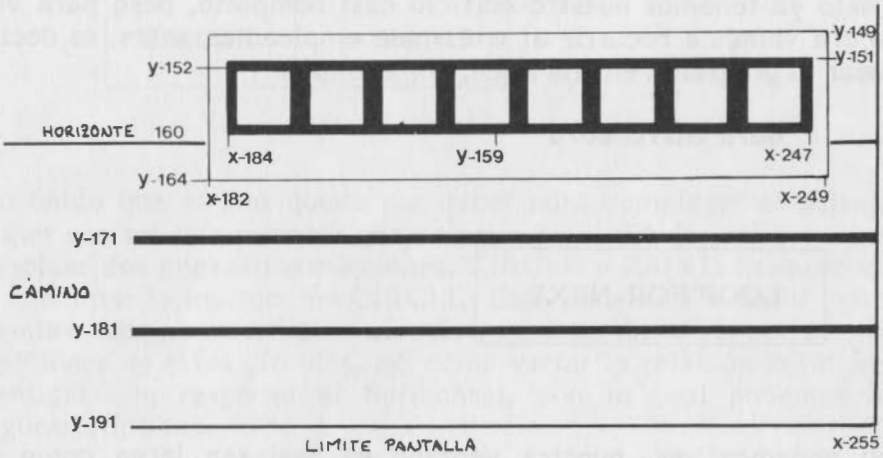


Fig. 12 Situación del Edificio de Control

Ya vemos en la figura cuál es la situación del edificio, así como los colores que vamos a emplear en su confección. Primero haremos la parte inferior, así que teclea la siguiente línea:

6040 LINE (182,152)-(249,164),10,BF

Ahora realizaremos el tejado, dado que el color elegido para las paredes según la tabla nº 1 es el amarillo oscuro (10), en el tejado emplearemos el rojo (6). En ambas instrucciones empleamos BF, ya que se trata de construir rectángulos **llenos.** Hagamos ahora el tejado por medio de la siguiente línea:

```
6050 LINE (182,149)-(249,151),6,BF
```

Bien, este es el momento de hacer las ventanas, así que escribe la siguiente instrucción:

```
6060 LINE (184,152)-(247,159),5,BF
```

Con esto ya tenemos nuestro edificio casi completo, pero para ver cómo era vamos a recurrir al truco que empleamos antes, es decir, **bloquear** el programa en una línea, por ejemplo:

```
6070 GOTO 6070
```

LOOP FOR-NEXT.

Como podemos ver, nuestra ventana es casi tan larga como el edificio y además carece de marco, lo cual puede ser muy peligroso con un cristal de ese tamaño. Vamos a solucionar este problema con un pequeño truco y nuestra instrucción LINE.

```
6070 FOR V=184 TO 240 STEP 8
6080 LINE (V,152)-(V+7,159),1,B
6090 NEXT V
```

El truco consiste en hacer un bucle o **loop**(V) que empieza en el punto X-184 y avanza hasta el X-240 en **saltos** o **steps** de 8 haciendo en cada uno de estos saltos un rectángulo de 7 x 8 de color negro. Podemos ver de nuevo cómo ha cambiado nuestro edificio con la inclusión de este detalle, tecleando:

```
6100 GOTO 6100
```

Supongo que recordarás que para parar hay que apretar STOP y CTRL al mismo tiempo...

CIRCLE	PAINT

Lo único que ya nos queda por hacer para completar el paisaje, es poner ese sol de amanecer que veíamos en el dibujo, así que vamos a emplear dos nuevas instrucciones, CIRCLE y PAINT. Primero vamos a examinar la instrucción CIRCLE. Esta instrucción como indica su nombre, nos permite hacer círculos, pero también nos permite hacer secciones de estos círculos, así como variar la relación del diámetro vertical con respecto al horizontal, con lo cual podemos hacer figuras elípticas.

Los parámetros que debemos suministrar a esta instrucción son los siguientes: Coordenadas X/Y para determinar el centro, longitud del radio, color con el que vamos a dibujarle, **ángulos** de comienzo y finalización (en el caso de que queramos hacer únicamente un sector) y relación entre diámetros.

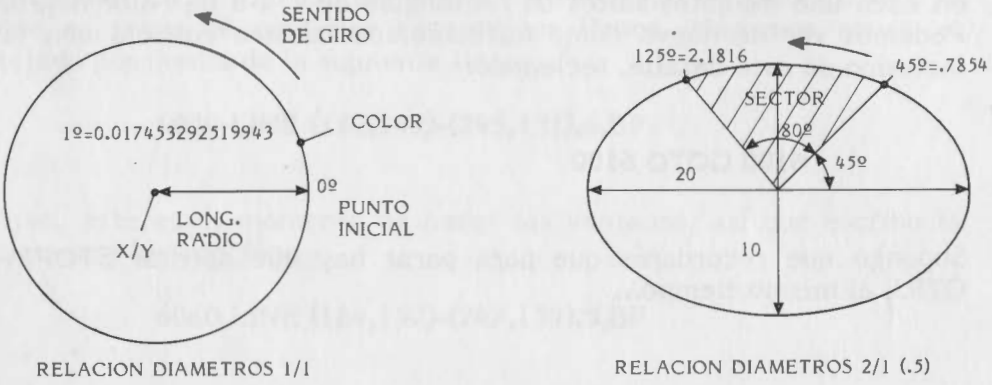


Fig. 13 Parámetros de la instrucción CIRCLE

Una de las características más importantes que tiene esta instrucción es la de poder hacer sectores de círculo con un ángulo determinado, para determinar este ángulo debemos saber cuál es el punto 0 y qué unidad debemos emplear para indicar el ángulo.

En la Fig. 13 tenemos la contestación a estas preguntas, el punto 0 está en el lado derecho de la circunferencia y el valor del círculo viene dado en radianes, por lo tanto, cada grado vale (3.1416 x 2)/360 = .017453292519943, o simplificando .0174533, es decir, que si queremos hacer un sector de 30 grados, éste debe valer 30 x .0174533 = .523599 ó .5239, dependiendo de la exactitud que busquemos. En el caso de que este sector tuviera un punto inicial distinto a 0º habría que sumar tanto al punto inicial como al final el valor del ángulo entre 0º y el inicio del sector.

Como ejemplo, haz este pequeño programa en el cual un sector de 80 grados comienza con un ángulo de 45º respecto al punto 0. Este ejemplo está reflejado en la Fig. 13 **sector.**

```
9000 SCREEN 2
9010 CIRCLE (122,80),30,1,-.7854,-2.1816
9020 GOTO 9020
```

Para poner en marcha el programa escribe GOTO 9000, y para pararle haz como has hecho hasta ahora. Observarás que han aparecido dos radios, esto es debido al signo '-' que hay delante de cada uno de los valores correspondientes a los grados. Puedes probar a suprimirlos por medio del editor con que cuenta tu ordenador dejando la línea 9010 de esta forma.

```
9010 CIRCLE (122,80),30,1,.7854,2.1816
```

Como verás, los radios han desaparecido. Una vez que ya sabemos cómo funciona la instrucción CIRCLE vamos a construir nuestro sol con ella, borra las líneas 9000, 9010 y 9020, tecleando su número y pulsando RETURN (una a una), o bien escribe DELETE 9000-9020. Ahora hagamos las siguientes líneas de programa.

```
6100 CIRCLE (75,159),20,11,-.001,-3.1416,.9
6110 PAINT (80,157),11
6120 CIRCLE (75,159),12,6,-.001,-3.1416,.9
6130 PAINT (80,157),6
```

Con la línea 6100 damos las coordenadas X-75 Y-159, situando el centro del círculo por encima de la línea de horizonte (160) y a la izquierda de nuestro dibujo, el radio que le adjudicamos es de 20 pixels, el color con que le trazamos es el amarillo claro (11). El ángulo elegido para iniciar el círculo ha sido el 0. Por este motivo usamos .001 que es inferior al valor que tiene un grado en la instrucción, y por lo tanto, no es reproducible en la pantalla con un radio como el que estamos empleando (20). El final del trazado es de 180 grados, o sea, 3.1416 (pi). Con estos parámetros trazaremos un semicírculo que empezará y terminará sobre el horizonte, pero para que se parezca más al sol que vemos al amanecer vamos a darle una relación de 10/9 (.9), es decir, el radio valdrá 18 en la vertical (20/10x9) y 20 en la horizontal.

PAINT

Después de ejecutar la instrucción CIRCLE pasamos a la línea 6110, donde utilizamos por primera vez la instrucción PAINT. Esta instrucción se utiliza para rellenar de color una zona **cerrada** de un dibujo determinado, en nuestro caso la emplearemos para rellenar el semicírculo.

Hay también otra circunstancia que debemos tener en cuenta cuando trabajemos con la instrucción PAINT, ésta es el **modo** gráfico que estamos utilizando.

SCREEN.

En nuestro ordenador disponemos de dos modos gráficos, el 1 correspondiente a la instrucción SCREEN 2 y el 2 correspondiente a la instrucción SCREEN 3.

RESOLUCION GRAFICA Y COLOR

Estos dos modos (1 y 2) son los que empleamos para trabajar con gráficos y la diferencia entre ambos está en su **resolución,** con el modo 1 (SCREEN 2) tenemos 256 x 192 pixels y 16 colores a nuestra disposición, pero a la hora de utilizar los colores debemos referirnos a la Fig. 9, donde representamos nuestra matriz dividida en bloques de 8 x 8, pues bien dentro de cada una de las **líneas horizontales** de estos bloques (8 x 8), no puede haber más de dos colores distintos. Por lo tanto, dado que tenemos 8 líneas, el número de colores que podemos tener en un bloque dos a dos es el de dieciséis, la distribución de los colores dentro de la línea es completamente arbitraria pudiendo ocupar cualquier número de puntos siempre que no coincida con el otro color, un ejemplo de esta distribución está representado en la figura 14. En el caso de emplear el modo 2 (SCREEN 3) la resolución de que disponemos es de 64 x 48 **grupos** de **4 x 4 puntos de imagen.** Lo que se hace en este caso es dividir nuestros bloques de 8 x 8 en bloques de 4 x 4 (Fig. 14a).

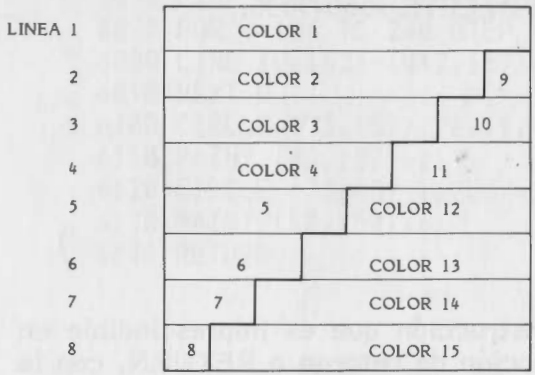
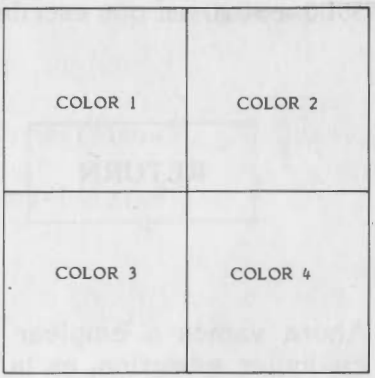


Fig. 14



EL Nº DE LOS COLORES
SE REFIERE UNICAMENTE
A LAS POSIBILIDADES

Fig. 14a

En el modo 2 (SCREEN 3) únicamente podemos disponer de un color por cada bloque (4 x 4) coloreándole totalmente sin que exista posibilidad de hacerlo parcialmente. La influencia que el modo elegido tiene sobre algunas instrucciones gráficas afecta características distintas a las de su resolución, este es el caso de la instrucción PAINT que nos ocupaba anteriormente. En la línea 6110 vemos que el color seleccionado es el mismo (11), que el empleado en la línea 6100, esto es porque cuando estamos en modo 1 (SCREEN 2) el perímetro del dibujo a rellenar y el color empleado en PAINT deben ser iguales. En el caso de emplear el modo 2 (SCREEN 3), el color de **relleno** puede ser distinto al de la línea de cierre, para ver esta posibilidad hagamos de nuevo otro pequeño programa. Escribe las líneas siguientes:

```
9000 SCREEN 3
9010 CIRCLE (125,94),55,1
9020 PAINT (125,94),11, 1
9030 GOTO 9030
```

Arranca este programa escribiendo GOTO 9000 y pulsando RETURN. Bien, ya tenemos este bloque terminado, sólo queda borrar las líneas 9000 a 9030 así que escribiremos DELETE 9000-9030.

RETURN

Ahora vamos a emplear una instrucción que es imprescindible en cualquier subrutina, es la instrucción de **retorno** o RETURN, con la cual volvemos al lugar del programa desde donde accedimos a ella. Pongamos esta instrucción en nuestro bloque.

```
6140 RETURN
```

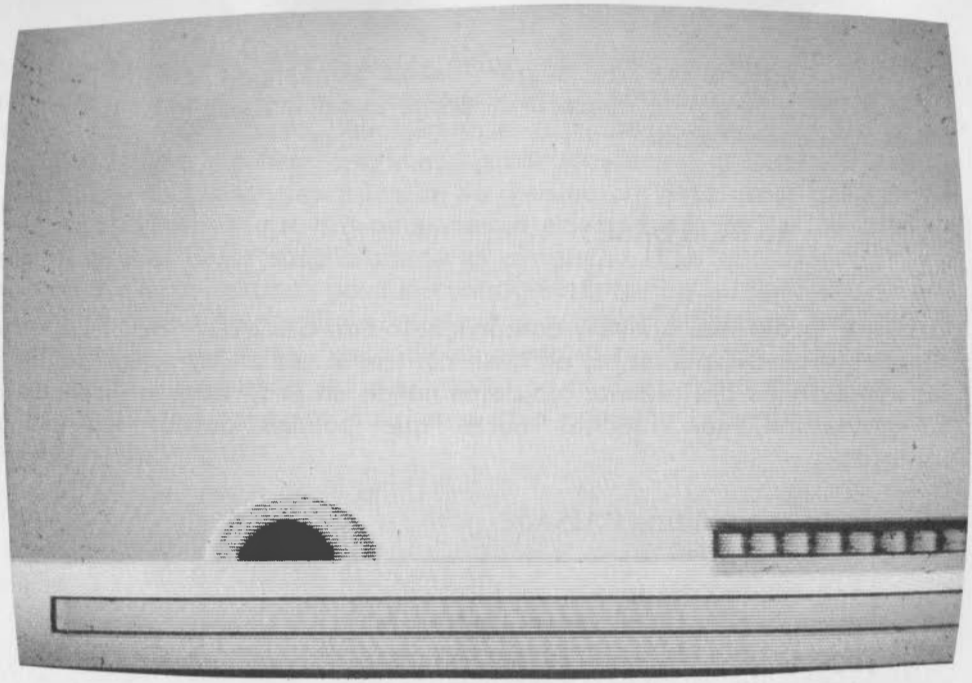
Recuerda que antes de seguir debes grabar el programa. NO REBOBINES de donde copiaste la parte anterior, cambia la línea 10 de la siguiente forma:

10 REM BASE3

Graba el programa ahora y compruébalo con CLOAD? "BASE3", si la respuesta es OK perfecto, en caso contrario revisa tu magnetofón y las conexiones del mismo. No dejes nunca un programa sin chequear su grabación, son muchas horas que puedes perder en pocos minutos...

Listado de la subrutina PAISAJE

```
6000 REM PAISAJE
6010 LINE (0,160)-(255,191),3,BF
6020 LINE (15,171)-(255,181),1,B
6030 LINE (16,172)-(255,180),14,BF
6040 LINE (182,152)-(249,164),10,BF
6050 LINE (182,149)-(249,151),6,BF
6060 LINE (184,152)-(247,159),5,BF
6070 FOR V=184 TO 240 STEP 8
6080 LINE (V,152)-(V+7,159),1,B
6090 NEXT V
6100 CIRCLE (75,159),20,11,-1E-05,-3.1416,.9
6110 PAINT (80,157),11
6120 CIRCLE (75,159),12,6,-1E-05,-3.1416,.9
6130 PAINT (80,157),6
6140 RETURN
```



Gráficos F-1

DRAW

Los gráficos que vamos a realizar ahora no son de gran complejidad, puesto que se trata de las nubes que adornan nuestro paisaje. Pero vamos a aprovechar su confección para utilizar una de las más potentes instrucciones gráficas que poseemos en nuestro ordenador MSX. Esta instrucción es DRAW (dibujo, en inglés) y realmente es un **macro-comando gráfico.** Hasta ahora hemos visto los parámetros que son necesarios para hacer una línea o un rectángulo con la instrucción LINE, también hemos visto los que se utilizan en la instrucción CIRCLE. En ambas instrucciones cada vez que necesitamos hacer uso de ellas tenemos que dar de nuevo todos y cada uno de los parámetros que necesitan.

Con la instrucción DRAW se hace un dibujo una vez y se le asigna a una constante **alfanumérica,** posteriormente sólo es necesario llamar a esta constante para que el dibujo sea realizado de nuevo, independientemente de su complejidad (y puede ser muy complicado). Esta es la parte **cómoda** para nosotros, pero esta instrucción tiene muchas cosas más.

ANGULOS

A diferencia de las coordenadas que necesariamente debemos suministrar cuando utilizamos LINE y donde tienen que estar indicados los puntos de comienzo y final de la línea, la instrucción DRAW posee una serie de ángulos fijos para su desplazamiento, los cuales corresponden cada uno a una letra, bastando mencionar ésta para que el ángulo quede seleccionado. Al mismo tiempo (después de la letra definitoria) damos la **longitud** del desplazamiento que deseamos sea realizado en la dirección señalada por el ángulo elegido. Esto puede parecer algo confuso, así que vamos a fijarnos en la Fig. 15 donde se han representado los ángulos y sus respectivas letras utilizadas en la instrucción DRAW.

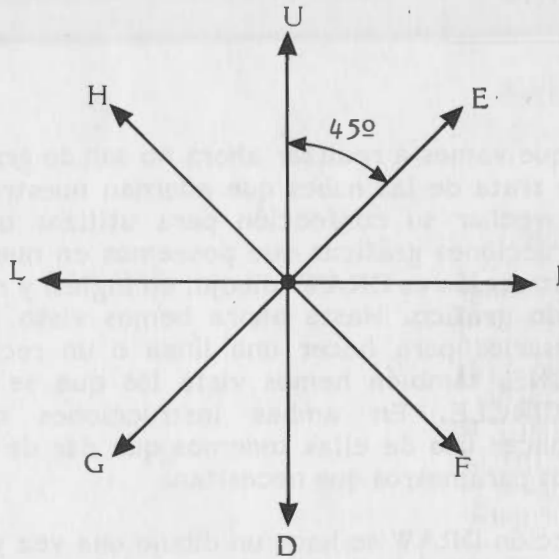


Fig. 15 Angulos de desplazamiento de la instrucción DRAW

En esta figura vemos un total de ocho posibles direcciones o ángulos, cada uno de ellos está separado 45 grados de sus adyacentes y estas direcciones son fijas, es decir, no podemos tener un ángulo de 60 grados debiendo tomar 45 ó 90. A pesar de esta limitación de la instrucción DRAW, sigue siendo de extraordinaria potencia gráfica como veremos.

Volviendo a la figura anterior, vemos en ella las letras que indican cada uno de los ángulos posibles, las letras correspondientes a los ángulos verticales y horizontales tienen una clara relación con la palabra inglesa que indica cada una de sus direcciones. Así U viene de **upper** (arriba), D viene de **down** (abajo), L viene de **left** (izquierda) y R de **right** (derecha), como vemos, la correspondencia es clara en estas letras, sin embargo, no sucede lo mismo con las otras cuatro, así que vamos a servirnos de un truco para poder memorizar sus posiciones y significados. Vemos que las letras que son la inicial de la palabra que indica su dirección en inglés, no tienen ningún orden alfabético ya que lo que se ha buscado es su significado. Sin embargo, las otras cuatro letras sí que están puestas en ese orden, E F G H, con lo cual si recordamos que la primera letra, la E, comienza a 45 grados de la U y que cada una de las siguientes (F G H) están situadas a 90 grados entre sí, ya tenemos una regla para memorizar cada una de sus posiciones.

En la figura 16 tenemos una ampliación de lo explicado hasta ahora. En esta figura hemos separado en dos partes el dibujo de la figura 15 con el fin de facilitar su memorización.

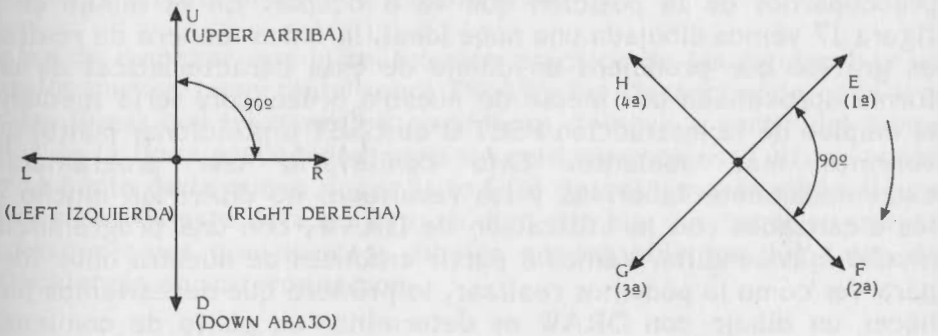


Fig. 16 Correspondencia entre dirección y letra en la instrucción DRAW

Con lo explicado hasta ahora ya tenemos una noción de cómo podemos realizar dibujos utilizando la instrucción DRAW. Vamos ahora a realizar por medio de esta instrucción un gráfico que nos sirva de ejemplo.

El gráfico elegido es una de nuestras nubes y para su confección utilizaremos una plantilla de papel milimetrado como la representada en la figura 17 (estas hojas de papel se pueden adquirir en cualquier papelería y son muy útiles para la creación de gráficos).

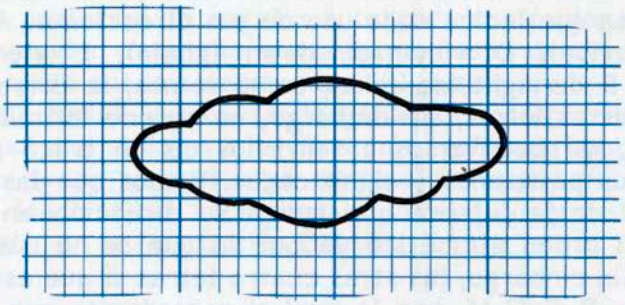


Fig. 17 Plantilla milimetrada para dibujo de gráficos

Cada una de las pequeñas cuadrículas de esta plantilla corresponde a uno de nuestros puntos de imagen o pixels y de momento, a diferencia de lo que sucede con LINE o CIRCLE, no necesitamos saber su posición relativa en nuestra pantalla.

Esta es otra de las ventajas de DRAW, el poder realizar el dibujo sin preocuparnos de la posición que va a ocupar. En el dibujo de la figura 17 vemos dibujada una nube **ideal**, la única manera de realizar un gráfico que produjera un dibujo de esas características de una forma aproximada por medio de nuestro ordenador, sería mediante el empleo de la instrucción **PSET** (Point **SET** o posicionar punto) que veremos más adelante. Esto conllevaría una programación extremadamente laboriosa y los resultados no diferirían mucho de los alcanzados con la utilización de DRAW, con una programación mucho más sencilla. Vamos a partir entonces de nuestra nube ideal para ver cómo la podemos realizar, lo primero que necesitamos para hacer un dibujo con DRAW es determinar un punto de comienzo, este punto es relativo al dibujo y nos servirá posteriormente para situar éste en la posición que elijamos dentro de nuestra pantalla.

El punto de partida va a ser el señalado en el margen izquierdo de la figura 18, donde hemos ampliado nuestra cuadrícula al fin de poder poner nuestras indicaciones en el dibujo.

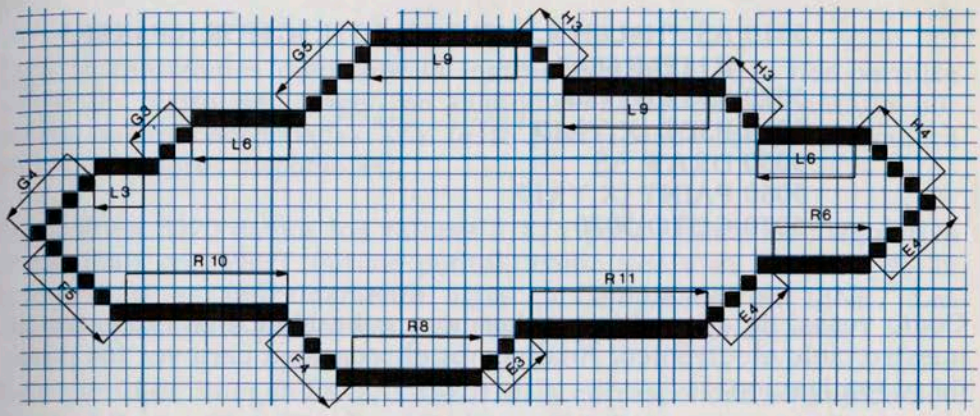


Fig. 18 Dibujo de la nube mediante la instrucción DRAW

Desde ese punto trazamos una línea de 4 puntos en la dirección H (H4), seguimos con otra de 6 puntos en la dirección L (L6), continuamos con otra de 3 puntos de nuevo en la dirección H (H3), después L9 H3 L9 G5 L6 G3 L3 G4 F5 R10 F4 R8 E3 R11 E4 R6 para finalizar con E4 en el punto donde iniciamos el trazado. La forma con que daríamos estos parámetros a la instrucción DRAW es prácticamente la misma empleada en esta explicación. H4L6H3L9H3 L9G5L6G3L3G4F5R10F4R8E3R11E4R6E4. ¡Un buen párrafo! Pero imagina por un momento esta misma figura realizada mediante LINE... ¿Bastante más complicada, verdad?

Antes de empezar con la realización práctica de las figuras hay un detalle que es importante sobre DRAW, las distancias de cada una de las líneas que trazamos se consideran siempre a partir del punto final de la línea precedente, pero sin contar **nunca** este último punto como parte de la nueva línea. Si te fijas detenidamente en la figura 18 observarás este detalle. Esto es algo que hay que tener en cuenta si no queremos que nuestros dibujos queden **abiertos** por falta de coincidencia en su terminación.

Bien, carga el programa (BASE3) y disponte a teclear las siguientes líneas a partir de la 5000.

```
5010 A$="H4L6H3L9H3L9G5L6G3L3G4F5R10F4R8E3
     R11E4R6E4"
5020 X=120:Y=80
5040 PSET (X,Y),15
5050 DRAW A$
5060 PAINT (X,Y),15
5070 GOTO 5070
```

Como verás, ha aparecido una hermosa nube blanca en el centro de nuestro cielo electrónico. Analicemos el programa para ver los pasos de su realización.

CADENA O CONSTANTE

En la línea 5010 adjudicamos a la constante **alfanumérica** A$ todos los parámetros de nuestro dibujo mediante un **string** o **ristra** de caracteres, que es lo que en inglés quiere decir esa palabra. Esta es la asignación a la cual nos referíamos al principio del capítulo, una vez que nuestros parámetros están asignados, debemos fijar el punto donde **comenzar** nuestro dibujo. Ya hemos dicho que en DRAW (durante su realización) no es necesario especificar posición, a diferencia de lo que sucede con LINE o CIRCLE, sin embargo, para situar el dibujo es necesario indicar una posición en la pantalla (o fuera de ella) u obtendremos un mensaje de error (ILLEGAL FUNCTION CALL...).

PSET

Esto lo realizamos mediante la instrucción PSET, que nos permite situar un punto de imagen en cualquier parte de la pantalla por medio de las coordenadas X/Y (este punto puede ser de cualquiera de los 16 colores disponibles).

En la línea 5020 damos los valores a X/Y para utilizarlos luego en la línea 5040 con PSET (situar punto), como verás el formato de la instrucción es prácticamente igual al empleado en LINE. Las coordenadas entre paréntesis y el color después de una coma.

Este punto de imagen (X-120 Y-80) va a corresponder a la ejecución de DRAW con el punto de partida empleado en nuestro dibujo, es decir, desde ese punto va a ser realizada nuestra nube siguiendo los parámetros indicados en A$.

En la línea 5050 se ejecuta DRAW, si escribimos la siguiente línea podremos ver cómo queda nuestra nube sin **pintar**.

```
5055 GOTO 5055
```

Verás que corresponde exactamente a la dibujada en la figura 18. Eliminemos la línea 5055 tecleando su número (después de parar el programa) y analicemos la línea 5060. En esta línea empleamos la instrucción PAINT para colorear nuestra nube, comprobarás que el color elegido (15) es el mismo que hemos empleado en PSET y que ha sido con este mismo color con el que se ha dibujado la nube.

En lo que corresponde a PAINT, el motivo es el mismo que explicábamos en el anterior capítulo al hablar sobre las características gráficas del **modo 1**, en cuanto a la instrucción DRAW, ésta realizará el dibujo con el último color utilizado (en este caso el dado en PSET), a no ser que se especifique otro dentro de la misma instrucción (posibilidad que más adelante veremos). Aparte de este detalle observarás que en PAINT hemos desplazado la coordenada X, restándola 3 puntos de imagen.

Esto es motivado porque PAINT rellena un área delimitada por una línea, para lo cual sus coordenadas deben situarse **dentro** de esta línea si no queremos que nuestra pantalla se llene de un mismo color. Comprueba este último extremo cambiando el signo en la coordenada X de PAINT ("-" por "+").

```
5060 PAINT (X+3,Y),15
```

¿Qué te parece toda la pantalla blanca con una nube azul? PAINT ha buscado los límites blancos y como la única figura cerrada por este límite es nuestra nube, ha coloreado todo lo demás excluyéndola a ella. Justo lo contrario de lo que pretendíamos.

Bien, restituye el signo '-' y vamos a cambiar algunas cosas en nuestro programa. Como veíamos en la figura 3, en el cielo había varias nubes, y nosotros únicamente hemos puesto una. No te preocupes, que no vas a tener que realizar las que nos faltan, de eso se va a encargar el ordenador y la instrucción DRAW. En esa figura veíamos que aparte del número, varía también el tamaño de las nubes, el cumplir estas dos premisas nos va a dar la oportunidad de utilizar otra de las posibilidades de DRAW, la de poder cambiar de escala un dibujo. Al mismo tiempo, vamos a usar dos nuevas instrucciones, READ y DATA. Teclea las líneas siguientes:

```
5020 FOR N=1 TO 8
5030 READ X,Y,S
5050 DRAW "S"+STR$(S)+A$
5070 NEXT N
5080 DATA 70,55,4,120,90,2,222,130,3,110,20,6,220,10,2
5090 DATA 130,110,2,180,67,3,40,150,2
5100 GOTO 5100
```

Estupendo, ya tenemos un hermoso y nubladísimo cielo; como podrás apreciar las nubes tienen diferentes tamaños y están colocadas en sitios distintos (lógicamente, en el mismo sitio no las veríamos). Veamos qué es necesario para que esto pase.

Primeramente, vamos a ver cómo ha quedado nuestro programa con la inclusión de estas líneas.

```
5000 GRAFICOS F-1
5010 A$="H4L6H3L9H3L9G5L6G3L3G4F5R10F4R8E3
     R11E4R6E4"
5020 FOR N=1 TO 8
5030 READ X,Y,S
5040 PSET (X,Y),15
5050 DRAW "S"+STR$(S)+A$
5060 PAINT (X-3,Y),15
5070 NEXT N
5080 DATA 70,55,4,120,90,2,222,130,3,110,20,6,220,10,2
5090 DATA 130,110,2,180,67,3,40,150,2
5100 GOTO 5100
```

Bien, ya que tenemos nuestro programa completo, analicemos su funcionamiento. En la línea 5020 hacemos un bucle o **loop** que nos va a repetir 8 veces las operaciones comprendidas entre esta línea y la 5070 (NEXT N).

READ/DATA

El número de repeticiones se corresponde con el de nubes, en la línea 5030 empleamos por primera vez la instrucción READ, esta palabra significa en inglés **leer** y no es otra cosa lo que con ella hacemos. Por medio de READ podemos realizar una asignación de valores a una o varias variables de acuerdo con el orden establecido en la instrucción DATA. ¿Complicado?

Veamos nuestro caso concreto, nosotros necesitábamos poner 8 nubes en distintos puntos de la pantalla, por lo tanto necesitamos 8 parejas de coordenadas para situar todas las nubes. Como vimos anteriormente, el posicionamiento lo realizábamos mediante PSET, esto equivaldría a la utilización de 8 de estas instrucciones con un valor distinto cada una...

```
PSET (70,55),15
PSET (120,90),15
PSET (222,130),15
PSET (110,20),15
PSET (220,10),15
PSET (130,110),15
PSET (180,67),15
PSET (40,150),15
```

...o la utilización de un sistema que fuera adjudicando secuencialmente los valores X Y a esta instrucción. Aquí es donde entra READ, como verás en la línea 5030 las dos primeras variables utilizadas son X/Y, las cuales corresponden a los dos primeros valores que tenemos en la instrucción DATA de la línea 5080.

La instrucción READ lee tres valores cada vez (X,Y,S), siguiendo esta secuencia comprobarás que los dos primeros números, de cada serie de tres, corresponden a los valores X/Y de la instrucción PSET. Suministrando estos valores secuencialmente evitamos el tener que repetir instrucciones de forma innecesaria, bastando con cambiar el valor de sus datos por la asignación de éstos a una variable, que a su vez tome su valor mediante la instrucción READ de un **fichero** DATA donde previamente los habremos puesto.

Bien, ya tenemos el valor de X/Y en la línea 5030; ahora lo único que tenemos que hacer es transferirlos a la línea 5040 donde está PSET, para esto basta con que utilicemos las mismas variables.

El valor de X/Y en 5030 sigue siendo el mismo en 5040 y no cambiará hasta que hagamos otra lectura al final del ciclo FOR NEXT, donde READ se desplazará automáticamente tres posiciones para tomar los siguientes valores. Cuando se trabaja con READ y DATA se debe tener especial cuidado en el orden en que los datos son puestos, ya que el carácter secuencial de estas instrucciones hace que un solo error cambie totalmente el significado de los datos posteriores. También es de suma importancia hacer coincidir el número de datos con el de **accesos** que se van a realizar, ya que si la instrucción READ no encuentra datos que leer, no toma el valor **0**, sino que da error de lectura (OUT DATA IN...) parando el programa.

En la línea 5050 hemos complicado algo la instrucción DRAW, primero hay una "S", esta letra es la inicial en inglés (todo en inglés...) de scale que significa escala, y con ella indicamos a la instrucción DRAW que el dibujo que va a realizar es en la escala que se incida después de esta letra (S). ¡Pero nosotros no hemos puesto un número, sino la **función** STR$!

ESCALAS EN DRAW

Bien, tenemos dos cosas que explicar sobre la escala (S) y su inclusión en nuestra constante **alfanumérica** A$. Empecemos por la escala, el valor que podemos dar a este **subcomando** es de 1 a 255, esto sin embargo, no quiere decir que podemos **ampliar** nuestros dibujos 255 veces, ya que con el fin de que también tengamos la posibilidad de reducción, el número inicial que éstos tienen (sin que nosotros hagamos ninguna asignación) es el 4. Es decir, si damos a "S" un valor de 4 obtendremos un dibujo de las mismas dimensiones que el original (éste es el caso de la nube situada en las coordenadas X-70 Y-55). También podemos decir que el factor de escala es el resultado de dividir el número empleado en "S" por 4. Ejemplo: S=12, 12/4=3, relación de escala 3/1.

En el caso de reducciones, cuando utilizamos valores inferiores a 4, como por ejemplo 1, obtendremos una escala de 1/4=.25=1/4. Por lo tanto, basta multiplicar el factor que deseemos por 4 para obtener el valor de "S", este número debe ser entero por lo que habrá que **redondearlo** en el caso de obtener decimales. Supongamos que queremos una escala de 12/10 (1.2/1) 1.2x4=4.8, por lo que el número que utilizaríamos en "S" sería el 5.

En la figura 19 se puede apreciar un ejemplo gráfico de la aplicación de distintos valores de escala al subcomando **S.**

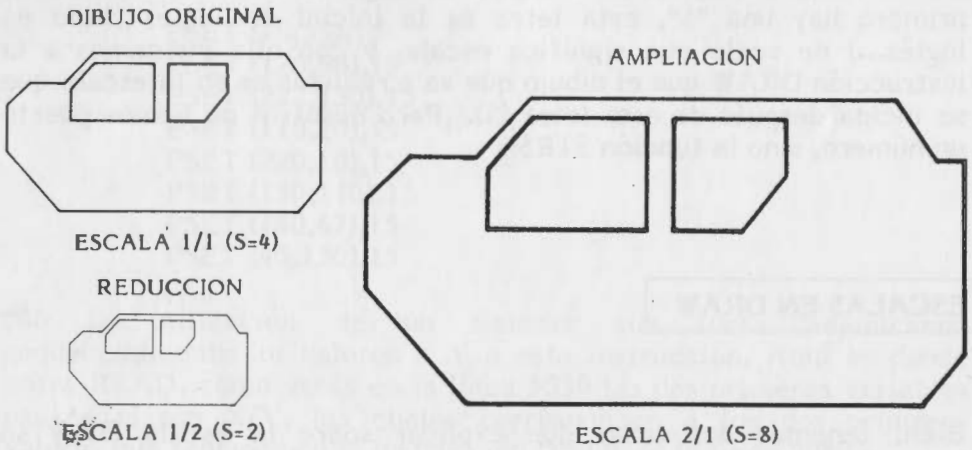


Fig. 19 Ejemplo de la utilización del subcomando S en DRAW

VARIABLES Y CONSTANTES

Una vez que conocemos el modo para determinar la escala, examinemos la forma de que ésta sea **ejecutada** en la instrucción DRAW. Primeramente diremos que DRAW ejecuta únicamente constantes alfanuméricas, y que éstas constantes o **cadenas** alfanuméricas se diferencian de las variables **numéricas** en que su contenido no tiene valor numérico, por lo que ponemos éste entre **comillas** (" ") para indicarle al ordenador que se trata de un contenido alfanumérico y no de una **variable.**

Por ejemplo, A es una variable numérica a la cual se la puede asignar un valor, como A=125, mientras que "A" es únicamente la primera letra de nuestro alfabeto, es una **constante alfanumérica** y su valor es únicamente de este tipo.

Para aclarar las ideas escribe lo siguiente en tu ordenador y después pulsa RETURN:

```
A=125*56:PRINT A:PRINT "A"
```

Como verás, los resultados han sido distintos a pesar de ser la misma letra (A), puesto que en la segunda el ordenador **identificó** la constante alfanumérica. La forma que utilizamos para **añadir** constantes entre sí, es mediante el signo "+". Teclea este otro ejemplo:

```
D$="PRU":E$="EBA":PRINT "ESTA ES LA "+D$+E$
```

Con esto hemos conseguido que las distintas constantes o cadenas alfanuméricas se **impriman** unidas en una instrucción PRINT, igualment podemos unir cadenas en la instrucción DRAW siempre que su contenido corresponda a los caracteres que pueden ser aceptados por esta instrucción. Hasta ahora hemos hablado de caracteres **alfanuméricos**, pero ¿qué sucede cuando queremos incluir en una constante o cadena alfanumérica una **variable** numérica?

STR$

Lo que sucede es que ésta no es aceptada por el ordenador si previamente no cambiamos su entidad numérica a alfanumérica, ésta es la labor que realiza la función STR$ ( ). Veamos dos ejemplos que nos aclaren este punto, escribe lo siguiente y pulsa RETURN:

```
A=210:B$="ESTO ":C$="VALE ":E$=B$+C$+A:PRINT E$
```

Obtenemos error, cambiemos la línea de esta manera:

```
A=210:B$="ESTO ":C$="VALE ":E$=B$+C$+"A":PRINT E$
```

Bien, admitió la línea pero en lugar de 210, que era lo que pretendíamos, ha escrito "A". Hagamos el último cambio:

```
A=210:B$="ESTO ":C$="VALE ":E$=B$+C$+STR$(A) PRINT E$
```

Correcto, en el primer ejemplo A era una variable numérica por lo que la cadena no era homogénea. En el segundo "A" es una letra que nada tiene que ver con la variable 'A'. Y por fin, en el tercer ejemplo convertimos el **valor** numérico de A en una constante alfanumérica que ya sí se puede sumar a las demás cadenas por ser esta suma homogénea.

Después de estas explicaciones, estamos ya en disposición de ver cómo integramos en la línea 5050 el valor del subcomando S por medio de STR$(S). Primeramente este valor es leído por READ (X,Y,S) en 5030, para posteriormente ser convertido y sumado en la instrucción DRAW.

De esta manera conseguimos introducir en una constante alfanumérica valores que podemos hacer cambiar externamente de acuerdo a las necesidades o circunstancias de nuestro programa.

Tal vez has observado ciertos **flecos** en las nubes que tienen una escala distinta de 1, vamos a solucionar este problema cambiando algunos parámetros en nuestra cadena A$. Añadamos al principio R5 y en la primera L sustituyamos el 6 por un 8. La cadena debe quedar ahora de esta forma:

```
5010 A$="R5H4L8H3L9H3L9G5L6G3L3G4F5R10F4R8
     E3R11E4R6E4"
```

Vamos a variar también el tamaño y posición de algunas nubes, cambia en las líneas 5080, 5090 la escala a las nubes números 1, 4, 6, 7 y en la número 8 la escala y posición de la siguiente forma:

```
5080 DATA 70,55,3,120,90,2,222,130,3,110,20,1,220,10,2
5090 DATA 130,110,1,180,67,4,30,30,1
```

CLEAR

Por último, vamos a añadir una instrucción en la línea número 20 (INICIALIZACION), se trata de CLEAR que sirve para poner todas nuestras variables a cero. Es importante usar esta instrucción para eliminar cualquier valor que pudiera haber en los **registros** del procesador de vídeo (VDP) y que podrían afectar la posición o tamaño de nuestros gráficos en algunas instrucciones. Escribe:

```
20 CLS:CLEAR
```

Ya tenemos casi acabada la subrutina GRAFICOS F-1; veamos qué aspecto tiene con las modificaciones efectuadas. Mejor ¿verdad?

Vamos a continuar poniendo la torre de lanzamiento.

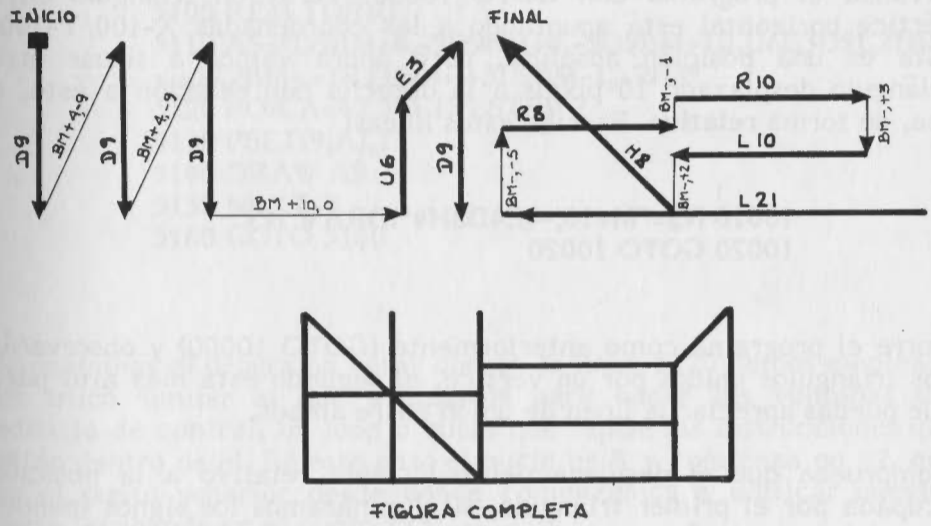


Fig. 20 Torre de lanzamiento, detalle de la misma

Para la construcción de la torre vamos a emplear de nuevo DRAW, pero esta vez con un nuevo **comando;** se trata de B y M. El comando B (blank, en blanco) sirve para no efectuar ningún trazo durante un desplazamiento, es decir, si ponemos este comando delante de cualquiera de las letras correspondientes a los ángulos de la instrucción DRAW, el desplazamiento que hayamos indicado se realizará sin **dibujar** en la pantalla (en Blanco).

