Autores

Ing. Eduardo de ANTUENO
Prof. Ignacio LUPPI

APRENDIENDO CON LOGO GRAFICO®

Ideas y Animaciones

- Edición Preliminar para Internet-

Acompañan a este libro material didáctico y programas

Edita y distribuye FUNDAUSTRAL
Gavilanes 2116 – 1416 - Buenos Aires - Argentina
Tel (01) 4581-2768 –
info@fundaustral.com.ar
http://www.fundaustral.com.ar
igluppi@igluppiweb.com.ar
http://www.igluppiweb.com.ar

Impreso en Argentina
Printed and published in Argentina
Imprimié et publié en Argentine
Stampato e publicato nell' Argentina
Impresso e editado na Argentina

Segunda Edición en Febrero de 1997
ISBN

Contenido

INTRODUCCION

Breve síntesis acerca de LOGO

El Lenguaje LOGO fue desarrollado en el Laboratorio de Inteligencia Artificial del Instituto Tecnológico de Massachusetts (MIT) en EEUU por Seymour Papert y fue difundido inicialmente en ese país, Inglaterra y Francia.

Aunque LOGO fue pensado para usarse en educación, es de hacer notar que es un lenguaje derivado del LISP, que es usado para inteligencia artificial. El mismo lenguaje LOGO tiene versiones específicas que se usan para ese fin y derivados de éste que se usan para programar robots en fábricas. Asimismo un módulo lunar fue programado con una variante del lenguaje LOGO.

Son poco conocidas por el público estas aplicaciones como también que la versión de LOGO Gráfico desarrollada en Argentina se usa en la Comisión de Energía Atómica para diseño de carga del reactor nuclear de Atucha.

En la Univ. de California (Berkeley - EEUU) el famoso matemático Brian Harvey, dedico tres voluminosos tomos a la "Ciencia de la Computación al estilo Logo" en el que trata temas como sistemas de inferencia, rompecabezas lógicos, backtracking y otros temas vinculados a la Inteligencia Artificial.(1)

LOGO no es solamente un lenguaje de computación o un software muy difundido en educación, sino que es también un conjunto de materiales disponibles para someter a prueba al usuario, ya sea niño, adolescente o adulto, sobre la coherencia - matemáticamente diríamos "consistencia" - de sus ideas al tratar de ordenar algo a la computadora.

Como afirma el Dr. Seymour Papert en su libro "Desafío a la Mente"(2), LOGO representa una modalidad en el uso de las computadoras y un instrumento para la exploración de ideas. LOGO se apoya en una teoría de conocimiento basada en los estudios de Piaget y explicitada por Papert (MIT).

El aspecto social del aprendizaje con LOGO, ha sido analizado por Gérard Bossuet, precursor en la introducción de la modalidad LOGO en Francia, el cual estudio el comportamiento de un grupo de trabajo colectivo compartiendo una computadora. Bossuet - de la Univ. de París VI - diferencia su experiencia en Montpélier (Francia) con la idea de la "computadora-pelota-para-compartir" que se contrapone a la experiencia norteamericana con la "computadora-lápiz-de-uso-individual" . Para este tema recomendamos leer La Computadora en la Escuela (3).

El uso frecuente de LOGO pone en evidencia el proceso intelectual realizado por quien lo utiliza evidenciando sus errores y por ello facilitando la autocorrección de fallas de razonamiento lógico.

En la terminología computacional se llama programación al proceso relativo a la comunicación con computadoras que produce un comportamiento particular. Para indicarle qué debe hacer la computadora debemos darle un conjunto de órdenes respetando determinada sintaxis.

El lenguaje LOGO está constituído por conjuntos de palabras llamadas "primitivas" a través de las cuales damos órdenes.

Al programar en LOGO se crean "palabras nuevas" llamadas procedimientos, con los cuales se manejan los materiales (recursos) incluidos en este lenguaje. Una vez creadas, estas nuevas palabras pasan a engrosar el vocabulario con el que nos comunicamos con la computadora y servirán para nombrar "procesos" creados por el usuario, el que las utilizará para graficar, resolver problemas, crear micromundos o juegos, etc. Esta es una característica de LOGO que se denomina extensibilidad del lenguaje.

¿Qué podemos hacer con LOGO?

Los materiales de que disponemos con LOGO dependen de la versión utilizada. Elegiremos para enumerarlos la versión de Logo Gráfico:

a- Un elemento graficador (tortuga) que construye dibujos desde una visión intrínseca, es decir con desplazamientos y giros.

La tortuga, matemáticamente hablando, es un vector, pues en todo momento tiene ubicación, dirección y sentido.

Al comandar la tortuga tratando de dibujar algo el operador debe descentrarse, ubicándose en el lugar de ésta, por lo que las líneas se construyen a partir de referencias concretas de su propio cuerpo. Es la llamada Geometría intrínseca.

La geometría que se utiliza en las gráficas cartesianas es, por el contrario, referida a ejes, ubicándose el observador separado del elemento graficador.

b- 20 actores o formas móviles (*) que permiten estudiar movimientos, simular animaciones para representar cualquier tipo de fenómeno o microcosmos material en la pantalla.