El comando M (move, mover, desplazar) se emplea para desplazar el **cursor** de una forma relativa o absoluta, este comando tiene gran similitud con PSET, ya que también hay que dar las coordenadas X/Y. La diferencia consiste en que dependiendo de que pongamos o no los signos "+" o "-" delante de las coordenadas X/Y, el desplazamiento será **relativo** o absoluto al punto en que estuviera anteriormente. Ejemplo de desplazamiento absoluto sería el siguiente:

```
10000 SCREEN 2:M$="M100,100E4D8H4":DRAW M$
10010 GOTO 10010
```

Arranca el programa con GOTO 10000. Verás un triángulo cuyo vértice horizontal está apuntando a las coordenadas X-100/Y-100, ésta es una posición absoluta, pero ahora vamos a situar otro triángulo desplazado 10 pixels a la derecha con relación a éste, o sea, de forma **relativa.** Escribe estas líneas:

```
10010 N$="M+10,-4E4D8H4":DRAW N$
10020 GOTO 10020
```

Corre el programa como anteriormente (GOTO 10000) y observarás dos triángulos unidos por un vértice, el segundo está más alto para que puedas apreciar la línea de unión entre ambos.

Comprueba que el desplazamiento ha sido relativo a la posición ocupada por el primer triángulo, si eliminàramos los signos (puedes hacerlo) la posición correspondería a las coordenadas X10/Y4. Es decir, la parte superior izquierda de la pantalla.

Utilicemos ahora el comando B para eliminar la línea de unión, modifica las líneas 10000 y 10010 de esta manera:

```
10000 SCREEN 2:M$="BM100,100E4D8H4":DRAW M$
10010 N$="BM+10,-4E4D8H4":DRAW N$
```

Como puedes comprobar, la línea de unión ha desaparecido. Ya conocemos dos comandos más de la instrucción DRAW, vamos ahora a utilizarlos.

En la figura 20 tienes el dibujo de una **sección** de lo que va a ser nuestra torre de lanzamiento, al igual que hicimos en la figura 18, tienes detalladas las letras correspondientes a los ángulos y la longitud de los desplazamientos que tenemos que realizar. En esta ocasión el dibujo está realizado en dos partes con el fin de que pueda ser seguido más fácilmente, figurando también la cadena alfanumérica del mismo. Borra las líneas que empleamos de prueba (DELETE 10000-10020) y escribe las siguientes:

```
5100 REM TORRE
5110 A$="D9BM+4,-9D9BM+4,-9D9BM+10,U6E3D9L21BM
+,-5R7BM-,-1R11BM+,+3L9BM-1,+2H8"
5120 FOR A=42 TO 175 STEP 7
5130 PSET(9,A),1
5140 DRAW A$
5150 NEXT A
5160 GOTO 5160
```

Si examinas el programa verás que en la línea 5120 hemos empleado un truco similar al que utilizamos para hacer las ventanas del edificio de control, un loop o bucle que repite las instrucciones que están dentro de él. En este caso el bucle es A y comienza en 42, que es el punto superior desde donde comenzamos a edificar nuestra torre, acabando en el punto 175.

En la línea 5130 empleamos nuestra vieja conocida PSET con un valor fijo de 9 para X y otro, que corresponde al que tiene el bucle A, para Y, este bucle aumenta en pasos (STEP) de siete. Después tenemos a DRAW, que va dibujando la cadena A$ (definida en 5110) a partir de las coordenadas y el color (1) indicados por PSET, finalmente recurrimos a la instrucción de enclavamiento en la línea 5160 para poder visualizar nuestro gráfico.

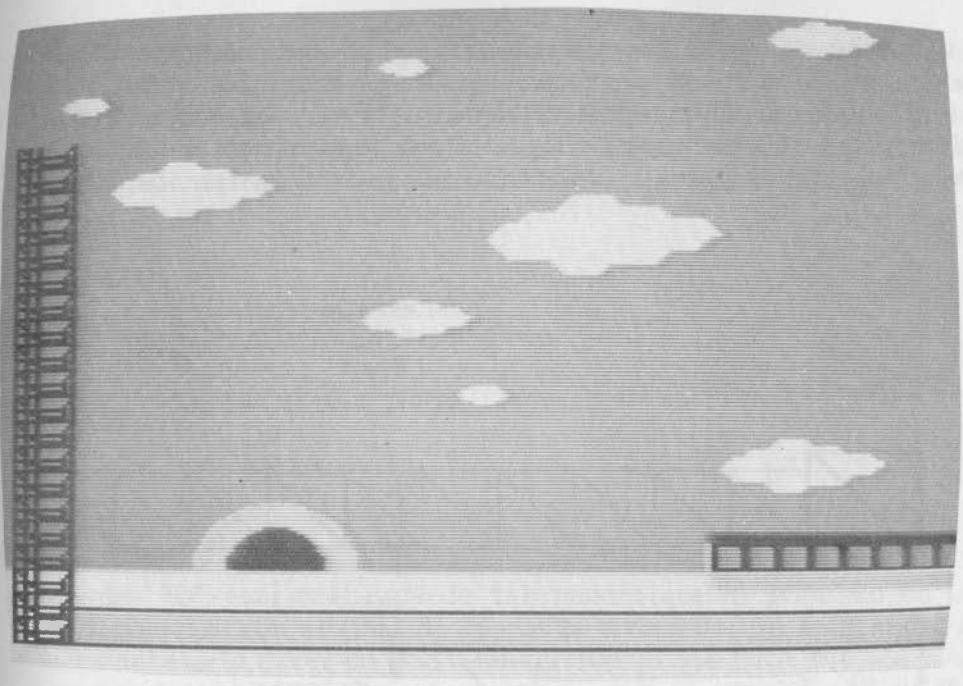
Observarás que en la cadena A$, en los valores correspondientes a BM hay veces que empleamos un signo ("+" o "-") sin que vaya seguido de ningún número. El motivo de esto es que cuando empleamos el desplazamiento relativo para M, como ahora en A$, podemos omitir su valor cuando éste es 0, pero **siempre** tendremos que poner un signo ("+" o "-") a X. En el caso de Y con valor 0, puede omitirse tanto el signo como el valor.

Rodemos el programa y veámos qué aparece en la pantalla. ¿Parece que la torre no queda muy airosa, verdad...?

Vamos a introducir una pequeña modificación al principio de la línea 5110, justo antes de BM+4 reduzcamos la escala de nuestro dibujo .75 (S3). Lista 5110 e inserta S3 de manera que quede así:

```
5110 A$="S3D9BM+4,-9D9BM+4,-9D9BM+10
        ,U6E3D9L21BM+,-5R7BM-,-1R11BM+,+3L9BM-1,+2H8"
```

Bien, corre de nuevo el programa y a ver qué te parece ahora. Lo que hemos hecho es reducir la escala para que se adapte a los desplazamientos que damos en A (7) y para que quede más proporcionada al conjunto del paisaje.



Lo construido hasta ahora es lo que podemos llamar la estructura principal de la torre, pero como verás es bastante simple y está abierta por su parte superior, vamos a cerrar esa parte al mismo tiempo que le añadimos algún detalle a su estructura. Realicemos los dibujos de la figura 21 utilizando de nuevo la instrucción DRAW.

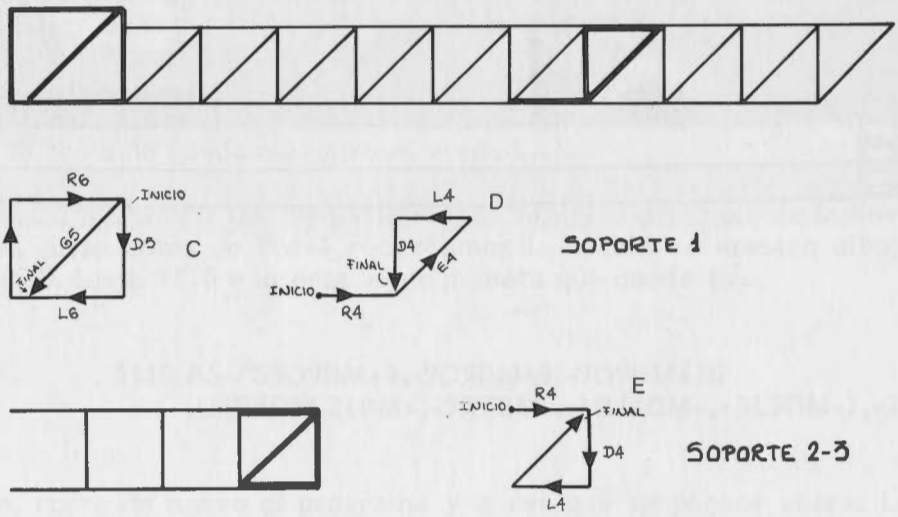


Fig. 21 Detalles de la torre de lanzamiento

Como puedes observar, las estructuras están realizadas mediante la repetición de una figura simple, así que lo que haremos es asignar los parámetros de esta figura a una cadena y mediante un bucle o loop, donde estará contenida DRAW, repetir este dibujo un número determinado de veces. Podemos comenzar dibujando la figura "D" escribiendo las líneas siguientes:

```
5160 REM SOPORTE 1
5170 A$="R4E4L4D4"
5180 PSET (9,41),1
5190 FOR A=1 TO 10
5200 DRAW A$
5210 NEXT A
5220 GOTO 5220
```

La figura "D" está realizada de tal manera que el principio de la misma coincide con el final de la que le antecede, de esta manera sólo es necesario hacer un PSET inicial puesto que DRAW empezará a dibujar desde el último pixel que hayamos utilizado. Con el fin de que no se solapen los dibujos, la posición vertical (Y) que damos en la línea 5180 (41) es un punto más alta que la utilizada en la torre (42), en cuanto a la horizontal verás que el soporte queda desplazado hacia la derecha, ya que en ese espacio pondremos la figura "C". Teclea estas líneas.

```
5220 A$="D5L6U5R6G5"
5230 PSET (11,37),1
5240 DRAW A$
5250 GOTO 5250
```

Ya tenemos el soporte superior (1) construido, ahora vamos a realizar los otros dos (2 y 3). Dado que éstos son iguales tenemos dos opciones para su ejecución, 1 realizar dos bucles, uno para cada soporte o 2 hacer una **subrutina** que nos sirva para los dos.

ENCESTADO DE SUBRUTINAS

Como lo que tratábamos era de hacer una programación estructurada, vamos a realizar esta última opción, cuando se llama a una subrutina desde otra subrutina se dice que éstas están **encestadas** (nested) para significar que una contiene a la otra.

Realicemos nuestra subrutina **encestada** para finalizar la torre de lanzamiento.

```
5250 REM SOPORTES 2-3
5260 A$="R4D4L4E4"
5270 PSET (24,90),1
5280 GOSUB 5390
5290 PSET (24,130),1
5300 GOSUB 5390
5310 GOTO 5310
5390 REM SUB SOPORTES
5400 FOR A=1 TO 4
5410 DRAW A$
5420 NEXT A
5430 RETURN
```

Primeramente, asignamos en A$ los parámetros correspondientes al dibujo "E" después, en la línea 5270 fijamos mediante PSET el punto inicial para el soporte número 2, con este punto fijado vamos a la subrutina 5390 donde por medio de un bucle de cuatro pasos realizamos el dibujo. Una vez acabado éste **retornamos** a la línea 5290 donde encontramos de nuevo la instrucción PSET, sólo que ahora hemos cambiado su posición vertical, de Y-90 ha pasado a Y-130 desplazándose hacia la parte inferior.

En la línea 5300 volvemos a ir a la subrutina 5390 para ejecutar de nuevo el mismo dibujo, sólo que ahora en distinta posición vertical, finalizando éste retornamos a la línea 5310 donde quedamos **enclavados.**

Con esto finalizamos el bloque GRAFICOS F-1 correspondiente a las nubes y torre de lanzamiento. Una vez que hayas visto cómo queda el paisaje cambia la línea 5310 de la siguiente forma:

```
5310 RETURN
```

Procede ahora a cambiar el título del programa en la línea número 10:

```
10 REM BASE4
```

Sigue el procedimiento empleado anteriormente para **salvar** el programa en la cassette, es decir, no rebobines y graba a partir del punto donde finalizaste al copiar la parte anterior (BASE3). Una vez grabado no te olvides de comprobar la grabación mediante CLOAD?"BASE4".

En la figura 22 tenemos la carta de flujo correspondiente a esta subrutina donde podemos apreciar, de una forma gráfica, las técnicas empleadas en la confección de la misma. Al final del capítulo figura el listado de este bloque.

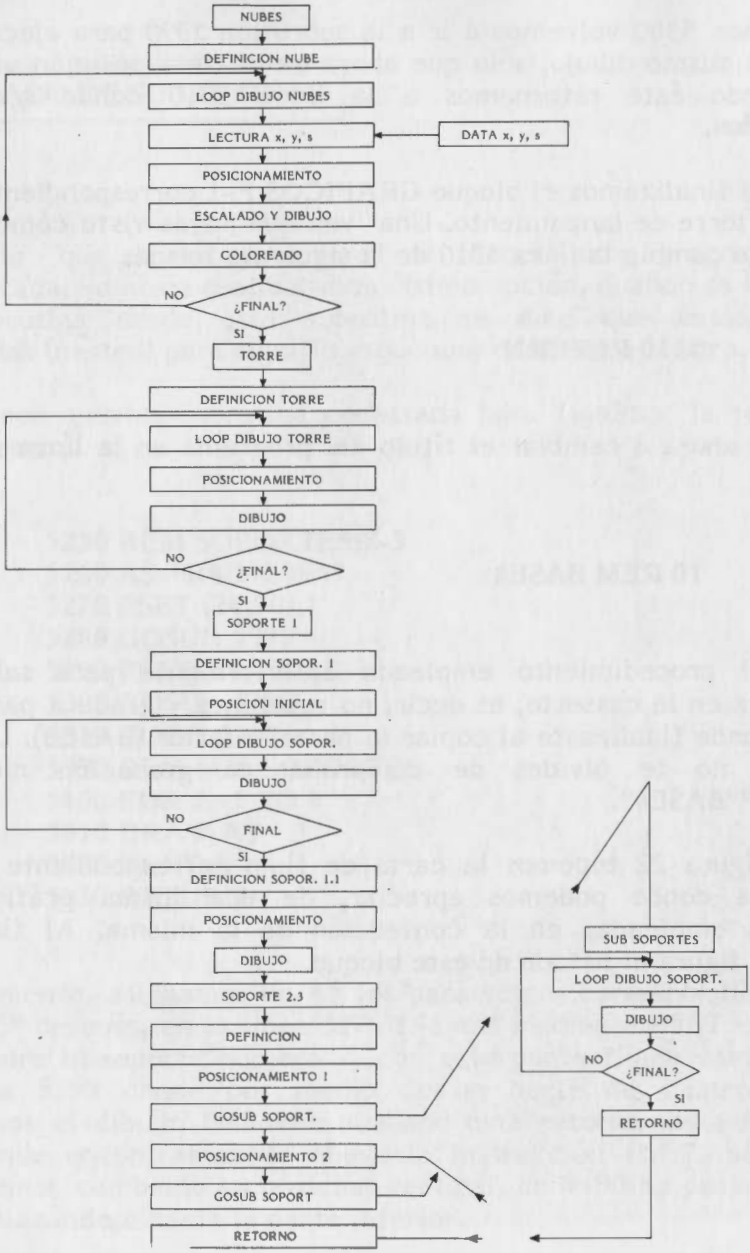
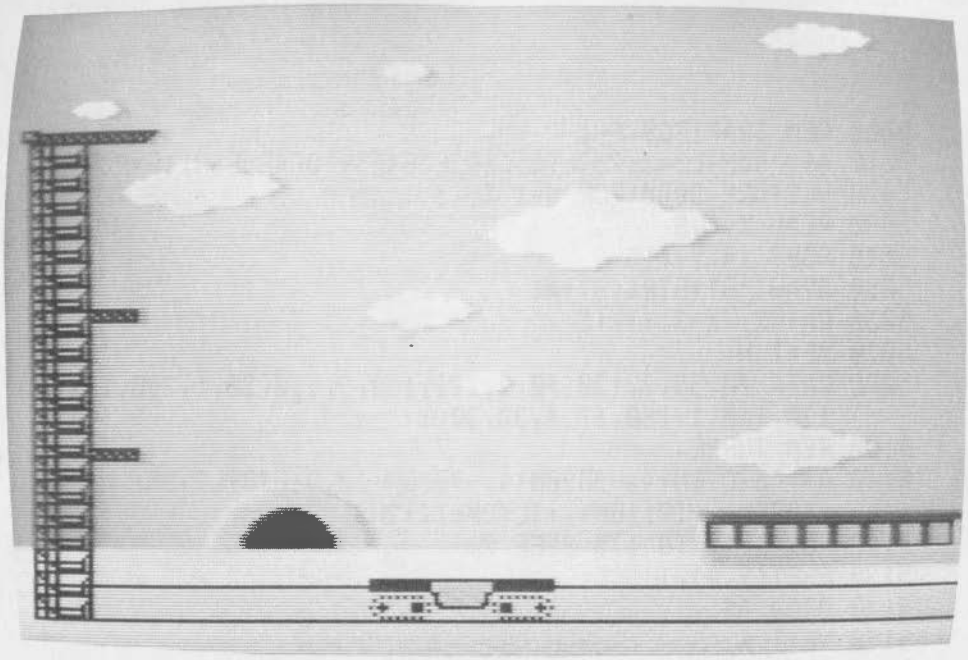


Fig. 22 Carta de flujo subrutina GRAFICOS F-1

```
5000 REM GRAFICOS F-1
5010 A$="R5H4L8H3L9H3L9G5L6G3L3G4F5R10F4R8E3R11E4R6E4"
5020 RESTORE 5080:FOR N=1 TO 8
5030 READ X,Y,S
5040 PSET (X,Y),15
5050 DRAW "S"+STR$(S)+A$
5060 PAINT (X-3,Y),15
5070 NEXT N
5080 DATA 70,55,3,120,90,2,222,130,3,110,20,1,220,10,2,
     130,110,1,180,67,4,30,30,1
5100 REM TORRE
5110 A$="S3D9BM+4,-9D9BM+4,-9D9BM+10,U6E3D9L21BM+,
     -5R7BM-,-1R11BM+,+3L9BM-1,+2H8
5120 FOR A=42 TO 175 STEP 7
5130 PSET(9,A),1
5140 DRAW A$
5150 NEXT A
5160 REM SOPORTE 1
5170 A$="R4E4L4D4"
5180 PSET(9,41),1
5190 FOR A=1 TO 10
5200 DRAW A$
5210 NEXT A
5220 A$="D5L6U5R6G5"
5230 PSET(11,37),1
5240 DRAW A$
5250 REM SOPORTES 2-3
5260 A$="R4D4L4E4"
5270 PSET(24,90),1
5280 GOSUB 5390
5290 PSET(24,130),1
5300 GOSUB 5390
5310 RETURN
5390 REM SUB SOPORTES
5400 FOR A=1 TO 4
5410 DRAW A$
5420 NEXT A
5430 RETURN
```



Dibujo de la Plataforma de Lanzamiento

SPRITES

En este capítulo construiremos nuestro primer dibujo **animado,** para lo cual vamos a utilizar por primera vez los famosos SPRITES o gráficos móviles. Estos SPRITES tienen unas posibilidades gráficas realmente extraordinarias que nos permiten la animación de figuras con una facilidad que no es posible en aquellos ordenadores que carecen de este sistema.

La realización de SPRITES requiere una técnica diferente a las que hasta ahora hemos utilizado, si bien es cierto que para su posicionamiento y coloreado nos servirá el conocimiento que ya tenemos de instrucciones gráficas utilizadas anteriormente. Primeramente vamos a ver en la figura 23 los tipos de SPRITES que puede haber, en esta figura tenemos cuatro tipos distintos A, B, C y D.

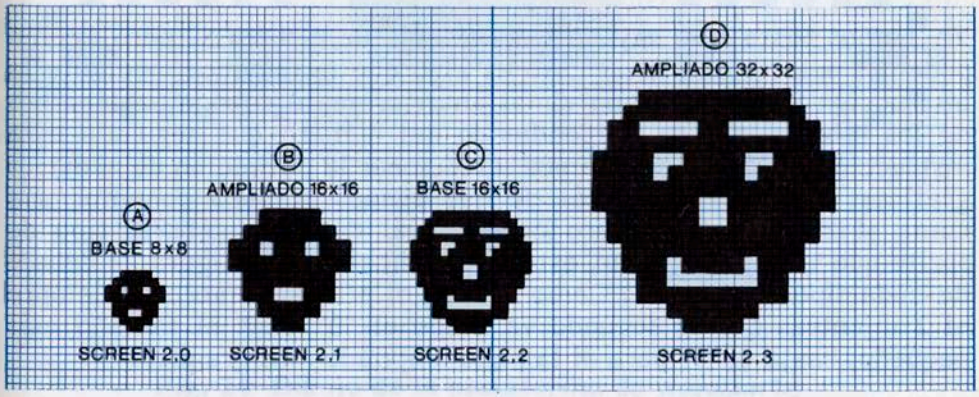


Fig. 23 Distintos tipos de SPRITES

Los que corresponden a "A" y "C" son los modelos **originales** o base, mientras que "B" y "D" son las **ampliaciones** de los anteriores.

Como podemos ver en la figura 23, los dos tipos de SPRITES, A y C, a partir de los cuales se realizan las ampliaciones B y D, están construidos dentro de una matriz de 8 x 8 y 16 x 16 pixels respectivamente, es decir, un SPRITE del modelo "A" no puede tener más de 8 pixels por cada lado y uno del modelo "C" más de 16. Los modelos o tipos, B y D, son una ampliación 2 a 1 de los anteriores, siendo este factor fijo, sin que se pueda variar como en el caso de DRAW (S).

Si te fijas en los tipos A y B, verás que la ampliación pierde **detalle** con respecto a su base, por lo que no siendo posible el definir directamente la figura ampliada, si queremos un dibujo detallado en una matriz de 16 x 16 tendremos que realizarlo directamente en un SPRITE del modelo "C", ya que si lo realizamos en uno del modelo "A" (8 x 8), para posteriormente ampliarlo a 16 x 16 (B), no podremos lograr el mismo detalle o definición.

SELECCION DEL MODELO

Ya hemos visto que hay varios modelos de SPRITES, pero ¿cómo podemos seleccionarlos...? Esto es algo que realizamos mediante la instrucción SCREEN; como vimos anteriormente cuando hablábamos de los **modos** gráficos, dependiendo del número que seguía a esta instrucción (SCREEN) podíamos seleccionar la definición de nuestros gráficos. Pues bien, después de ese primer número ponemos una coma y un segundo número con el cual determinamos el tipo de sprite que vamos a utilizar, esto está indicado en la figura 23 debajo de cada tipo de SPRITE. Como vemos, el primer número es siempre el 2 mientras que el segundo va desde el "0" para el tamaño más pequeño, hasta el "3" para el tamaño ampliado más grande.

Una vez que sabemos el sistema vamos a utilizarlo, el tamaño de SPRITE elegido es el grande sin ampliar, es decir, el tipo "C" de la figura 23, por lo tanto, el segundo número de SCREEN debe ser el 2.

Como la instrucción SCREEN estaba en nuestro bloque de INICIALIZACION vamos a cargar el programa para listar este bloque (CLOAD "BASE4").

Una vez cargado el programa lista la línea 30 e inserta en la misma el segundo número para que quede de la siguiente forma:

```
30 SCREEN 2,2
```

Con esto ya tenemos seleccionado el tipo de SPRITE; ahora vamos a ver cómo llevamos a cabo el dibujo. En la figura 24 está representada la plataforma que debemos realizar, el área que ocupa es la correspondiente a tres sprites del tipo C unidos lateralmente, esto es, 48 x 16 pixels.

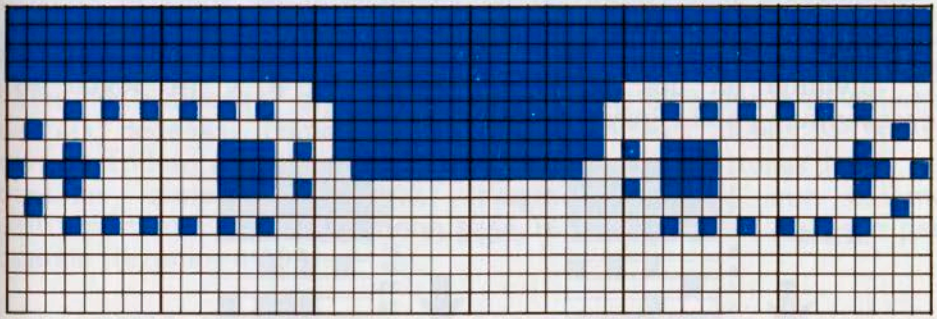


Fig. 24 Dibujo de la plataforma de transporte

DEFINICION DE SPRITES

Ahora debemos indicar al ordenador cuáles son los puntos de imagen, en estas matrices, que están **llenos** y cuáles son los que están **vacíos.** Imagínate que la matriz que contiene un SPRITE es un tablero que cuenta con un número de bombillas igual al número de pixels de esa matriz y distribuidas de la misma forma. Si nosotros quisiéramos reproducir el dibujo del SPRITE número 1 tendríamos que encender las cuatro primeras filas superiores, dejaríamos apagada la quinta, en la sexta no encenderíamos las tres primeras lámparas para encender, a partir de la cuarta, una si y otra no, hasta la catorceava, en la séptima fila encenderíamos únicamente la segunda lámpara, etc., hasta acabar con todas las filas de nuestro SPRITE siguiendo el trazado de su dibujo.

Como es lógico, en nuestra pantalla no se forman las imágenes con bombillas, pero la idea de la matriz del cartel luminoso es válida para nosotros ya que lo que tenemos que hacer es **encender** dentro de nuestro ordenador los pixels que contienen imagen en cada matriz correspondiente a un SPRITE y para esto debemos indicarle, exactamente, qué fila y qué pixels deben ser **encendidos.**

Lo primero que vamos a hacer es considerar nuestro SPRITE **modelo** "C" dividido en dos secciones iguales de 8 x 16 según está representado en la figura 25.

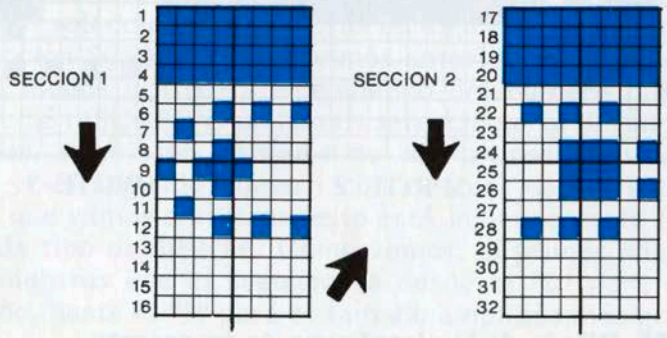


Fig. 25 División del SPRITE número 1

HEXADECIMAL

Como vemos, hay dos columnas cuyas bases tienen 8 pixels, a cada uno de estos grupos de 8 pixels que forman una línea horizontal vamos a llamarlos **octetos** (por aquello de **ocho**), y a su vez vamos a dividirlos en dos grupos de cuatro, con lo cual nos quedan dos columnas de 8 x 16 y cuatro secciones de 4 x 16, como puedes ver en el dibujo.

Lógicamente, debemos dar nuestras indicaciones en un determinado orden. Este orden es el que está indicado por las flechas en la figura 25, empezamos por la columna de la derecha y por su fila superior para, una vez acabada ésta, seguir con la de la izquierda. Ya sólo nos falta saber en qué **lenguaje** debemos comunicar nuestros datos, el que vamos a utilizar es el HEXADECIMAL... (para aquéllos que les suene a chino, tranquilos que no muerde...).

El sistema que normalmente usamos para contar y realizar nuestras operaciones matemáticas es el sistema **decimal**; en este sistema las cantidades están representadas por números que van del "1" al "0". Cuando estamos en la columna de las unidades y alcanzamos el número diez, ponemos en esa columna un "0" y pasamos a la de las decenas un uno, cuando de nuevo en la columna de las unidades volvemos a alcanzar el número diez pasamos otro uno a la columna de las decenas, como allí había ya un "1" ambos se suman convirtiéndose en "2". ¡Bien, no creo que a estas alturas tenga que explicar cómo se cuenta...! Imaginemos ahora que estamos utilizando un sistema que tiene base 16 en lugar de 10... ¿qué pasa cuando llegamos a 10?

Lógicamente, no podemos pasar a la columna de los dieciseisavos, tenemos que permanecer en la de las unidades. Pero, ¿entonces qué número utilizamos? que se sepa no hay más números que los diez comprendidos entre el "1" y el "0".

Esto es cierto, así que hay que recurrir a otro tipo de símbolos (al fin y al cabo los números no son más que eso, símbolos). Ya que tenemos los diez primeros, necesitamos otros seis para completar los dieciséis que necesitamos para nuestro sistema HEXADECIMAL.

Dibujo de la plataforma de lanzamiento

Para este menester se utilizan las seis primeras letras de nuestro alfabeto, A, B, C, D, E y F, puesto que el último valor que podíamos representar era el "9" la letra "A" adquiere el 10, la "B" el 11, la "C" el doce, así hasta la "F" que tiene el valor 15. Cuando superamos este valor, por ejemplo con 16, pasa exactamente lo mismo que en sistema decimal, pasamos un uno a la columna siguiente, que en este caso no tiene el factor 10 si no el 16.

Dado que nosotros no vamos a utilizar números superiores a 15, no vamos a extendernos más en este tema que puede ser bastante largo y no es el objeto de este libro. En la figura 26 tienes una tabla con la correspondencia entre el símbolo **hexadecimal** y su valor **decimal.**

HEX.	DEC.
0	0
1	1
2	2
3	3
4	4
5	5
6	6
7	7

HEX.	DEC.
8	8
9	9
A	10
B	11
C	12
D	13
E	14
F	15

Fig. 26 Tabla de equivalencias hexadecimal/decimal

Bueno y ahora que ya sabemos que C es igual a 12 ¿qué pasa con nuestro SPRITE? Un poco de calma, porque ahora tenemos que explicar el sistema binario...

BINARIO

Este sistema es el que todos los ordenadores utilizan internamente ya que estos equipos con lo único que trabajan es con unos y ceros. ¿Y a cuento de qué viene todo este lío? te preguntarás, pues aunque no te lo creas, para **simplificar** las cosas...

En el sistema binario no existe un símbolo único que represente el dos, como tampoco existe el **número** diez en el decimal (en cuanto a símbolo único), ni el dieciséis en el hexadecimal. Cuando en la primera columna se alcanza el valor dos, automáticamente se pasa un uno a la segunda, pero en este caso, al tener ésta ya en uno y sumar dos se convierte en cero pasando el uno a la tercera.

Es decir, cada columna tiene un factor de dos con respecto a la anterior, así un uno en la columna primera vale 1, un uno en la columna segunda vale 2, un uno en la tercera vale 4, en la cuarta 8, en la quinta 16, y así hasta que nos cansemos. (En el caso de que tuviéramos ocho columnas con un "1" (FF), el valor sería de 255 (128+64+32+16+8+4+2+1) que es el valor máximo con ocho bits).

Nosotros vamos a utilizar únicamente cuatro columnas, por lo tanto el valor máximo que podemos alcanzar es el de 15. Si las cuatro columnas tienen un "1" y aplicamos a cada una de ellas su factor, veremos de dónde sale este 15, esto lo tienes representado en la figura 27 donde se establece la relación entre el sistema hexadecimal, decimal, binario y el valor de cada columna en este último cuando está ocupada por un "1".

Dibujo de la plataforma de lanzamiento

HEX.	DEC.	BINARIO			
0	0	0	0	0	0
1	1	0	0	0	1
2	2	0	0	1	0
3	3	0	0	1	1
4	4	0	1	0	0
5	5	0	1	0	1
6	6	0	1	1	0
7	7	0	1	1	1
8	8	1	0	0	0
9	9	1	0	0	1
A	10	1	0	1	0
B	11	1	0	1	1
C	12	1	1	0	0
D	13	1	1	0	1
E	14	1	1	1	0
F	15	1	1	1	1

Fig. 27 Equivalencias hexadecimal/decimal/binario

Si ahora comparamos la parte derecha del dibujo de la figura 27, con cualquiera de las cuatro secciones en las que dividimos nuestro SPRITE en la figura 25, comprobarás que **todas** las líneas de estas secciones tienen un equivalente en dicho dibujo. Así por ejemplo, las cuatro primeras líneas de la sección 1 corresponden al 15, la quinta corresponde al 0, la sexta al 1, la séptima al 4, la octava al 1 de nuevo, la novena al 11, etc., etc.

Creo que ya habrás visto cuál es la correspondencia que buscamos.

El dibujo está formado por valores binarios, cada pixel **lleno** es un uno y un cero los vacíos, al utilizar únicamente cuatro columnas el valor máximo es de 15, como anteriormente dijimos. ¡Bueno, ahora lo único que tenemos que hacer es pasar este valor al código **hexadecimal...**!

En el ejemplo de nuestra primera columna, ésta quedaría **codificada** de la siguiente manera, "F F F F 0 1 4 1 B 1 4 1 0 0 0 0". Si comparas esta codificación con las tablas de las figuras 26 y 27, podrás apreciar la correspondencia entre estos valores y el dibujo de la columna.

Con el fin de despejar cualquier duda que todavía pueda haber, en la figura 28 están las cuatro columnas del SPRITE número 1 con la representación de cada una de sus líneas en el sistema decimal y hexadecimal (la representación binaria está constituida por el mismo dibujo).

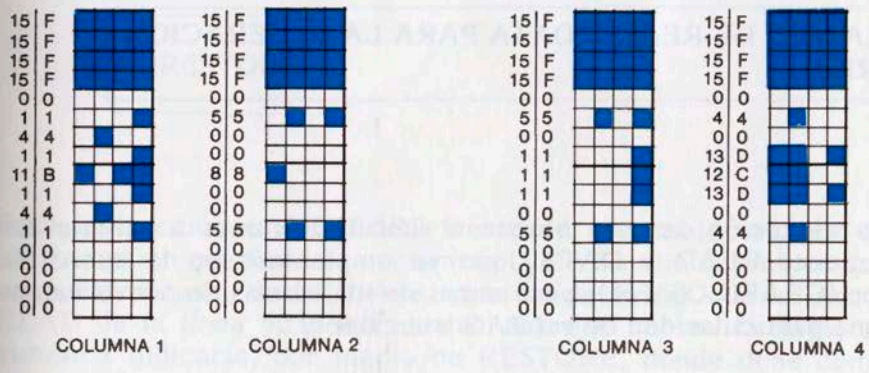


Fig. 28 Representación decimal y hexadecimal del SPRITE nº 1

Tal vez todavía no ves claro del todo este lío de numeración en tres bases distintas; pero es realmente necesario su conocimiento para realizar los dibujos en un SPRITE. Al principio (si no conocías este sistema) te puede parecer algo lioso, pero realmente no lo es, únicamente debes seguir este método: divide los SPRITES en columnas de 8 pixels (en el caso de los que tienen formato 8 x 8 lógicamente no es necesario), dividir a su vez éstas en nuevas columnas o secciones de 4 pixels, dar la correspondencia decimal a cada uno de los códigos binarios contenidos en estas líneas de 4 pixels y, por último, dar el código hexadecimal a cada uno de estos valores. Puede parecer largo pero verás que no lo es en cuanto lo practiques un poco. Bien, creo que es el momento para practicar toda esta teoría y empezar a hacer nuestros primeros SPRITES.

UTILIZACION DE READ Y DATA PARA LA GENERACION DE SPRITES

Para la ejecución de este bloque o subrutina vamos a utilizar las instrucciones READ y DATA, que ya empleamos en la subrutina anterior (GRAFICOS F-1), pero antes de utilizarlas de nuevo vamos a ver una particularidad de estas instrucciones.

Cuando en el transcurso de un programa, éste se encuentra con una instrucción READ, automáticamente va a **leer** a la primera posición de DATA que **no** ha sido leída **independientemente** de que esta posición se encuentre en una línea inferior a la de la instrucción READ. Es decir, si compruebas la posición relativa de la subrutina PLATAFORMA, verás que comienza en la línea 3500 mientras que GRAFICOS F-1 comienza en la línea 5000.

Dado que GRAFICOS F-1 es accedido antes (ver Fig. 8) la instrucción READ contenida en esta subrutina leerá la **primera posición** de DATA que no haya sido leída... y esta posición no será la de su subrutina, sino la que nosotros vamos a realizar ahora, puesto que está en una línea inferior a la suya. Simplificando, podemos decir que el ordenador, cada vez que tiene que ejecutar una instrucción READ, **busca** desde el principio del programa, línea por línea, donde se encuentra una instrucción DATA, una vez hallada **lee** el número de veces que se le indique en READ, **y toma nota** de la última posición.

La próxima vez que tenga que ir a una DATA irá exactamente a la posición siguiente de donde fue anteriormente. Ahora vemos el porqué cualquier DATA que pongamos en una línea inferior a GRAFICOS F-1 será accedida por la instrucción READ contenida en este bloque (línea 5030) a no ser que utilicemos una instrucción que indique al ordenador dónde debe empezar a leer, en lugar de que él busque por su cuenta...

RESTORE

Esta instrucción es RESTORE (restituir, reestablecer), con la cual indicamos en qué línea se encuentra la DATA que queremos acceder, en nuestro caso, de lo que se trata es de evitar que la instrucción READ de la línea 5030 **lea** otra DATA que no sea la suya, así que vamos a indicarle, por medio de RESTORE, **dónde** debe comenzar, en este caso, el comienzo está en la línea 5080, así que escribe lo siguiente:

```
5020 RESTORE 5080:FOR N=1 TO 8
```

Una vez que ya tenemos ordenada nuestra **lectura,** vamos a cambiar un poco la pauta seguida hasta ahora en los capítulos anteriores (en la variación está el gusto).

En lugar de hacerlo al final, vamos a ver ahora las características de esta subrutina mediante su carta de flujo, para lo cual debemos remitirnos a la Fig. 29.

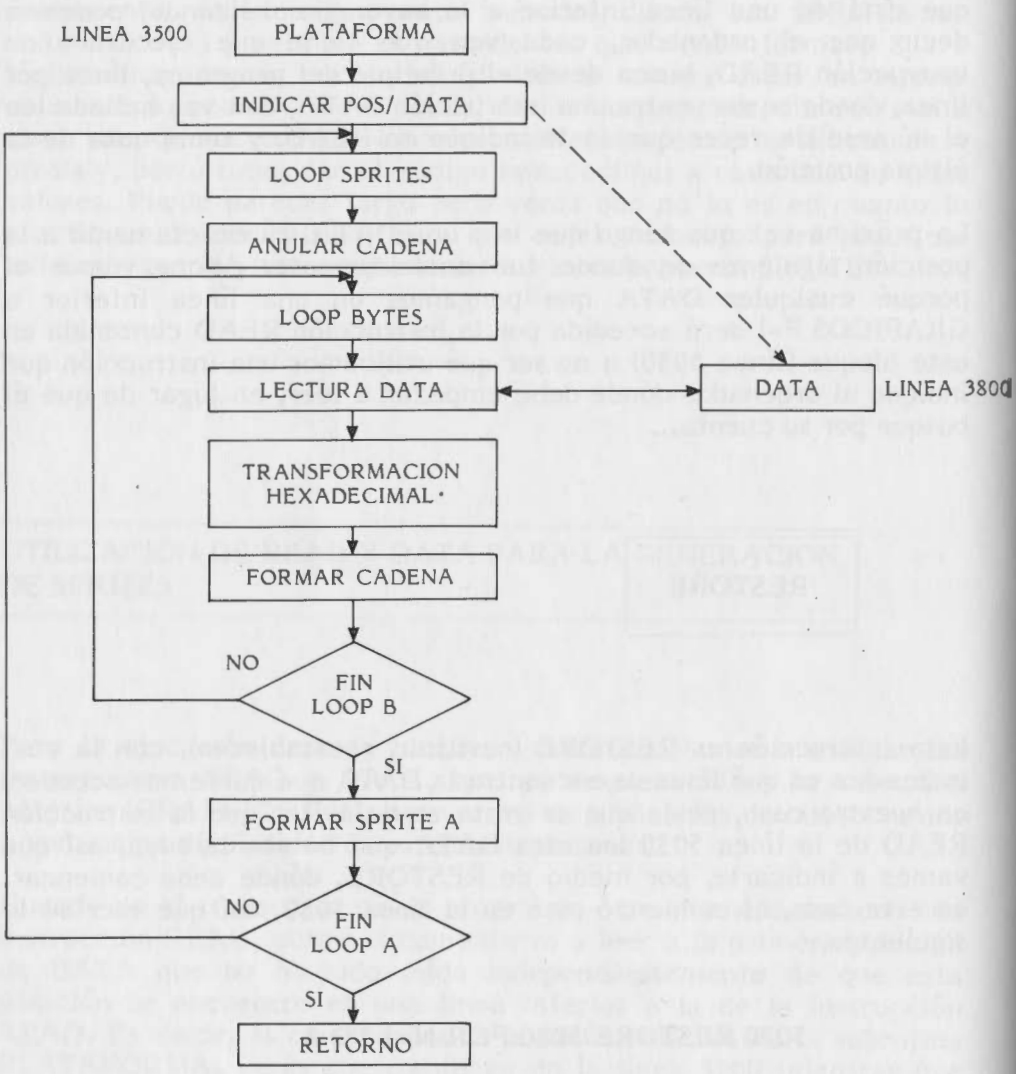


Fig. 29 Flow chart de la subrutina PLATAFORMA

BYTE/BIT

Bien, ya tenemos **casi** todo lo necesario para hacer este bloque del programa, pero como el que no quiere la cosa, se nos ha colado de rondón una palabrita que seguro que hemos oído más de una vez, BYTES. Vemos que está en el segundo loop. Pero, ¿qué es un BYTE? Pues un BYTE no es ni más ni menos que ocho **bits...** ¡Claro que a lo mejor alguno se pregunta que qué es un BIT...!

Para aquéllos que lo pregunten, diremos que se llama BIT a la unidad elemental de información, y que este nombre, que nos hace poner la boca de forma tan rara para pronunciarla, es la abreviatura de la palabra inglesa (¡cómo no!) **BI**nary digi**T** (dígito binario), que simplificando simplificando, no es ni más ni menos que cada uno de los puntos que están contenidos en la matriz de nuestros SPRITES...

Como puedes ver en la Fig. 25, habíamos dividido el SPRITE nº 1 en **dos** columnas (que a su vez se subdividían en dos secciones), comprobarás que cada una de las líneas de estas columnas tienen ocho puntos de imagen, pues bien, cada uno de estos puntos es un BIT y la línea de ocho un BYTE... ¡Binario, decimal, hexadecimal, bytes, bits...! Todo esto es para hacer un SPRITE, ¿no es demasiado lío?

No, realmente no es como puede parecerte (si es que no conocías estos términos), el método que dimos anteriormente sigue siendo válido, lo único nuevo es que la información de nuestro dibujo la debemos dar como está representada en la Fig. 30.

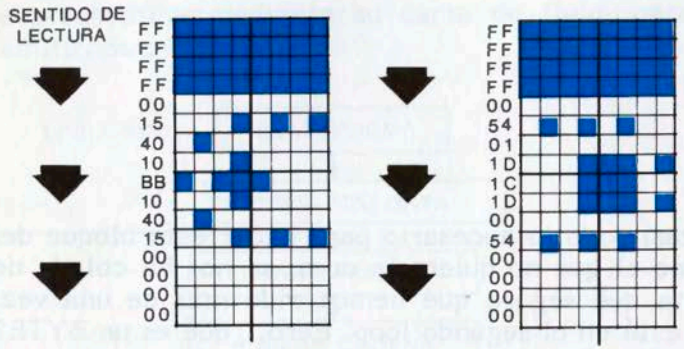


Fig. 30 Representación en BYTES del SPRITE nº 1

Como puedes ver, lo único que diferencia esta figura (30) de la figura 28, es que hemos unido los valores de las secciones de cada columna. De esta manera, formando **bytes** es como vamos a poner cada una de sus líneas en nuestra DATA.

Vamos ahora a teclear el programa, escribe:

```
3500 REM PLATAFORMA
3510 RESTORE 3700
3520 FOR A=1 TO 4
3530 A$=""
3540 FOR B=1 TO 32
3550 READ C$
3560 X$="&H"+C$
3570 C=VAL(X$)
3580 A$=A$+CHR$(C)
3590 NEXT B
3600 SPRITE$(A)=A$
3610 NEXT A
```

En la línea 3510 tenemos una instrucción RESTORE que nos indica dónde empezará la lectura la instrucción READ y por lo tanto, dónde debe estar la primera instrucción DATA de este bloque.

En la línea 3520 está el loop o bucle que contiene las instrucciones necesarias para formar los SPRITES, en este caso cuatro (del cuarto ya hablaremos), seguidamente tenemos la línea 3530 donde **borramos** cualquier asignación que tuviera la constante alfanumérica "A$". Después nos encontramos con el loop **bytes,** que realiza las instrucciones en él contenidas, 32 veces.

Veamos primero el porqué de este número. Si sumamos el número de líneas que tienen las dos columnas de la figura 30, vemos que el total es el de 32. Como lo que tenemos que hacer para confeccionar un SPRITE es una cadena formada por todas estas líneas, tendremos que **leer** 32 datos para confeccionar esta cadena.

Esto es lo que realizamos en la línea 3550, por medio de READ, asignando el contenido de la lectura a la constante "C$". En la línea 3560 **sumamos** "C$" con el **string** o cadena alfanumérica "&H", esta parte es importante pues sirve para indicar al ordenador, en la próxima instrucción, que estamos tratando con valores **hexadecimales.**

VAL

En la línea 3570 encontramos una nueva **función,** se trata de VAL, con ella demos un valor **numérico** a una expresión **alfanumérica,** en nuestro caso, X$; en la línea 3560, que puede ser algo así como "&HFF". Lo que nosotros necesitamos ahora es convertir esta expresión en un valor **numérico,** es decir, lo contrario que hacíamos con la **función STR$.** Veamos unos ejemplos prácticos de esta función, escribe el siguiente párrafo y pulsa RETURN.

```
X$="&HFF":PRINT X$:A=X$:PRINT A
```

Habrás obtenido una repuesta como ésta:

```
&HFF
Type mismatch
```

Esto es debido a que no es posible asignar una expresión alfanumérica a una variable numérica (A), vamos a escribir de nuevo la siguiente línea introduciendo algunos cambios (pusa RETURN al final).

```
X$="&HFF":PRINT X$:A=VAL(X$):PRINT A
```

Distinto ¿verdad...? Ahora X$ tiene un valor numérico de 255, que es el valor máximo que se puede alcanzar con ocho bits, como anteriormente habíamos explicado.

También podíamos haber puesto en la DATA el valor de cada una de las líneas de ocho bits directamente en decimal, pero esto implicaría una mayor dificultad, teniendo que trabajar mentalmente con valores de hasta 255, mientras que con cuatro bits, el valor máximo es de 15, mucho más fácil de obtener directamente para que sea el ordenador (que para eso está) el que obtenga el número decimal correspondiente a los ocho bits.

Una vez que ya tenemos nuestro BYTE convertido a su valor decimal en 3570, vamos a formar la cadena alfanumérica o constante "A$", esta cadena es similar a la que utilizábamos en DRAW, si bien en ésta los parámetros para la construcción de nuestro SPRITE pueden ser cualquiera de los 255 **caracteres** de los que disponemos en nuestro ordenador. Por este motivo, no es posible la realización de esta cadena de forma similar a como realizábamos la de DRAW (¡cualquiera se aprende las 255 equivalencias...!)

CHR$

Esa equivalencia se la damos por medio de la **función** CHR$ y la variable "C", en la línea 3580. En esta línea ves cómo se forma una cadena que tendrá una longitud de 32 caracteres (loop "B"), ya que cada vez que se realice el bucle "B" se sumará un nuevo carácter a "A$" hasta que se complete el loop.

En 3600, una vez completado el bucle "B", asignamos la cadena de SPRITE(A) por medio de la **función del sistema** SPRITE$ (ya vímos que el loop "A" va de 1 a 4). En la línea 3610 volvemos al inicio de loop "A", donde esta variable se incrementa +1, pasando de nuevo a 3530, donde "A$" queda anulada, así se repite el proceso hasta que el loop "A" iguala su valor máximo (4), en cuyo momento, al alcanzar la línea 3610 pasará a la siguiente en lugar de retornar a la 3520.

Ya tenemos realizada la parte de la subrutina que va a ejecutar todas las operaciones necesarias para hacer cuatro SPRITES, como únicamente tenemos los tres primeros, vamos ahora a dibujar el cuarto, para posteriormente confeccionar la DATA correspondiente a cada uno.

NORMAS PARA LA CONFECCION DE SPRITES

Antes de proceder al dibujo del cuarto SPRITE, vamos a ver algunas de las condiciones que es necesario cumplir con este sistema de representación gráfica.

1. La utilización de SPRITES por medio del MSX BASIC, está restringida a los **modos** gráficos 1 y 2.
2. No podemos utilizar distintos tipos de SPRITES en la misma imagen.
3. El número **máximo** de SPRITES que podemos poner simultáneamente en una línea (entendiendo como línea cualquiera de las 192 líneas horizontales que tiene la matriz de nuestra pantalla) es de cuatro, independientemente del tamaño de los mismos.
4. Cada SPRITE no puede tener más que un color.

Examinando las condiciones 1 y 2 vemos que las cumplimos plenamente, ya que el modo gráfico que utilizamos es el "1" (SCREEN 2) y tenemos únicamente un tipo de SPRITE (16 x 16).

En cuanto a lo que se refiere a las condiciones 3 y 4, vamos a examinarlas para ver si nuestro dibujo se ajusta a las mismas.

La condición número 3 la cumplimos hasta ahora, puesto que sólo tenemos tres SPRITES coincidentes en una misma línea, es decir, nos podemos permitir el lujo de poner otro más.

La condición número 4 es precisamente la que motiva nuestro cuarto SPRITE, ya que únicamente podemos dar un color y nuestra plataforma quedaría un poco **desangelada,** así que vamos a aprovechar la capacidad que tienen los SPRITES para superponerse y poner **dos** en la parte central.

Lógicamente, si ponemos dos iguales, uno sobre el otro, no veremos nada más que aquél que tenga mayor **prioridad** (volveremos sobre este tema). Ya que nuestra intención es la de dar un poco más de variedad cromática a nuestro dibujo, al mismo tiempo que le añadimos detalle, vamos a hacer que el SPRITE central **posterior** sea amarillo, dado que este color resalta poco con el verde y el gris, vamos a **enmarcarlo** en negro con el SPRITE **anterior,** tal y como está representado en la Fig. 31.

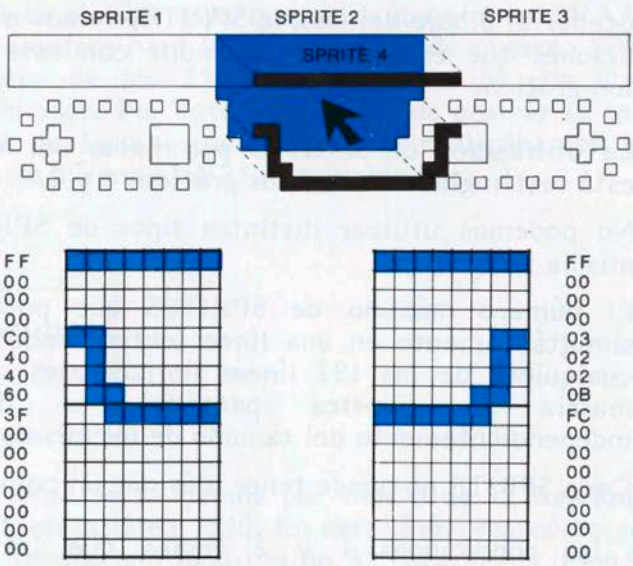


Fig. 31 SPRITE nº 4 y superposición del mismo

Una vez que ya tenemos nuestro cuarto SPRITE vamos a confeccionar la DATA que nos falta para completar este bloque o subrutina (otro más de nuestros **ladrillos...**). Como creo que ya hemos hablado suficiente sobre cómo se codifica (¿demasiado...?), podemos escribir directamente las siguientes líneas, en cualquier caso, puedes comprobar la DATA con cada uno de los SPRITES.

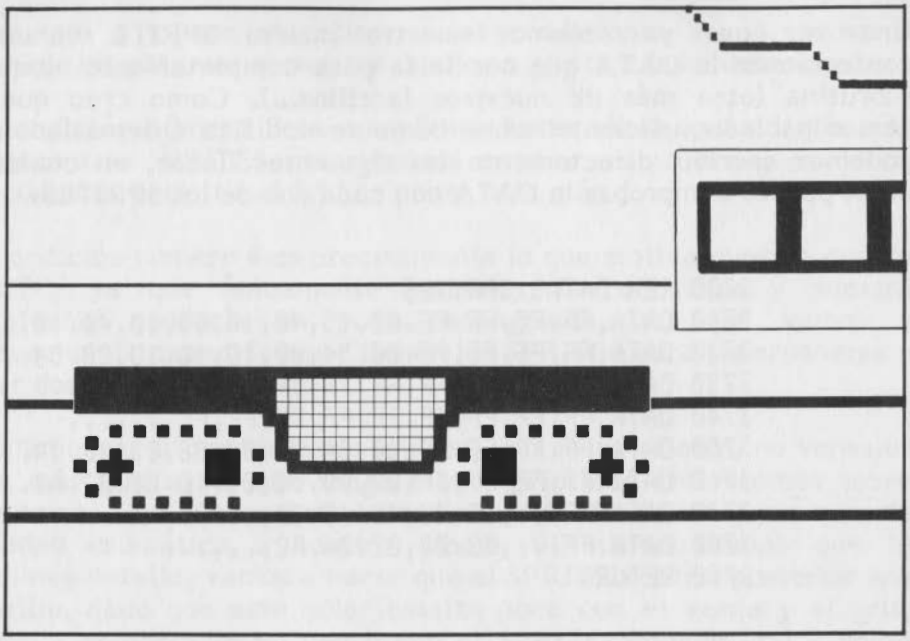
```
3700 REM DATAS SPRITES
3710 DATA FF,FF,FF,FF,00,15,40,10,B8,10,40,15,,,,
3720 DATA FF,FF,FF,FF,00,54,00,1D,1C,1D,00,54,,,,
3730 DATA FF,FF,FF,FF,FF,7F,7F,7F,3F,,,,,,,
3740 DATA FF,FF,FF,FF,FF,FE,FE,FE,FC,,,,,,,
3750 DATA FF,FF,FF,FF,00,2A,00,B8,38,B8,00,2A,,,,
3760 DATA FF,FF,FF,FF,00,A8,02,08,1D,08,02,A8,,,,
3770 DATA FF,,,,C0,40,40,60,3F,,,,,,,
3780 DATA FF,,,,03,02,02,06,FC,,,,,,,
3790 RETURN
```

Bien, ya está acabada esta parte, pero todavía no podemos visualizar nuestra plataforma, para eso necesitamos hacer la subrutina del capítulo siguiente. ¡Tranquilo que es más corta! Antes de acabar, únicamente una observación, las comas que hay en las líneas de DATA, sin nada entre ellas, es debido a que el valor correspondiente a esta posición es "00", por lo tanto podemos omitir poner este valor, pero no así la coma que indica una posición de dato.

Llegó la hora de **salvar** nuestro programa, cambiemos la línea 10 de esta manera:

```
10 REM BASE5
```

Pongamos ahora a cero el contador y grabemos el programa mediante CSAVE "BASE5". No te olvides de comprobar la grabación antes de apagar el ordenador.



Input - Comunicándonos con Nuestro Programa

Esta es la sección que nos permitirá comunicarnos directamente con el programa por medio del teclado o un **joystick**, para ello vamos a emplear la **función** STICK (palo o palanca en inglés). Antes de seguir vamos a explicar algo sobre dos expresiones que venimos utilizando, **instrucción** y **función.**

INSTRUCCION

Instrucción es aquella **orden** que damos al ordenador para que ejecute una determinada tarea, esta orden debe ser dada haciendo uso de las expresiones correctas, en nuestro caso, las contenidas en el BASIC MSX, no podemos decir a nuestro ordenador **ESCRIBE "PEPE",** sino que habremos de **decirle PRINT "PEPE",** ya que la primera expresión no la acepta al no tener equivalencia en el BASIC MSX. Las instrucciones pueden usarse **directamente,** basta con incluirlas en el programa y dar los parámetros adecuados, cuando éstos son necesarios, para que la instrucción se ejecute.

FUNCION

Una función, a diferencia de lo que ocurre con las instrucciones, no puede ser empleada **directamente**, debiendo ser **ejecutada** mediante una instrucción, de la cual formará parte, o estar asignada a una variable o una constante.

STICK

Normalmente, una función cuenta con dispositivos y subrutinas internas para la realización de tareas específicas, en el caso de la que vamos a utilizar ahora, STICK, el ordenador tiene una serie de circuitos electrónicos dedicados única y exclusivamente a esta función, así como una subrutina que hace uso de los datos entregados por estos circuitos (esta subrutina no es como las que nosotros estamos realizando, ya que está confeccionada en código máquina).

Por aquello de que los ejemplos nos ayudan a comprender mejor las cosas, vamos a teclear las siguientes líneas haciendo uso de la función STICK.

```
STICK (0)
```

Pulsa RETURN y obtendrás el siguiente mensaje:

```
Syntax error
```

Probemos ahora a asignar esta **función** a una **instrucción:**

```
PRINT STICK (0)
```

Pulsa RETURN de nuevo:

```
0
Ok
```

Como ves, esta vez ha sido aceptado por el ordenador, vamos a hacer ahora una asignación de esta función a una variable, escribe:

```
A=STICK (0)
```

RETURN de nuevo, podemos comprobar que el ordenador acepta esta expresión, ya que nos contesta "Ok".

Bien, una vez aclarada la diferencia entre las funciones y las instrucciones, volvamos con nuestro programa y la función STICK.

Esta función va seguida de un número entre paréntesis, con este número lo único que indicamos es de dónde van a provenir los datos, del teclado o del joystick con arreglo a la siguiente tabla.

Número	Significado
0	La introducción de datos se hace desde las teclas del cursor.
1	Los datos se introducen desde el joystick nº 1.
2	Los datos se introducen desde el joystick nº 2.

En nuestro caso, vamos a utilizar el teclado, por lo tanto, el número debe ser "0", pero si quieres usar el joystick no tienes más que cambiar este número.

Esta **función** lo que nos da es un valor que depende de qué tecla del cursor está pulsada, o de qué posición tiene el joystick, la equivalencia entre este valor y las teclas está representada en la Fig. 32.

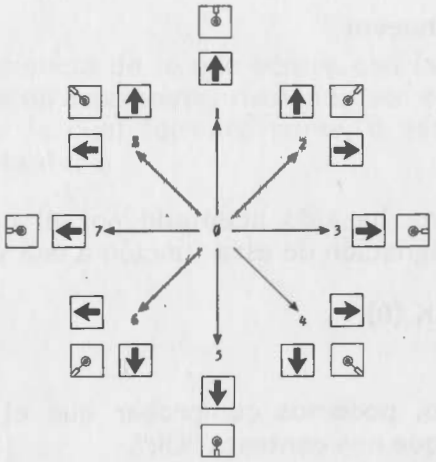


Fig. 32 Valores correspondientes a las teclas del cursor en la función STICK

Como puedes apreciar, el "0" central corresponde al valor que obtenemos cuando no está pulsada ninguna tecla (o el mando del joystick está en reposo). Los números pares, 2, 4, 6 y 8 se obtienen pulsando dos teclas a la vez, como está indicado en el dibujo.

La memorización de cada uno de los números y sus posiciones no representa dificultad, lo único que debemos hacer es recordar que el "1" se encuentra en la parte superior, y que los demás están distribuidos cada 45 grados, siendo el último el nº 8.

Aprovechando que todavía no hemos cargado nuestro programa, vamos a ver un corto ejemplo sobre STICK, escribe estas líneas:

```
10 LOCATE 20,12:PRINT STICK(0)
20 GOTO 10
```

Ahora pulsa RETURN y podrás comprobar (pulsando las teclas del cursor) cómo en el centro de la pantalla aparecen los números que corresponden a éstas, según el dibujo de la Fig. 32.

LOCATE

LOCATE es una **instrucción** con la cual situamos el cursor (ese cuadradito que está parpadeando en nuestra pantalla) en una posición determinada por los valores X/Y que la siguen. Esta instrucción se emplea únicamente en TEXTO, por lo que estos valores son de 0 a 39 para X y de 0 a 23 para Y, o sea, los valores máximos y mínimos de nuestra matriz de caracteres.

Había prometido que este capítulo sería corto y que veríamos nuestros SPRITES, acabemos entonces rápidamente ya que no es en éste, sino en el próximo donde los veremos... ¡Un poco de paciencia que ya queda poco!

Carga el programa (BASE5) y teclea las siguientes líneas:

```
100 REM INPUT
110 IF STICK (0)=3 THEN D=D+1:GOTO 150
120 IF STICK (0)=7 THEN D=D+1:GOTO 150
```

IF-THEN-ELSE

En las líneas 110 y 120 acabamos de utilizar una nueva **instrucción** IF-THEN. Esta es una instrucción fundamental en cualquier BASIC, ya que es la que nos permite la **bifurcación condicional** o la **ejecución selectiva** en nuestro programa (¡menudas expresiones!).

En un programa normalmente son usadas dos tipos de bifurcaciones, **condicionales** e **incondicionales,** ejemplo de bifurcaciones incondicionales son las que logramos mediante las instrucciones GOTO y GOSUB, las cuales son ejecutadas de forma inmediata, esto es, cuando el ordenador se encuentra en el programa cualquiera de estas instrucciones **va** directamente a la dirección indicada en las mismas, sin que nada le pueda condicionar a realizar algo distinto.

Por el contrario, en una **bifurcación condicional,** el ordenador irá (no te preocupes que no se va a mover de tu mesa) a la dirección indicada **únicamente** cuando se cumpla la condición puesta detrás de IF.

El caso de **ejecución selectiva** es el mismo que el de bifurcación, sólo que en esta ocasión de lo que se trata es de ejecutar una instrucción, esto es lo que realizamos en las líneas 110 y 120, "D" no cambiará de valor a no ser que se **cumpla** la condición de que la función STICK(0) sea igual a 3 ó 7, que como observarás en el dibujo de la Fig. 32, estos números corresponden a los desplazamientos laterales.

MULTI-INSTRUCCIONES

IF tiene también otra particularidad, las líneas que hemos tecleado son líneas **multi-instrucción,** contienen un GOTO después de dos puntos, este GOTO se ejecutará solamente cuando la condición STICK(0) sea cierta o coincidente con la que nosotros hemos fijado. Es decir, que a pesar de tratarse únicamente de una ejecución selectiva, realizamos al mismo tiempo una **bifurcación selectiva** ya que desde la línea 110 **saltaremos** a la 150 únicamente en el caso de que STICK(0) sea igual a "3".

Para clarificar un poco más todo lo anterior en la Fig. 33, está representada la carta de flujo de esta parte del programa.

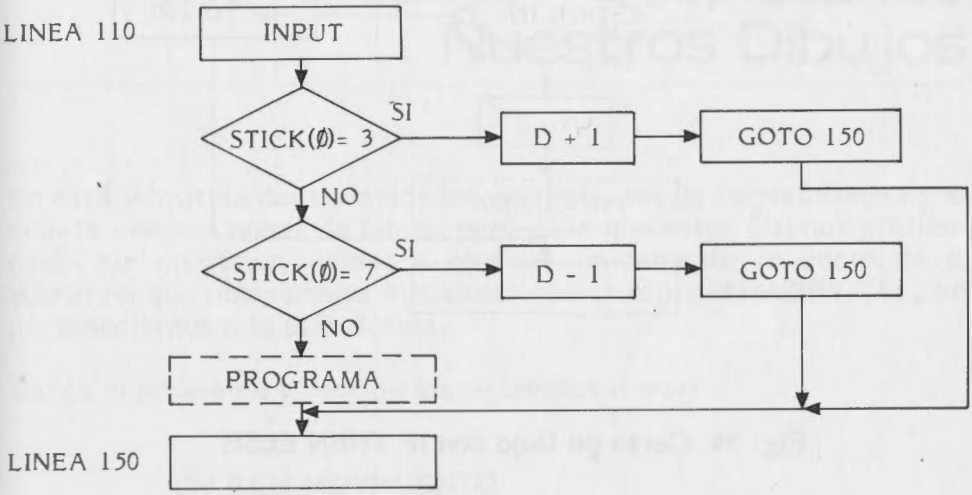


Fig. 33 INPUT Carta de flujo con IF THEN

Podemos conseguir lo contrario de lo que está representado en la Fig. 33 empleando otra versión de **IF**, IF THEN ELSE, IF es el **si** interrogativo (en inglés), THEN significa **entonces,** dentro de un condicional y ELSE, en el contexto de esta instrucción, viene a ser algo así como **si no has hecho lo anterior, haz lo siguiente** (traducción muy libre...).

Si cambiáramos la línea 110 de la siguiente manera:

110 IF STICK (0)=3 THEN D=D+1 ELSE GOTO 150

El significado sería el siguiente, "SI STICK (0) es igual a 3 ENTONCES D es igual a D+1, pero si esto no es así (STICK (0) distinto a 3) VETE a la línea 150". En la Fig. 34 tienes la carta de flujo de IF THEN ELSE aplicada a nuestro caso.

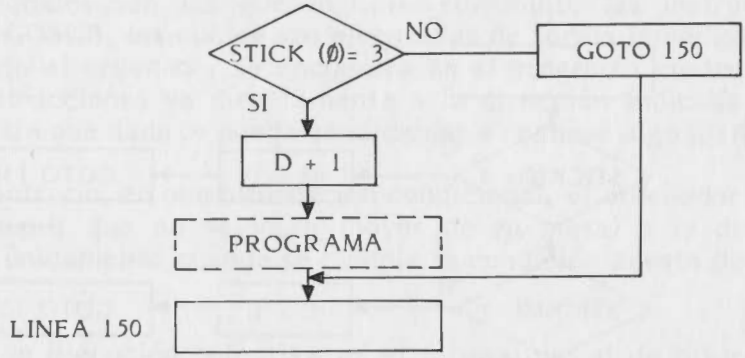


Fig. 34 Carta de flujo con IF THEN ELSE

Como verás el flujo es distinto al que obteníamos en la Fig. 33, ahora vamos a la línea 150 cuando **no** se cumple la condición. Espero que no cambiarás la línea en 110, en cualquier caso debe quedar como estaba en el ejemplo anterior.

Salva el programa cambiando la línea 10 de nuevo:

```
10 REM BASE6
```

No te olvides de comprobar la grabación antes de apagar el ordenador, ¿que soy un pesado...?

Movimiento - Desplazando Nuestros Dibujos

En esta subrutina controlamos los movimientos de la plataforma y el cohete con sus nubes de humo, pero dado que estos últimos gráficos están sin construir, vamos a realizar únicamente la parte de la subrutina que nos permita visualizar nuestros primeros SPRITES, los pertenecientes a la plataforma.

Carga el programa y escribe las siguientes líneas:

```
150 REM MOVIMIENTO
200 PUT SPRITE 4,(216+D,168),1,1
210 PUT SPRITE 3,(232+D,168),10,2
220 PUT SPRITE 2,(248+D,168),1,3
230 PUT SPRITE 1,(232+D,168),1,4
240 GOTO 110
```

Arranca el programa ahora y podrás ver la plataforma en el lado izquierdo de la pantalla, utiliza las teclas del cursor que señalan hacia derecha o izquierda y podrás desplazarla en ambos sentidos.

Ya has visto cómo puedes mover los SPRITES por **encima** del camino, sin que éste resulte destruído y sin que ambos gráficos se mezclen, pero también te habrás dado cuenta de que si mantienes pulsada una tecla, la plataforma **desaparece por un extremo apareciendo por el otro.**

Esto es algo que tenemos que solucionar, para lo cual emplearemos nuestra vieja conocida "IF", pero antes de empezar a hacer añadidos a nuestro pequeño programa realicemos un análisis de las instrucciones que hemos empleado para poner la plataforma en la pantalla, y conseguir que ésta se mueva bajo nuestro control.

PUT SPRITE

Hay dos tipos de instrucciones relacionadas con los **sprites,** las que nos sirven para **posicionarlos** en la pantalla, y las relativas a las **colisiones** entre ellos.

Relacionada con las primeras únicamente tenemos PUT SPRITE (put significa poner) que es la utilizada en nuestro programa. Esta instrucción utiliza el mismo método que empleamos en PSET y CIRCLE para posicionar nuestro **sprite,** las coordenadas horizontales y verticales dentro de un paréntesis y separadas por una coma, con estos parámetros lo que nosotros definidos es un punto, ¿pero a qué parte del **sprite** corresponde este punto...?

El punto indicado en la instrucción PUT SPRITE corresponde a la parte superior izquierda de la **matriz** en la que está contenido nuestro **sprite,** independientemente del dibujo de éste, tal y como está representado en la Fig. 35.

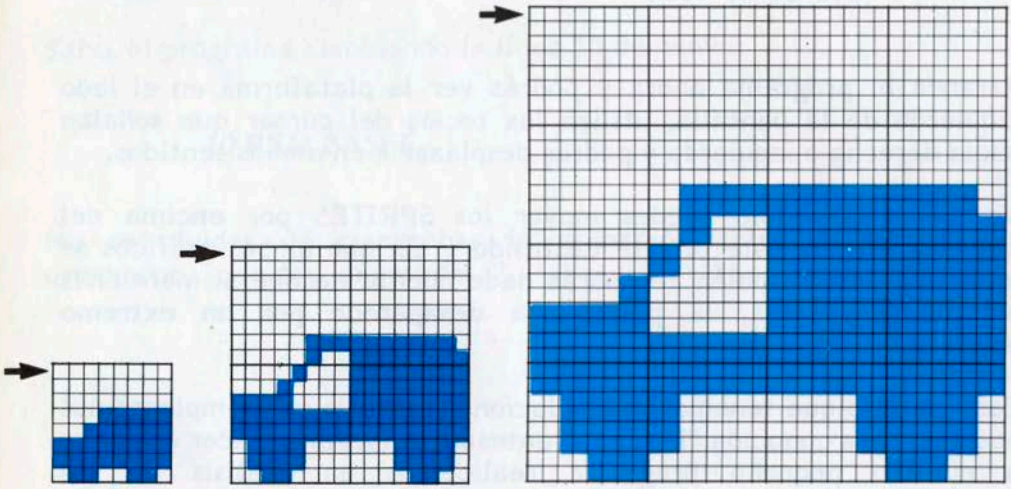


Fig. 35 Punto de referencia para posicionar un 'sprite'

PRIORIDAD-COLOR-NUMERO

Delante del paréntesis vemos un número seguido de una coma, con este número indicamos la **prioridad** o posición relativa que tendrá un **sprite** cuando coincida o se solape con otro.

El **sprite** que posea el número más bajo quedará en primer plano **superpuesto** al de número más alto, este número puede ser cualquiera entre 0 y 31 ya que el máximo de **sprites** que podemos poner en pantalla simultáneamente es de 32.

Inmediatamente detrás del paréntesis, precedido de una coma, encontramos otro número. Con éste indicamos el color que va a tener nuestro dibujo en las partes que hayamos **rellenado** de la matriz.

Por último tenemos el número que está al final de la instrucción con el cual definimos cuál es el **sprite** que se va a poner en la posición indicada con la prioridad y el color señalado en ésta, este número es el mismo que asignamos a la **variable del sistema** SPRITE$ durante su confección.

En las cuatro instrucciones utilizadas para posicionar nuestra plataforma vemos que en la coordenada horizontal (X) sumamos el valor de la variable "D", ésta es la misma variable que utilizábamos en la subrutina INPUT, en las líneas 110 y 120.

Dada la finalidad que tiene en nuestro programa podemos denominarla "Desplazamiento", puesto que dependiendo de su valor conseguiremos mover hacia derecha o izquierda nuestra plataforma.

COLISIONES

En el segundo grupo, el relacionado con la colisión entre **sprites**, tenemos dos tipos de instrucciones.

1ª SPRITE/ON/OFF/STOP, con la cual activamos (ON) la **detección** de colisiones, o bien desactivamos (OFF) esta detección.

2ª ON SPRITE GOSUB, que dirige el programa a una subrutina cuando habiendo activado la instrucción, SPRITE ON, se produce una colisión.

Tenemos todavía la instrucción SPRITE STOP, que no quiere decir que cuando colisionen dos **sprites** el programa se va a parar.

Con esta instrucción el sistema de detección sigue trabajando, lo que sucede es que al registrarse una colisión, ésta queda **memorizada** por nuestro ordenador sin que se ejecute ON SPRITE GOSUB, sin embargo, debido a esta memorización, al encontrarse posteriormente en el programa la instrucción SPRITE ON, se ejecutará **inmediatamente** la subrutina indicada por ON SPRITE GOSUB, independientemente del tiempo transcurrido desde que la colisión aconteció.

SPRITE$

Aparte de estas instrucciones tenemos la **variable del sistema** SPRITE$, que es la que utilizamos para la confección de éstos, los **sprites** forman parte inherente del procesador de vídeo con que cuentan todos los equipos MSX, este procesador es el modelo TMS9118A, perteneciente a la casa Texas Instruments.

Una vez que hemos visto las instrucciones relacionadas con los **sprites** vamos a poner límite al desplazamiento de nuestra plataforma, como ya dijimos al principio de este capítulo, vamos a utilizar la instrucción IF.

LIMITES

Primeramente, vamos a ver cuáles son las posiciones máxima y mínima que podemos dar a nuestro desplazamiento, en sentido horizontal la posición máxima que podemos emplear en la instrucción, PUT SPRITE, es la de 255, dado que el **sprite** número 3 tiene ya asignado un valor de 248, el valor máximo que debemos permitir a "D" es de 7, ya que 248+7=255, por lo tanto, vamos a escribir una instrucción que compruebe el valor de "D" y le limite a un máximo de 7, escribe:

```
160 IF D > 7 THEN D=7
```

("Si D es superior a 7 entonces **hagamos** D igual a 7"). Una vez que tenemos limitado nuestro desplazamiento máximo, vamos a hacer lo mismo con el mínimo, este mínimo va a venir determinado por la posición que ocupa la torre de lanzamiento. En la línea 5130 fijábamos la posición horizontal para el inicio del dibujo de la torre en 9 (PSET (9,A),1), por lo tanto si a 9 le sumamos los 15 pixels que nuestra torre tiene de ancho (el dibujo base tenía 21 pero al aplicarle el factor de escala 3 (S3) quedó reducido a 15) nos da un total de 24.

El **sprite** con menos valor horizontal es el número 1 (línea 200) con 216, restando a esta cantidad los 24 pixels del extremo derecho de nuestra torre nos dará un valor de 192 para "D", este valor debe ser negativo, ya que "D" se suma al valor de "X" ya existente. Escribamos una instrucción que nos limite el desplazamiento mínimo a estos valores.

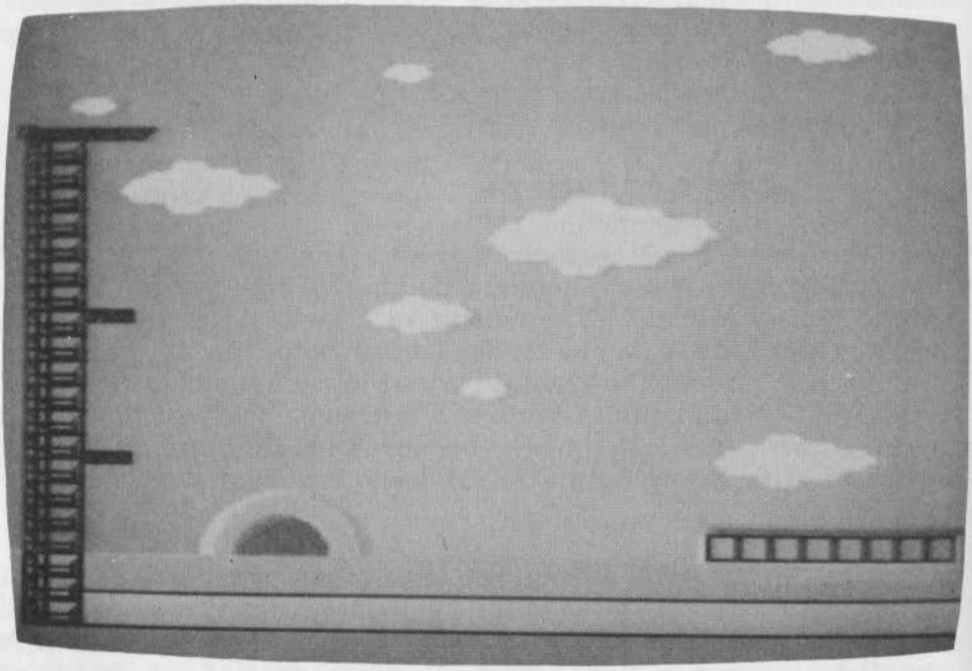
```
170 IF D < -192 THEN D=-192
```

Corre ahora el programa y podrás comprobar cómo la plataforma queda limitada en sus movimientos extremos.

Esta subrutina tendremos que retocarla posteriormente, cuando hagamos nuestro cohete, ya que es aquí desde donde ordenaremos su despegue. Lista ahora la línea 10 y cámbiala de esta manera:

```
10 REM BASE7
```

Salva ya el programa bajo este nombre (CSAVE "BASE7") y procede a la comprobación del mismo como en las veces anteriores.



Cohete - La Nave Columbia

REEDICION

Este capítulo nos va a servir para que utilicemos una de las facilidades que nos brinda nuestro ordenador MSX, se trata de la posibilidad de **reeditar** un programa.

A veces, dentro de un programa determinado, se emplean secciones que son similares entre sí y que tenemos que estar repitiendo con muy pocas variaciones entre ellas. Gracias al sistema de edición con que contamos en nuestros equipos, nosotros podemos evitarnos estas repeticiones reduciendo el tiempo de confección de nuestros programas.

Como vale más un ejemplo que mil palabras, vamos a realizar uno que nos sirva para construir la subrutina **COHETE.**

Carga el programa y lista desde la línea 3500 a la línea 3700; únicamente teclea:

LIST 3500 - 3700

Pulsa RETURN y te aparecerá en la pantalla la parte del programa que utilizamos para realizar la plataforma. Este programa puede servirnos perfectamente cambiando algunos detalles para construir los **sprites** de nuestro cohete, únicamente falta la DATA, pero lógicamente ésta es distinta a la que empleamos anteriormente.

Una vez que tenemos listado el programa, vamos a situar el cursor mediante las teclas justo **sobre** la **L** de **LIST;** ahora pulsa la tecla correspondiente a **AUTO** (normalmente F2) o bien escribe directamente este comando (AUTO) seguido de 2000.

Ahora **borra** el resto de la línea (-3700) pulsando la tecla CTRL y la E. Pulsa RETURN una vez por cada una de las líneas, como verás éstas van tomando la nueva numeración, cuando ésta alcance el número 2120 pulsa CTRL y STOP para desconectar el comando AUTO.

Borra ahora la pantalla (CLR) y procede a listar el programa. Comprobarás cómo tenemos las nuevas líneas sin que hayan desaparecido las que nos sirvieron de **patrón** (si por cualquier circunstancia se te borrara una parte del programa, procede a cargarlo de nuevo y repite cuidadosamente las operaciones descritas).

Bien, ahora lista la parte **insertada:**

```
LIST 2000-2110
```

Si utilizáramos en esta subrutina este programa tal y como está, volveríamos a generar de nuevo los mismos **sprites** de la plataforma y eso no es lo que pretendemos. Lo primero que debemos cambiar son los parámetros de la línea 2020 para que queden de esta forma:

```
2020 FOR A=5 TO 8
```

Esta corrección puedes hacerla también directamente posicionando el cursor sobre el número a modificar y escribiendo el nuevo, siempre que emplees este sistema de edición ten en cuenta que para que ésta sea efectiva, debes pulsar RETURN al acabar la corrección en una línea.

El cohete que vamos a construir es la célebre **lanzadera espacial** COLUMBIA que se utiliza para poner satélites y otras zarandajas alrededor de nuestro planeta. Como sabes, esta lanzadera está acoplada a dos impulsores que junto con un depósito gigantesco alimenta los motores que la ponen en órbita. El dibujo de este conjunto es algo parecido al representado en la Fig. 36.

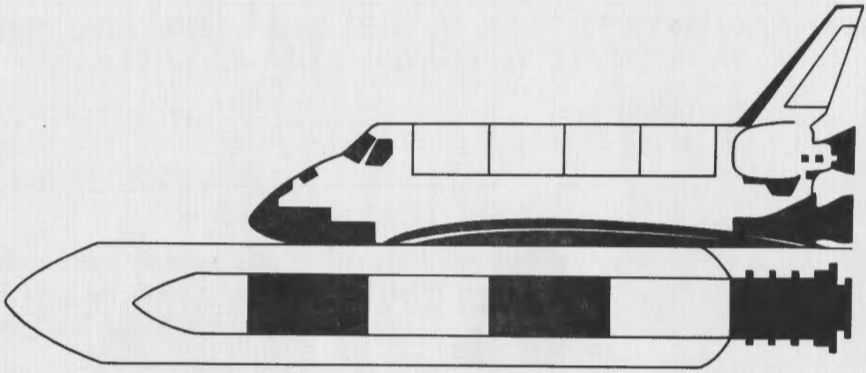


Fig. 36 Conjunto Columbia con su impulsor

El total de **sprites** que vamos a utilizar es de catorce (¡No te preocupes, que no vas a tener que **codificarlos** de nuevo...!) y van a ir repartidos en cuatro grupos, dos correspondientes a los **fondos** de las figuras y dos a los **detalles.**

En la Fig. 37 tienes la distribución de los **sprites** que forman ambos conjuntos, el número que figura al lado es el correspondiente al **sprite** como puedes observar en los dibujos del **fondo** de la nave y el tanque empleamos el mismo **sprite** (6) varias veces, con lo cual simplificamos la realización de esta parte del dibujo. En el impulsor también recurrimos al mismo sistema utilizando el **sprite** número 17 cuatro veces.

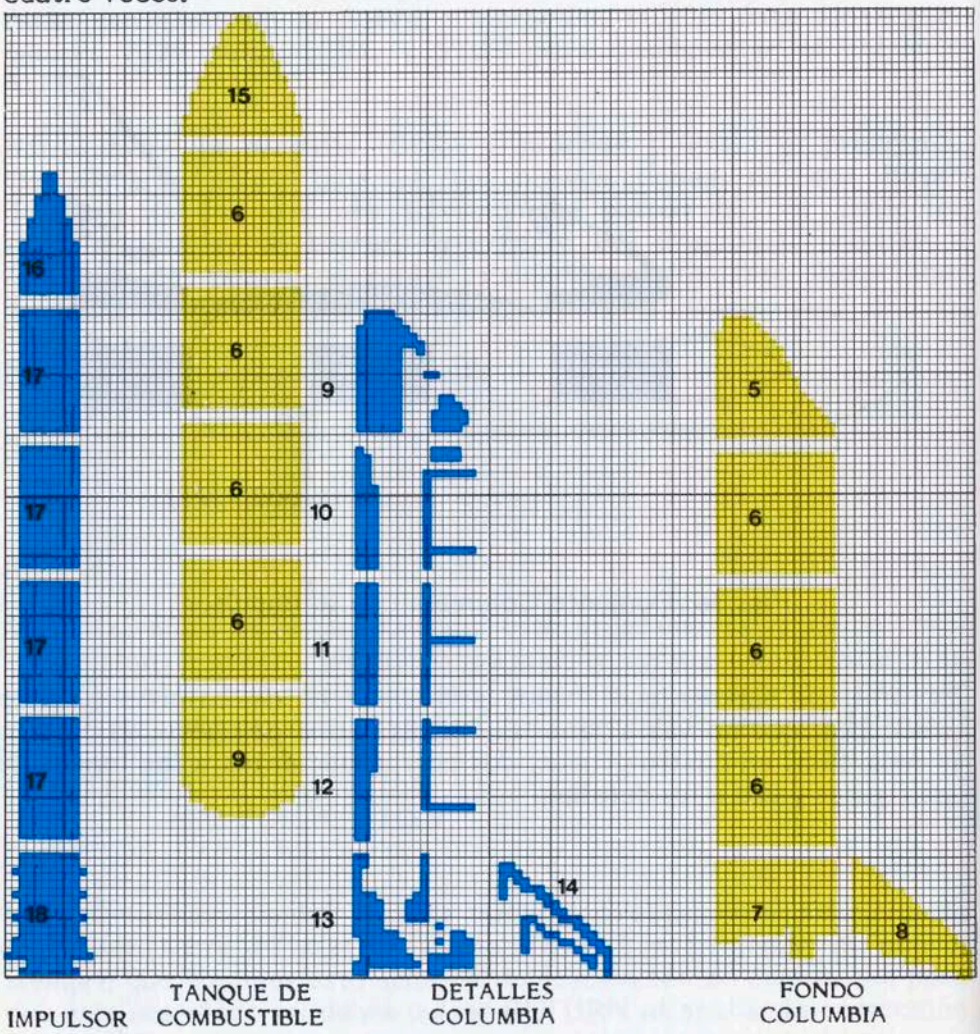


Fig. 37 Detalle sprites Columbia

Vamos a teclear las líneas de DATA correspondiente al **fondo** del cohete, escribe:

```
2170 REM **FONDO COHETE**
2180 DATA 78,7C,FE,FF,FF,FF,FF,FF,FF,FF,FF,FF,FF,FF,FF,FF
     ,,,,,,80,80,C0,C0,E0,E0,F0,F8,FC,FE,FF,FF
2190 DATA FF,FF,FF,FF,FF,FF,FF,FF,FF,FF,FF,FF,FF,FF,FF,FF
     ,FF,FF,FF,FF,FF,FF,FF,FF,FF,FF,FF,FF,FF,FF,FF,FF
2200 DATA FF,FF,FF,FF,FF,FF,FF,FF,FF,FF,FF,E0,,,,,,FF,FF,FF,
     FF,FF,FF,FF,FF,FF,BF,38,38,38,,,
2210 DATA 80,E0,F0,F8,FE,FF,FF,FF,FF;FF,3F,3F,0F,01,,,,,,
     ,,,80,C0,F0,F8,FC,FE,FF,FF,1F,03


2360 RETURN
```

Puedes comprobar la equivalencia entre la DATA y el dibujo teniendo en cuenta que el **sprite** 5 está en la línea 2180, el 6 en la 2190 y así sucesivamente.