(*) hasta 60 en la versión más moderna.

c- Imágenes fijas de cualquier tamaño (llamadas bitmaps) que es posible visualizar en cualquier lugar de la pantalla.

d- Sonidos grabados (digitalizados) previamente por medio de un micrófono o disponibles de una colección.

e- Porciones de sonidos, músicas y videos realizados por otros programas y ejecutados desde LOGO.

f- Dispositivos externos conectados a la computadora como interruptores, motores, sensores físicos, ópticos o térmicos, etc.

g- Lugar donde almacenar la información a procesar. La forma en que LOGO guarda la información - y luego dispone de ella - es a través de objetos llamados "palabras" y conjuntos llamados "listas".

El manejo de este tema es el que posibilita el real dominio del lenguaje. Actualmente el porcentaje de docentes que conoce este tema es muy bajo, lo que hace que sólo "arañen" la superficie del LOGO.

h- LOGO tiene alrededor de 30 primitivas (funciones) para cálculos matemáticos de todo tipo y 27 otras para trabajar con palabras y listas.

i - 40 Botones de distintos tipos que permiten activar procedimientos y 10 controles que pueden controlar datos numéricos en otras tantas variables.

j - Entornos físicos con velocidad, fricción, aceleración, gravitación, y otros conceptos que permiten introducirse en el mundo de las simulaciones. Esta es una característica casi propia de Logo Gráfico.

¿Para qué nos sirve LOGO?

Para ayudarnos a aprender conceptos lógicos y matemáticos tan importantes como toda la aritmética y geometría curricular (la lógica no suele ocupar un lugar importante hasta el ciclo universitario).

Puede hacerse un paralelismo entre el aprendizaje de la matemática escolar "tradicional" y la que se aprende experimentando con LOGO comparándola con el conocimiento que se obtendría de un lugar geográfico mirándolo a través de fotos o videos, versus el conocimiento logrado "in situ" desplazándose en el propio terreno.

Papert lo menciona en su libro cuando habla de la diferencia entre aprender matemáticas y construir las "matemáticas". Es decir entre estudiar matemáticas y ser matemático.

El aprendizaje con Logo es activo, exploratorio y vivencial.

Al observar programas de los estudiantes hechos con LOGO podemos encontrar formas de comprender cómo han trabajado sus mentes en el proceso del desarrollo de los programas, además del nivel de conocimiento, las estrategias y el estilo que usaron para resolver problemas.

Actualmente, los productos informáticos que más se usan en los colegios - sobre todo de nivel medio - son los programas utilitarios. Ello se debe a la presión de los medios y del mercado de consumo.

Muchos de los adultos que apoyan esta postura creen que los alumnos adolescentes poseen la misma dificultad que ellos en manejar un procesador de texto. Los adultos que han crecido en una cultura precomputacional tienen miedos que les dificultan su relación con las máquinas, y por lo tanto traban el aprendizaje de esta herramienta.

Es un hecho que un adolescente de inteligencia media, con ganas de aprender un programa utilitario (o por necesidad laboral) puede hacerlo en una semana.

Hemos notado que muchas veces el uso de estos programas en el medio educativo sólo tiende a una mejor presentación o acabado de un trabajo y con menor esfuerzo.

La forma de usar la computadora con utilitarios o con LOGO no se contrapone, se complementa. Ya que en el primer caso la máquina se usa para hacer algunas operaciones o procesos más fácilmente[4] y en el segundo se usa para entender como funcionan los procesos.[5]

¿Por qué programar?

Cualquier actividad intelectual puede verse como una ventana hacia la mente. Sin embargo, hay aspectos importantes de la programación con los que no contábamos antes que nos permiten penetrar nuestra mente. La discusión sobre "por qué programar" es un tema actual de debate.

En el trabajo ¿"Está obsoleta la programación?" (Clements y Meredith, 1993) se resume una reunión de la Asociación Estadounidense para la Investigación Educativa en la cual algunos importantes investigadores opinan sobre el tema.

La programación es "el mejor apoyo con que contamos para representar las actividades cognitivas", según Andrea diSessa (MIT).

Un importante investigador de la Universidad de Campinas[6] dice "la ventana hacia la mente surge cuando se ve el proceso de programación como un ciclo que consiste en describir-ejecutar-reflexionar-corregir-describir".

Nosotros preferimos la analogía de la computadora como un "espejo de la mente", pues la tarea de programación es un reflejo del pensamiento (Computación en el Colegio- Antueno, Naveira y Thompson -[7]

Citando al experto brasileño José Valente diremos que: "Cuando un estudiante usa Logo sus ideas iniciales sobre cómo resolver el problema las transmite a la computadora. Así, el estudiante actúa sobre el objeto-computador".

Se puede decir que es un microcosmos obediente que facilita el proceso de descripción de ideas.

El LOGO no enseña un tema determinado: Es una herramienta para investigar...

La computadora es la primera herramienta que hace pasar la acción del nivel material al lógico.