Añadamos ahora las instrucciones necesarias en la subrutina MOVIMIENTO para poder visualizar los **sprites** que hemos realizado, escribe las siguientes instrucciones:

```
290 PUT SPRITE 9,(248+D,152+H),15,8
300 PUT SPRITE 19,(232+D,152+H),15,7
310 PUT SPRITE 18,(232+D,136+H),15,6
320 PUT SPRITE 17,(232+D,120+H),15,6
330 PUT SPRITE 16,(232+D,104+H),15,6
340 PUT SPRITE 15,(232+D,88+H),15,5
```

Dado que MOVIMIENTO es una subrutina y que cualquier subrutina que se precie debe tener un RETURN, vamos a ponerle a ésta el suyo. Elimina la línea 240 escribiendo este número y pulsando RETURN, escribe ahora:

```
480 RETURN
```

Corre el programa y observarás cómo al aparecer la plataforma se borra la pantalla dándonos el error siguiente.

RETURN without GOSUB in 340

Esto es debido a que hemos **roto** el lazo que existía desde la línea 240 (GOTO 110) a la sección INPUT (100 - 120) y ahora seguimos **entrando** directamente a la subrutina MOVIMIENTO (150) por lo que al llegar en ésta a la línea 340 y encontrarse con la instrucción RETURN sin que haya habido previamente una instrucción GOSUB para el programa y nos da el error que hemos obtenido.

Vamos a cambiar un poco nuestra sección INPUT para que esto no suceda, lista 100 -120.

```
LIST 100 - 120
```

Modifica las líneas 110 -120 y añade la 140.

```
110 IF STICK(0)=3 THEN D=D+1:BEEP:GOTO 140
120 IF STICK(0)=7 THEN D=D-1:BEEP:GOTO 140


 140 GOSUB 150:GOTO 110
2010 RESTORE 2170
```

Prueba ahora y verás cómo aparece un COLUMBIA blanco sobre nuestra plataforma, desplazándose con ésta cuando pulses las teclas del cursor. Bueno, ya que tenemos el fondo de la nave ¿qué te parece si hacemos los detalles...? Escribe las siguientes líneas de DATA:

```
2220 REM **DETALLE COHETE**
2230 DATA 78,7C,FE,FF,FF,FC,FC,FC,FC,FC,FC,FC,F8,F8,F8,F8
     ,,,,,80,80,,,60,,,18,1C,3E,3E,3C
2240 DATA 80,C0,C0,C0,C0,E0,E0,E0,E0,E0,E0,E0,E0,E0,E0,E0
     ,3C,3C,,7F,40,50,50,50,50,40,40,40,40,7F,40,40
2250 DATA E0,E0,E0,E0,E0,E0,E0,E0,E0,E6,E6,E0,E0,E0,E0,E0,E0
     ,40,40,40,40,40,40,40,40,7F,40,40,40,40,40,40,40
2260 DATA EC,EC,EC,EC,EC,EC,EC,CC,CC,CC,CC,CC,CC,CC,CC,CC
     ,40,7F,40,40,40,40,40,40,40,40,40,7F,,,1C,20
2270 DATA C0,C0,80,80,80,E1,F3,F3,F3,F0,FC,7E,7E,7F,FF,C0
     ,40,40,40,40,40,C0,C0,C0,C0,10,06,17,07,3F,3F,30
2280 DATA 0,60,70,58,4E,43,01,,0C,A,9,8,88,80,C0,,,,,,,,
     80,C0,70,18,8C,66,13,D,5,1
```

De nuevo tenemos que volver a la subrutina MOVIMIENTO para poder visualizar nuestros **sprites,** teclea las siguientes líneas:

```
240 PUT SPRITE 14,(232+D,152+H),1,13
250 PUT SPRITE 13,(232+D,136+H),1,12
260 PUT SPRITE 12,(232+D,120+H),1,11
270 PUT SPRITE 11,(232+D,104+H),1,10
280 PUT SPRITE 10,(232+D,88+H),1,9
```

Lista ahora la línea 2020 y modifica el número final para que se adecúe a la nueva DATA, añadamos los nuevos 'sprites' modificando la línea 2020:

```
2020 FOR A=5 TO 13
```

¿Qué te parece como va quedando nuestra nave...? Bueno, creo que ha llegado el momento de hacer un alto y **salvar** nuestro programa según está, no vaya a ser que tengamos un apagón y... ¡mejor no pensarlo!

Lista la línea 10 y cámbiala así:

```
10 REM BASE8
```

Salva ahora el programa siguiendo las reglas dadas anteriormente.

Siempre que realices un programa de esta longitud es aconsejable que realices grabaciones parciales del mismo como precaución.

LIMITACION POSICIONAMIENTO

Te habrás dado cuenta que la cola de nuestra nave carece del detalle que hemos construido para ella en el **sprite** nº 14. Realmente no hemos puesto este **sprite** en la subrutina MOVIMIENTO, la razón es la de que, como recordarás, únicamente podemos poner **cuatro** sprites en la misma línea y actualmente ya tenemos tres, así que si ponemos otro más no podremos posicionar el cohete impulsor. Bien, una vez conocida la razón, nada nos impide que posicionemos **provisionalmente** este **sprite** (14):

```
350 PUT SPRITE 7,(248+D,152),1,14
```

SPRITES - POSIBILIDADES

Tiene buen aspecto ¿verdad? Bueno, pues una vez que tenemos la nave vamos a ver algunos detalles en el sistema que hemos empleado para posicionarla. Primeramente vemos que en las líneas 310, 320 y 330 ponemos el mismo **sprite,** concretamente el número 6, en distintas posiciones. Realmente, el número de sprite lo único que determina es una figura y ésta la podemos repetir tantas veces como queramos hasta un máximo de 32 puestos en la pantalla, si bien podemos **almacenar** hasta 256.

¿Quiere esto decir que **todos** los sprites con el número 6 deben ser iguales? En la forma sí, pero no en el color, ya que podemos utilizar cualquiera de los 16 disponibles para colorear cualquiera de ellos. Asimismo, la **prioridad** debe ser distinta entre **todos** los que estén en la pantalla.

Después de esta explicación si observas de nuevo la subrutina MOVIMIENTO, comprobarás cómo todas las prioridades (el primer número delante del paréntesis) son distintas en las instrucciones PUT SPRITE. En esta subrutina acabamos de incorporar una nueva variable, se trata de "H". El empleo que posteriormente la daremos será el de conseguir que nuestra nave y su impulsor se eleven; éste es el motivo por el cual está en la coordenada vertical. El principio es el mismo que empleamos con "D" para el desplazamiento horizontal.

Referente al posicionamiento vertical, verás que esta coordenada está separada por una distancia de 16 pixels de la que le antecede en la misma figura (fondo o detalles), esto es lógico ya que empleamos sprites de 16 x 16 y de lo que se trata es de ponerlos uno a continuación del otro sin que se solapen.

Creo que es el momento para poner el impulsor en escena, así que como esta DATA es más corta, vamos a realizar el tanque y el impulsor al mismo tiempo, escribe:

```
2290 REM **FONDO IMPULSOR**
2300 DATA 1,3,7,7,F,F,1F,1F,3F,3F,7F,7F,7F,FF,FF,FF,80,C0
     ,E0,E0,F0,F0,F8,F8,FC,FC,FE,FE,FE,FF,FF,FF
2310 REM *DETALLE IMPULSOR**
2320 DATA 1,1,1,3,3,3,7,7,7,F,F,F,F,F,F,F,80,80,80,C0,C0,
     C0,E0,E0,E0,F0,F0,F0,F0,F0,F0,F0
2330 DATA F,F,F,F,F,F,F,F,F,F,F,F,F,F,F,F,F0,F0,F0,F0,F0,
     F0,F0,F0,F0,F0,F0,F0,F0,F0,F0,F0
2340 DATA F,F,1F,F,F,1F,F,F,1F,F,F,1F,1F,3F,7,F,F0,F0,F8,
     F0,F0,F8,F0,F0,F8,F0,F0,F8,F8,FC,E0,F0
2350 DATA FF,FF,FF,FF,FF,FF,FF,FF,FF,FF,FF,FF,7F,7F,1F,7,
     FF,FF,FF,FF,FF,FF,FF,FF,FF,FF,FF,FE,FE,F8,E0
```

Cambiemos ahora la línea 2020 con el fin de que sea leída esta nueva DATA:

2020 FOR A=5 TO 19

Bien, y éste es el momento de irnos de nuevo a nuestra conocida subrutina MOVIMIENTO para que nos posicione estos nuevos sprites:

```
350 PUT SPRITE 26,(216+D,136+H),14,19
360 PUT SPRITE 27,(216+D,120+H),14,6
370 PUT SPRITE 28,(216+D,104+H),6,6
380 PUT SPRITE 29,(216+D,88+H),14,6
390 PUT SPRITE 30,(216+D,72+H),14,6
400 PUT SPRITE 31,(216+D,56+H),6,15
410 IF DS=5 THEN RETURN
420 PUT SPRITE 20,(216+D,152+H),1,18
430 PUT SPRITE 21,(216+D,136+H),15,17
440 PUT SPRITE 22,(216+D,120+H),1,17
450 PUT SPRITE 23,(216+D,104+H),15,17
460 PUT SPRITE 24,(216+D,88+H),1,17
470 PUT SPRITE 25,(216+D,72+H),15,16
```

En la línea 410 vemos que se debe cumplir la condición de que DS es distinta a 5 para que sean ejecutadas las líneas siguientes, esta variable (DS) la utilizaremos después, cuando hagamos el **desacople** y como observarás, está situada separando la parte correspondiente al tanque (líneas 420-470) y los impulsores (líneas 350-400) de tal manera que si hacemos DS=5 y accedemos a la subrutina por la línea 350 únicamente se desplazarán los impulsores ya que **retornará** en la línea 410 antes de llegar al posicionado del tanque de combustible.

Modifiquemos ahora la línea 170 con el fin de obtener un sonido cuando el cohete coincide con la torre de lanzamiento:

```
170 IF D<-179 THEN D=-179:PLAY "S10M500C"
```

Si no quieres hacer todo el recorrido con la plataforma cuando se inicia el programa puedes dar un valor inicial a "D" con el fin de que se sitúe en un lugar más próximo a la torre prueba por ejemplo con -172 (-179 es justo sobre la torre y por lo tanto el valor mínimo que puedes poner).

Cambia el bloque de iniciación de esta forma:

```
20 CLEAR:D=-172
30 SCREEN 2,2
40 COLOR 1,5,1:CLS
```

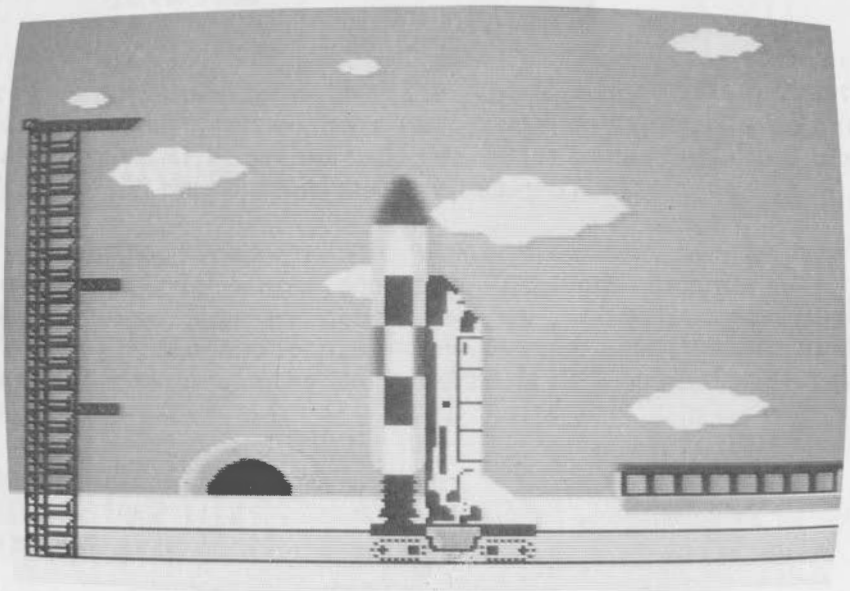
El cambio de la línea 20 es con el fin de que siempre obtengamos el mismo color de fondo al arrancar el programa después de haber usado otra selección de colores. Arranca el programa ahora y verás dónde te aparece el cohete.

Sigues teniendo la posibilidad de moverle hacia ambos lados, la única diferencia es la posición inicial. Si te gusta otra posición puedes cambiar el valor de "D" entre 0 y -179.

Finalizada la subrutina con la que confeccionamos los sprites de nuestro cohete, lo que necesitamos ahora es hacerle despegar, pero esto será en el capítulo que viene. Lista ahora la línea 10 y cambia el título así:

```
10 REM BASE9
```

Salva el programa como siempre y no te olvides de comprobar que la carga ha sido correcta.



Gráficos Móviles (M1)

En esta subrutina vamos a construir las llamas y el humo que se desprenden en el despegue y el ascenso y para su realización vamos a emplear el mismo sistema de **reedición** que utilizamos anteriormente en la subrutina COHETE. Comienza listando las líneas 2000 - 2170:

```
LIST 2000 - 2170
```

Sitúa ahora el cursor justo sobre la letra "L" de LIST como hiciste anteriormente y pulsa o escribe el comando AUTO seguido del número 1000.

```
AUTO 1000
```

Ahora borra el resto de la línea bien con CTRL y "E" o mediante la barra espaciadora, de manera que quede únicamente el ejemplo superior. Pulsa RETURN en cada una de las líneas hasta que llegues al número 1130 donde debes pulsar CTRL y STOP.

Ya tenemos numerado nuestro nuevo programa, vamos ahora a adaptarle para la nueva subrutina; debemos cambiar las tres primeras líneas que es donde está el nombre, la instrucción RESTORE y el bucle o loop, también es necesario cambiar la línea correspondiente al nombre de la DATA (1120). Procedamos a estos cambios:

```
1000 REM GRAFICOS M1/VENTANA
1010 RESTORE 1120
1020 FOR A=20 TO 23

1120 REM **LLAMAS COHETE**
```

Los sprites que vamos a realizar son los que están representados en la figura 38 y como puedes ver son tres, el nº 20 que corresponde al humo y los números 21 y 22 con los que realizamos las llamas. Por lo tanto, el loop "A" debe ser desde 20 hasta 23. Te habrás dado cuenta que los programas con los cuales realizamos los sprites son prácticamente iguales y que podíamos haber utilizado únicamente uno de ellos con un bucle "A" que fuera de 1 hasta 22 poniendo toda la DATA junta.

Efectivamente, esto es completamente factible de realizar, pero lo que se ha pretendido haciéndolo de esta manera es la de asignar una función completa a una subrutina o bloque para que sea más fácil de comprender su funcionamiento dentro del conjunto que forma un programa.

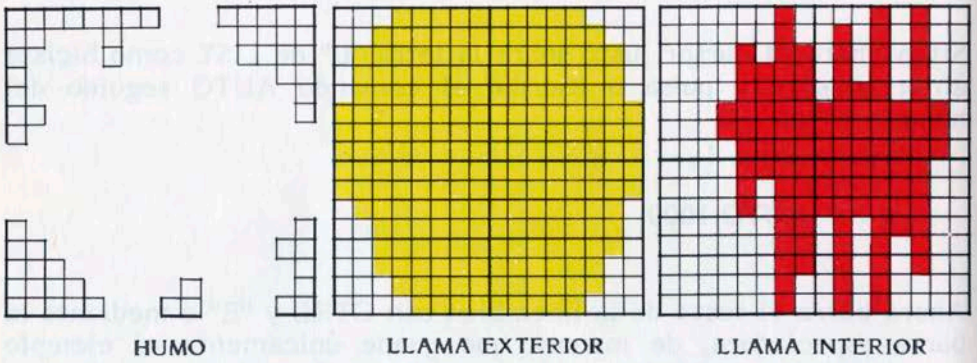


Fig. 38 Las llamas y humo de nuestro cohete

Tecleemos ahora la data correspondiente a estos dibujos:

```
1130 DATA ,3B,7B,7F,7F,3F,7F,FF,FF,FF,FF,7F,3F,1F,F,7,E0,
     F0,F0,FC,FE,FE,FF,FF,FF,FF,FF,FE,FC,FE,BE,38
1140 DATA 1F,3F,7F,7F,7F,FF,FF,FF,FF,FF,7F,7F,7F,7F,3F,1F
     ,F8,FC,FE,FE,FE,FF,FF,FF,FF,FF,FE,FE,FE,FC,FC,F8
1150 DATA 2,A,B,1F,1F,1F,2F,2F,1F,1F,17,16,6,6,4,4,8,48,
     48,48,68,FC,FC,FC,F8,F8,F8,B8,A8,A8,A0,A0

1280 RETURN
```

Con esto ya tenemos finalizada esta subrutina (¿corta verdad...?) es el momento de despegar, pero eso es algo que hacemos en MOVIMIENTO... así que de momento lista la línea 10 y cambia de nuevo el nombre:

```
10 REM BASE 10
```

GRABACION

Dado que ya tenemos 10 **subprogramas** grabados secuencialmente y que es una cantidad bastante respetable, puedes rebobinar la cinta hasta el principio y comenzar de nuevo desde ese punto, pues aunque cometieras un error ahora siempre tendrías los programas que grabaste después del número 1 hasta el último, que es el 9 (grabado). Si por otro lado quieres seguir como hasta ahora, no hay ningún inconveniente, así que salva el programa y comprueba su grabación... (¿pesado otra vez?).

Movimiento (2)

Ya dijimos anteriormente, cuando realizamos esta subrutina, que era parcial, vamos ahora a plantearnos todas las funciones que debe tener y su distribución.

En el capítulo en que hablábamos de la división en bloques de un programa, poníamos como ejemplo que describiera la función de esta subrutina el de la sección o bloque que se encargaba de controlar el **tráfico** de nuestro programa, a medida que vayamos construyéndola verás lo adecuado de este ejemplo.

Empecemos estudiando cuáles son las funciones que debe realizar nuestro programa a partir de este momento. Actualmente podemos mover el cohete en su plataforma de derecha a izquierda y posicionarla junto a la torre de lanzamiento.

¿Qué es lo que hace el Columbia cuando le sitúan junto a la torre de lanzamiento...? Te lo he puesto fácil ¿verdad? Efectivamente, lo que hace es despegar, así que la siguiente parte dentro de este bloque la llamaremos **despegue.**

DESPEGUE

Después del despegue ¿qué es lo que hace la nave? (Después de esto, de cabeza al UN, DOS, TRES...). ¡Efectivamente, **volar**! Pues éste será el nombre de la parte que sigue.

VOLANDO

Bueno, esta pregunta es un poco más difícil (no todas iban a ser facilitas).

Cuando llega a cierta altura ¿qué pasa con el cohete y la nave? Para aquéllos que hayan respondido adecuadamente, les acaba de corresponder un BYTECHOLLO MSX y para los que no acertaron diremos que lo que sucede es que ambos se **desacoplan...** así que el nombre está claro...

DESACOPLAMIENTO

Por último, cuando el cohete y la nave están subiendo, pasan a través de capas de nubes. Si has montado alguna vez en avión, habrás tenido oportunidad de ver esto mismo, nosotros vamos a hacer que nuestro cohete también atraviese su capa de nubes (¡faltaría menos...!).

NUBES MOVILES

Vamos ahora a fijar el inicio y la longitud de cada una de estas **subpartes** que vamos a añadir a nuestra subrutina y veamos en la figura 39 cómo queda su distribución.

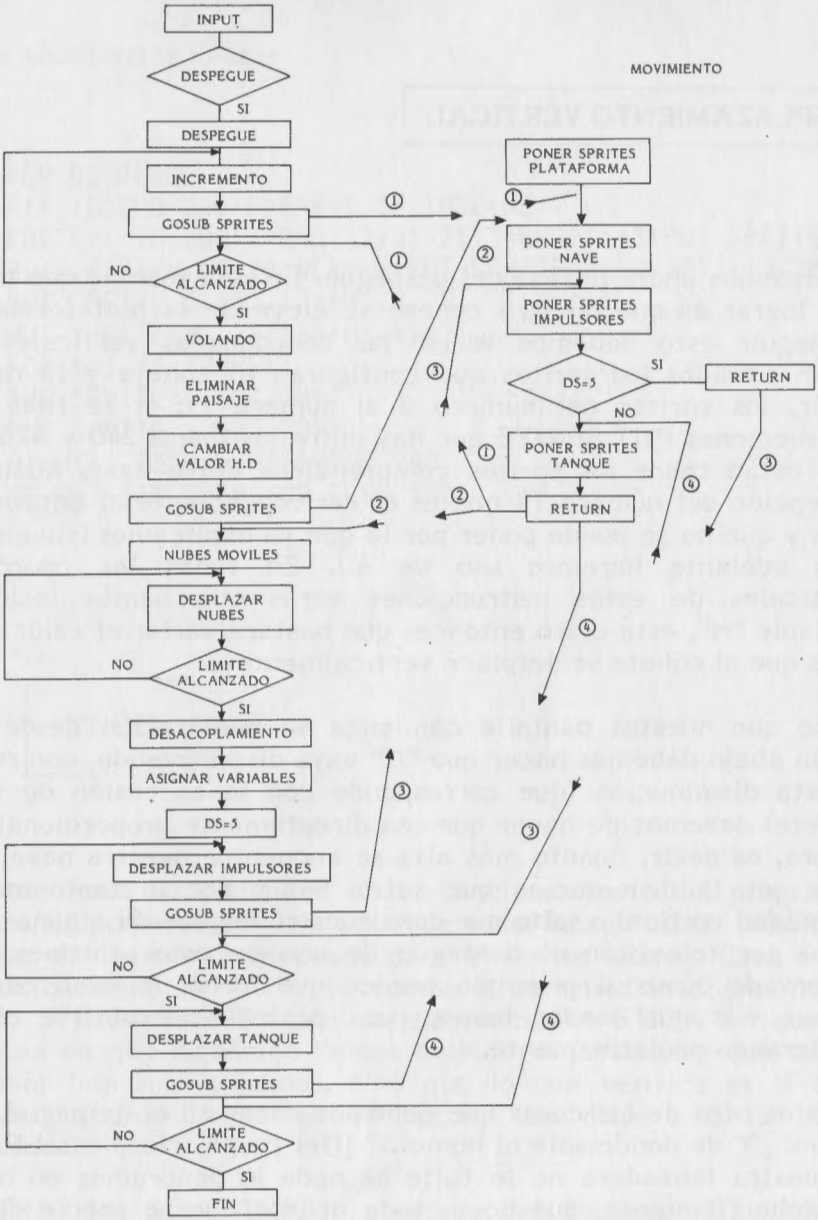


Fig. 39 Distribución simplificada subrutina MOVIMIENTO en posición de DESPEGUE

DESPLAZAMIENTO VERTICAL

Realicemos ahora la fase del despegue. En esta fase, lo que tenemos que lograr es que nuestro cohete se eleve de la plataforma. Para conseguir esto debemos variar las coordenadas verticales donde están situados los sprites que configuran el cohete y la nave, es decir, los sprites del número 5 al número 19; si te fijas en las instrucciones PUT SPRITE que hay entre las líneas 240 y 470, verás que están todos los sprites comprendidos entre estos números, a excepción del número 14 que es el correspondiente al detalle de la cola y que no se puede poner por lo que ya explicamos (sin embargo, más adelante haremos uso de él). En todas las coordenadas verticales de estas instrucciones verás que hemos incluido la variable "H", está claro entonces que bastará variar el valor de ésta para que el cohete se desplace verticalmente.

Dado que nuestra pantalla comienza su numeración desde arriba hacia abajo debemos hacer que "H" vaya disminuyendo, con respecto a esta disminución (que corresponde con la ascensión de nuestro cohete) debemos de hacer que sea directamente proporcional con la altura, es decir, cuanto más alta se encuentre nuestra nave, mayor debe ser la disminución que sufra "H" y por lo tanto **mayor** la velocidad vertical o **salto** que dará nuestro cohete. Si alguna vez has visto por televisión el despegue de uno de estos chismes, habrás observado cómo al principio parece que no se mueven casi y no hacen más que lanzar humo, para posteriormente irse elevando acelerando paulatinamente.

Esta es otra de las cosas que debemos hacer en el despegue, lanzar humo. ¿Y de dónde sale el humo...? ¡Del fuego, claro está! Para que a nuestra lanzadera no le falte de nada le pondremos un hermoso penacho flamígero, que como todo cohete que se precie de serlo, debe tener cuando está volando.

Escribe ahora estas líneas:

```
500 REM DESPEGUE
510 H1=1:GOSUB 240:FOR R=1 TO 1000:NEXT
520 FOR R=1 TO 200:X=RND(1)*35+25:Y=RND(1)*5:PUT SPRITE
    0,(X,168+Y),15,20:NEXT R:PUT SPRITE 0,(S,209),15,20
530 PUT SPRITE 5,(37,176+H),9,22
540 PUT SPRITE 6,(37,168+H),11,21
550 FOR R=1 TO 40:NEXT
560 PUT SPRITE 5,(37,209)
570 PUT SPRITE 6,(37,209)
580 H1=H1*1.05:H=H-H1:GOSUB 240
590 IF H=<-115 THEN 600 ELSE 530
```

STRIG

Verás que nada ha cambiado en el funcionamiento del programa, seguimos sin poder despegar a pesar de tener la subrutina para ello, nos falta dar la orden adecuada para ir a este bloque. Esta orden debemos darla en INPUT y la vamos a dar mediante una nueva función, STRIG. Esta función se diferencia de STICK en que no determina en qué dirección hemos movido el joystick o qué teclas del cursor han sido pulsadas, sino que lo que verifica es si esta pulsación ha sido hecha en el botón de **disparo** o la barra espaciadora.

Con STRIG (de State TRIGer o estado del disparador) sucede lo mismo que con STICK en cuanto a que es posible seleccionar cualquiera de los dos joysticks o la barra espaciadora, dependiendo del número que pongamos en esta instrucción. El significado lo tienes en la tabla de la figura 40.

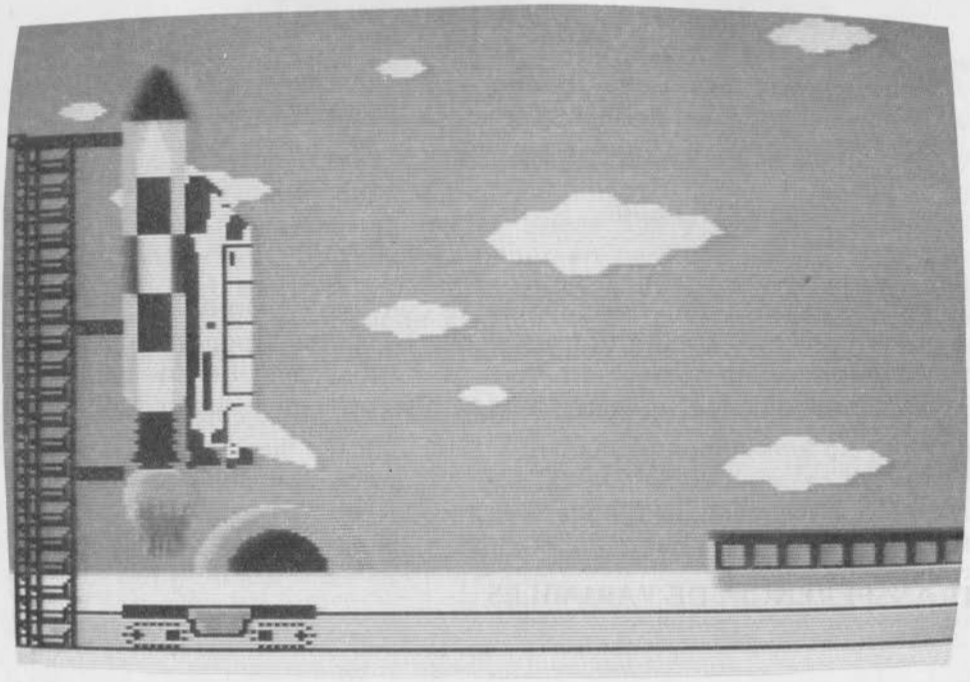
NUMERO	SIGNIFICADO
0	Barra espaciadora
1	Botón de disparo izquierdo del joystick nº 1
3	Botón de disparo derecho del joystick nº 1
2	Botón de disparo izquierdo del joystick nº 2
4	Botón de disparo derecho del joystick nº 2

Fig. 40 Tabla función STRIG

Esta función nos entrega un valor de -1 cuando está activada y de 0 cuando está en reposo (botón o barra sin pulsar). Una vez conocida, vamos a ponerla en el bloque INPUT:

```
130 IF STRIG(0)=-1 AND D=-179 THEN GOTO 500
```

Observarás que hemos condicionado el despegue de nuestra nave a que se cumplan dos premisas, primero que STRIG esté activado (barra pulsada) y segundo que "D" sea igual a -179, es decir, que nuestro cohete esté sobre la torre de lanzamiento. (Esto es lógico, pues si lanzamos el cohete fuera de la torre se puede formar una gorda...). El número que hemos utilizado para la función es el "0", dado que vamos a usar la barra espaciadora, pero si tú quieres usar un joystick no tienes más que cambiarle por el número adecuado.



¿Qué te parece como sube? Bueno, vamos a evitar provisionalmente la parada del programa poniéndole un tope en la línea 600:

```
600 GOTO 600
```

Al cohete le falta algo tan importante como es el sonido, pero eso se lo pondremos más adelante, pues el que le vamos a poner es tan real que corremos el riesgo de que se nos escape de la pantalla. Examinemos primero qué hemos hecho en nuestro bloque DESPEGUE.

TRANSFERENCIA DE VARIABLES

Primeramente, en la línea 510 damos a la variable que vamos a utilizar para el cambio de la posición vertical (H1) el valor "1", esto es debido a que esta variable es posteriormente **multiplicada** en la línea 580 y de no haberle asignado antes un valor éste sería cero, con lo que el resultado de la multiplicación sería siempre el mismo (0). En esta misma línea (580) procedemos a cambiar el valor de "H1", para que te quede claro qué es lo que sucede con estos valores, escribe lo siguiente en la pantalla y luego pulsa RETURN:

```
H1=1: FOR A=1 TO 100:H1=H1*1.05:H=H-H1:PRINT
H:NEXT
```

Habrás observado cómo la separación entre números iba haciéndose cada vez mayor, puedes ver esto de una forma más gráfica si escribes estas líneas:

```
H1=1:SCREEN2:FOR A=1 TO 50:H1=H1*1.05:H=H-H1:
PSET(A,200+H),1:BEEP:NEXT
```

Te habrás dado cuenta cómo la línea trazada no era recta y los puntos se separaban cada vez más.

En la línea 590 ponemos el límite para el desplazamiento vertical y al mismo tiempo indicamos en qué momento debemos pasar a la siguiente fase (VOLANDO), en las líneas 530 y 540 utilizamos las instrucciones PUT SPRITE para poner en pantalla nuestras llamas. En el caso del **sprite** 21 (540) su posición es 16 pixels debajo del cohete, mientras que el **sprite** 22 (530) se sitúa únicamente 8 pixels debajo del 21, es decir, se **monta** encima de éste.

Esta superposición es posible dado que el número 22 tiene una prioridad menor que el número 21. En 550 introducimos un loop de retardo, es decir, una operación que lo único que hace es consumir tiempo. El motivo de este loop es el de que el **parpadeo** de las llamas sea más real. Puedes anularlo provisionalmente insertando un REM delante del FOR para ver la diferencia.

ANULACIONES SPRITES

En las líneas 560 y 570 volvemos a utilizar de nuevo la instrucción PUT SPRITE, únicamente que esta vez para lo que la usamos no es para **poner**, sino para **quitar** nuestros sprites flamígeros.

Observarás que las coordenadas verticales de ambos sprites son las mismas (209) siempre que demos esta coordenada a un **sprite** éste desaparecerá de la pantalla y eso es lo que tratamos con el fin de dar una sensación de **parpadeo** a las llamas.

Por último, (a pesar de que estaba antes) en la línea 520 creamos el movimiento juguetón que tienen las nubes de humo en el despegue.

Estas nubes están presentes únicamente cuando el cohete está abajo. Es decir, mientras se está ejecutando el loop R de la línea 520, ya que al final de este loop se encuentra la misma instrucción que explicábamos en las líneas 560 y 570, PUT SPRITE con la coordenada vertical 209 que nos quita nuestra nube de la pantalla.

RND

El movimiento **aleatorio** de las nubes lo conseguimos mediante la función RND. Con esta función creamos un número comprendido entre 0 y 1, este número le multiplicamos por 35 y le sumamos 25 para la coordenada X, con lo cual conseguimos que el desplazamiento máximo sea 60 (1*35+25) y el mínimo de 25 (0*35+25) quedando de esta manera dentro de los límites más o menos previsibles en que estaría la nube de humo generada por el cohete.

Para la coordenada Y recurrimos a la misma función, sólo que en este caso los límites de desplazamiento son de 0 a 5.

Puedes comprobar el funcionamiento de RND mediante esta línea:

```
FOR A=1 TO 25:PRINT RND(1)*10:NEXT
```

Llegados a este punto creo que es hora de que salves el programa. Cambia la línea 10 de nuevo y sigue el procedimiento habitual de comprobación... (¡sí, ya lo sé ...soy un pesado!).

Esta parte del programa que acabamos de realizar forma parte del bloque MOVIMIENTO y no es más que una subrutina **encestada** dentro de otra como ya hicimos en GRAFICOS F1. En la figura 41 tienes la carta de flujo correspondiente a esta subrutina secundaria (la principal es MOVIMIENTO).

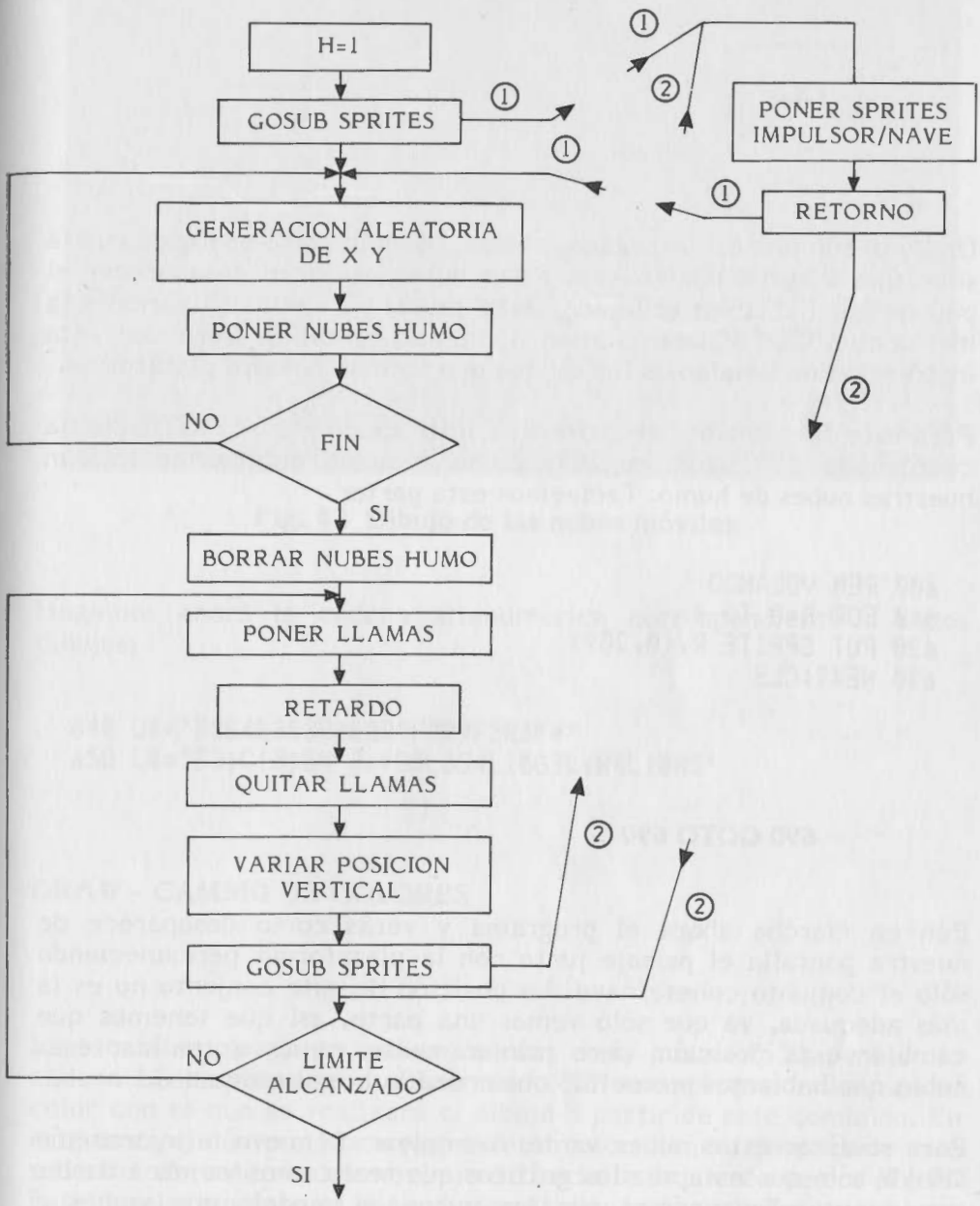


Fig. 41 Carta de flujo de la subrutina DESPEGUE

CLS

Una vez que hemos despegado, vamos a seguir como es lógico con la subrutina VOLANDO. En esta parte debemos hacer desaparecer el paisaje que habíamos utilizado hasta ahora, para ello utilizaremos la instrucción CLS (CLear Screen o limpiar pantalla). Pero con esta instrucción no **limpiamos** los sprites que forman nuestra plataforma.

Para este fin debemos recurrir a la instrucción PUT SPRITE con la coordenada "Y" igual a 209, como hicimos anteriormente con nuestras nubes de humo. Tecleemos esta parte:

```
600 REM VOLANDO
610 FOR R=0 TO 4
620 PUT SPRITE R,(0,209)
630 NEXT:CLS
```

690 GOTO 690

Pon en marcha ahora el programa y verás cómo desaparece de nuestra pantalla el paisaje junto con la plataforma permaneciendo sólo el conjunto cohete/nave. La posición de este conjunto no es la más adecuada, ya que sólo vemos una parte, así que tenemos que cambiar esta posición, pero primeramente vamos a realizar esas nubes que habíamos prometido que cruzaríamos al elevarnos.

Para realizar estas nubes vamos a emplear de nuevo la instrucción DRAW sólo que esta vez los gráficos que realicemos vamos a darles movimiento. Empecemos por las nubes, el modelo que vamos a utilizar es similar al que ya empleamos, pero esta vez el dibujo está dividido en dos partes como puedes ver en la figura 42:

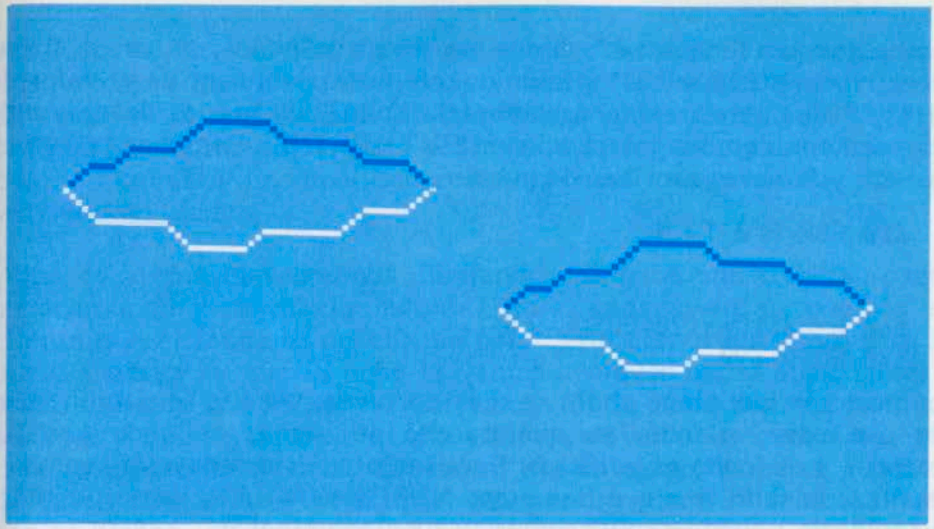


Fig. 42 Dibujo de las nubes móviles

Hagamos ahora la cadena alfanumérica correspondiente a estos dibujos:

```
640 U$="S3E4R3E3R6E5R9F3R9F3R6F4"
650 L$="S3;C15;BM-2,+G5L6G4L10G3L6H4L10H5"
```

DRAW - CAMBIO DE COLORES

La escala empleada en ambas cadenas es la misma "S3", pero en la cadena L$ empleamos el subcomando "C" con el que indicamos el color con el que se realizará el dibujo a partir de este comando. En este caso, el número 15 corresponde al color blanco, después de este subcomando tenemos un movimiento hacia la izquierda de dos pixels. Este desplazamiento es para conseguir que la parte inferior sea más corta y que quede dentro de la parte superior para lograr el efecto que después veremos.

Bien, ya que tenemos el dibujo de nuestras nubes, vamos a situar nuestro cohete. Para esto, lo único que vamos a variar es el valor de "H" y "D" y enviar el programa a la línea 240, que es desde donde empezamos con las instrucciones PUT SPRITE relacionadas con el cohete y la nave, escribamos pues:

```
610 FOR R=0 TO 4


660 H=-35:D=-120:GOSUB 240
690 GOTO 690
```

La posición que tiene ahora es más correcta, pero la sensación no es de que esté volando, ya que carece de llamas saliendo por sus toberas y el fondo es estático. Para mejorar esta sensación, vamos a realizar la subrutina que llamamos NUBES MOVILES, vamos primero a teclearla y luego explicaremos su funcionamiento:

```
850 REM NUBES MOVILES
860 FOR V=1 TO 180
870 PUT SPRITE 2,(96,133),11,21
880 PUT SPRITE 1,(96,141),9,22
890 PSET (30,-15+V),5
900 DRAW U$+L$
910 PUT SPRITE 2,(96,209)
920 PUT SPRITE 1,(96,209)
930 PSET (85,5+V),5
940 DRAW U$+L$
950 PSET (175,-35+V),5
960 DRAW U$+L$
970 NEXT V
980 COLOR 1,4:CLS:RETURN
```

Como vemos, se trata de un loop entre las líneas 860-970. En este loop ponemos primeramente dos sprites (870-880) en la pantalla, estos sprites son los correspondientes a las llamas (21 y 22) posteriormente utilizamos la instrucción PSET para determinar un punto; este punto será de color azul y desde él realizaremos el dibujo de la primera nube. En la coordenada vertical de PSET observarás que hemos incluido una variable "V", que es la misma que empleamos en el loop.

Por lo tanto, la posición Y de esta instrucción (PSET) irá variando al mismo ritmo que lo hace el loop o bucle "V", dado que el punto dibujado es de color azul (5) la primera parte del dibujo (U$) se realizará en dicho color, pero al llegar a la segunda parte (L$) este color cambiará como consecuencia del subcomando "C15", volviéndose blanco.

¿Qué es lo que conseguimos dibujando una nube que tiene su parte superior azul y la inferior blanca? En sí nada, pero si tenemos en cuenta que el fondo es del mismo color azul con el que dibujamos la parte superior, y que la nube la vamos a mover hacia abajo pixel a pixel, la cosa puede ser ya diferente. Si la nube tuviera todo su perímetro blanco, al desplazarla iríamos dejando un **rastro** de este color sobre el fondo azul, al utilizar sin embargo, la parte superior de color azul, conseguimos que esta parte **rellene** el rastro dejado por el blanco, en la figura 43 tienes las distintas fases por las que pasa la creación de nuestra **nube móvil:**

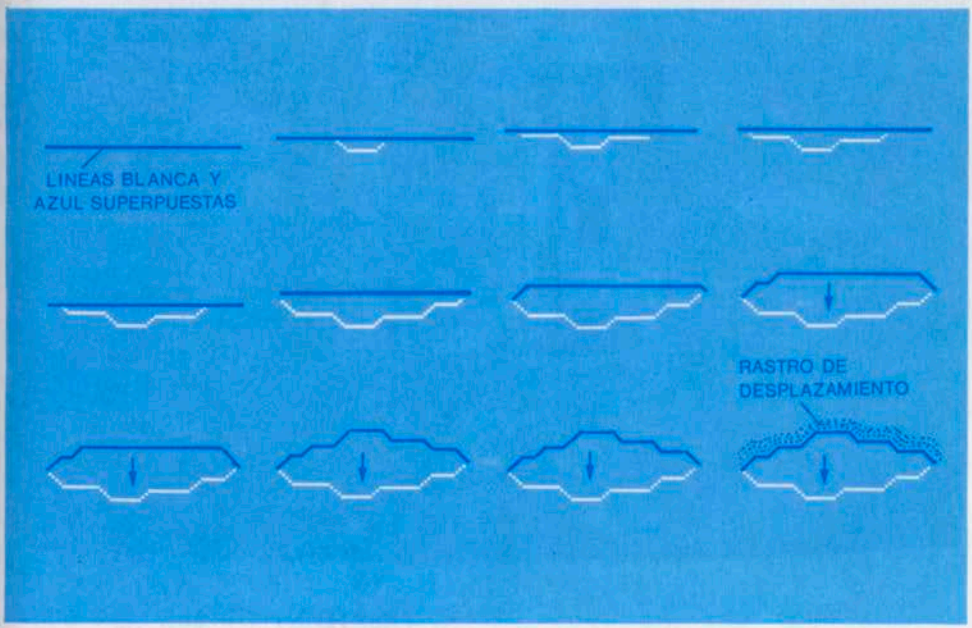
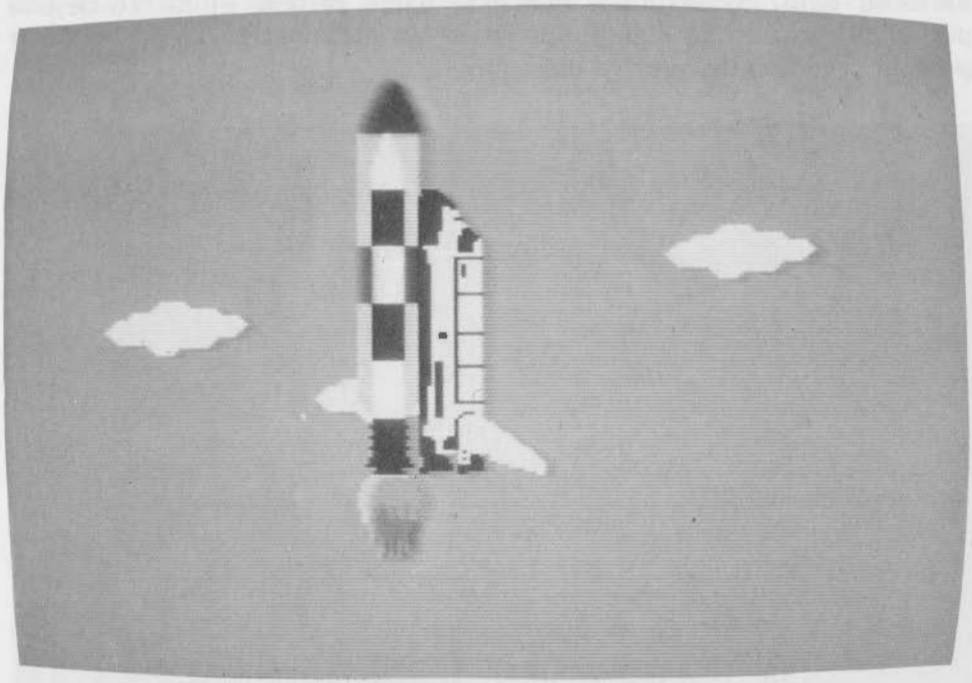


Fig. 43 Generación de la 'nube móvil'

Una vez puesta nuestra primera nube, procedemos a quitar los sprites de las llamas con el fin de conseguir un efecto de parpadeo, esto lo realizamos como hicimos anteriormente dando a la posición vertical el valor 209. Luego, posicionamos otras dos nubes con el mismo sistema, únicamente que en distinta posición cada una, esta operación la repetimos 180 veces, que es la longitud del bucle, en cuyo momento pasamos a la siguiente subrutina. Antes de pasar a ésta vamos a dar la instrucción necesaria para acceder la que acabamos de ver (NUBES MOVILES), esta instrucción debe ser GOSUB 850 y debe estar en la línea 660:

```
660 H=-35:D=-120:GOSUB 240:GOSUB 850
```



Con estos últimos elementos que hemos añadido, ha cobrado más realismo nuestro programa, le sigue faltando el sonido pero todo llegará.

Habrás comprobado cómo al final el cohete se ha quedado donde estaba, las nubes han desaparecido y las llamas se han apagado, todo parece igual, pero sin embargo, hay algo que sí ha cambiado. El color de nuestro cielo es más oscuro, esto es debido a la línea 980, donde limpiamos la pantalla y cambiamos de color, es porque esta fase está ya consumida, estamos ya en el punto en el cual el Columbia se desprende del cohete que le ha impulsado hasta esa altura y comienza su andadura en solitario. Es el momento de que entre la subrutina DESACOPLAMIENTO.

Esta subrutina lo que va a realizar es la separación de los impulsores y el tanque de la nave, por lo tanto, accederemos desde ella primeramente a la línea 420, que es donde empiezan los sprites correspondientes al impulsor, previamente habremos puesto la variable "DS" en la línea 710 a un valor igual a 5, posteriormente accederemos a la subrutina por la línea 350, después de un tiempo de retardo (línea 800), con lo cual se desprenderá también el tanque, quedando sola la nave. Al encontrarse el programa con la variable "DS" igual a 5 en la línea 410 'retorna' desde ese punto, no accediendo al resto de las líneas (420-470), ya que estas pertenecen a los impulsores.

```
690 GOTO 690

710 X=D:Y=H:DS=5:FOR F=1 TO 90
720 PUT SPRITE 2,(112,133),11,21
730 PUT SPRITE 1,(112,141),9,22
740 FOR R=1 TO 40:NEXT R
750 PUT SPRITE 2,(112,209)
760 PUT SPRITE 1,(112,209)
770 D=X-F:H=Y+F:GOSUB 420:NEXT F
780 FOR R=20 TO 25:PUT SPRITE R,(0,209):NEXT

800 FOR R=1 TO 1000:NEXT R

820 FOR F=1 TO 128:D=X-F:H=Y+F:GOSUB 350:NEXT F
840 GOTO 840
```

Este es, como puedes comprobar, un programa muy sencillo, su única particularidad estriba en la transferencia del valor del loop "F", (que es el que utilizamos para el movimiento del cohete) a las variables D y H.

Para esta transferencia utilizamos las variables X e Y, las cuales toman el mismo valor que tienen D y H al principio del loop, para posteriormente ir disminuyendo y aumentando respectivamente (770 y 800) y transfiriendo su valor a D y H.

Podría parecerte que esta transferencia no es necesaria, que bastaría con que sumáramos o restáramos F de D y H directamente, pero esto no es así, el valor que adquirirían estas variables no aumentaría o disminuiría de forma lineal y el efecto de desprendimiento sería bastante extraño.

Si quieres hacer la prueba, cambia la línea 770 de la siguiente manera (déjala luego como estaba).

```
770 D=D-F:H=H-F:GOSUB 420:NEXT
```

Bastante raro ¿verdad? En la línea 830 es donde ponemos el detalle que habíamos realizado para la cola de nuestra nave, el **sprite** número 14, como ves su posicionamiento coincide con el final del bucle, y por lo tanto, al haber desaparecido ya de la pantalla el cohete, podemos poner este cuarto **sprite**, que de otra manera nos sería rechazado.

```
830 PUT SPRITE 7,(128,117),1,14
```

¡Bien, hemos llegado al final...! Al final de la subrutina MOVIMIENTO claro está, así que ya no te digo lo que debes hacer con el cassette... únicamente que la línea 10 debe ser así:

```
10 REM BASE12
```

En la figura 44 tienes el listado completo de esta subrutina. En el próximo capítulo pondremos sonido a nuestro cohete.

```
150  REM MOVIMIENTO
160 IF D>7 THEN D=7:PLAY "S12M5000C"
170 IF D<-179 THEN D=-179:PLAY "S10M500C"
200 PUT SPRITE 4,(216+D,168),1,1
210 PUT SPRITE 3,(232+D,168),10,2
220 PUT SPRITE 2,(248+D,168),1,3
230 PUT SPRITE 1,(232+D,168),1,4
240 PUT SPRITE 14,(232+D,152+H),1,13
250 PUT SPRITE 13,(232+D,136+H),1,12
260 PUT SPRITE 12,(232+D,120+H),1,11
270 PUT SPRITE 11,(232+D,104+H),1,10
280 PUT SPRITE 10,(232+D,88+H),1,9
290 PUT SPRITE 9,(248+D,152+H),15,8
300 PUT SPRITE 19,(232+D,152+H),15,7
310 PUT SPRITE 18,(232+D,136+H),15,6
320 PUT SPRITE 17,(232+D,120+H),15,6
330 PUT SPRITE 16,(232+D,104+H),15,6
340 PUT SPRITE 15,(232+D,88+H),15,5
350 PUT SPRITE 26,(216+D,136+H),14,19
360 PUT SPRITE 27,(216+D,120+H),14,6
370 PUT SPRITE 28,(216+D,104+H),6,6
380 PUT SPRITE 29,(216+D,88+H),14,6
390 PUT SPRITE 30,(216+D,72+H),14,6
400 PUT SPRITE 31,(216+D,56+H),6,15
410 IF DS=5 THEN RETURN
420 PUT SPRITE 20,(216+D,152+H),1,18
430 PUT SPRITE 21,(216+D,136+H),15,17
440 PUT SPRITE 22,(216+D,120+H),1,17
450 PUT SPRITE 23,(216+D,104+H),15,17
460 PUT SPRITE 24,(216+D,88+H),1,17
470 PUT SPRITE 25,(216+D,72+H),15,16
480 RETURN
500 REM DESPEGUE
510 H1=1:GOSUB 240:FOR R=1 TO 1000:NEXT:
520 FOR R=1 TO 200:X=RND(1)*35+25:Y=RND(1)*5:PUT SPRITE
    0,(X,168+Y),15,20:NEXT R:PUT SPRITE 0,(S,209),15,20
530 PUT SPRITE 5,(37,176+H),9,22
540 PUT SPRITE 6,(37,168+H),11,21
550 FOR R=1 TO 40:NEXT
560 PUT SPRITE 5,(37,209)
570 PUT SPRITE 6,(37,209)
```

```
580 H1=H1*1.05:H=H-H1:GOSUB 240
590 IF H=(-115 THEN 600 ELSE 530
600 REM VOLANDO
610 FOR R=0 TO 31
620 PUT SPRITE R,(0,209)
630 NEXT:CLS
640 U$="S3E4R3E3R6E5R9F3R6F4"
650 L$="S3;C15;BM-2,+G5L6G4L10G3L6H4L10H5"
660 H=-35:D=-120:GOSUB 240:GOSUB 850
690 REM DESACOPLAMIENTO
710 X=D:Y=H:DS=5:FOR F=1 TO 90
720 PUT SPRITE 2,(112,133),11,21
730 PUT SPRITE 1,(112,141),9,22
740 FOR R=1 TO 40:NEXT R
750 PUT SPRITE 2,(112,209)
760 PUT SPRITE 1,(112,209)
770 D=X-F:H=Y+F:GOSUB 420:NEXT F
780 FOR R=20 TO 25:PUT SPRITE R,(0,209):NEXT
800 FOR R=1 TO 1000:NEXT R
820 FOR F=1 TO 128:D=X-F:H=Y+F:GOSUB 350:NEXT F
830 PUT SPRITE 7,(128,117),1,14
840 FOR R=1 TO 1500:NEXT:GOTO 8000
850 REM NUBES MOVILES
860 FOR V=1 TO 180
870 PUT SPRITE 2,(96,133),11,21
880 PUT SPRITE 1,(96,141),9,22
890 PSET (30,-15+V),5
900 DRAW U$+L$
910 PUT SPRITE 2,(96,209)
920 PUT SPRITE 1,(96,209)
930 PSET (85,5+V),5
940 DRAW U$+L$
950 PSET (175,-35+V),5
960 DRAW U$+L$
970 NEXT V
980 COLOR 1,4:CLS:RETURN
```

Fig. 44 Listado de MOVIMIENTO

Sonido

Uno de los sistemas más potentes que posee nuestro ordenador es el de creación de sonido. Para este fin todos los ordenadores MSX cuentan con un **Generador de Sonido Programable** (Program Sound Generator o PSG en inglés). Este circuito posee tres **voces** o canales que podemos programar independientemente en sus tonos, volúmenes y matices, permitiéndonos la emisión de los más variados efectos acústicos.

Las posibilidades de este sistema son tantas que haría falta un libro completo para describirlas, teniendo en cuenta además, que es necesario un conocimiento previo de música para poder hacer un correcto uso de los comandos relacionados con este tema. En este capítulo vamos a dar una serie de indicaciones o guías prácticas para usar las instrucciones dedicadas a la generación de sonidos que nos permitan aplicar éstas a nuestro programa o a cualquier otro cuyas características se vean mejoradas con la inclusión del elemento sonoro. Ya hemos visto cómo nuestro cohete queda falto de realismo por la carencia del sonido adecuado que acompañe su ascenso.

BEEP

La instrucción más sencilla con que contamos para generar un sonido es BEEP. Esta instrucción lo único que produce es un ruido que suena igual que su pronunciación en inglés (de ahí su nombre) **biip.** En esta instrucción no es posible ningún tipo de control en lo que se refiere a volumen, frecuencia o duración; simplemente emite un corto pitido y ahí acaba todo. De todas maneras, no por eso deja de ser útil, es de muy fácil programación y nos puede servir para indicarnos acústicamente determinadas situaciones como una especie de alarma o avisador.

La primera instrucción importante que nos permite la generación de música es PLAY. Esta instrucción tiene una forma de programación similar a la que empleamos en DRAW, es decir, empleamos una cadena alfanumérica para determinar las notas, tonos, volúmenes, envolventes, etc., etc., con los que es posible programar una canción o melodía.

La otra instrucción relacionada con este tema es SOUND (sonido), esta difiere en varios aspectos de PLAY. Las diferencias más importantes son que su programación está basada en el acceso directo a los **registros** del generador de sonido y que no hace uso de cadenas alfanuméricas.

PLAY

Empecemos por PLAY. Esta instrucción, como decíamos anteriormente, es utilizada para la generación de notas musicales; a tal efecto posee una escala de este tipo (musical). Esta escala está constituida por las letras comprendidas entre la "A" y la "G", ambas inclusive. Al igual que ocurre con los instrumentos musicales, tenemos la posibilidad de cambiar el tono de determinada nota haciéndole agudo mediante el signo "+" o "#" y grave mediante el signo "-".

La representación gráfica para aquéllos que sepan música, de la equivalencia entre las letras empleadas en nuestra instrucción y la escala musical está reflejada en la figura 45. En esta figura están asimismo, señalados los signos que nos permiten cambiar el tono de las notas junto a las letras en las que es posible su aplicación, ya que a letra "C" no es posible aplicarle el signo "-", por citar un ejemplo.

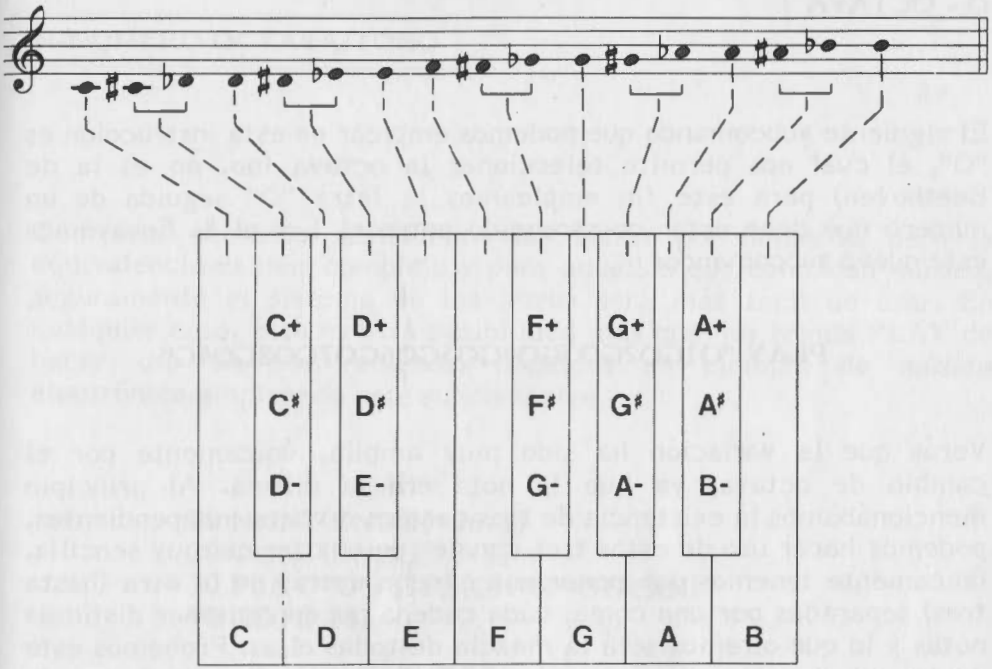


Fig. 45 Escala musical

Para mostrarte el funcionamiento de esta instrucción, escribe esta línea y pulsa RETURN para que se ejecute:

```
PLAY "CCGGAAGEFGAC"
```

Prueba ahora con la misma cadena, pero haciendo uso de los signos:

```
PLAY "C+C+G+A+A+G+E-F+G+A+C+"
```

Notarás que la segunda cadena tiene un tono distinto a la anterior a pesar de contener las mismas notas.

O - OCTAVA

El siguiente subcomando que podemos emplear en esta instrucción es "O", el cual nos permite seleccionar la **octava** (no, no es la de Beethoven) para este fin empleamos la letra "O" seguida de un número que debe estar comprendido entre el 1 y el 8. Ensayemos este nuevo subcomando:

```
PLAY "O1GO2GO3GO4GO5GO6GO7GO8GO4G"
```

Verás que la variación ha sido muy amplia, únicamente por el cambio de octava, ya que la nota era la misma. Al principio mencionábamos la existencia de tres canales o voces independientes, podemos hacer uso de estos tres canales de una forma muy sencilla, únicamente tenemos que poner una cadena detrás de la otra (hasta tres) separadas por una coma; cada cadena puede contener distintas notas y lo que oiremos será la mezcla de todas ellas. Probemos este punto, escribe estas líneas y pulsa RETURN al finalizar cada una de ellas:

```
PLAY "O4CEFGE"
PLAY "O2FBO3GC"
PLAY "O3EFEO2A"
```

Pongamos ahora estas tres cadenas cada una en un canal y oigámos las tres a la vez:

```
PLAY "O4CEFGE","O2FBO3GC","O3EFEO2A"
```

Como verás, el resultado es completamente distinto de cuando oíamos cada una de las cadenas por separado, también es posible la generación de música empleando números en lugar de letras; esto se consigue mediante el subcomando "N".

N-NUMERO OCTAVA/TONO

Con este comando sustituimos las letras por números, pero la equivalencia es más compleja y para aquéllos que conozcan música, seguramente el sistema de las letras será más fácil de usar. En cualquier caso, ésta es otra posibilidad más que nos brinda PLAY de hacer uso de sus recursos, hagamos un ejemplo de **música electrónica** empleando este subcomando:

```
10 N=INT(RND(1)*96+1)
20 O=INT(RND(1)*8+1)
30 PLAY "O"+STR$(O)+"N"+STR$(N)
40 GOTO 10
```

INT

Pulsa RUN y podrás oir una obra muy de nuestro tiempo... No es gran cosa, pero puede servir para aburrir a un grillo... (¡por ejemplo!). Bien, bromas aparte, con este programa acabamos de ver cómo se puede introducir una variable en la cadena alfanumérica de PLAY, lo mismo que hicimos con DRAW. Aparte de esto, también hemos usado una nueva **función**, INT. Esta función la empleamos cuando queremos redondear un número que tiene decimales ya que lo que hace es quitar esto, dejando el número entero **inferior** al valor dado.

L-LONGITUD

Aquéllos que saben música estarán pensando que además de las notas y las octavas, es necesario cambiar más cosas para elaborar una melodía, por ejemplo la duración de las notas. Pues se ve que los que construyeron los MSX también lo pensaron, ya que contamos con el subcomando "L" que nos permite cambiar esta duración.

Este subcomando puede adoptar valores que van desde 1 hasta 64. Este valor se aplica como divisor del tiempo que una nota estaría sonando normalmente, es decir, si aplicamos un valor de 23 el resultado será que la nota (o notas) afectada sonará durante un veintetresavo de su tiempo (1/23). Algo que hay que tener muy en cuenta cuando se utilice este subcomando (como también sucede con el de octavas) es que el valor que demos se aplicará a todas las notas que le sigan hasta que encuentre un nuevo subcomando "L".

El valor que tiene "L" cuando encendemos nuestro ordenador es el de 4, el mismo que tiene "O".

Hagamos una prueba con este nuevo subcomando:

```
PLAY "O4L4CDAC".
```

Probemos ahora cambiando la duración.

```
PLAY "L16CDAC"
```

Distinto ¿verdad? Además de esto "L" tiene la posibilidad de emplearse únicamente en la nota que le **antecede**, para ello lo único que hay que hacer es poner el número correspondiente **sin** la letra "L" detrás de la nota en cuestión.

```
PLAY "L16CD1AC"
```

R-RETARDO

Lo mismo que tenemos necesidad de cambiar la duración de nuestras notas, también necesitamos (los que son capaces de componer) **pausas** entre ellas. Para este cometido tenemos el subcomando "R" con valores iguales a los empleados por "L", es decir, 1 hasta 64. El significado de este valor es el mismo que el explicado anteriormente, únicamente que aplicado a la pausa entre notas:

```
PLAY "L4CDR4FG"
```

. - DURACION

Como verás, en este acorde hay un espacio en silencio a la mitad del mismo, también es posible aumentar la duración de una nota añadiéndole puntos (.) detrás de ella, cada punto que le añadamos alargará su duración .5 de la que tuviera.

```
PLAY "C....."
```

T-TIEMPO

En música (¡que es una cosa muy complicada...!) es necesario también establecer el **tiempo** o **cadencia** con que se ejecuta una estrofa, pues para esto también tenemos subcomando (como verás, que por subcomandos no quede...).

Se trata de "T" con valores que van de 32 a 255, el valor fijado como base al conectar el ordenador es de 120, este valor lo que nos indica es el número de **cuartas** por minuto:

```
PLAY "T200CDECR1":PLAY "T40CDEC:PLAY "T120"
```

V-VOLUMEN

La diferencia es fácilmente perceptible entre los dos acordes. Bien, y ya para "casi" terminar con este tema tenemos el **control de volumen** que es el subcomando (otro más) "V". Este puede tener valores que van desde "0" para inaudible, hasta 15 para máximo volumen:

```
PLAY "V7DCFGR8":PLAY "V14DCFG"
```

Con estos dos últimos subcomandos ("T" y "V") también sucede lo mismo que ocurría con "L", esto es, que permaneen hasta que encuentran otro de su **especie** que cambie su valor; por lo tanto, esto hay que tenerlo muy en cuenta ya que si no puede cambiarnos totalmente nuestra "composición".

Debido a la variedad de subcomandos, creo que es necesario tener una tabla que nos sirva de guía sobre los mismos, esta tabla la tienes en la figura 46.

LETRA	SIGNIFICADO	VALOR
O	Determina la octava en que se ejecutará la nota	1-8
N	Valor numérico que comprende octavas y notas simultáneamente	0-95
L	Longitud de una nota	0-64
.	Multiplica la duración de una nota por $\frac{1}{2}$	
R	Pausa entre notas	0-64
T	Determinación de Tiempo o cadencia	32-255
V	Volumen	0-15
S	Forma o perfil de la envolvente	0-15
M	Determinación del período de repetición	1-65535

Fig. 46 Tabla de subcomandos empleados por PLAY

M-PERIODO

En la figura 46 vemos que el último comando es la letra "M" y hace referencia a la duración de la envolvente, esto nos lleva a una de las funciones que nos permiten aproximarnos en la imitación de ciertos instrumentos. Hasta ahora hemos visto que la variación de sonidos era únicamente sobre una nota o conjunto de éstas, sin que variáramos el volumen especificado para ella, que le afectaba de plano sin que hubiera ninguna variación en el mismo.

Esto difiere de lo que sucede en la realidad con ciertos instrumentos musicales, así por ejemplo, la trompeta comienza con un volumen bajo para subir muy rápidamente, sin embargo, el piano comienza con un volumen alto para ir decayendo lentamente.

S-ENVOLVENTE

Con el fin de adaptar nuestros sonidos lo más posible a los emitidos por los instrumentos que tratamos de imitar, tenemos el subcomando "S" con el cual seleccionamos uno de los ocho perfiles o envolventes que posee nuestro generador.

En la figura 47 tienes representadas estas **envolventes** con el número que corresponde al subcomando "S":

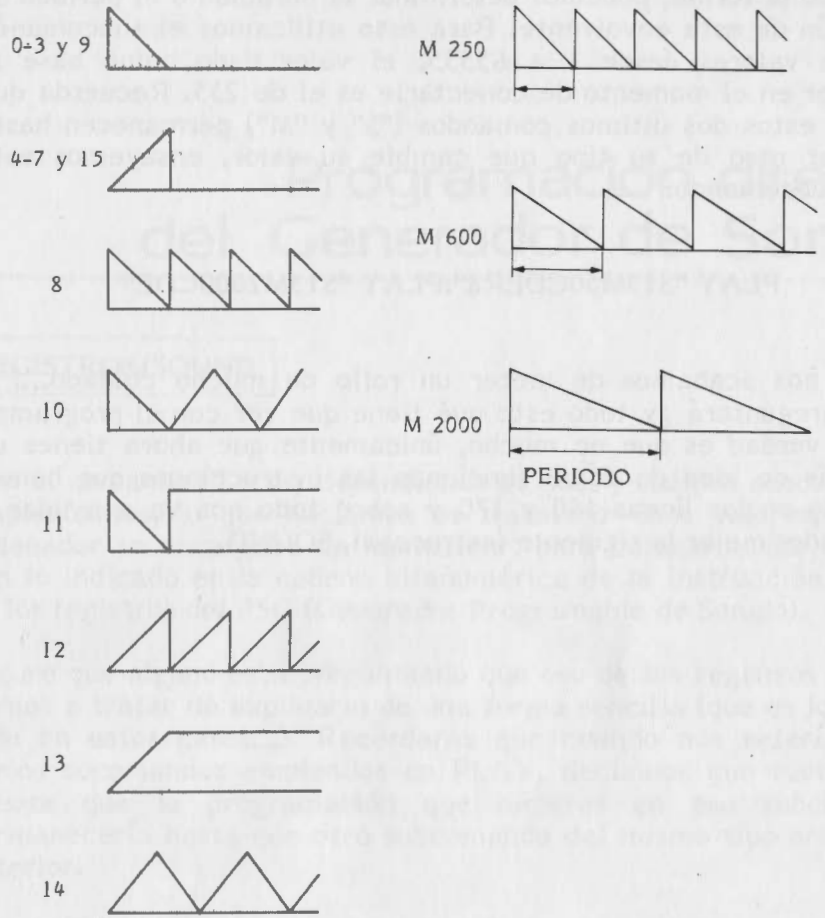


Fig. 47 Envolventes y períodos

Veamos cómo afecta una envolvente a una estrofa determinada:

```
PLAY "S8CDER8":PLAY "S13CDE"
```

Aparte de la **forma,** podemos determinar la duración o el período de repetición de esta envolvente. Para esto utilizamos el subcomando "M" con valores desde 1 a 65535, el valor dado como base al ordenador en el momento de conectarle es el de 255. Recuerda que también estos dos últimos comandos ("S" y "M") permanecen hasta encontrar otro de su tipo que cambie su valor, ensayemos este último subcomando:

```
PLAY "S13M50CDER8":PLAY "S13M2000CDE"
```

¡Bueno, nos acabamos de meter un rollo de mucho cuidado...! y alguno preguntará ¿y todo esto qué tiene que ver con el programa? Pues la verdad es que no mucho, únicamente que ahora tienes un poco más de idea de cómo funcionan las instrucciones que hemos empleado en las líneas 160 y 170 y sobre todo nos va a ayudar a comprender mejor la siguiente instrucción, SOUND.

Programación directa del Generador de Sonido

REGISTROS/SOUND

Con el comando PLAY y dependiendo de cuál y cuántos subcomandos empleábamos, lo que hacíamos es transferir unos valores, que el ordenador se encargaba de identificar, para colocarlos (de acuerdo con lo indicado en la cadena alfanumérica de la instrucción PLAY) en los registros del PSG (Generador Programable de Sonido).

Seguro que alguno está preguntando que eso de los registros qué es. Vamos a tratar de explicarlo de una forma sencilla (que es lo que se dice en estos casos...). Recordarás que cuando nos referíamos a varios sucomandos empleados en PLAY, decíamos que tuvieras en cuenta que la programación que hicieras en ese subcomando permanecería hasta que otro subcomando del mismo tipo anulase al anterior.

Esto era debido a que este subcomando quedaba **fijado** en un registro y ese registro únicamente era accesible por otro subcomando de la misma especie, imagínate un edificio cerrado completamente y en su interior una persona que la única comunicación que tiene con el exterior son cuatro cuadros de luces donde a su vez existen cuatro lámparas, él no puede encender ni apagar ninguna de las lámparas, pero le han enseñado para que actúe según las que estén encendidas.

En el tejado del edificio existen cuatro altavoces conectados con un potente amplificador que hay en el interior. Asimismo, conectado al amplificador hay un magnetofón y para que nada falte, la persona que hay en el interior tiene cuatro cassettes de música distintos.

Esta persona sabe que el primer cuadro de luces le indica que encienda o apague el amplificador y el volumen que debe poner en el mismo (mediante una combinación que él conoce); también sabe que el segundo cuadro sirve para indicarle cuál de los cuatro altavoces debe conectar, el tercero le señala cuál de las cuatro cintas debe poner en el magnetofón, y por último, el cuarto cuadro de luces le muestra cuánto tiempo debe estar conectada la música.

Bien, dentro del edificio tenemos a nuestro hombre esperando que se encienda alguna bombilla ...y mientras tanto... ¿qué hay en el exterior?

Rodeando el edificio (que es circular) hay un grupo de cuatro personas, estas personas no hacen otra cosa que dar vueltas al edificio con un papel en la mano; cuando llegan a cierto lugar de la pared se paran, miran su papel, hacen una maniobra y la música empieza a sonar, otras veces sin embargo, se para o baja de volumen... claro que también hay veces que suena más alta.

¿Qué es lo que sucede? Nos acercamos al grupo de personas para interrogarles y nos dicen que son el **grupo programa** y que su labor consiste en actuar de acuerdo con lo que tienen escrito en su papel, nos explican un poco más; cada uno de ellos tiene acceso únicamente a uno de los cuatro cuadros que hay en la pared, estos cuadros están numerados para que no se confundan y tienen (como seguramente ya habrás adivinado) cuatro interruptores cada uno.

Ellos no saben exactamente (a pesar de que oyen la música) para qué sirven los interruptores, lo único que entienden es lo que pone el papel que llevan en la mano. En este papel lo que está indicado es el número de vueltas que deben dar y los interruptores que deben conectar o desconectar en cada una de ellas.

Así que aquí tenemos al **conjunto programa** dando vueltas como tontos y sacudiendo cada uno a sus interruptores para arriba y para abajo. ¡Claro que cada vez que dan a un interruptor la persona de dentro rápidamente ejecuta una acción...! Esto es debido a que cada uno de los cuadros de interruptores de exterior se corresponden con los cuadros de luces del interior...

¿Sabes cómo llaman al edificio los del grupo programa? Le llaman "El Generador de Sonido" (seguramente por el follón que forma), pero lo más raro es cómo llaman a los cuadros de interruptores, **registros...**

No sé si te habrá gustado el cuento, pero esto es lo que pasa entre la PSG y el programa. La única comunicación que tienen son los registros y el valor que se deposita en ellos permanece allí hasta que el subcomando indicado (el miembro del **conjunto programa**) le cambia por otro valor.

Los registros que tiene el Generador de Sonido no son lógicamente a base de lámparas e interruptores, son por el contrario un conjunto de circuitos microscópicos que están encapsulados en el mismo "chip" que el PSG; es en estos circuitos donde se almacenan las instrucciones para ser ejecutadas. Una de las características que tienen los registros es su capacidad o número de **bits** que pueden contener (el equivalente a nuestros interruptores), en nuestro caso son registros de ocho bits.

El número de registros con que contamos es de 14, numerados del 0 al 13 y los podemos dividir en cuatro grupos principales:

GRUPO 1 Registro 0 - 6 Determinación de frecuencia

GRUPO 2 Registro 7 Conexión del canal correspondiente

GRUPO 3 Registro 8 - 10 Volumen del canal

GRUPO 4 Registro 11 - 13 Determinación de forma y longitud envolvente.

FRECUENCIA

Como verás, hay cierto parecido con alguno de los subcomandos empleados anteriormente en PLAY. Empecemos con el número número 1. En este grupo tenemos el control de las frecuencias que van a ser usadas en cada una de las tres vías o canales de que disponemos, por lo tanto este grupo se divide en tres secciones, cada una dedicada a un canal:

REGISTRO 0 - 1 Frecuencia Canal Nº 1

REGISTRO 2 - 3 Frecuencia Canal Nº 2

REGISTRO 4 - 5 Frecuencia Canal Nº 3

El registro número 6 es el que determina la frecuencia del **ruido blanco.** Sobre este tema hablaremos más adelante.

Como vemos, cada uno de los tres canales tiene dos registros. Veamos cómo se emplean éstos. Los registros 1, 3 y 5 emplean únicamente cuatro de sus ocho bits, los llamados **menos significativos** por estar a la izquierda y tener menor valor que los de la derecha (esto nada tiene que ver con la política n.a.). con estos cuatro bits el valor máximo que podemos alcanzar es de 15 (ver Codificación Hexadecimal en PLATAFORMA).

Mediante estos registros (i, 3 y 5) hacemos una aproximación **gruesa** de la frecuencia que deseamos en cada canal.

Con los registros 0, 2 y 4 de mismo grupo, seleccionamos el tono dentro de un margen mucho más amplio, ya que estos registros tienen 8 bits, lo que nos permite 256 posibilidades. En cualquier caso, el **afinamiento** de la frecuencia que hagamos con estos últimos registros, será siempre dentro del margen seleccionado en los anteriores (1, 3 y 5).

Con el fin de recordar mejor el número de bits y el uso de cada registro, en la figura 48 tienes una tabla con los registros pertenecientes al grupo de control de tono:

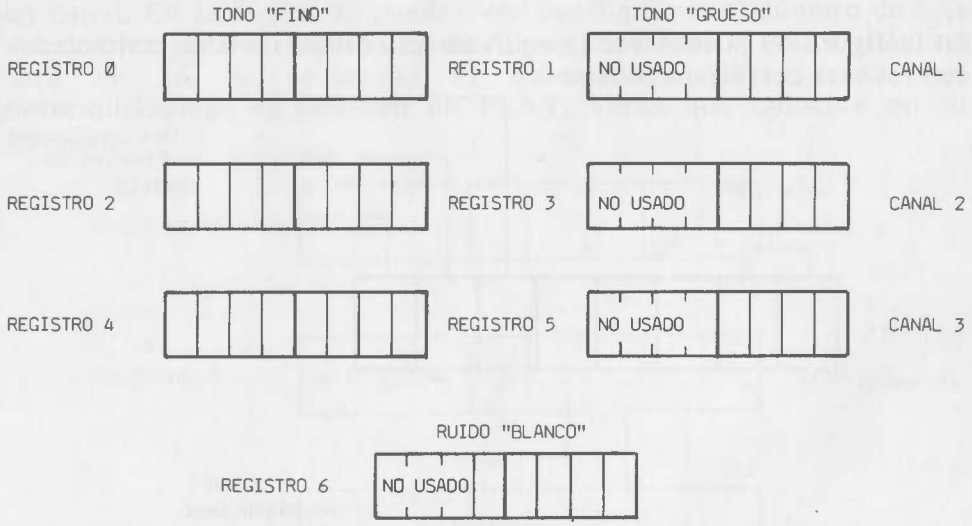


Fig. 48 Registros de control de tono

Para **cargar** estos registros utilizamos la instrucción SOUND seguida del número de registro y (separado por una coma) el valor que queremos introducir, por ejemplo:

SOUND 0,123:SOUND 1,3:SOUND 2,250:SOUND 3,0

No trates de ejecutarlas ya que no pasará nada. Esta es sólo la instrucción sobre el tono o frecuencia, pero además hay que **autorizar** el canal y fijar el volumen.

SELECCION DE CANAL

Veamos ahora el segundo grupo donde únicamente está el registro número 7. Este registro emplea solamente 6 de sus ocho bits para el control de los tres canales, los otros dos bits (7-6) tienen otra finalidad y no son empleados en la generación de sonido. Estos seis bits están divididos en dos grupos, del bit 0 al 2 son utilizados para conectar los tres canales en cuanto a generación de tonos se refiere. Los bits 3, 4 y 5 se utilizan para conectar los tres canales en la generación de sonido **blanco**; estas **conexiones** se realizan poniendo a "0" el canal correspondiente.

En la figura 49 puedes ver la equivalencia de los canales controlados con los bits correspondientes:

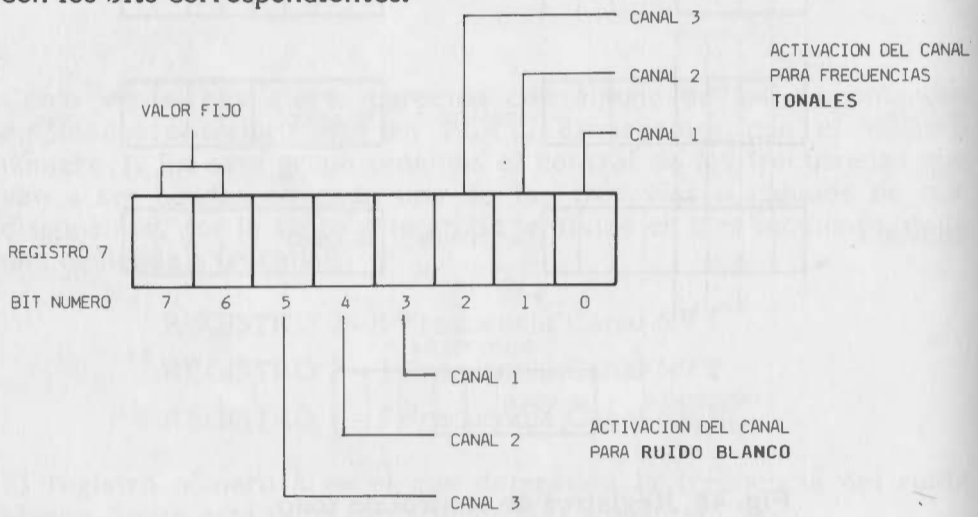


Fig. 49 Registro nº 7. Conexión de canales

Este registro es también accedido por la instrucción SOUND, pero en este caso es preferible utilizar directamente un número **binario** para indicar cuáles son los registros que conectamos, por ejemplo:

```
SOUND 7,&B 10111101
```

Con esta instrucción indicamos que el canal número 2 está abierto para la emisión de tonos y cerrado para la emisión de ruido, los otros dos canales están cerrados para ambas cosas.

VOLUMEN

El grupo número 3 compuesto por los registros 8, 9 y 10 es el que se encarga de indicar el volumen con que debe ser reproducida una nota determinada, podemos ver que el resto de los grupos no sirven para nada, mientras no se conecte o autorice el canal (o canales) correspondientes en el registro número 7.

Los tres registros que controlan el volumen pertenecen cada uno a un canal. En la figura 50 puedes ver también que el número de bits disponibles para el control es de 5, por lo tanto, el valor máximo será de 16, si recuerdas el subcomando "V" con el que determinábamos el volumen en PLAY, verás que coincide en su función.

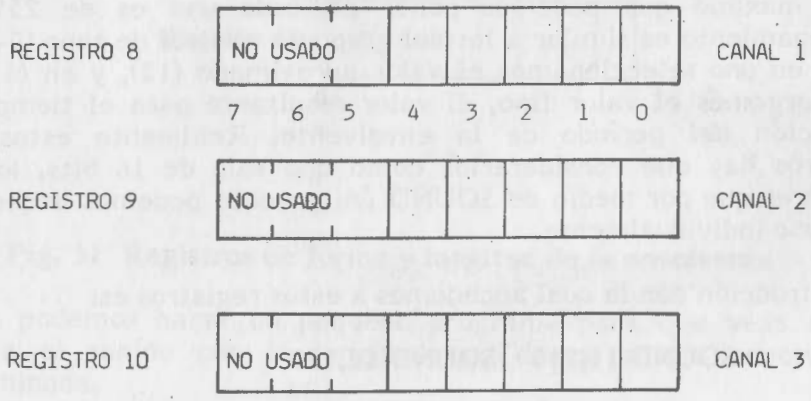


Fig. 50 Registros de volumen

Como es natural, estos registros (8, 9 y 10) son también accedidos por la instrucción SOUND, con un valor máximo de 16 y mínimo de 0, correspondiente éste último (como pasaba con "V") a sonido inaudible. Ya que tenemos los elementos necesarios para producir un poco de ruido, vamos a hacer un pequeño programa que nos muestre lo que podemos hacer con SOUND:

```
10 SOUND 7,&B 10101111:SOUND 9,14
20 FOR F=0 TO 31:SOUND 6,F
30 FOR R=1 TO 500:NEXT R:NEXT:SOUND 9,0
```

¿Qué te parece? Hasta casi puede servir para nuestro cohete...

PERIODO

Siguiendo con el orden de los registros vamos a ver ahora los números 11 y 12, correspondientes al cuarto grupo. Estos dos registros tienen la misión de determinar la **longitud** de la envolvente seleccionada, si recuerdas, esto lo realizábamos en PLAY mediante el subcomando "M", y realmente es lo mismo, ya que es en estos dos registros donde **almacenamos** el valor dado con este subcomando.

En estos registros utilizamos en ambos sus ocho bits, por lo que el valor máximo que podemos poner en cada uno es de 255, su funcionamiento es similar a los del grupo de control de tono (0-5) es decir, en uno seleccionamos el valor aproximado (12), y en el otro (11) cargamos el valor **fino**, el valor resultante para el tiempo de repetición del período de la envolvente. Realmente estos dos registros hay que considerarlos como uno solo de 16 bits, lo que sucede es que por medio de SOUND únicamente podemos acceder a cada uno individualmente.

La instrucción con la cual accedemos a estos registros es:

```
SOUND 11,210   SOUND 12,120
```

Los valores, como ya dijimos, puedes ser entre 0 y 255.

ENVOLVENTE

Para poder utilizar correctamente estos registros es necesario fijar primero la **forma** de la envolvente (lo mismo que sucedía con "M"), esto lo realizamos mediante el último registro del cuarto grupo, el número 13.

Las envolventes que podemos seleccionar con este registro son las mismas que seleccionábamos con el subcomando "S" de PLAY, es decir, 15. Por lo tanto, únicamente necesitamos cuatro bits para poder hacer esta selección, y esos son los que usamos en este registro, cuatro bits. En la figura 51 tienes representados estos tres últimos registros.

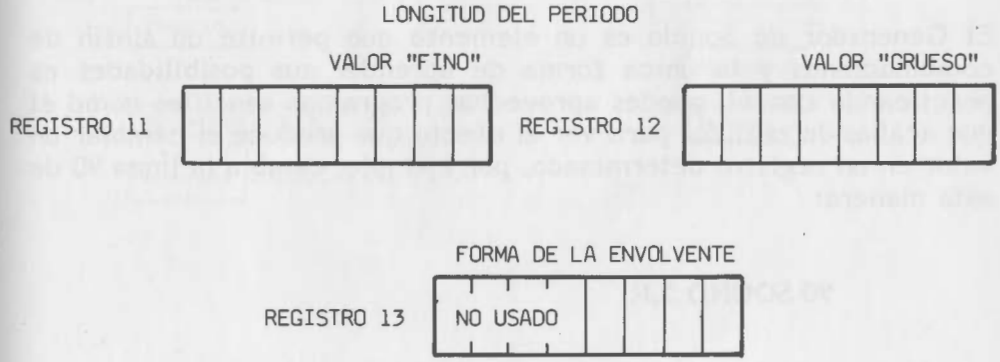


Fig. 51 Registros de forma y longitud de la envolvente

Ahora podemos hacer un pequeño programa para que veas cómo cambia el sonido con la longitud que demos a una envolvente determinada.

Una vez que lo hayas tecleado, ponlo en marcha y pulsa RETURN, cada vez que lo hagas notarás cómo cambia el sonido, prolongándose el tiempo de **ascensión** del volumen, esto es debido al cambio de valor que cargamos en el registro número 12, este valor es escrito en la pantalla con el fin de que te pueda servir de orientación a la hora de tener que seleccionar un sonido.

```
10 SOUND 7,&B 10111011
20 SOUND 13,10
30 SOUND 11,255
40 SOUND 12,0
50 SOUND 4,255
60 SOUND 5,0
70 SOUND 10,16
80 INPUT A:R=R+1:PRINT R
90 SOUND 12,R
100 GOTO 80
```

El Generador de Sonido es un elemento que permite un sinfin de combinaciones y la única forma de aprender sus posibilidades es practicando con él, puedes aprovechar programas sencillos como el que acabas de realizar para ver el efecto que produce el cambiar un valor en un registro determinado, por ejemplo, cambia la línea 90 de esta manera:

```
90 SOUND 5,R
```

Para clarificar en lo posible el funcionamiento de las distintas variantes de SOUND y de la PSG en general, en la figura 52 está representado el **camino** que recorre el sonido y la influencia que tiene cada registro e instrucción el el resultado final.

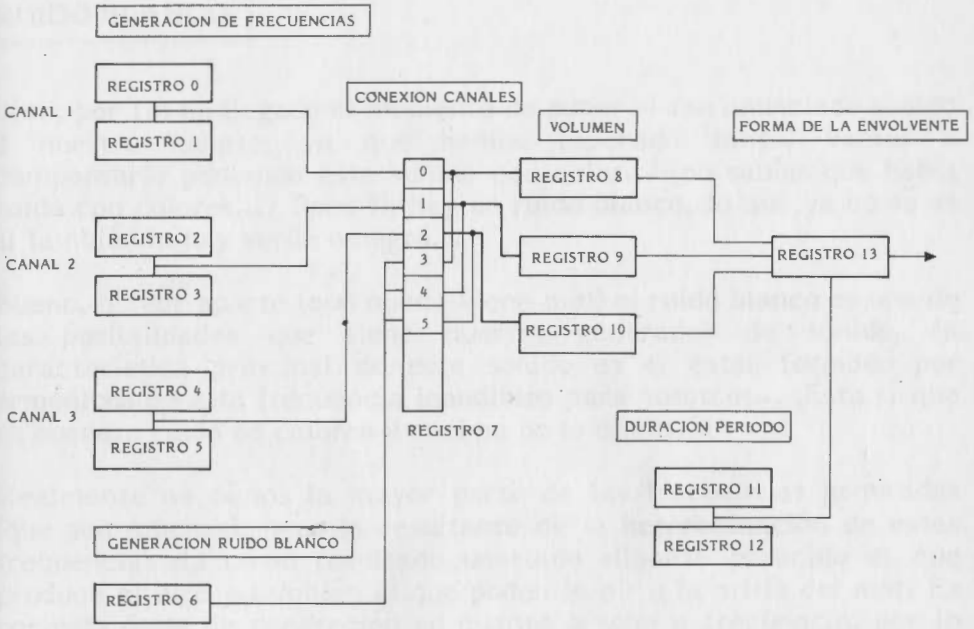


Fig. 52 Recorrido del sonido a través de los registros

Con esto hemos acabado con la introducción al Generador de Sonido Programable, en el próximo capítulo haremos uso de sus posibilidades para poner el sonido adecuado a nuestro programa.

Poniendo Sonido a Nuestro Cohete

RUIDO BLANCO

Bien, por fin ha llegado el momento de poner el tan anunciado sonido a nuestro cohete; ya que hemos esperado tanto, vamos a compensarlo poniendo este sonido con color... ¿no sabías que había ruido con colores...? Pues sí, hay un **ruido blanco**, lo que ya no sé es si también lo hay verde o negro...

Bueno, humor aparte (que nunca viene mal) el **ruido blanco** es una de las posibilidades que tiene nuestro generador de sonido, la característica principal de este sonido es el estar formado por **armónicos** de alta frecuencia **inaudibles** para nosotros... ¡Esta sí que es buena... ruido de colores y encima no lo oímos...!

Realmente no oímos la mayor parte de las frecuencias generadas (que son muchas), pero la resultante de la **heterodinación** de estas frecuencias da como resultado un ruido silbante parecido al que produce el aire y también al que podemos oir a la orilla del mar. Es por esta falta de concreción en cuanto a tono o frecuencia, por lo que a la hora de buscar una denominación para este ruido, la de **blanco** le viene tan bien como cualquier otra, si bien, parece que este color, junto con el gris, son colores discretos y poco llamativos, característica ésta que también tiene nuestro ruido, mientras no subamos mucho el volumen...

Como vímos en el anterior capítulo, el registro número 7 es el encargado de autorizar la emisión del ruido blanco en los bits 3, 4 y 5 correspondientes a los canales 1, 2 y 3. Pero a diferencia de lo que sucedía con la generación de tonos, en donde cada canal tenía dos registros para la determinación de la frecuencia, para la generación de **ruido blanco** existe únicamente un registro común para los tres canales, es decir, la única modificación que podemos hacer de un canal a otro, en el caso de usar más de uno para la emisión de ruido, es el de poner distintos volúmenes en cada canal.

FRECUENCIAS DE RUIDO

En el registro número 6 disponemos únicamente de 5 bits para seleccionar el tipo de frecuencia base para nuestro ruido, con estos 5 bits el margen es (como ya sabrás, después de lo de la codificación de los sprites...) de 0 a 31, o sea, 32 posibilidades. La frecuencia que obtenemos con un valor de 31 es de aproximadamente 3.6 Khz, mientras que con un valor de 1 esta frecuencia se convierte en 111 Khz. Para que te sirva de referencia, en la figura 53 tienes una tabla de equivalencias entre el valor del registro número 6 y la frecuencia base obtenida.

VALOR REG. Nº 6	FRECUENCIA BASE
0	111.860 Khz.
1	111.860 Khz.
2	55.930 Khz.
3	37.286 Khz.
4	27.965 Khz.
5	22.372 Khz.
6	18.643 Khz.
7	15.980 Khz.
8	13.982 Khz.
9	12.428 Khz.
10	11.186 Khz.
11	10.169 Khz.
12	9.321 Khz.
13	8.604 Khz.
14	7.990 Khz.
15	7.457 Khz.

VALOR REG. Nº 6	FRECUENCIA BASE
16	6.991 Khz.
17	6.580 Khz.
18	6.214 Khz.
19	5.887 Khz.
20	5.593 Khz.
21	5.326 Khz.
22	5.084 Khz.
23	4.863 Khz.
24	4.660 Khz.
25	4.474 Khz.
26	4.302 Khz.
27	4.142 Khz.
28	3.995 Khz.
29	3.857 Khz.
30	3.728 Khz.
31	3.608 Khz.

Fig. 53 Frecuencias de ruido según el valor del registro nº 6

Como vemos, las frecuencias más bajas corresponden a los números más altos y entre los valores 0 y 1 no existe diferencia como podrás ver seguidamente. Después de lo explicado seguramente no tenemos todavía claro (a pesar del color) lo del **ruido blanco,** así que, aprovechando que no tenemos cargado nuestro ordenador vamos a hacer un pequeño programa que nos muestre cómo suenan cada una de estas frecuencias, teclea las líneas siguientes:

```
10 CLS
20 FOR S=0 TO 31
30 SOUND S,0
40 NEXT S
50 SOUND 7,7
60 SOUND 8,16:SOUND 9,16:SOUND 10,16
70 SOUND 13,13
80 FOR S=0 TO 31
90 SOUND 6,S
100 LOCATE 15,15:PRINT S
110 FOR R=1 TO 1500:NEXT R
120 NEXT S:BEEP
```

Bueno, ya que tenemos el **ruido blanco** un poco más "aclarado", vamos a utilizarlo (¡por fin!) en nuestro programa. La subrutina para la incorporación del sonido la tenemos situada a partir de la línea 7000. Esta subrutina nos va a servir para generar distintos tipos de ruidos dependiendo de la situación de nuestro programa, así en la parte que llevamos realizada hasta ahora tenemos varios sonidos que podemos definir claramente.

1º En el momento que antecede al despegue se produce una explosión, antes de salir las llamas por las toberas de los cohetes impulsores.

2º El despegue comienza con la ignición de los cohetes, pero vemos que el movimiento de la nave es paulatino, no abandonamos la plataforma de una forma brusca, por lo tanto el ruido debe variar también a medida que nos vayamos elevando.

3º El desacople de los cohetes y el tanque de combustible se realiza mediante la explosión de los pernos que los fijan entre sí y al COLUMBIA respectivamente.

Estos son los tres primeros **ruidos** o efectos sonoros que vamos a incorporar: explosión de salida, ignición e impulsión de los cohetes y explosiones en el desacople.

Como ya hemos dicho, esta subrutina nos va a servir para distintos tipos de sonidos y el sistema que vamos a emplear en su confección va a consistir en un loop común para la carga de los registros del generador, mediante las instrucciones READ y DATA, el tipo de sonido lo seleccionaremos mediante la instrucción RESTORE y el acceso a la línea adecuada.

Por aquello de las imágenes que dijeron los chinos, veamos en la figura 54 cómo es la carta de flujo de nuestra subrutina SONIDO.

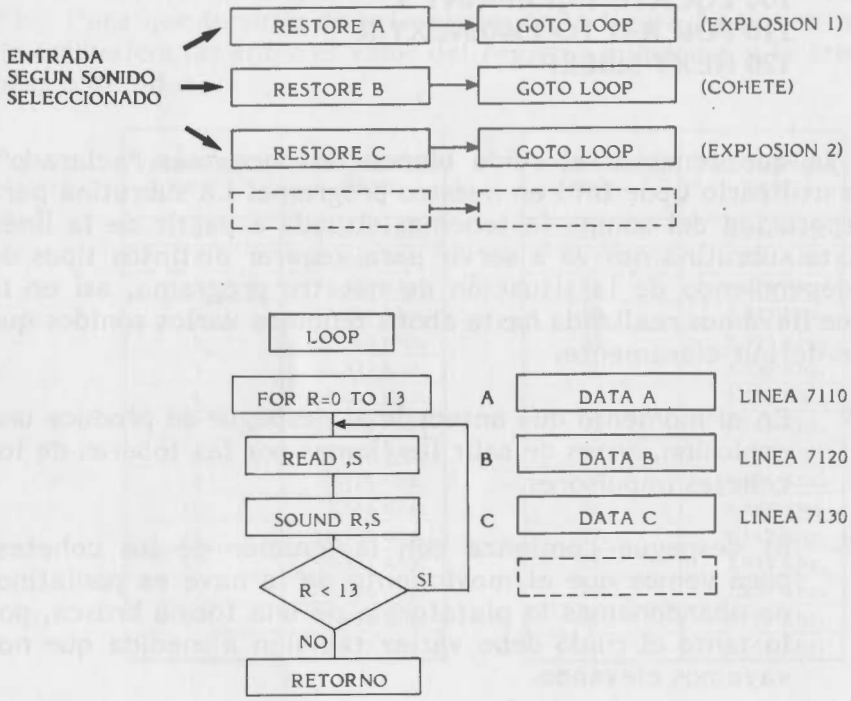


Fig. 54 Carta de flujo subrutina SONIDO

Como vemos, es una subrutina muy sencilla que carga nuestros 14 registros con el valor adecuado a cada sonido, la diferencia de esta subrutina con respecto a las que hemos realizado hasta ahora es la de tener entradas múltiples que condicionan su funcionamiento, es decir, el resultado que obtendremos de ella dependerá de la línea elegida para accederla.

Una vez explicada, vamos a teclearla, escribe NEW y pulsa RETURN; carga el programa (BASE12) y empecemos por el loop de carga de registros.

```
7000  REM SONIDO

7100 FOR R=0 TO 13:READ S:SOUND R,S:NEXT:R=0:RETURN
```

EXPLOSION 1

Lo primero que vamos a seleccionar para generar nuestra explosión es la forma de onda o envolvente, dado que el ruido que produce una explosión se genera de una forma brusca y se va reduciendo de una forma paulatina, la envolvente adecuada es la número 1, éste es el valor que daremos al registro número 13.

Después vamos a fijar una duración a este período, el valor de este tiempo lo cargaremos en los registros 11 y 12, ya sabes que cuanto más alto sea el valor de ambos registros (12 valor **grueso**, 11 valor **fino**) mayor será el período de la envolvente, el valor que daremos a nuestros registros será de 90, para el 12 y 0, para el 11.

CONTROL DE LA ENVOLVENTE

Con respecto a la envolvente, cuando empleamos SOUND en lugar de PLAY, hay que tener en cuenta que para la aplicación de la misma a un canal determinado, éste debe tener en su registro de volumen correspondiente (8, 9, 10) el valor 16. Cualquier valor inferior dará como resultado la anulación de la envolvente y la obtención de un volumen más bajo en ese canal.

Ahora fijaremos el valor del volumen en los registros 8, 9 y 10, este valor será de 16 en los tres registros. En el registro número 7 autorizaremos los canales 1, 2 y 3 para la obtención de **ruido blanco** mediante el valor 7.

Finalmente, en el registro número 6 pondremos una frecuencia de 3.857 Khz. correspondiente a un valor de 29.

Tecleemos esta data:

```
7010  RESTORE 7110:GOTO 7100

7110  DATA 0,0,0,0,0,0,29,7,16,16,16,0,90,1
```

Oigamos ahora cómo suena; para ello escribe esta instrucción y pulsa RETURN:

```
GOSUB 7000:PRINT A
```

Para parar el sonido pulsa CTRL y STOP simultáneamente, la segunda parte de la instrucción puede parecerte sin sentido, pero no es así. Siempre que accedemos a una subrutina debemos tener una parte del programa donde ésta pueda retornar, en el caso de que únicamente pusiéramos GOSUB 7000 la subrutina **no sabría** dónde volver y nos daría error, parando la ejecución del sonido, puedes hacer la prueba omitiendo :PRINT A.

Vamos a incorporar ahora nuestra explosión al programa.

```
510 H=1:GOSUB 240:GOSUB 7010:FOR R=1 TO 1000:
    NEXT
```

¿Qué te parece el efecto? Bueno, una vez que hemos explicado el método a seguir para la confección de una explosión, vamos a poner ahora el ruido de los cohetes.

IGNICION

Aquí no vamos a pormenorizar tanto, únicamente destacar que la forma de la envolvente que se adecúa a nuestras pretensiones es la número 13. Efectivamente vemos que esta envolvente tiene una subida paulatina para estabilizarse posteriormente, para que el tiempo de subida sea más largo daremos un valor alto a los registros 11 y 12, puedes examinar el valor de cada uno de los registros y su significado, teniendo en cuenta que el registro número 0 es el primero y el número 13 el último.

```
7020  RESTORE 7120:GOTO 7100

7120  DATA 60,5,8,13,100,1,24,4,10,16,16,8,200,13
```

EXPLOSION 2

El principio de esta explosión es el mismo de la número 1, únicamente variamos la duración y el volumen.

```
7030  RESTORE 7130:GOTO 7100

7130  DATA 0,0,0,0,0,0,01,7,06,06,06,0,00,0
```

Introduzcamos ahora estos dos últimos sonidos en nuestro programa:

```
510 H1=1:GOSUB 240:GOSUB 7010:FOR R=1 TO 1000:NEXT:
    GOSUB 7020

700 GOSUB 7010:FOR R=1 TO 300:NEXT:GOSUB 7020:SOUND 12,20

790 SOUND 13,9:SOUND 8,11

810 GOSUB 7010:SOUND 12,40:FOR R=1 TO400:NEXT:
    GOSUB 7030
```

En la línea 700 lo que hacemos es después recurrir a la subrutina de sonido, variar la duración (R12) de la envolvente para cambiar el sonido, ahora nuestra nave queda sola, sin impulsores ni motores principales, moviéndose gracias a la inercia adquirida.

Es hora de empezar la parte correspondiente a nuestra nave en órbita así que salva el programa cambiando el nombre antes de empezar con la siguiente parte, teclea:

```
CSAVE "BASE13"
```

Recuerda que debes comprobar la grabación antes de apagar el ordenador.

Listado SUBRUTINA SONIDO

```
7000  REM SONIDO
7010  RESTORE 7110:GOTO 7100
7020  RESTORE 7120:GOTO 7100
7030  RESTORE 7130:GOTO 7100
7100  FOR R=0 TO 13:READ S:SOUND R,S:NEXT:R=0:RETURN
7110  DATA 0,0,0,0,0,0,29,7,16,16,16,0,90,1
7120  DATA 60,5,8,13,100,1,24,4,10,16,16,8,200,13
7130  DATA 0,0,0,0,0,0,01,7,06,06,06,0,00,0
```

En Orbita

Para esta parte de nuestro programa vamos a utilizar de forma intensiva la instrucción DRAW, con ella vamos a construir un COLUMBIA bastante más grande que el que realizamos con los sprites, no podremos desplazarlo, pero podremos hacerlo más detallado. Cada instrucción tiene sus ventajas y sus inconvenientes, por eso lo bueno consiste en tener una gran variedad de estas instrucciones y usar la más adecuada en cada momento.

La práctica que vamos a adquirir haciendo el siguiente dibujo nos servirá posteriormente para la realización de la cabina, cuya complejidad es bastante más grande.

En este capítulo, como su nombre indica, vamos a simular al COLUMBIA orbitando nuestro planeta, en la pantalla aparecerá en primer término nuestra nave viéndose al fondo la luna, que se desplazará por el espacio donde están parpadeando las estrellas; debajo veremos nuestro planeta de color azul con una aureola más clara, una imagen igual a la representada en la figura 55.

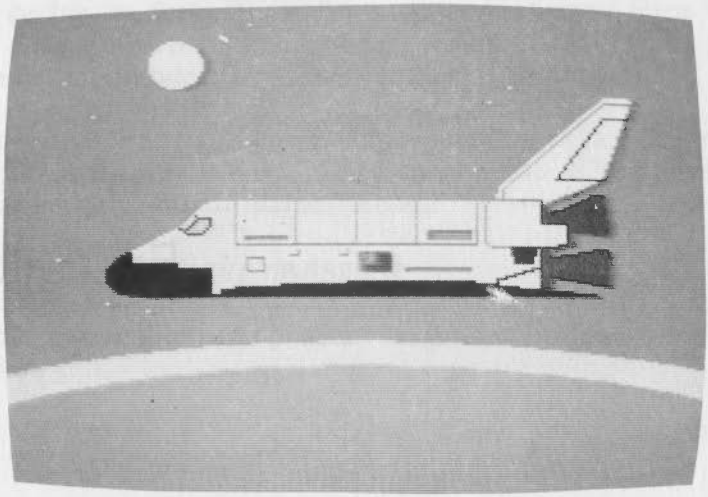


Fig. 55 El COLUMBIA en órbita

ASIGNAMIENTO DE CADENAS

Primeramente vamos a dibujar la nave (que es la parte más laboriosa) y para ello, como ya dijimos, utilizaremos la instrucción DRAW. Pero esta vez vamos a hacer la asignación de todas las cadenas que necesitamos para el dibujo a distintas letras, de manera que luego únicamente tengamos que indicar la posición donde deben dibujarse cada una de ellas, la carta de flujo de esta parte del programa es tan sencilla como la representada en la figura 56.

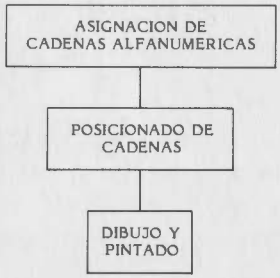


Fig. 56 Carta de flujo para dibujar el COLUMBIA

Como estarás sospechando, el programa es un poco más largo que el representado en esta figura, pero no mucho más y sobre todo verás que COLUMBIA te sale en la pantalla, hasta se lo podríamos alquilar a los americanos como reserva de los suyos (es broma...).

Para la realización de un gráfico de este tipo es necesaria la utilización de una plantilla milimetrada donde dibujaremos el motivo que queremos reproducir, la técnica es la misma que ya empleamos en la confección de nuestra torre, nubes y nave (sprites), por lo que no es necesario repetir lo ya explicado ni seguir paso a paso la realización de las cadenas, lo que sí es importante es poder identificar cada una de ellas con el dibujo que tiene asignado así como la posición desde donde empezamos a dibujarlas mediante DRAW.

POSICIONAMIENTO

Para lograr este fin, en la figura 57 están representados todos los dibujos que van a constituir nuestra nave, junto al dibujo hay una letra, la cual corresponde a la cadena alfanumérica donde va a estar **almacenado** este dibujo. Asimismo, vemos también dos números separados por un punto, corresponden (como ya habrás imaginado) a las coordenadas X,Y desde donde se empieza a trazar la instrucción DRAW.

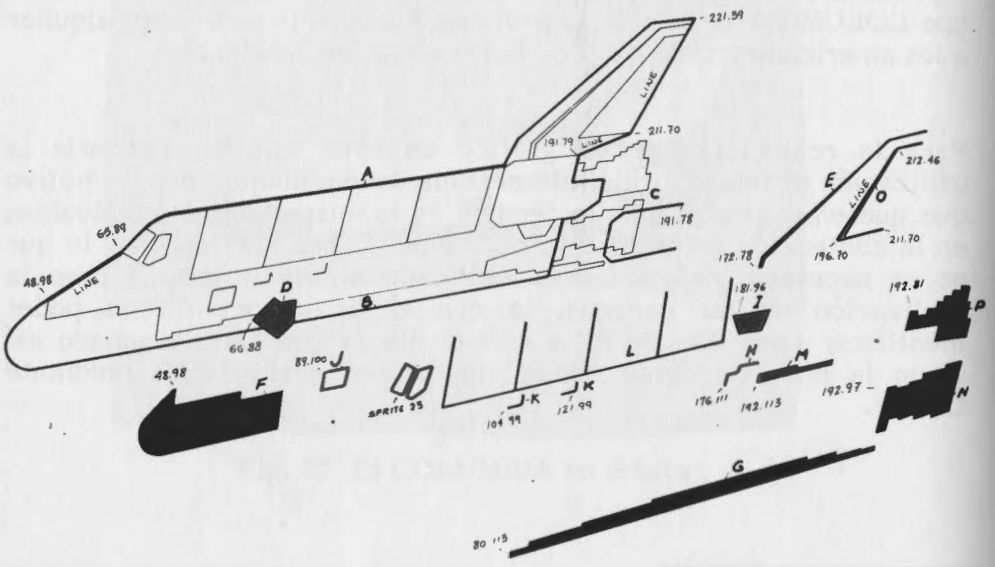


Fig. 57 Posicionamiento de los dibujos

Como verás, en el fuselaje y timón hay varias instrucciones LINE, esto es debido al ángulo de estas líneas. Como sabes con la instrucción DRAW los ángulos con los que podemos trazar líneas son fijos, así que cuando, como ahora, tenemos que trazar líneas con ángulos distintos a los que posee DRAW debemos recurrir a LINE para el trazado de éstas.

De todas formas, a pesar de no haber realizado el dibujo totalmente con DRAW, el porcentaje del mismo ejecutado con esta instrucción es superior al 95% y en el caso de haber tenido que utilizar una instrucción como LINE para la confección de este dibujo te puedo garantizar que la longitud del programa excedería ampliamente al que vamos a realizar ahora.

SEGUNDA FASE

Y hablando de programas, ha llegado el momento de introducir las líneas correspondientes al dibujo de la figura 57, pero antes de proceder a teclear vamos a repasar los bloques realizados hasta ahora. Si vemos el capítulo EL PORQUE DE LOS BLOQUES, observaremos que el último es el correspondiente al SONIDO, efectivamente con este bloque finalizábamos la primera fase, pero ahora empezamos la segunda; ya tenemos nuestra nave en el espacio y estamos en órbita, así que vamos a realizar una distribución de esta segunda fase (no confundir con la segunda parte).

GRAFICOS COLUMBIA

En esta parte realizaremos el dibujo de la figura 57 y añadiremos algunos detalles más, la definición de estos gráficos por comparación con la anterior distribución, es la de gráficos fijos.

PAISAJE ESTELAR

Aquí crearemos nuestra tierra, la luna y las estrellas y haremos que éstas parpadeen así como que se mueva la luna, para lo cual esta subrutina hará uso a su vez de otras dos subrutinas:

MOVIMIENTO LUNA y **MOVIMIENTO ESTRELLAS**

CAPTURA SATELITE

Con este programa trataremos de atrapar a un satélite averiado que está flotando en el espacio, para ello deberemos ajustar la velocidad de nuestra nave a la que tiene el satélite y cogerle mediante el brazo articulado que posee el COLUMBIA, para eso deberemos abrir previamente las dos grandes compuertas que posee esta nave, una vez capturado será el momento de guardarlo en la bodega, cerrar las puertas y regresar a la tierra.

Este bloque en sí es un programa completo donde tendremos secciones como INPUT INICIALIZACION DIBUJO BRAZO MOVIMIENTO , etc.

COLISION

Esta será una subrutina donde comprobaremos en qué condiciones se ha producido el contacto de nuestro brazo con el satélite, si nos traemos un satélite más a la tierra o un brazo menos...

SALIDA ORBITA

Una vez que hayamos conseguido capturar nuestro satélite (o cargárnosle...) será el momento de pasar a la continuación de este programa, saliendo de la órbita en que estamos e iniciando la fase de aterrizaje.

Con todo esto la distribución por bloques de esta parte del programa queda como está representada en la figura 58.

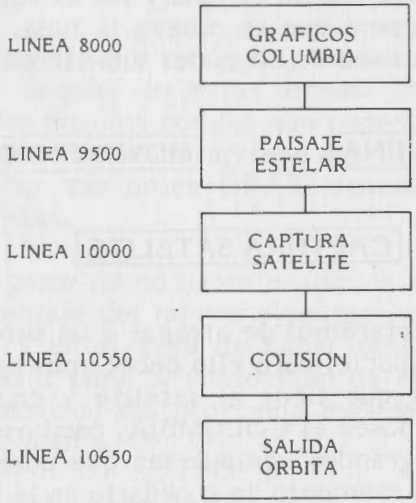


Fig. 58 Distribución en bloques de la 2ª fase

Gráficos Columbia

En esta parte vamos a proceder a teclear directamente las líneas correspondientes al primer bloque de esta sección, la asignación de las cadenas alfanuméricas correspondientes a los dibujos de las figuras 57 y 58. Antes de cargar el programa, veámos la carta de flujo de este bloque en la figura 59.

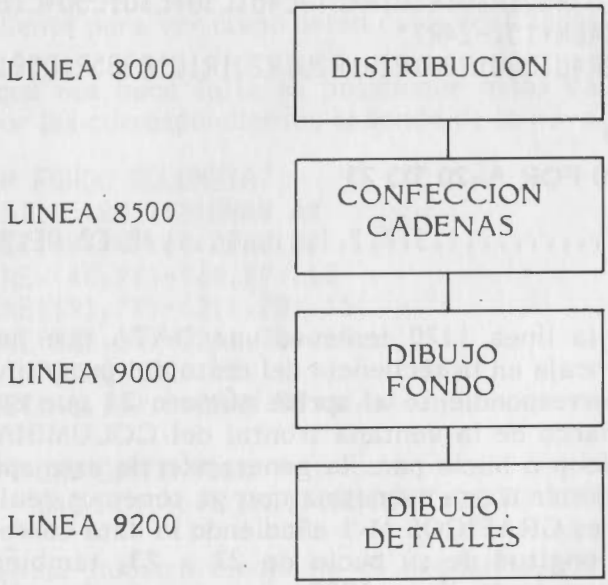


Fig. 59 Distribución de GRAFICOS COLUMBIA

Vamos a empezar por la distribución y la confección de cadenas, carga ahora el programa (BASE13) y escribe las siguientes líneas:

```
8000 REM 2.PARTE COLUMBIA
8010 CLEAR
8030 GOSUB 8500
8040 GOSUB 9000
8050 GOSUB 9200
8500 REM GRAFICOS COLUMBIA
8510 A$="S4E9R1E1R100U3E1U2E35R8"
8520 B$="S4L1D1L2G5D5F4R6F1R140U18R5E1R2E1U7L9U9"
8530 C$="L18D1G1D14F2R16"
8540 D$="D1F2R5E3U3L5H1G5"
8550 E$="R6U4E34R9"
8560 F$="D4R16D2R12D9"
8570 G$="U1R5U1R16U1R23U1R44D1R7D1R8D1R7"
8580 H$="U3R3U1R3U1R3U1R3U1R3"
8590 I$="D3F1D2R6U2E1U3L8"
8600 J$="D5R6U5L6"
8610 K$="R3U2"
8620 L$="D16R21U16D16R21U16D16R21U16D16R21U16"
8630 M$="R9D1R8L17"
8640 N$="R1D3R5U1R6U1R5U1R4D13L4U1L5U1L6U1L5D4L1U13"
8650 O$="L14BM+15,-24R7"
8660 P$="D2R4U1R3U1R2U1R2U1R2U1R2U1R1U1R3D5R1D5R1D4L1D1L3
     U1L8"
8700 RETURN
```

1020 FOR A=20 TO 23

```
1170 DATA ,,,,,,,,,,1,3,7,2,1,,,,,,,,,,40,E0,DE,BE,7E,FC
     ,F8,
```

Vemos que en la línea 1170 tenemos una DATA que junto con la línea 1020 no encaja en la secuencia del resto del programa, se trata de la DATA correspondiente al **sprite** número 23 que es el que se superpone al marco de la ventana frontal del COLUMBIA, en lugar de hacer otro loop o bucle para la generación de este **sprite** lo que hacemos es recurrir a una subrutina que ya tenemos realizada para este fin, como es GRAFICOS M-1 añadiendo la data correspondiente y variando la longitud de su bucle de 22 a 23, también podemos variar el nombre en la línea 1000 para que quede reflejado este nuevo **sprite:**

1000 REM GRAFICOS M1/VENTANA

Con el fin de que puedas verificar la DATA con el **sprite** número 23, en la figura 59 tienes la distribución dentro de su matriz del dibujo de dicho **sprite.**

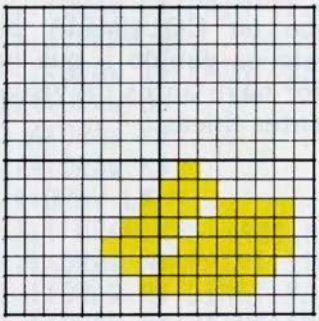


Fig. 60 Sprite ventana frontal

Puedes comprobar cada una de las cadenas con su dibujo correspondiente para ver cómo están éstos realizados.

Ahora lo que nos hace falta es posicionar estas cadenas, vamos a empezar por las correspondientes al **fondo** de la nave.

```
9000 REM FONDO COLUMBIA
9010 PSET(65,89),15:DRAW A$
9020 PSET(48,98),15:DRAW B$
9030 LINE (48,98)-(64,89),15
9040 LINE(191,79)-(211,70),15
9050 LINE(221,39)-(211,70),15
9060 PAINT(112,102),15
9070 PSET(48,98),1:DRAW B$
```

9080 GOTO 9080
840 FOR R=1 TO 1500:NEXT:GOTO 8000

Para visualizar nuestro dibujo no es necesario que **ruedes** todo el programa, teclea las siguientes instrucciones y pulsa RETURN.

COLOR 1,4:CLS:SCREEN 2,2:GOTO 8000

Como verás, aparece una silueta de nuestra nave totalmente blanca y carente de detalles, pulsa ahora RUN y llega hasta esta fase para que veas qué sucede.

Como has visto, aparece un COLUMBIA pequeño sobre el que acabamos de realizar, esto es debido a que no hemos **limpiado** los sprites de nuestra anterior nave antes de visualizar ésta, hagámoslo ahora mediante las siguientes líneas.

```
8670 FOR R=7 TO 19
8680 PUT SPRITE R,(0,209)
8690 NEXT R
```

La anulación de los sprites la realizamos después de confeccionar las cadenas, ya que si la hiciéramos antes, el tiempo en el cual no hay dibujo en la pantalla sería todavía mayor que el actual, pues habría que sumarle el de esta parte del programa.

Es importante cuando realices gráficos móviles que hagas coincidir el borrado de uno con el posicionamiento del otro que le sustituye, este ejemplo tendremos oportunidad de emplearlo cuando hagamos el bloque de CAPTURA SATELITE.

DETALLES COLUMBIA

Una vez que ya tenemos el fondo de nuestra nave dibujado, no hay nada que nos impida (creo...) ponerle todos los detalles del dibujo (y alguno más que se te ocurra) para lo cual vamos a escribir las siguientes líneas:

```
9080 RETURN
9200 REM DETALLES COLUMBIA
9210 PSET(191,78),4:DRAW C$
```

```
9220 PSET(66,88),1:DRAW D$:PAINT(75,88),1
9230 PSET(172,78),4:DRAW E$
9240 PSET(48,98),1:DRAW F$:PAINT(60,107),1
9250 PSET(80,113),1:DRAW G$:PAINT(120,113),1
9260 PSET(176,111),1:DRAW H$
9270 PSET(181,96),1:DRAW I$:PAINT(182,99),1
9280 PSET(89,100),4:DRAW J$
9290 PSET(104,99),4:DRAW K$
9300 PSET(121,99),4:DRAW K$
9310 PSET(85,79),4:DRAW L$
9320 PSET(192,113),1:DRAW M$
9330 PSET(192,97),1:DRAW N$:PAINT(194,102),1
9340 PSET(211,70),1:DRAW O$:LINE(196,70)-(212,46),1
9350 PSET(192,81),1:DRAW P$:PAINT(196,84),1
9360 PUT SPRITE 7,(61,75),11,23
```

9490 GOTO 9490

Visualiza esta parte como te indiqué anteriormente:

COLOR 1,4:CLS:SCREEN 2,2:GOTO 8000

Bonito ¿verdad? Bueno, autoelogios aparte, creo que queda bastante bien, en cualquier caso y por eso de que en la programación hay duendes (como en las imprentas) comprueba que el dibujo queda como el de la figura 61, si por casualidad durante la generación de la nave la pantalla comienza a **llenarse** de blanco o de negro, revisa detenidamente las cadenas y las instrucciones PSET y PAINT, ya que esto es debido a que un dibujo ha quedado **abierto** y por ahí se **escapa** el color, para ver cuál es puedes utilizar la tecla STOP, mediante esta tecla puedes hacer avanzar el programa poco a poco (dependiendo de lo rápido que des las pulsaciones) y ver cuál es el dibujo que origina el problema, el paso siguiente es sencillo, identifica la cadena correspondiente en la figura 57 y revísala en el programa listando éste.

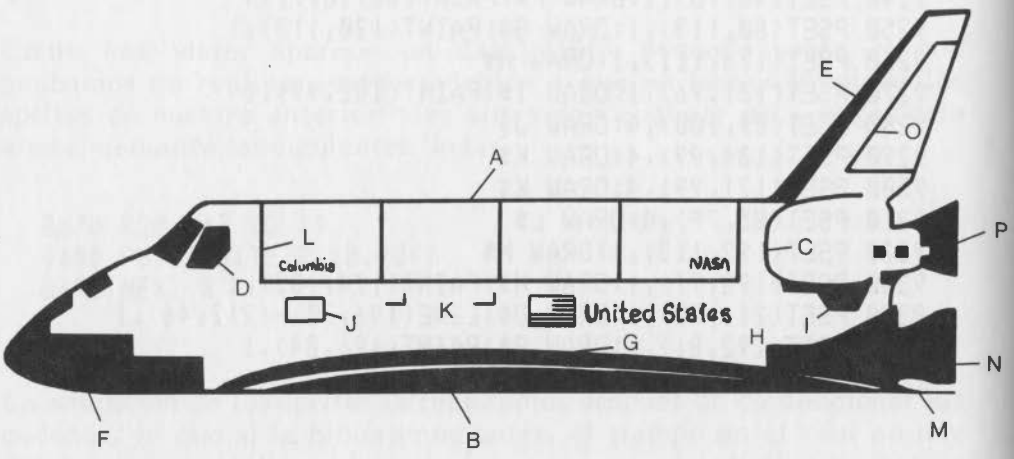


Fig. 61 COLUMBIA con sus detalles

MEZCLA DE CARACTERES

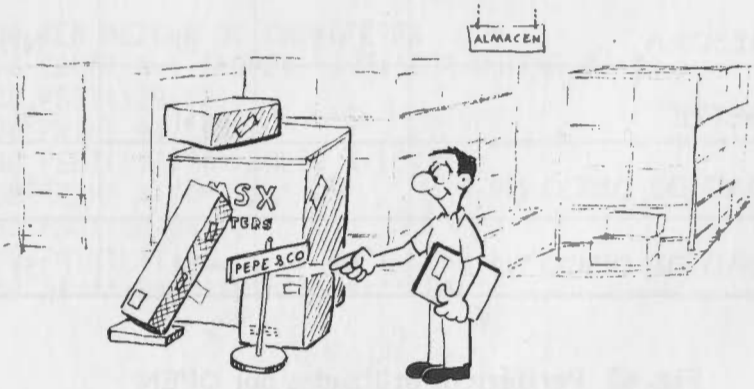
Hasta ahora todos nuestros dibujos han sido realizados utilizando únicamente las instrucciones gráficas que poseemos, LINE, DRAW, CIRCLE, PSET, etc., sin embargo, no hemos usado para nada el **texto** en nuestras realizaciones, y cuando me refiero al texto quiero significar el conjunto de caracteres de que dispone nuestro ordenador, signos, números, letras y caracteres gráficos.

De estos últimos, nuestro ordenador posee un buen repertorio de ellos y mediante su utilización es posible también la creación de dibujos y figuras.

OPEN

Estamos hablando de utilizar el **texto** en nuestros dibujos y esto parece imposible si nos atenemos a las explicaciones dadas cuando tratamos los **modos** gráficos en la instrucción SCREEN. Como recordarás, esta instrucción tenía los números 0 y 1 para texto (32 y 40 caracteres) y 2 y 3 para gráficos (alta y baja resolución), pero no existía en ninguna de estas cuatro posibilidades la que contemplara el uso simultáneo de texto y gráficos, sin embargo, esta posibilidad existe mediante el uso de la instrucción OPEN.

OPEN significa **abrir** o **abierto** en inglés y esto es lo que hacemos con esta instrucción, **abrir** un **buffer** para entrada o salida de datos en nuestro ordenador. Esto de **buffer** suena como cuando pisamos la cola a un gato (si tienes uno, haz la prueba...), la verdad es que hay cada palabreja que se las trae... Bromas aparte, diremos que **buffer** viene a significar algo así como almacén o depósito auxiliar. Imagínate una gran fábrica de productos, esta fábrica almacena toda su producción en un almacén general que tiene anexo, pero como tiene varios clientes importantes, dentro del almacén tiene reservados espacios específicos para estos clientes donde deposita los productos que éstos le encargan, cada cliente no tiene más que ir a la sección que le corresponde y retirar de allí su mercancía, sin tener que estar buscándola por todo el almacén.



Pues más o menos esto es lo que hacemos con OPEN, reservamos o **asignamos** un espacio determinado para poner nuestro producto, que en este caso son datos o texto. Primeramente debemos saber de qué **cliente** se trata para que nuestro ordenador sepa en qué área debe almacenar estos datos, los clientes en nuestro caso reciben el nombre genérico de **periféricos** y pueden ser desde la pantalla de texto hasta una unidad de disco, pasando por el cassette y la impresora.

También debemos dar al **cliente** la posibilidad de que nos deje productos a nosotros, una fábrica no únicamente produce, también consume materiales para elaborar sus productos, lo mismo pasa con nuestro ordenador, él genera los datos a partir de las informaciones que nosotros le suministramos así que en OPEN tenemos las dos posibilidades, sacar datos de nuestro ordenador (OUTPUT) o introducirlos (INPUT). En la figura 62 tienes una tabla con los nombres de los periféricos junto con su abreviatura y las posibilidades de INPUT/OUTPUT que tiene cada uno de ellos.

PERIFERICO	ABREV.	OUTPUT	INPUT
PANTALLA DE TEXTOS	CRT	SI	NO
PANTALLA DE GRAFICOS	GRP	SI	NO
IMPRESORA	LPT	SI	NO
CASSETTE	CAS	SI	SI
UNIDAD DE DISCO Nº 1	A	SI	SI
UNIDAD DE DISCO Nº 2	B	SI	SI

Fig. 62 Periféricos utilizados por OPEN

Podemos ver que los tres primeros **periféricos** únicamente tienen la posibilidad de **sacar** información del ordenador, esto es lógico si tenemos en cuenta su tipo, la pantalla y la impresora son eminentemente periféricos de **salida**, únicamente en el caso de la pantalla es posible su utilización como periférico de **entrada** mediante el uso de un lapicero óptico, pero esto requeriría un programa especial para su utilización y no la instrucción OPEN.

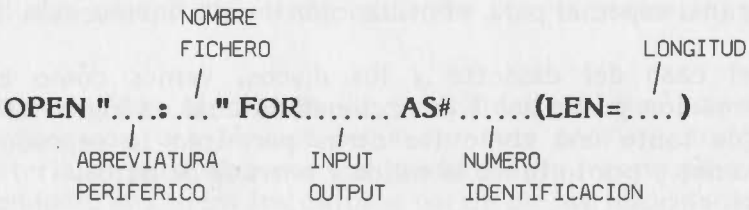
En el caso del cassette y los discos, vemos cómo el flujo de información puede ser bidireccional, lo cual es lógico en este caso, ya que tanto uno como los otros permiten la reproducción y la grabación y por lo tanto la salida y entrada de datos.

Bueno, creo que ya está bien y que la parrafada ha sido bastante larga para que veamos de una forma práctica el uso de esta nueva instrucción, lo que vamos a hacer es añadir algunos detalles a nuestra nave, habrás visto que los americanos están muy orgullosos de su bandera y la ponen por todos lados y como es lógico, el COLUMBIA tiene la suya, así que vamos a ponérsela. También habrás observado que esta nave está llena de cartelitos como "UNITED STATES", "NASA", "RESCUE", etc., pues vamos a simulárselos también (dada la escala no podemos reproducirlos).

Escribe estas líneas teniendo en cuenta que los símbolos que figuran entre comillas son **caracteres gráficos** y que debes pulsar la tecla GRAPH para su obtención (una vez obtenido el carácter púlsala de nuevo para seguir con texto normal), asimismo, guarda los espacios que hay entre caracteres:

```
9400 REM MEZCLA DE CARACTERES
9410 COLOR 8,1,1:OPEN "GRP:"FOR OUTPUT AS #1
9420 PSET(128,98),1
9430 PRINT #1,"▀"
9440 PSET(128,98):COLOR 4,15,1
9450 PRINT #1,"▀ ▬"
9460 PSET(88,86),1
9470 PRINT #1,"▬      ▄":CLOSE
9480 A$="":B$="":C$="":D$="":E$=""
```

Examinemos el programa, primeramente damos una instrucción COLOR, con ello determinamos el color con el que vamos a imprimir los caracteres (en este caso que usamos la pantalla como salida), después viene nuestra instrucción OPEN, la sintaxis de esta instrucción es la siguiente:



En nuestro programa hemos omitido el nombre del fichero ya que cuando se utiliza esta instrucción donde únicamente es posible la salida (OUTPUT) no es necesaria su útilización, después de la abreviatura del periférico vemos las expresiones FOR OUTPUT AS el símbolo "# " y el número "1", esto lo podíamos traducir como "PARA (FOR) SALIDA (OUTPUT) COMO (AS) BLOQUE Nº 1", es como si a un área asignada a un cliente en el almacén de nuestro ejemplo la denomináramos con un número (el "1" en este caso) y la destináramos para salida de nuestros productos únicamente, todos los productos destinados a este cliente los pondríamos siempre en este área, de donde serían retirados por él.

LEN

Esta es una función que nos sirve para determinar el número de caracteres que están contenidos en una cadena, por ejemplo, escribe lo siguiente y pulsa RETURN

```
A$="ESTA ES UNA PRUEBA":PRINT LEN (A$)
```

Como verás, obtenemos el número de caracteres que forman la cadena "A$", en el caso de OPEN la función LEN es usada para determinar la longitud de nuestro fichero (el número de paquetes que vamos a poner en nuestro área reservada), en el caso de omitir LEN la longitud que se dará automáticamente será de 256 caracteres.

POSICIONAMIENTO

Vemos en la línea 9420 la instrucción PSET indicando unas coordenadas, estas coordenadas corresponden con la parte superior izquierda de la **matriz** del carácter que vamos a imprimir, como ya dijimos al tratar de la definicin gráfica de nuestra pantalla, los caracteres están realizados dentro de matrices de 8 x 8 pixels, tal y como está representado en la figura 63.

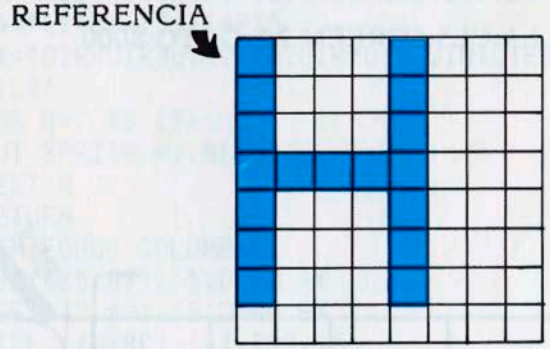


Fig. 63 Situación de la letra H dentro de su matriz

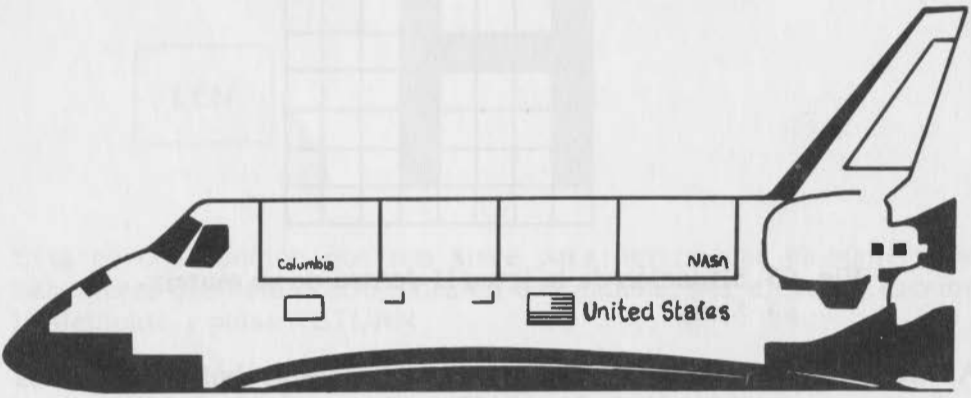
Al utilizar la pantalla gráfica como archivo o **buffer** debemos indicar a partir de cuál de las 49152 posiciones (192 x 256) posibles queremos situar nuestra cadena de caracteres, es para este fin para lo que usamos la instrucción PSET.

PRINT#

Posteriormente, en la línea 9430 vemos la instrucción PRINT # 1, seguida de un carácter gráfico, el símbolo "#" en la instrucción PRINT nos indica que los caracteres contenidos entre comillas van a ser escritos en un fichero, con el número "1" estamos identificando el fichero de que se trata (qué área dentro de nuestro almacén), en este caso el "1" que es el mismo que asignamos en OPEN.

El resto de las líneas es una repetición de lo anterior, únicamente reseñar que mediante la superposición de los caracteres gráficos en las líneas 9430 y 9470 se ha buscado simular la bandera americana, el resto de los caracteres imitan a los textos a los que aludíamos al principio, en la figura 64 tienes el listado completo de GRAFICOS COLUMBIA, puedes ver esta parte del programa con las siguientes instrucciones.

```
COLOR 1,4:CLS:SCREEN 2,2:GOTO 8000
```



Salva ahora el programa bajo el título "BASE14" y acuérdate de seguir todas las recomendaciones sobre grabación...

```
8000 REM 2.PARTE COLUMBIA
8010 CLEAR
8030 GOSUB 8500
8040 GOSUB 9000
8050 GOSUB 9200
8500 REM GRAFICOS COLUMBIA
8510 A$="S4E9R1E1R100U3E1U2E35R8"
8520 B$="S4L1D1L2G5D5F4R6F1R140U18R5E1R2E1U7L9U9"
8530 C$="L18D1G1D14F2R16"
8540 D$="D1F2R5E3U3L5H1G5"
8550 E$="R6U4E34R9"
8560 F$="D4R16D2R12D9"
8570 G$="U1R5U1R16U1R23U1R44D1R7D1R8D1R7"
8580 H$="U3R3U1R3U1R3U1R3U1R3U1R3"
8590 I$="D3F1D2R6U2E1U3L8"
8600 J$="D5R6U5L6"
8610 K$="R3U2"
8620 L$="D16R21U16D16R21U16D16R21U16D16R21U16"
8630 M$="R9D1R8L17"
8640 N$="R1D3R5U1R6U1R5U1R4D13L4U1L5U1L6U1L5D4L1U13"
8650 O$="L14BM+15,-24R7"
8660 P$="D2R4U1R3U1R2U1R2U1R2U1R2U1R1U1R3D5R1D5R1D4L1D1L3
     U1L8"
8670 FOR R=7 TO 19
8680 PUT SPRITE R,(0,209)
8690 NEXT R
8700 RETURN
9000 REM FONDO COLUMBIA
9010 PSET(65,89),15:DRAW A$
9020 PSET(48,98),15:DRAW B$
9030 LINE (48,98)-(64,89),15
9040 LINE(191,79)-(211,70),15
9050 LINE(221,39)-(211,70),15
9060 PAINT(112,102),15
9070 PSET(48,98),1:DRAW B$
9080 RETURN
9200 REM DETALLES COLUMBIA
9210 PSET(191,78),4:DRAW C$
```

```
9220 PSET(66,88),1:DRAW D$:PAINT(75,88),1
9230 PSET(172,78),4:DRAW E$
9240 PSET(48,98),1:DRAW F$:PAINT(60,107),1
9250 PSET(80,113),1:DRAW G$:PAINT(120,113),1
9260 PSET(176,111),1:DRAW H$
9270 PSET(181,96),1:DRAW I$:PAINT(182,99),1
9280 PSET(89,100),4:DRAW J$
9290 PSET(104,99),4:DRAW K$
9300 PSET(121,99),4:DRAW K$
9310 PSET(85,79),4:DRAW L$
9320 PSET(192,113),1:DRAW M$
9330 PSET(192,97),1:DRAW N$:PAINT(194,102),1
9340 PSET(211,70),1:DRAW O$:LINE(196,70)-(212,46),1
9350 PSET(192,81),1:DRAW P$:PAINT(196,84),1
9360 PUT SPRITE 7,(61,75),11,23
9400 REM MEZCLA DE CARACTERES
9410 COLOR 8,1,1:OPEN "GRP:"FOR OUTPUT AS #1
9420 PSET(128,98),1
9430 PRINT #1,"■"
9440 PSET(128,98):COLOR 4,15,1
9450 PRINT #1,"  ▬"
9460 PSET(88,86),1
9470 PRINT #1,"▬      ■":CLOSE
9480 A$="":B$="":C$="":D$="":E$=""
```

Fig. 64 Listado GRAFICOS COLUMBIA

Paisaje Estelar

Ya que tenemos acabada nuestra nave, vamos a proceder a crear el entorno donde va a desarrollarse su próxima misión, empecemos por grabar el programa principal (BASE14) para comenzar a dibujar nuestro planeta, escribe estas líneas:

```
8060 GOSUB 9500
9490 RETURN
9500 REM PAISAJE ESTELAR
9510 CIRCLE (127,190),194,7,,,.3
9520 LINE(0,150)-(255,191),7,BF
9530 PAINT(120,145),7
9540 CIRCLE (127,190),164,5,,,.3
9550 LINE(0,160)-(255,191),5,BF
9560 PAINT(120,158),5
```

9570 GOTO 9570

Como ves, es un programa tan sencillo que casi no necesita explicación, la única particularidad es el uso en la instrucción CIRCLE de una relación entre el radio vertical y el horizontal de .3 con el fin de **achatar** la imagen de nuestro planeta y que quede más real.

Para visualizar esta parte recurriremos al mismo sistema que empleamos anteriormente.

COLOR 1,4:CLS:SCREEN 2,2:GOTO 8000

POSICION ESTRELLAS

Lo que tratamos con esta subrutina es de crear una serie de coordenadas (30) donde poner nuestras estrellas, estas coordenadas no deben coincidir con nuestra nave, ya que **borraría** el dibujo de la misma, ni con la tierra por el mismo motivo, por lo tanto, las coordenadas no pueden ser generadas de una forma aleatoria y la mejor manera es asignar éstas mediante un loop y las instrucciones READ y DATA.

El número de estrellas que vamos a poner en nuestro firmamento es de 30, por lo tanto necesitamos almacenar 30 coordenadas X,Y, la forma de conseguirlo es asignando a una variable determinada (en este caso X,Y) distintos valores según la posición que ocupe dentro de una tabla. La cosa parece complicada pero no lo es, si miras la figura 65 verás más claro lo que quiero decir.

VARIABLE	POSICION	VALOR
X	(1)	10
X	(2)	32
X	(3)	56
X	(4)	90
X	(5)	156
X	(6)	167

Fig. 65 Valor de la variable X según su posición

La variable "X" tiene ahora asignadas varias posiciones y cada una de estas tiene un valor diferente (o igual), así que cuando queramos referirnos a un valor determinado de X tendremos que indicar también su posición junto con la variable.

DIM

Cuando utilizamos un sistema **multidimensional,** como ahora, es necesario indicar al ordenador la longitud que va a tener éste, es decir, el número de veces que una variable (numérica o alfanumérica) está contenida en una tabla, esto lo realizamos mediante la instrucción DIM. Cuando utilicemos una variable de valor múltiple sin esta instrucción declarando **previamente** la longitud de la tabla donde va a estar contenida el número máximo de elementos que esta tabla podrá tener será de 10, a partir de este número es necesaria la utilización de DIM si no queremos tener un error (y si queremos que el programa funcione).

Tecleemos ahora las siguientes líneas:

```
8020 DIM X(35),Y(35),C(35)
9950 REM POSICION ESTRELLAS
9960 FOR R=1 TO 30:READ X(R),Y(R):NEXT R:RETURN
9970 DATA 10,80, 32,11, 56,5, 90,9, 156,20, 167,12, 173,
     40, 245,3, 2,43, 25,31, 16,125, 35,45, 89,56, 125,7
     0, 110,65, 132,118, 189,123, 54,120, 250,110, 23,10
     0, 2,60, 10,67, 23,105, 40,117, 89,3, 92,5, 180,34,
     205,39, 250,70, 100,117
```

Vamos a ver ahora la asignación de los valores X e Y, escribe estas instrucciones:

```
CLS:DIM X(30),Y(30):GOSUB 9950:FOR A=1 TO
30:PRINT "X";A;"=";X(A),"Y";A;"=";Y(A):NEXT
```

Habrás podido comprobar la asignación de los distintos valores a cada una de las posiciones de X,Y, estos valores están **almacenados** en un área del ordenador de manera que si ahora escribes:

```
PRINT X(20),Y(20)
```

Comprobarás que el valor es el mismo que obtuviste para esta posición (20) en el anterior listado. Bueno, ya tenemos coordenadas, ahora es necesario utilizarlas para posicionar nuestras estrellas, pero esto lo haremos en la siguiente subrutina.

MOVIMIENTO LUNA

Con esta subrutina ponemos en nuestra pantalla una luna que se va desplazando hacia la derecha (sin significado político n.a.) en el fondo de la imagen, realmente lo que sucede es que nuestra nave se mueve hacia la izquierda (igualmente sin significado político n.a.), al mismo tiempo hacemos **parpadear** las estrellas que se ven en el espacio y como fondo sonoro vamos a poner un **beep** tipo satélite "ECO", que para aquéllos que no lo sepan diremos que el "ECO" fue uno de los primeros satélites que se pusieron en órbita y que lo único que hacía era beep, beep, servía para dormirte por la noche, previa sintonización radiofónica... (es broma n.a.).

Bien, dejando el humor a un lado (esto es muy serio... ¿o no?) vamos a ver primeramente la carta de flujo o flow chart de nuestra subrutina en la figura 66.

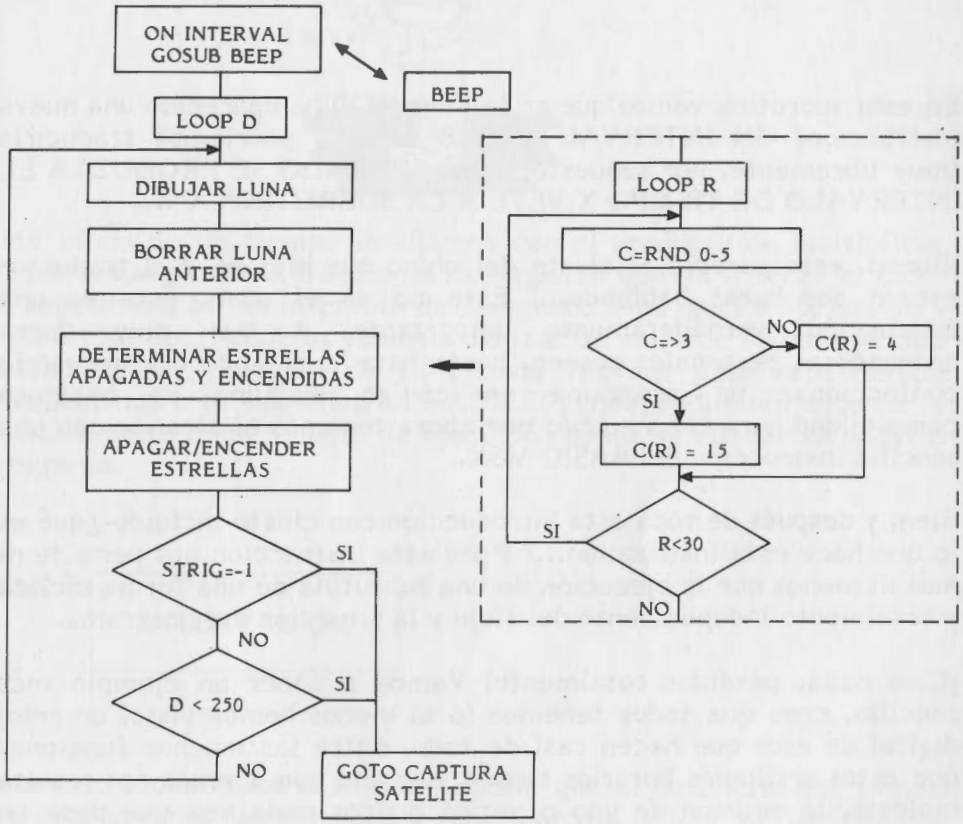


Fig. 66 Carta de flujo subrutina MOVIMIENTO LUNA

ON INTERVAL GOSUB

En esta subrutina vemos que en la línea 9720 ha aparecido una nueva instrucción, ON INTERVAL GOSUB la cual podriamos traducirla (muy libremente, por supuesto) como "CUANDO SE PRODUZCA EL INTERVALO DE TIEMPO X VETE A LA SUBRUTINA A"...

¡Bueno, esto parecía el chiste del chino que dijo ## y el traductor estuvo dos horas hablando...! Este no es el caso, ésta es una instrucción verdaderamente importante y que muy pocos ordenadores personales poseen, hasta hace muy poco era necesario confeccionar un programa en código máquina de bastante complejidad para conseguir lo que ahora tenemos al alcance con una sencilla instrucción del BASIC MSX.

Bien, y después de toda esta introducción con chiste incluido ¿qué es lo que hace esta instrucción...? Pues esta instrucción nos permite ni más ni menos que la ejecución de una subrutina de una forma **cíclica** y totalmente independiente del flujo y la situación del programa.

¡Casi nada, perdidos totalmente! Vamos a poner un ejemplo más sencillo, creo que todos tenemos (o al menos hemos visto) un reloj digital de esos que hacen casi de todo, entre las muchas funciones que estos artilugios horarios tienen, hay una que a veces nos resulta molesta, la emisión de uno o varios pitidos cada vez que pasa un determinado lapso de tiempo, este lapso o ciclo de tiempo es ajustable y una vez programado se repite incansable, independientemente de que nos guste o no, la única forma de pararlo es desprogramándolo. Pues esto más o menos es lo que podemos hacer con ON INTERVAL GOSUB, fijar el período de tiempo para que una vez transcurrido éste se ejecute una subrutina, esta ejecución, como ocurría con el reloj, no cesará hasta que desprogramemos esta instrucción.

Este intervalo de tiempo lo fijamos con el producto de multiplicar 50 por los segundos o fracciones de segundo que va a tener el ciclo de repetición, así un intervalo de 1 segundo será igual a 50, uno de 3 valdrá 150, etc. Nosotros vamos a utilizar un valor de 75, con el cual obtenemos un intervalo de 1.5", cada vez que pase este tiempo **accederemos** a la subrutina LUNA/BEEP, independientemente de la posición de la línea o punto de ejecución donde se encuentre nuestro programa.

INTERVAL ON

Esta instrucción hace la misma función que el botoncito que tienen los cronómetros para empezar una cuenta, inicia el período de tiempo establecido en ON INTERVAL GOSUB.

INTERVAL OFF

Anula la instrucción ON INTERVAL GOSUB.

INTERVAL STOP

Queda una última instrucción relacionada con ON INTERVAL GOSUB, es INTERVAL STOP. Esta instrucción desconecta la ejecución de ON INTERVAL GOSUB, pero sigue midiendo el tiempo transcurrido de tal manera que si el programa se encuentra con una instrucción INTERVAL ON y el tiempo transcurrido es mayor al fijado en ON INTERVAL GOSUB, la subrutina se ejecutará **inmediatamente** sin necesidad de que transcurra otro intervalo de tiempo.

Creo que el "intervalo" ha sido ya suficiente , así que vamos a teclear el programa de esta subrutina.

```
8070 GOTO  9700
9570 RETURN
9700 REM MOVIMIENTO LUNA ESTRELLAS
9710 RESTORE 9970:GOSUB 9950
9720 ON INTERVAL=75 GOSUB 9900:INTERVAL ON
9730 FOR D=0 TO 160
9740 GOSUB 9850
9750 IF STRIG(0)=-1THEN GOTO 10000
9760 GOSUB 9850
9770 NEXT D:GOTO 10000
9850 REM PARPADEO ESTRELLAS
9860 FOR R=1 TO 30:C=INT(RND(1)*5):IF C=)3 THEN C(R)=15 ELSE
     C(R)=4
9870 NEXT
9880 FOR R=1 TO 30:PSET (X(R),Y(R)),C(R):NEXT :RETURN
9900 REM SUBRUTINA LUNA/BEEP
9910 CIRCLE (X,25),10,4
9920 CIRCLE (X,25),9,15
9930 PAINT(X,25),15
9940 X=X+1:BEEP:RETURN
```

El sistema empleado para dibujar nuestra luna en la subrutina LUNA/BEEP y que ésta se mueva, es parecido al que ya usamos en nuestras nubes móviles, en la línea 9940 la variable "X" va incrementando su valor cada vez que es accedida la subrutina (1.5") variando una coordenada, en este caso la horizontal (X), el dibujo de un primer círculo con el radio mayor que el segundo y el color igual al fondo donde se traza (4), posteriormente hacemos otro círculo menor de color blanco y lo **rellenamos** de este color con PAINT, al variar la coordenada y trazar de nuevo los círculos, el exterior **borra** parte del interior (él no necesita borrarse ya que es del mismo color que el fondo) antiguo y al dibujarse el nuevo concéntricamente da la sensación de desplazamiento que vemos, esto lo tienes un poco más aclarado en la figura 67.

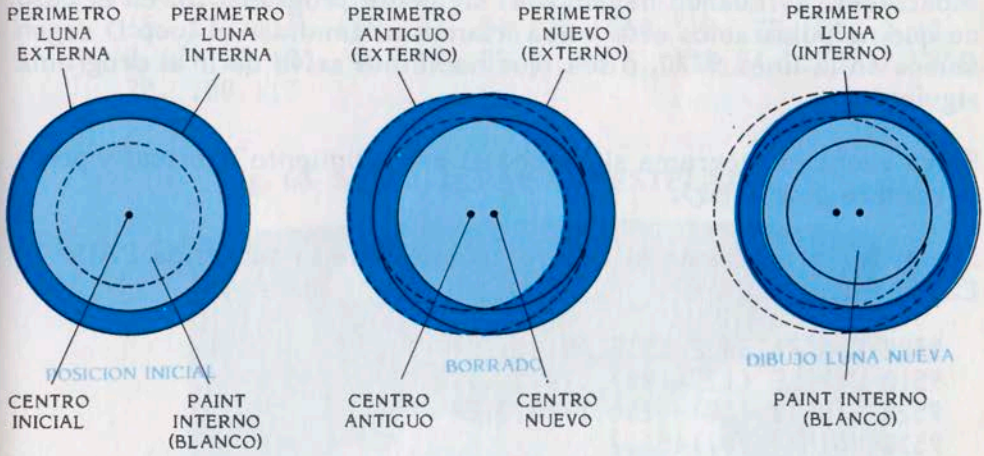


Fig. 67 Desplazamiento de la Luna

Este efecto (desplazar la luna) podríamos haberlo logrado también mediante el uso de SPRITE, pero he querido enseñarte cómo se puede lograr un efecto de animación en objetos sencillos sin recurrir a la programación de nuestros **sprites.** Visualiza este nuevo bloque con el mismo conjunto de instrucciones que utilizaste anteriormente.

La otra parte que se puede destacar de este programa es el loop de la línea 9860 donde mediante la función RND obtenemos los valores 15 ó 4 que asignamos posteriormente a C(R), estos valores son usados en el loop de la línea 9880 para **pintar** nuestras estrellas de blanco (15) o azul (4), en este último caso se hacen invisibles ya que éste es el color que tiene el fondo, con lo cual y dado el carácter aleatorio con el que se seleccionan debido a RND, obtenemos una sensación de parpadeo asíncrono.

Para finalizar, vemos la función STRIG (0) unida a la instrucción IF que nos enviará al siguiente programa cuando pulsemos la barra espaciadora (y cuando hagamos el siguiente programa...), en el caso de que no pulsáramos esta barra iríamos al finalizar el loop D según vemos en la línea 9770, o sea, que nadie nos salva de ir al programa siguiente.

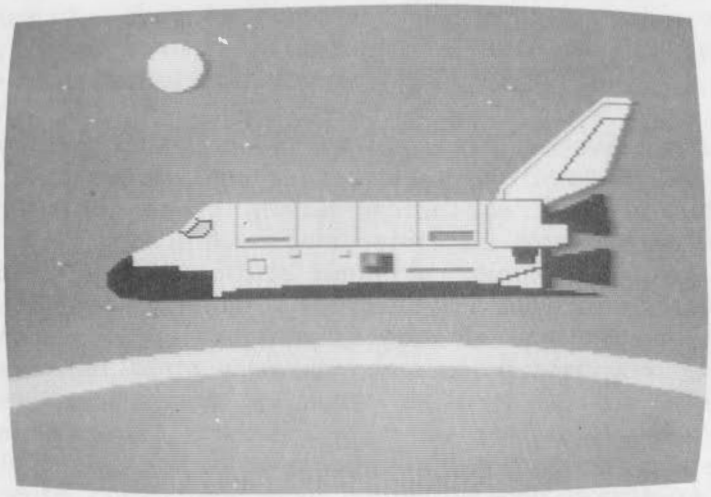
Salva ahora el programa siguiendo el procedimiento habitual y ponle el nombre de BASE15.

En la figura 68 tienes el listado completo de la subrutina PAISAJE ESTELAR.

```
9500 REM PAISAJE ESTELAR
9510 CIRCLE (127,190),194,7,,,.3
9520 LINE(0,150)-(255,191),7,BF
9530 PAINT(120,145),7
9540 CIRCLE (127,190),164,5,,,.3
9550 LINE(0,160)-(255,191),5,BF
9560 PAINT(120,158),5
9570 RETURN
9700 REM MOVIMIENTO LUNA ESTRELLAS
9710 RESTORE 9970:GOSUB 9950
9720 ON INTERVAL=75 GOSUB 9900:INTERVAL ON
9730 FOR D=0 TO 160
```

```
9740 GOSUB 9850
9750 IF STRIG(0)=-1THEN GOTO 10000
9760 GOSUB 9850
9770 NEXT D:GOTO 10000
9850 REM PARPADEO ESTRELLAS
9860 FOR R=1 TO 30:C=INT(RND(1)*5):IF C=>3 THEN C(R)=15 ELSE
     C(R)=4
9870 NEXT
9880 FOR R=1 TO 30:PSET (X(R),Y(R)),C(R):NEXT :RETURN
9900 REM SUBRUTINA LUNA/BEEP
9910 CIRCLE (X,25),10,4
9920 CIRCLE (X,25),9,15
9930 PAINT(X,25),15
9940 X=X+1:BEEP:RETURN
9950 REM POSICION ESTRELLAS
9960 FOR R=1 TO 30:READ X(R),Y(R):NEXT R:RETURN
9970 DATA 10,80, 32,11, 56,5, 90,9, 156,20, 167,12, 173,40,
      245,3, 2,43, 25,31, 16,125, 35,45, 89,56, 125,70, 110,
      65, 132,118, 189,123, 54,120, 250,110, 23,100, 2,60, 1
      0,67, 23,105, 40,117, 89,3, 92,5, 180,34, 205,39, 250,
      70, 100,117
```

Fig. 68 Subrutina PAISAJE ESTELAR



La Captura del Satélite

Antes de empezar a teclear esta parte del programa, vamos a describir su realización y cuáles son los objetivos que tenemos que cumplir. Se trata de simular la captura de un satélite artificial por la nave COLUMBIA. Esta nave cuenta dentro de su bodega de carga con un brazo mecánico, con el que es posible realizar una gran variedad de labores manejándole desde el interior, una vez abiertas sus compuertas. Entre éstas labores se encuentra la de capturar satélites, no se trata de cazar satélites como el que caza conejos, sino de recuperar uno que se encuentra averiado para repararlo (en la tierra o en la nave) poniéndole en órbita de nuevo.

Con lo descrito hasta ahora, ya tenemos dos cosas que debe hacer nuestro programa: abrir las compuertas de la nave y sacar el brazo articulado, lógicamente para capturar el satélite debemos ser capaces de controlar su movimiento.

A pesar de la articulación de nuestro brazo, todavía necesitamos más cosas para poder coger a un satélite artificial. Primeramente debemos situarnos en la misma órbita que él y seguidamente tendremos que adquirir su misma velocidad. Nosotros asumiremos que estamos en la misma órbita, siendo la velocidad, la otra función que tendremos que controlar. Esta puede ser superior o inferior a la del satélite, por lo que debemos disponer de sistemas de impulsión que nos permitan tanto avanzar como retroceder, con el fin de situarnos exactamente en el punto de captura. Un satélite no permite ser golpeado, por lo que si no ajustamos bien nuestra posición y le 'sacudimos' un porrazo, lo más probable es que el pobre satélite explote, convirtiéndose en humeante sopa de satélite.

Este es pues, otro de los objetivos, comprobar las posiciones relativas y ver si ha habido **colisión** entre el brazo y el satélite, dependiendo de las características de esta colisión, el satélite estallará o bien le podremos bajar a nuestra bodega para su reparación, incrementando nuestra cuenta de satélites capturados..

Con el fin de dar movimiento propio a nuestros satélites contaremos con una subrutina que se encargará de **darles vueltas.** Una vez que los hayamos capturado (o antes, si es que nos hemos cansado del tema...) deberemos cerrar nuestra bodega con el fin de regresar a la tierra. Para poder realizar esto contaremos con un sistema que comprobará que el brazo está abajo (¡imagínate que cierras con el brazo subido...!) antes de comenzar la operación de cierre.

Todas estas acciones deben ser llevadas a cabo dentro de un tiempo limitado, del que deberemos tener indicación, ya que una vez que expire éste no podremos capturar más satélites. Como final, saldremos de nuestra órbita para comenzar la fase de aterrizaje, pero esto será en la segunda parte de este libro (ATERRIZA CON TU MSX).

Una vez que hemos descrito nuestro programa podemos examinar en la figura 69 la representación simplificada de su carta de flujo.

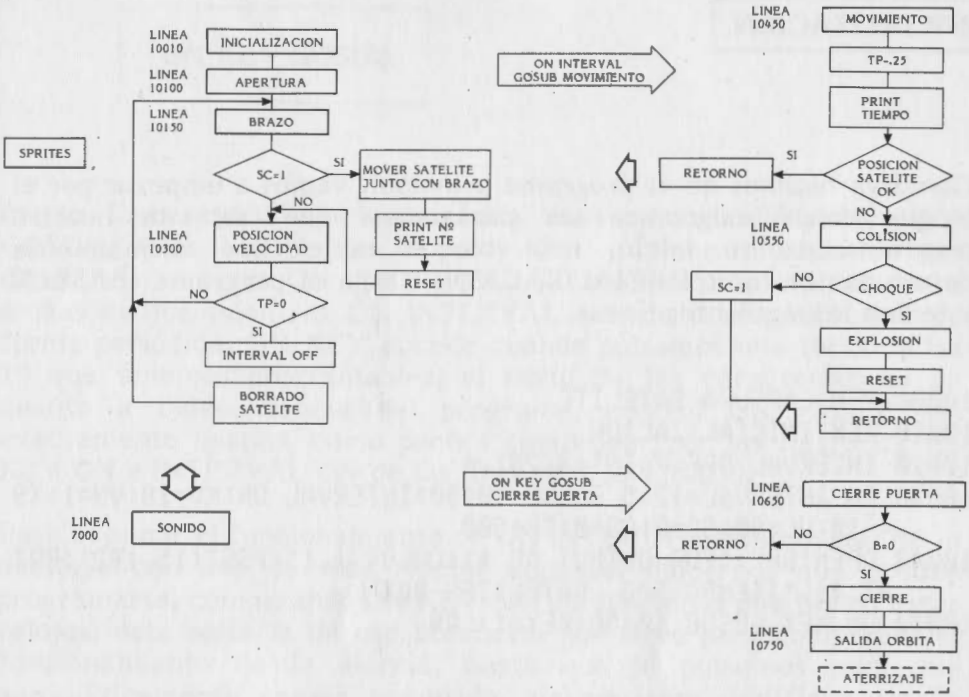


Fig. 69 Carta de flujo CAPTURA SATELITE

INICIALIZACION

Como ya hicimos en el programa principal, vamos a empezar por el bloque donde asignamos los parámetros que necesita nuestro programa en su inicio, este bloque es el que normalmente denominamos como INICIALIZACION. Carga el programa (BASE15) y teclea las siguientes líneas.

```
10000 REM CAPTURA SATELITE
10010 REM INICIALIZACION
10020 INTERVAL OFF:PAINT(X,25),4
10030 ON INTERVAL=12.5 GOSUB 10450:INTERVAL ON:MV=28:VN=1:XS
    =-10:VC=80:SC=0:CS=0:TP=500
10040 OPEN"GRP:"FOR OUTPUT AS #1:COLOR 1,15:PSET(15,182):PRI
    NT #1,"TIEMPO 500  SATELITES 00"
10050 ON KEY GOSUB 10650:KEY(1) ON
```

Como vemos en la línea 10020 desconectamos la función INTERVAL y utilizamos la instrucción PAINT, esto es con el fin de **apagar** nuestra luna y dejar libre el sistema de **interrupciones**, en la próxima línea (10030) volvemos a conectarlo, pero esta vez para acceder a otra subrutina situada en la línea 10450, que como puedes ver en la carta de flujo, corresponde a MOVIMIENTO.

Esta subrutina (MOVIMIENTO) es accedida cada 1/4 de segundo, como se desprende del valor de 12.5 que damos en la instrucción ON INTERVAL (1 segundo=50), posteriormente, y en esta misma línea, damos unos valores a una serie de variables de las que vamos a destacar de momento TP, esta variable corresponde al tiempo de que disponemos para la captura de nuestros satélites, 500 segundos.

En la línea 10040 **abrimos** nuestra pantalla imprimiendo nuestro tiempo y puntuación iniciales.

ON KEY GOSUB

En la línea 10050 empleamos una nueva instrucción, ON KEY GOSUB. Esta instrucción es muy útil y tiene muchas de las características de ON INTERVAL, la diferencia principal entre ambas es que mientras ON INTERVAL accede a una subrutina de forma periódica, ON KEY accede cuando pulsamos una tecla de las 10 que tenemos programables, el resto de las características en cuanto a independencia del programa, control ON y OFF son exactamente iguales, como puedes comprobar por las instrucciones KEY ON e INTERVAL ON de las líneas 10030 y 10050.

Para explicar el funcionamiento de ON INTERUP hacíamos uso de la analogía con uno de esos relojes digitales modernos que pueden programarse, comparando ON KEY con las funciones que tienen estos relojes, ésta sería la de ese botoncito que sirve para comprobar el funcionamiento de la alarma, basta que le pulsemos para que automáticamente genere un pitido, sin que esto signifique ningún cambio en su programa horario.

Con el fin de poder probar el programa a medida que le construimos, vamos a señalar, mediante sendas instrucciones REM, las dos subrutinas que hemos declarado, teclea estas líneas:

```
10450 REM MOVIMIENTO
10650 REM CIERRE PUERTA
10660 RETURN
```

De esta manera podemos acceder a las direcciones indicadas, ya que al tener la instrucción RETURN al final, siempre regresaremos al programa, es el mismo sistema que empleamos al principio cuando describimos REM.

APERTURA

En este bloque es donde haremos que nuestra puerta se abra y aparezca el brazo articulado, debido a que este con su articulación es relativamente complejo de realizar, vamos a usar uno **falso** en esta fase para posteriormente cambiarle por el de **verdad.**

INKEY$

¿Qué letra podemos pulsar mejor que la "A" para dar la orden de apertura de sus compuertas a nuestra nave? Creo que ninguna (es un decir...) pues aparte de ser la inicial de "abrir" nos va a servir para utilizar una nueva **función** de nuestro ordenador, INKEY$.

Mediante esta función podemos introducir en nuestro programa caracteres alfanuméricos, tanto para la confección de cadenas como para, mediante la instrucción IF, realizar una acción determinada dependiendo de la tecla que hayamos pulsado. Este es nuestro caso, así que vamos a escribir esta parte del programa para analizarla posteriormente.

```
10100 REM APERTURA
10110 A$=INKEY$:IF A$()"A" AND A$()"a" THEN GOTO 10110
10120 GOSUB 7040:FOR Y=79 TO 95:LINE (86,Y)-(168,Y),4:FOR D=
    1 TO 100:NEXT D:NEXT Y:GOSUB 7050
10150 REM BRAZO
10160 LINE (92,90)-(126,94),14,BF:CIRCLE(127,92),2,14:PAINT(
    128,92),14
10170 PUT SPRITE 0,(86,82),8,27
```

Como vemos en la línea 10100 a no ser que la tecla pulsada sea la "A" (mayúscula o minúscula) no pasaremos a la línea siguiente, una vez en esta línea (10110), accedemos a la subrutina 7040.

Si listamos nuestro programa veremos que entre 7030 y 7100 no hay ninguna línea, sin embargo el área indicada (7040) está situada en la subrutina SONIDO, y de esto se trata, vamos a poner sonido a la apertura de nuestra puerta, teclea lo siguiente.

```
7040  RESTORE 7140:GOTO 7100
7140  DATA 0,0,0,0,0,0,30,7,16,00,00,060,00,14
```

Una vez que tenemos otro ruido vamos a abrir las compuertas, para esto vamos a emplear un loop que nos varíe la coordenada "Y" (vertical) desde el punto 79 al 95, correspondientes a la parte superior e inferior de la compuerta, las coordenadas "X" (horizontales) son 86 y 168 en los extremos, con estos parámetros y la instrucción LINE trazaremos una serie de líneas paralelas que irán **borrando** nuestro dibujo en la parte central de la nave, este efecto de borrado se consigue gracias a que el color con el que son trazadas es el mismo que utilizamos para nuestro cielo (4), como verás este truco lo hemos empleado ya varias veces.

Inserto en el loop "Y" encontramos otro loop o bucle, el "D", este segundo loop lo único que hace es **retardar** el trazado entre líneas para dar mayor realismo, si no le pusiéramos las compuertas prácticamente **desaparecerían** de golpe. Al final de la línea tenemos un nuevo GOSUB, que como habrás supuesto por el número, también pertenece a SONIDO, con esta nueva llamada a la subrutina de SONIDO queremos crear el efecto del golpe que dan las compuertas al abrirse completamente, como el efecto anterior, tampoco está grabado, así que teclea de nuevo:

```
7050  RESTORE 7150:GOTO 7100
7150  DATA 0,0,0,0,0,0,25,7,00,00,16,0,80,00
```

Si quieres ver cómo se abren las compuertas, teclea las siguientes instrucciones:

```
SCREEN 2,2:COLOR 1,4:CLS:GOTO 8000
```

Habrás observado cómo al final desaparecía todo, esto es debido a que no pusimos ningún **bloqueo** al programa (10300 GOTO 10300) y éste se encuentra todavía **abierto.**

En la línea 10160 es donde realizamos el brazo **falso** tal y como está representado en la figura 70.

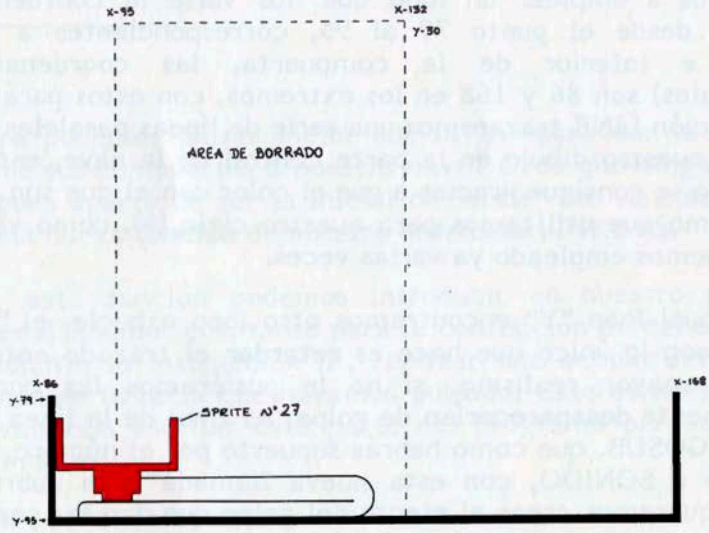


Fig. 70 Dibujo brazo articulado (falso)

En esta última figura podemos ver un dibujo en forma de horquilla correspondiente al sprite 27, éste es el mismo número que mencionamos en la línea 10170, si recuerdas (y si no, compruébalo) el último sprite que realizamos fue el número 23, correspondiente a la ventana de nuestra nave. Esto quiere decir que tenemos que crear más sprites.

Cuando leíamos la descripción de esta parte del programa veíamos la necesidad de adecuar la velocidad de nuestra nave, tanto en avance como en retroceso, esto lo consigue el COLUMBIA mediante el uso de cohetes auxiliares que lleva instalados en la cola y el morro, nosotros vamos a simular estos cohetes mediante el uso de sprites con forma de llamas, como ya hicimos con nuestra nave en el bloque DESPEGUE, estos sprites los tienes dibujados en la figura 71.

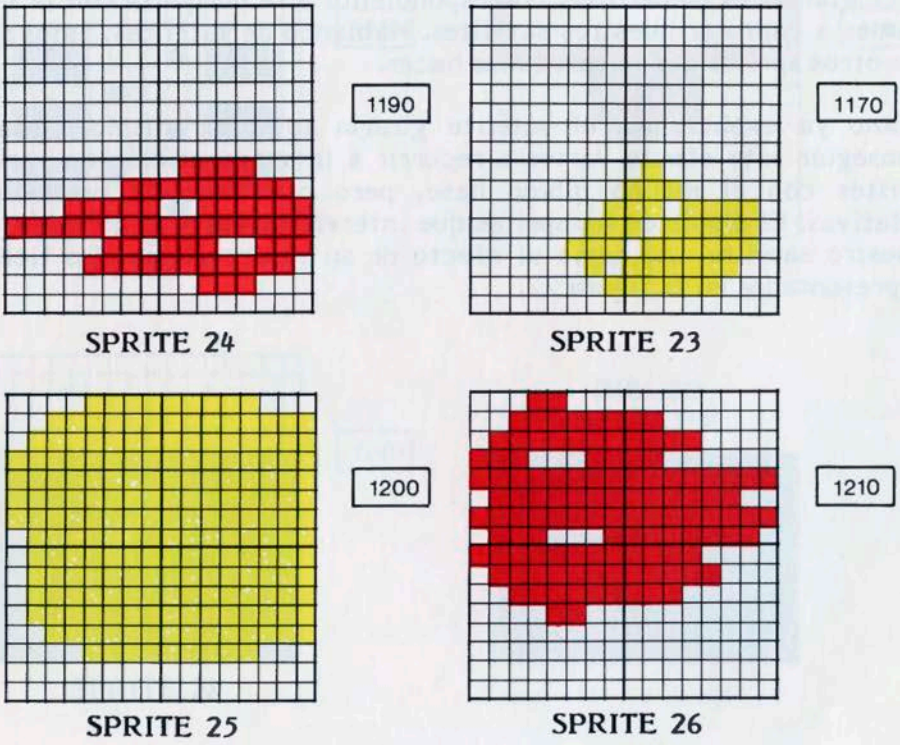


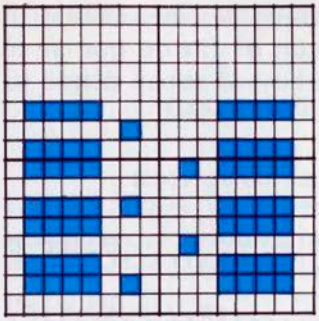
Fig. 71 Sprites llamas cohetes auxiliares

Nuevamente vamos a utilizar la subrutina 1000 (GRAFICOS M1/VENTANA) para la generación de estos nuevos sprites, ya que son también gráficos **móviles** la denominación sigue siendo válida (M1), escribe las siguientes líneas.

```
1180 REM LLAMAS
1190 DATA ,,,,,,,,,3,1F,FB,6,F,,,,,,,,,,,78,FE,3F,FF,E7,FE,3
     8,
1200 DATA F,3F,7F,FF,FF,FF,FF,7F,7F,7F,7F,7F,3F,F,,,F8,FF,FF
     ,FF,FF,FF,FF,FF,FF,FF,FF,FF,FF,F8,,
1210 DATA C,3F,7F,F1,FF,7F,FF,78,FF,7F,3F,C,,,,,,E0,F8,E0,FF
     ,FC,FF,FE,F0,F8,E0,,,,,
```

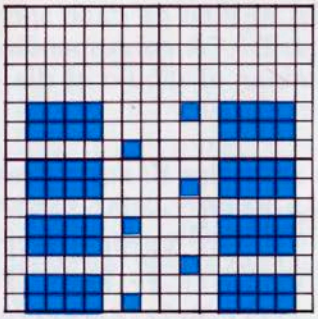
Con estas líneas hemos realizado hasta el sprite 26, pero nosotros necesitamos el número 27, correspondiente a la horquilla con la que vamos a capturar nuestros satélites. Hablando de satélites, éstos son los otros sprites que necesitamos hacer.

Como ya explicamos, el satélite giraría sobre sí mismo, y para conseguir este efecto vamos a recurrir a la superposición de varios sprites con el mismo dibujo base, pero con distintas posiciones relativas. El dibujo de los sprites que intervienen en la generación de nuestro satélite, así como el efecto de su superposición, los tienes representados en la figura 72.



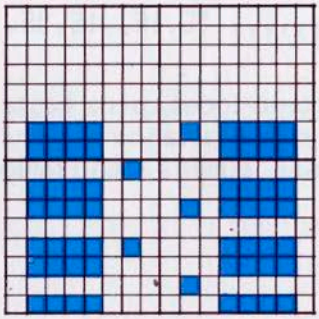
1250

SPRITE 29



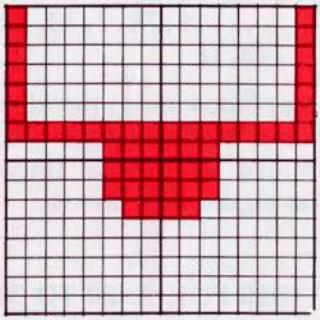
1260

SPRITE 30



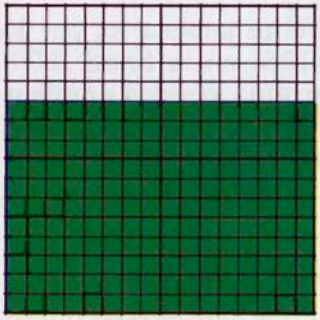
1270

SPRITE 31



1230

SPRITE 27



1240

SPRITE 28

DIBUJO SATELITE

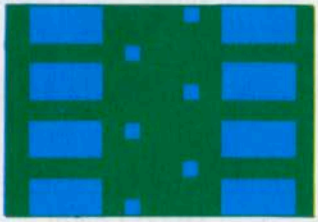


Fig. 72 Sprites y dibujo satélite artificial

Como verás, al lado de cada sprite está el número de línea donde se encuentra su DATA correspondiente, esto es con el fin de que si tú quieres cambiar el dibujo el día de mañana (u hoy...) sepas dónde se encuentra cada uno.

Tecleemos ahora las líneas correspondientes a estos sprites:

```
1220 REM SATELITE
1240 DATA ,,,,,FF,FF,FF,FF,FF,FF,FF,FF,FF,FF,FF,,,,,,FF,FF,F
     F,FF,FF,FF,FF,FF,FF,FF,FF,FF
1250 DATA ,,,,,78,2,78,78,,7A,78,,78,7A,,,,,,,1E,,1E,5E,,1E,
     1E,40,1E,1E,
1260 DATA ,,,,,78,78,2,78,78,,7A,78,,78,7A,,,,,,5E,1E,,1E,5E
     ,,1E,1E,40,1E,1E
1270 DATA ,,,,,,78,78,2,78,78,,7A,78,,78,,,,,,,5E,1E,,1E,5E,
     ,1E,1E,40,1E
```

Bueno, no hacemos más que dibujar sprites como locos, pero hasta ahora no hemos visto ninguno y sobre todo nos sigue faltando nuestro célebre número 27 de la línea 10170. Vamos pues a realizarlo mediante las siguientes líneas:

```
1020 FOR A=20 TO 31
1220 REM SATELITE
1230 DATA 80,80,80,80,80,80,FF,7,7,7,3,,,,,,1,1,1,1,1,1,FF,E
     0,E0,E0,C0,,,,,
```

Para poderlo visualizar es necesario primeramente generarlo, por lo tanto debemos acceder a la subrutina GRAFICOS, en la línea 1000. Con el fin de no tener que **rodar** todo el programa puedes emplear este conjunto de instrucciones, que para no tener que teclearlas cada vez que quieras verlo, las vamos a poner en una línea que no utilizaremos, por ejemplo, la 40000; escribe:

```
40000 CLEAR:SCREEN 2,2:COLOR 1,4:GOSUB 1000:
      GOTO 8000
```

Pongamos ahora un cierre a nuestro programa:

```
10200 GOTO 10200
```

"Arranca" ahora tecleando:

```
GOTO 40000
```

Una vez que ya tenemos nuestro brazo **falso**, vamos a realizar el **verdadero**, primeramente veámos el **flow chart** de la parte correspondiente al dibujo y movimiento del brazo, representada en la figura 73.

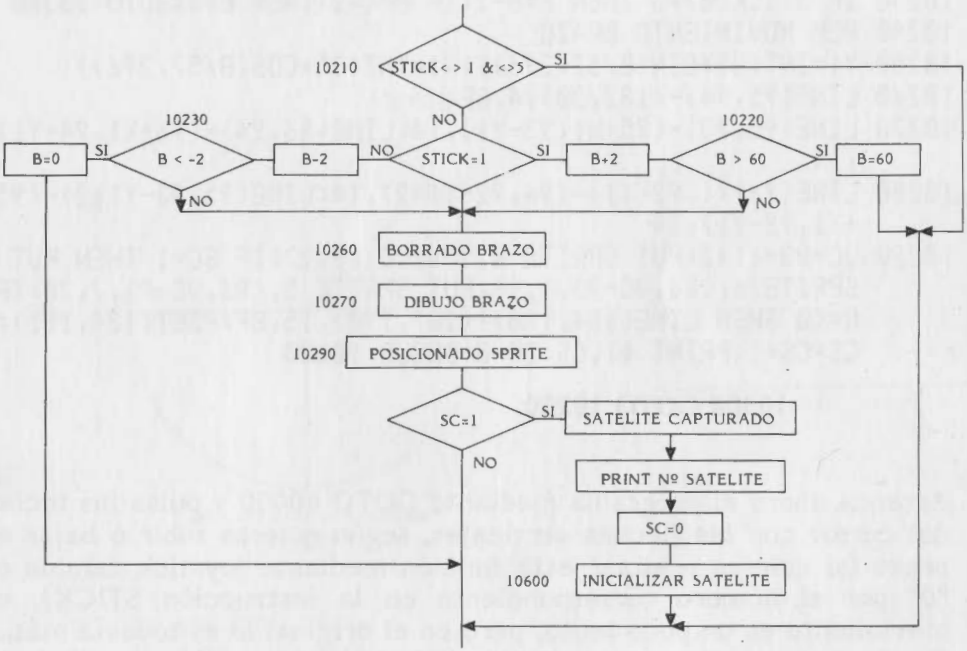


Fig. 73 Carta de flujo INPUT/MOVIMIENTO BRAZO

Como vemos es un programa sencillo, únicamente no sabemos qué significado tiene la variable "SC", esta variable está situada (como veremos posteriormente) en la subrutina COLISION, y toma el valor "1" cuando la **captura** ha sido correcta, permitiendo así que el satélite sea **cogido** por el brazo y el contador nos apunte un tanto más.

Tecleemos ahora esta parte:

```
10200 REM INPUT BRAZO
10210 IF STICK(0)<>1 AND STICK(0)<>5 THEN GOTO 10300
10220 IF STICK(0)=1 THEN B=B+2:IF B=>60 THEN B=60:GOTO 10300
10230 IF STICK(0)=5 THEN B=B-2:IF B=<-2 THEN B=0:GOTO 10300
10240 REM MOVIMIENTO BRAZO
10250 Y1=INT(35*SIN(B/57.296)):X1=INT(35*COS(B/57.296))
10260 LINE(95,94)-(132,30),4,BF
10270 LINE(95,93)-(95+X1,93-Y1),14:LINE(96,94)-(96+X1,94-Y1)
      ,14
10280 LINE(96+X1,92-Y1)-(96,92-Y1*2),14:LINE(95,93-Y1*2)-(95
      +X1,93-Y1),14
10290 VC=83-Y1*2:PUT SPRITE 0,(86,VC),8,27:IF SC=1 THEN PUT
      SPRITE 6,(86,VC-9),1,28:PUT SPRITE 5,(86,VC-9),7,30:IF
      B=<0 THEN LINE(184,180)-(207,190),15,BF:PSET(184,182):
      CS=CS+1:PRINT #1,CS:SC=0:GOSUB 10600
```

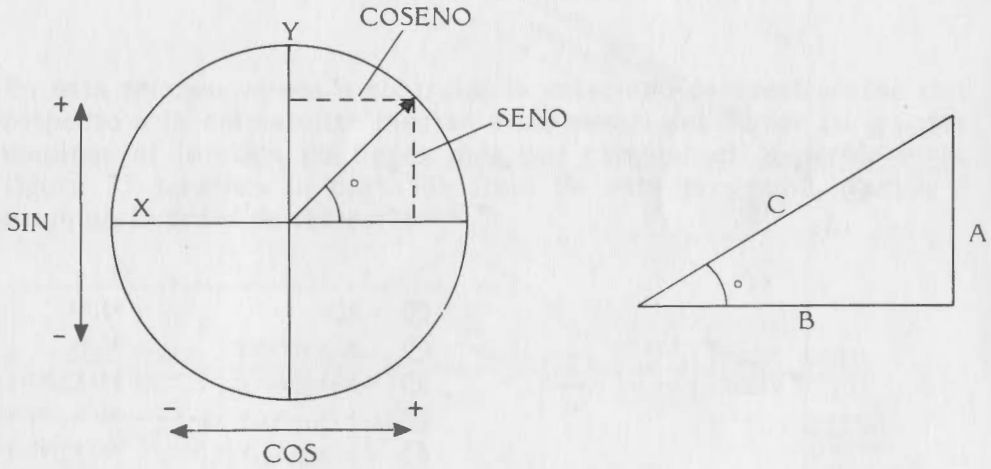
10300 GOTO 10200

Arranca ahora el programa mediante GOTO 40000 y pulsa las teclas del cursor con las flechas verticales, según quieras subir o bajar el brazo (si quieres realizar esta función mediante joystick cambia el "0" por el número correspondiente en la instrucción STICK), el movimiento es un poco lento, pero en el original lo es todavía más... De todas maneras es necesario darse cuenta de la cantidad de operaciones que tiene que realizar nuestro ordenador para mover el brazo (¡el de la nave, claro...!)

El borrado del brazo lo hacemos en la línea 10260 mediante el dibujo de un rectángulo que cubra todo el área donde se mueve, este área se **llena** (BF) con el mismo color que el cielo (4), en la figura 67 puedes ver el **área de borrado.**

SIN/COS

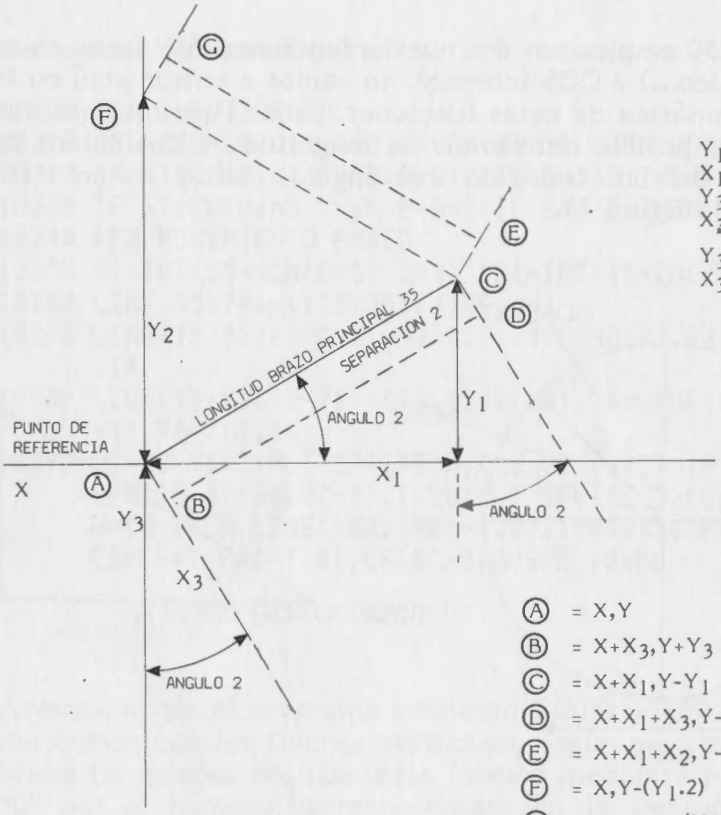
En las líneas 10250 empleamos dos nuevas **funciones,** SIN (seno en su sentido matemático...) y COS (coseno), no vamos a entrar aquí en la explicación matemática de estas funciones, pero sí que diremos que mediante ellas es posible determinar la magnitud de cualquiera de los tres lados de un triángulo rectángulo, tal y como está representado en la figura 74.



A = C° SIN (°/57.296)
B = C° COS (°/57.296)
C = A/SIN (°/57.296)
C = B/SIN (°/57.296)

Fig. 74 SENO y COSENO

Ahora podemos relacionar el movimiento de nuestro brazo con el empleo de estas funciones, pues conociendo la longitud de "C" y los grados (variable "B") podemos hallar la altura "A". La formulación empleada para el dibujo la tienes representada en la figura 75.



Y_1= 35 . SIN (°/57.296)
X_1= 35 . COS (°/57.296)
Y_3= 2 . COS (°/57.296)
X_3= 2 . SIN (°/57.296)

Y_3 = 1
X_3 = 1

Ⓐ = X,Y = 95,93
Ⓑ = X+X_3,Y+Y_3 = 96,94
Ⓒ = X+X_1,Y-Y_1 = 95+X_1,93-Y_1
Ⓓ = X+X_1+X_3,Y-Y_1+Y_3 = 96+X_1,94-Y_1
Ⓔ = X+X_1+X_2,Y-Y_1-Y_3 = 96+X_1,92-Y_1
Ⓕ = X,Y-(Y_1.2) = 95,93-(Y_1.2)
Ⓖ = X+X_3,Y-(Y_1.2)-Y_3 = 96,92-(Y_1.2)

Fig. 75 Dibujo brazo mediante funciones SIN/COS

El resultado de las ecuaciones "X3" e "Y3" han sido sustituidas por "1", ya que su diferencia en las distintas posiciones, con la escala que estamos empleando, no hacía práctica la utilización de estos resultados.

En la línea 10290 vemos que la variable "SC" (satélite capturado) toma el valor "0" cuando el brazo alcanza su posición más baja (B=0), después hay un acceso a la subrutina 10600, esta parte está todavía sin hacer, pero la labor que realiza es la de **reinicializar** la posición del satélite para que empecemos una nueva captura.

POSICION/VELOCIDAD

En esta sección vamos a controlar la velocidad de nuestra nave con respecto a la del satélite mediante las teclas del cursor (si quieres emplear el joystick no tienes más que cambiar el número), en la figura 73 tenemos la carta de flujo de este programa, vamos a examinarla antes de teclearlo:

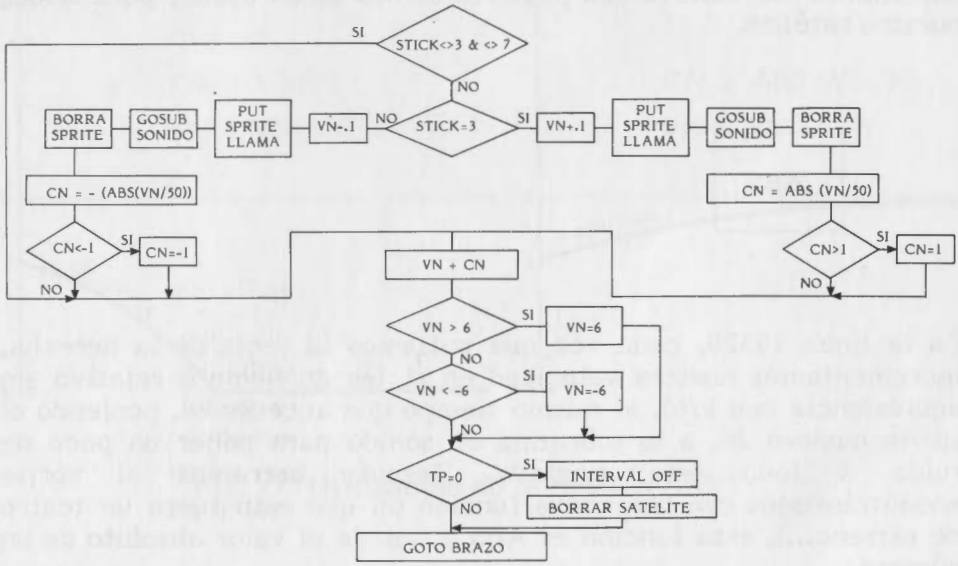


Fig. 76 Flow chart VELOCIDAD/POSICION

Como puedes ver, es un programa de lo más sencillo (sobre todo para nosotros que ya somos unos expertos...), en él se comprueban el cumplimiento de unas condiciones y se limitan unos valores, aprovechamos el accionamiento de las teclas del cursor para poner el sprite correspondiente a las llamas de los cohetes auxiliares, vamos a teclear ahora el programa para repasar sus líneas:

```
10300 REM INPUT VELOCIDAD
10310 IF STICK(0)<>3 AND STICK(0)<>7 THEN GOTO 10340
10320 IF STICK(0)=3 THEN VN=VN+.1:PUT SPRITE 1,(201,84),9,26
      :GOSUB 7050:PUT SPRITE 1,(0,209):CN=ABS(VN)/50:IF CN>1
       THEN CN=1 ELSE GOTO 10340
10330 IF STICK(0)=7 THEN VN=VN-.1:PUT SPRITE 1,(32,94),9,24:
      GOSUB 7050:PUT SPRITE 1,(0,209):CN=-(ABS(VN)/50):IF CN<
      -1 THEN CN=-1 ELSE GOTO 10340
10340 REM CALCULOS/POS. SATELITE
10350 VN=VN+CN:IF VN>6 THEN VN=6
10360 IF VN<-6 THEN VN=-6
10370 IF TP=<0 THEN TP=0:INTERVAL OFF:FOR D=1 TO 6:PUT SPRIT
      E D,(0,209):NEXT D
10380 GOTO 10200
```

La línea 10310 determina si las teclas pulsadas son las correctas, en caso contrario nos envía a la posición 10340. Aquí es donde calculamos los valores que posteriormente serán usados para situar nuestro satélite.

ABS

En la línea 10320, cada vez que pulsamos la tecla de la derecha, incrementamos nuestra velocidad en .1 (es un número relativo sin equivalencia con k/h), al mismo tiempo que accedemos, poniendo el sprite número 26, a la subrutina de sonido para poner un poco de ruido a todo este tinglado. Después **borramos** el sprite encontrándonos con una nueva **función** (ni que esto fuera un teatro de estreno...), esta función es ABS y nos da el valor **absoluto** de un número.

Ya explicamos que la nave puede ir más deprisa o más despacio que el satélite, dado que las velocidades son **relativas** entre ambos. Cuando estas dos velocidades sean iguales su diferencia será cero, cuando la nave vaya más deprisa su velocidad será positiva, mientras que cuando vaya más despacio será negativa (con relación al satélite).

Hasta aquí todo más o menos claro (¡menos...!), pero sabemos que cuando se mueve un objeto hay algo que le llama inercia, que impide tanto el inicio de un movimiento como la parada de éste.

Esta inercia es la que tratamos de simular con la variable "CN", esta variable toma el valor de 1/50 de la velocidad de la nave (VN), este valor debe ser positivo cuando aceleramos, línea 10320, y negativo cuando deceleramos, línea 10330.

Dado que los valores de "VN" pueden ser tanto positivos como negativos en cualquiera de las dos líneas, nos podríamos encontrar con el efecto contrario del buscado. Examinemos la figura 74 donde se ha representado gráficamente una velocidad negativa y lo que sucedería en la línea 10350 con y sin el uso de la función ABS.

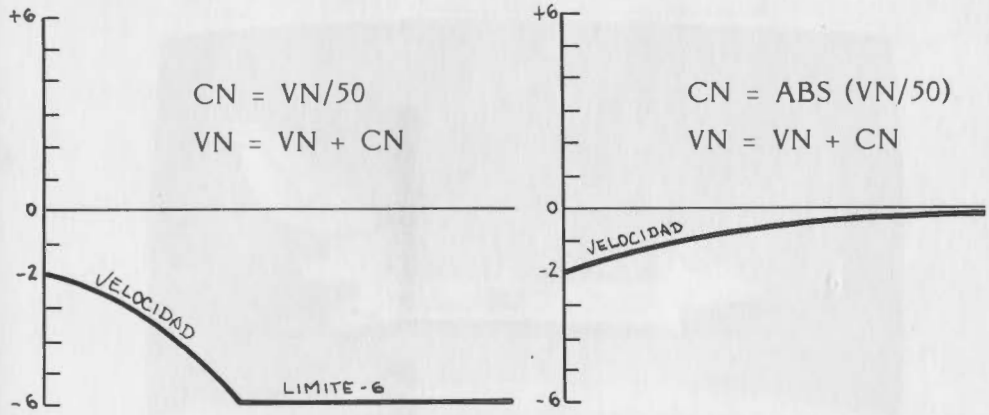


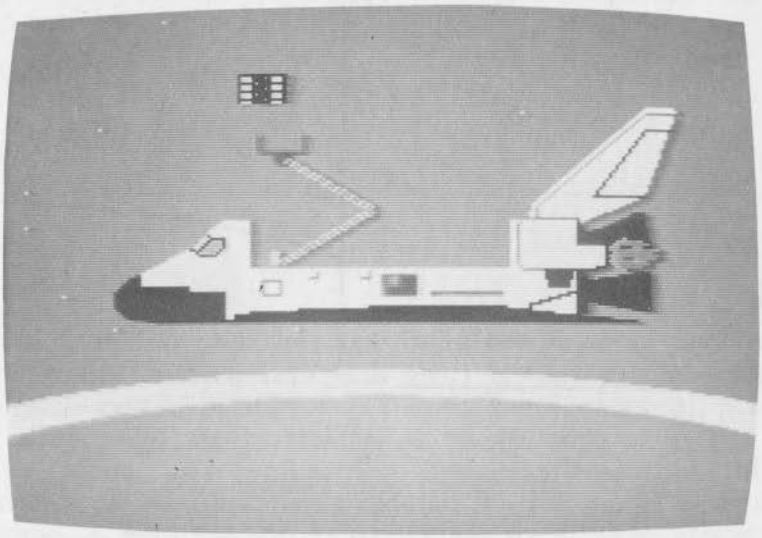
Fig. 77 Utilización de la función ABS

En esta figura están representadas dos líneas que parten de la misma velocidad negativa (-2), ahora pulsamos una vez para acelerar, y vemos que en el dibujo de la izquierda (sin ABS) en lugar de aumentar, disminuye la velocidad, mientras que en el de la derecha la velocidad aumenta pero no sobrepasa la línea de "0" (partimos de la base que únicamente has dado una pulsación).

Este último caso es lógico, puesto que al ser "CN" igual a VN/50, a medida que disminuye "VN" también lo hace "CN", y en el caso de repetir indefinidamente esta operación el valor máximo que obtendríamos sería "0", ya que al llegar "VN" a este valor (0) "CN" sería también "0" (0/50=0), y de ahí no nos moveríamos a no ser que pulsáramos de nuevo una tecla, tanto para acelerar como para reducir la velocidad.

Lo más importante es que veas cómo un impulso de aceleración puede transformarse en frenado sin el uso de ABS.

¡Bueno, yo creo que está bien de disertaciones seudomatemáticas! Es hora de que salvemos nuestro programa para pasar a ver nuestro satélite, sigue el proceso habitual y graba el programa con el título "BASE16".



Movimiento del Satélite

Esta es la subrutina que es accedida de forma cíclica por nuestro programa cada 1/4 de segundo mediante la instrucción ON INTERVAL GOSUB, con esta subrutina haremos dar vueltas a nuestro satélite y cambiaremos su posición, dependiendo de la velocidad de la nave (VN). También es aquí donde comprobaremos si el satélite ha sido capturado o si chocó con el brazo, siguiendo la práctica establecida, vamos a ver en la figura 78 su carta de flujo.

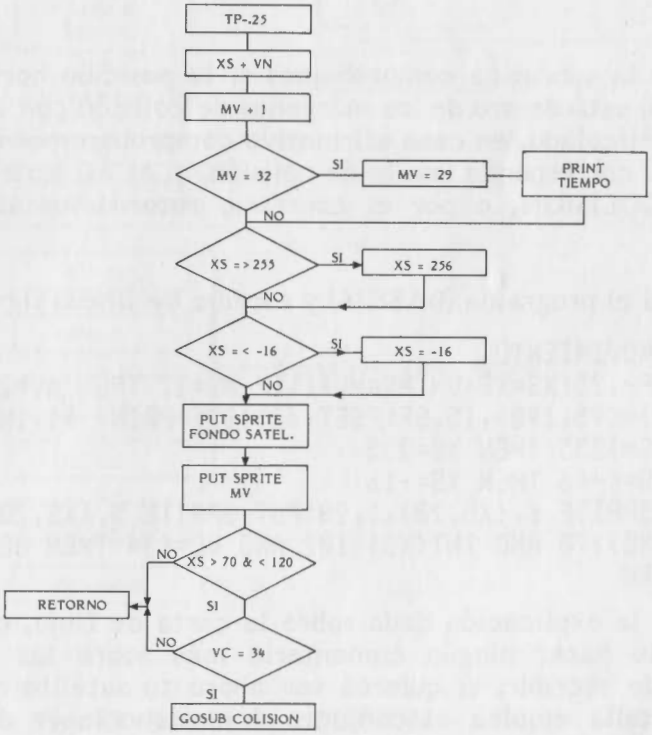


Fig. 78 Carta de flujo subrutina MOVIMIENTO

La subrutina se inicia restando .25 al tiempo total (TP), después sumamos a "XS" el valor de la velocidad relativa (VN), "XS" es la variable que utilizaremos para posicionar horizontalmente nuestro satélite (la posición vertical es fija), la variable "MV" (movimiento) la usaremos para hacerle **girar.**

Esta variable (MV) tenía un valor de 28 al iniciarse este programa (línea 10030), por lo que al sumarle "1" obtenemos 29, este es el valor del primer sprite móvil que **superponemos** al que representa el fondo del satélite. Cuando la subrutina es accedida de nuevo, "MV" se incrementa adquiriendo el valor 30, siendo entonces el sprite de este número el que se superpone, así hasta llegar al número 32, en cuyo momento el valor se reduce a 29, comenzando el ciclo nuevamente.

Dada la velocidad (1/4") con que se producen estos cambios, y la disposición de los dibujos, el efecto que obtenemos es de un giro constante.

Al final de la subrutina comprobamos si la posición horizontal del satélite (XS) está dentro de los márgenes de colisión con la horquilla del brazo articulado, en caso afirmativo comprobaremos si la altura de éste (VC) corresponde con la de colisión, si es así accedemos a la subrutina COLISION, o por el contrario retornamos al programa principal.

Carga ahora el programa (BASE16) y escribe las líneas siguientes:

```
10450 REM MOVIMIENTO
10460 TP=TP-.25:XS=XS+VN:MV=MV+1:IF MV=32 THEN MV=29:LINE(64
      ,180)-(95,190),15,BF:PSET(62,182):PRINT #1,INT(TP)
10470 IF XS=>255 THEN XS=255
10480 IF XS=<-16 THEN XS=-16
10490 PUT SPRITE 6,(XS,20),1,28:PUT SPRITE 5,(XS,20),7,MV:IF
      INT(XS))70 AND INT(XS)<102 AND VC=<34 THEN GOSUB 10550
10500 RETURN
```

Después de la explicación dada sobre la carta de flujo, creo que no es necesario hacer ningún comentario más sobre las líneas que acabamos de escribir, si quieres ver ahora tu satélite moviéndose por la pantalla emplea el conjunto de instrucciones de la línea 40000.

```
GOTO 40000
```

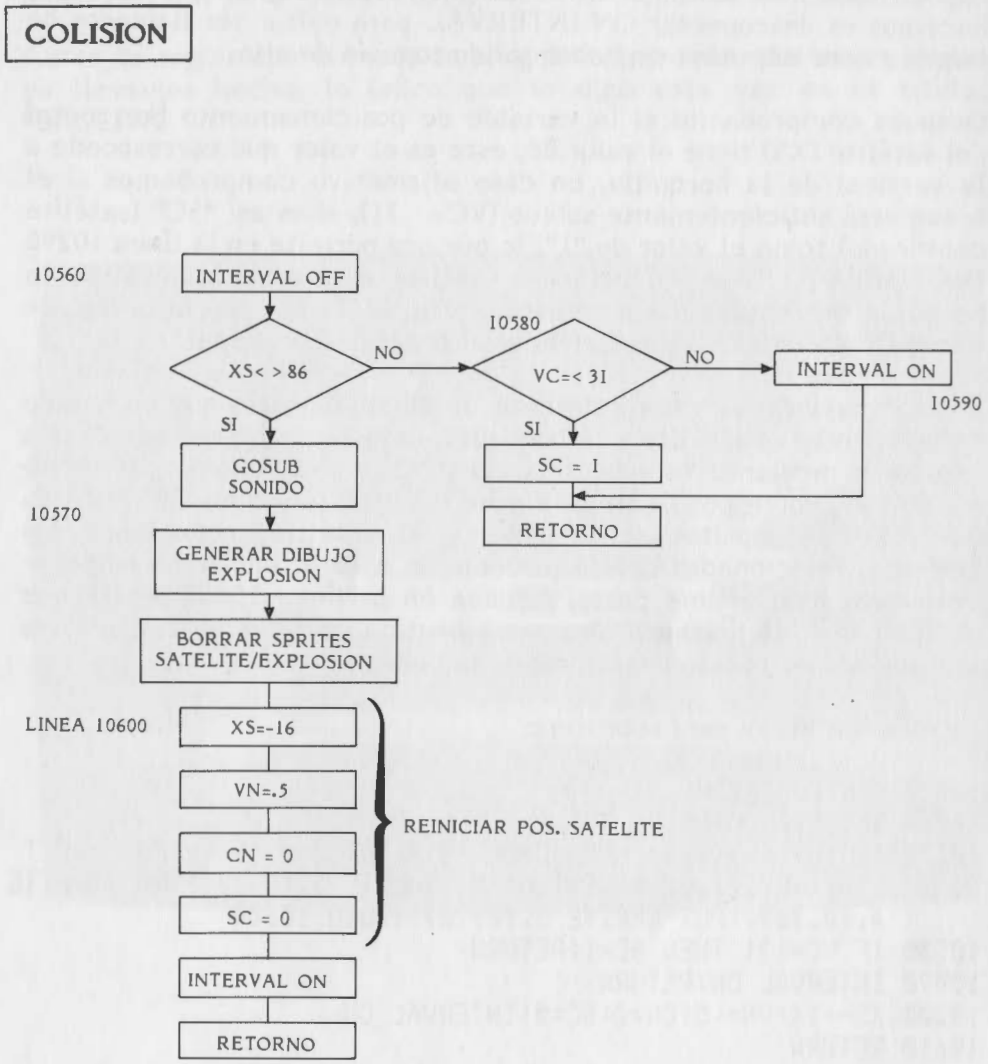


Fig. 79 Subrutina COLISION

A esta subrutina hemos accedido después de cumplir las condiciones especificadas en la línea 10490 de MOVIMIENTO; lo primero que hacemos es desconectar ON INTERVAL para evitar ser **llamados** de nuevo a esta subrutina sin haber salido todavía de ella.

Después comprobamos si la variable de posicionamiento horizontal del satélite (XS) tiene el valor 86, este es el valor que corresponde a la vertical de la horquilla, en caso afirmativo comprobamos si el brazo está suficientemente subido (VC= 31), si es así "SC" (satélite capturado) toma el valor de "1", lo que nos permite en la línea 10290 (MOVIMIENTO BRAZO) bajar el satélite a nuestra bodega. Si la horquilla no está suficientemente alta (VC 31), conectamos de nuevo ON INTERVAL y retornamos al programa principal.

Si "XS" tuviera un valor diferente de 86 significaría que ha habido colisión entre el satélite y la horquilla, en este caso producimos una explosión mediante la subrutina SONIDO y posicionamos el sprite número 20, correspondiente a la nube que utilizamos en DESPEGUE. Borramos los sprites de la nube y el satélite, **reiniciando** las variables relacionadas con la posición de este último para empezar de nuevo, esta última parte, situada en la línea 10600, es la que utilizamos en la línea 10290 como subrutina de reinicialización, una vez que hemos puesto el satélite en la bodega.

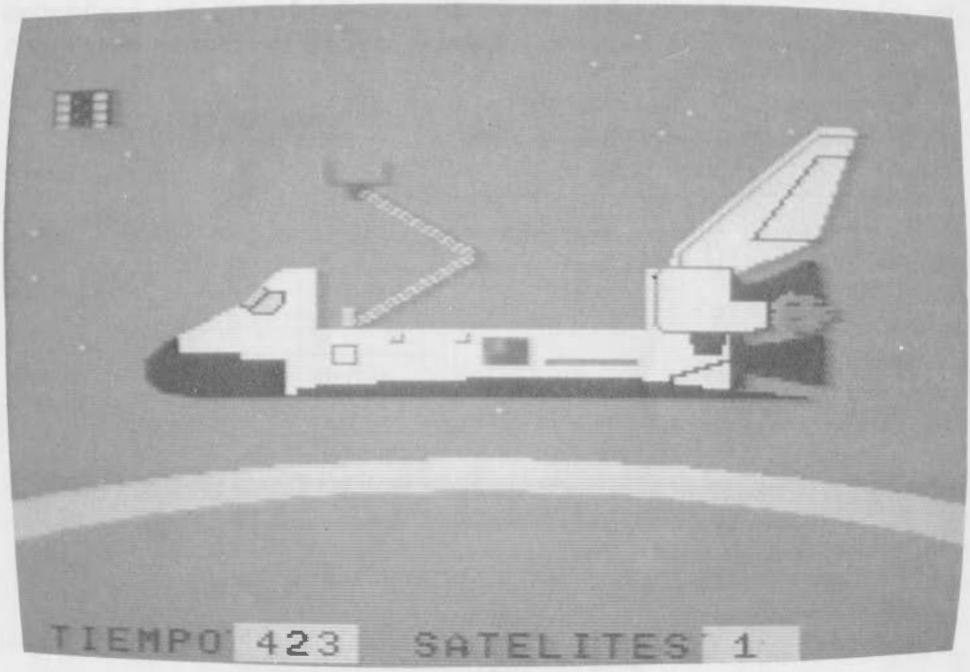
Escribamos ahora esta subrutina:

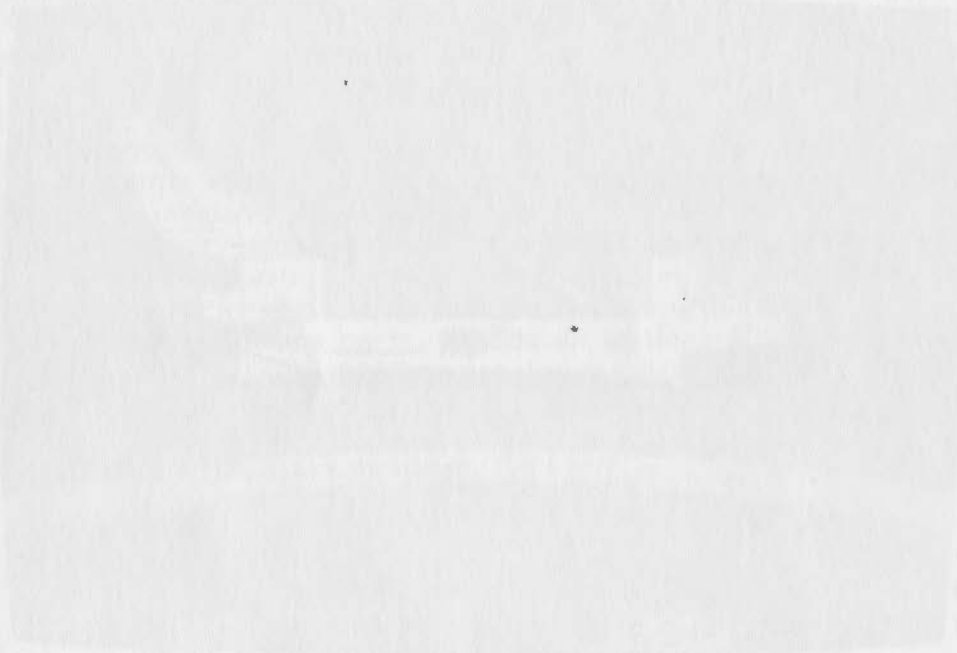
```
10550 REM COLISION
10560 INTERVAL OFF
10570 IF INT(XS)<>86 THEN GOSUB 7010:FOR D=6 TO 11:PUT SPRIT
      E 3,(XS,20),D,20:NEXT D:PUT SPRITE 5,(0,209):PUT SPRITE
      6,(0,209):PUT SPRITE 3,(0,209):GOTO 10600
10580 IF VC=<31 THEN SC=1:RETURN
10590 INTERVAL ON:RETURN
10600 XS=-16:VN=.5:CN=0:SC=0:INTERVAL ON
10610 RETURN
```

Lo único destacable que no ha sido explicado de todas estas líneas es el loop "D" en 10570, mediante este loop o bucle hacemos cambiar de color al sprite de la nube (20), los colores van desde el número 6, rojo oscuro, hasta el número 11, amarillo claro.

Mediante este cambio de colores conseguimos un efecto de parpadeo, que junto con el ruido, hacen la explosión más real.

Antes de seguir con el resto de programa conviene que salves lo que ya llevamos hecho, lo único que te digo esta vez es el título, "BASE17".





Cierre de la Puerta

Este bloque será accedido cada vez que pulsemos la tecla F1, el que se ejecute la acción que tiene programada, cerrar las compuertas, dependerá únicamente de que el brazo esté en su posición inferior, veámos el flow chart de esta parte en la figura 80.

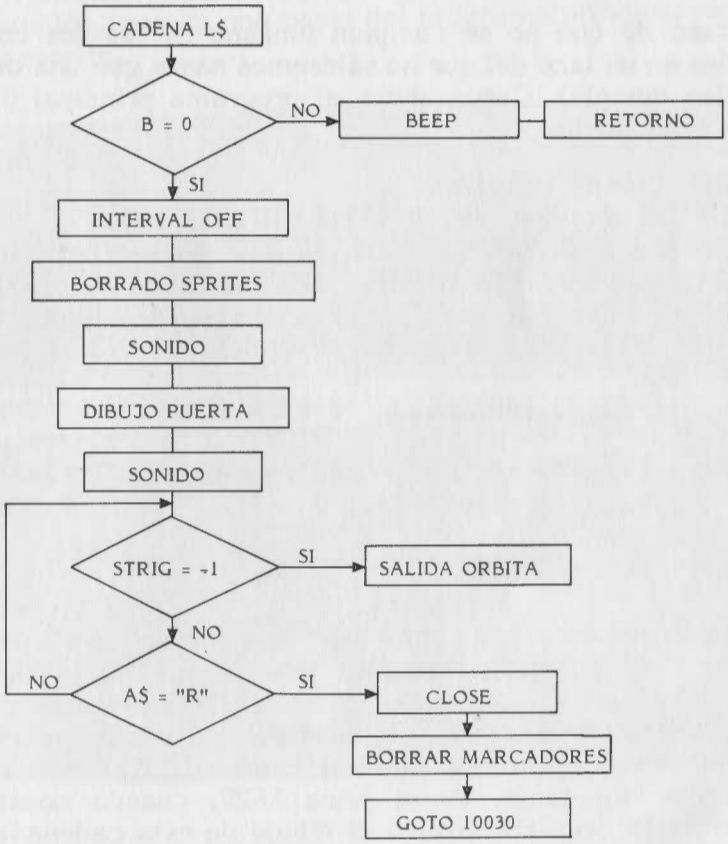


Fig. 80 Carta de flujo CIERRE PUERTA

Como vemos, este programa es pan comido (que es lo que se dice para levantar el ánimo...), si el brazo no está abajo (B<>0) obtendremos un **beep** de aviso retornando al programa principal, en caso contrario desconectamos ON INTERVAL, accediendo a la subrutina de sonido y dibujando la puerta, el sistema empleado es el mismo que usamos para borrarla, un bucle que va dibujando líneas, únicamente que en este caso el dibujo se hace de abajo hacia arriba.

Después nos encontramos con dos condiciones "IF", si se cumple la primera (STRIG=-1) vamos al programa SALIDA ORBITA, si no es éste el caso, y es la segunda la que se cumple (A$="R") iniciamos de nuevo el programa de captura.

En el caso de que no se cumplan ninguna de las dos condiciones, quedamos en un lazo del que no saldremos hasta que una de ellas sea cierta (se cumpla). Carga ahora el programa principal (BASE17) y escribe estas líneas:

```
10650 REM CIERRE PUERTA
10660 L$="D16R21U16D16R21U16D16R21U16D16R21U16"
10670 IF B=0 THEN GOTO 10680 ELSE BEEP:RETURN
10680 INTERVAL OFF:FOR D=0 TO 6:PUT SPRITE D,(0,209):NEXT D
10690 GOSUB 7040:FOR Y=95 TO 79 STEP-1:LINE (86,Y)-(168,Y),15
      :PSET(85,79),4:DRAW L$:FOR D=1 TO 50:NEXT D:NEXT Y:GOSU
      B 7050
10700 CLOSE:GOSUB 9400:GOSUB 7030:IF STRIG(0)=-1 THEN GOTO
      10750
10710 A$=INKEY$:IF A$="R" OR A$="r" THEN CLOSE:LINE(64,180)-
      (95,190),5,BF:LINE(184,180)-(207,190),5,BF:GOTO 10030
10720 GOTO 10700
10750 REM SALIDA ORBITA
```

Observarás que en el listado no hay nada que no haya sido explicado sobre el flow chart, únicamente señalar que en la línea 10690, dentro del loop "Y" con el que trazamos las líneas de color blanco que van **cerrando** nuestras compuertas, hay una instrucción DRAW que dibuja después de cada línea la cadena "L$". Esta cadena es la misma que empleamos en la línea 8620, cuando construimos la segunda parte del COLUMBIA, el dibujo de esta cadena **no se ve** en la pantalla por ser del mismo color (4), pero al trazar las líneas blancas, éstas son **borradas** en la parte en que coinciden con la cadena, apareciendo de esta forma el dibujo original.

Salva ahora el programa con el nombre que sea más de tu agrado, pues éste es el último capítulo de la primera parte, en la segunda parte (ATERRIZA CON TU MSX) continuaremos con el programa SALIDA ORBITA y ATERRIZAJE, que como ya explicamos es un simulador de vuelo del COLUMBIA.

Desearía haber cumplido la finalidad que me propuse al escribir este libro, y que hayas aprendido, de una forma más amena que leyéndote el "Manual de Referencia", algunas de las particularidades del sistema MSX. Los programas que hemos construido no son piezas inamovibles y puedes hacer cambios en ellos, para eso cuentas en este libro con documentación más que suficiente, ten únicamente una precaución, guarda una copia del programa principal tal y como aquí está escrito.

Glosario de Comandos

DESPEGA CON TU MSX

***DESPEGA CON TU MSX** es una forma práctica y amena de introducción al MSX basic, mediante un programa que simula las distintas fases del viaje de la nave espacial COLUMBIA. El posicionamiento para el despegue, la recuperación de satélites artificiales en órbita o cualquiera de las fases intermedias como el desacople de los cohetes impulsores, se convierten en el marco ideal para analizar los secretos de la programación en este potente lenguaje.*

En DESPEGA CON TU MSX, Medel consigue crear un programa estructurado en bloques didácticos, de forma que facilita su comprensión, al mismo tiempo que los resultados se pueden ir visualizando desde el primer momento sin teclear el listado completo.

***DESPEGA CON TU MSX** viene completado por un segundo libro del mismo autor que transformará la cabina de mando del Columbia en un simulador de alto nivel.*