Valor pedagógico de LOGO

El LOGO es un material formativo, como lo son los rompecabezas, el meccano o los juegos de ingenio.

Visto desde este punto de vista no puede decirse de LOGO que sea anacrónico ni pasado de moda, de la misma manera que nadie diría que los rompecabezas o los materiales constructivos son anacrónicos. Igual que en un "meccano" lo importante es el armado del mismo y no el producto final.

Los conceptos pedagógicos incluidos en LOGO permiten:

- Plantearse y resolver problemas.
    a. Proponiendo soluciones.
    b. Descomponiéndolo en partes simples (análisis)
    c. Construir la solución total a partir de pequeñas soluciones (síntesis).
- Desarrollar la capacidad de describir.
- Explorar y descubrir.
- Valorar los componentes del lenguaje.
- Desarrollar la capacidad de abstraer.
- Encarar actividades que ayudan a la construcción del espacio.

Citando a los profesores Hoyles y Noss, del Instituto de Educación de la Univ. de Londres "LOGO ofrece a los niños una forma de "hacer" matemáticas o de usar ideas matemáticas como un preludio para entender. LOGO ofrece la posibilidad de desarrollar control sobre su propio aprendizaje, donde ellos mejor que sus maestros, se preguntan y se contestan en su propio estilo"[8].

LOGO como lenguaje computacional

La parte más divulgada del lenguaje LOGO es su facilidad para graficar: la famosa "tortuga".

Pero tal vez esa facilidad para lograr algunos dibujos, han sido también propicios para crearle una imagen de lenguaje sólo para los más chicos.

Se confunde simpleza en el manejo y aprendizaje con falta de potencia de programación. Han contribuido a ellos varios factores:

1. La capacitación superficial que han tenido muchos docentes, ya que por distintos motivos sólo han hecho cursos cortos y por distintos motivos no han investigado lo suficiente.
2. La falta de material en castellano.
3. El desconocimiento del tema por muchos profesionales de la informática, educados con lenguajes como el COBOL, FORTRAN, BASIC, etc.
4. El desconocimiento del tema de muchos comunicadores sociales y de la prensa en general.

LOGO tiene estructuras de control muy simples. La modularidad y la recursión son los elementos que lo hacen muy potente.

Un procedimiento LOGO que utiliza varios subprocedimientos con varios niveles de anidamiento y de forma recursiva es capaz de expresar en pocas líneas ideas tan complejas que a lectores no acostumbrados a los lenguajes de inteligencia artificial, les costará comprender. Un programa como el citado equivale a cientos de instrucciones de otros lenguajes.(9)

Esta versión de LOGO

Esta versión ha sido desarrollada con el objetivo de reforzar el "principio de poder" del que habla Papert en su libro Desafío a la Mente, ya que desde la creación del LOGO hasta ahora, el entorno computacional que rodea a los niños se ha modificado en gran medida, por lo que un niño de los años '90 no experimentaría la misma sensación de poder (y de poder hacer) al experimentar con LOGO que hace 20 años.

A fines de los '90 el micromundo de la tortuga debe dejar paso a un micromundo animado más sintónico con el mundo actual, es decir, creemos que debe poner el acento en la noción de movimiento. O sea que reemplazamos un paradigma basado en la geometría por otro basado en la Física.

Es por ello resulta razonable, trabajar con idénticos objetivos constructivistas, pero utilizando entre otras herramientas la animación simultánea y los eventos. La programación por eventos permite desarrollar estrategias de anticipación en niños y adolescentes(10)

Un evento puede ser introducido por el usuario -por ej. tocando una tecla- o ser una consecuencia de la evolución del movimiento de un actor -por ej. una colisión- u ocurrir por el mero paso del tiempo (ver §7.5.1.)

La programación orientada hacia la acción responde bien a la cultura actual en la que son comunes los video-juegos y la animación generada por computadora, moneda corriente en TV y cine.

Se propone también esta herramienta como masilla modelable por docentes o profesionales para crear micromundos donde mediante la simplificación de leyes que rijan ese universo sea posible explorar determinados comportamientos.

De hecho LOGO GRAFICO incluye un micromundo físico con primitivas utilizables en cinemática y dinámica.

LOGO en el mundo...

En estos últimos años han aparecido nuevos desarrollos de versiones de LOGO en América y Europa.

Importantes grupos de investigación se ocupan del tema en Rusia, Inglaterra, Bulgaria, Hungría, Eslovaquia, Grecia, Portugal, etc. Particularmente podemos citar el proyecto de la Unión Europea desarrollando un Entorno Multimedia de Autor para Niños llamado MATCh

(Multimedia Authoring environmenT for CHildren) en el que intervienen 7 países que esta basado en LOGO.[10a]

- En Inglaterra se acaba de editar el Super-Logo desarrollado en la Universidad de Comenius (Eslovaquia), en EEUU han aparecido varias versiones nuevas: Micromundos, el UCB Logo, el Berkeley MSW Logo, etc.

- En la Facultad de Matemáticas de la Univ. de Sofía, en Bulgaria B. Sendov y s. Yalamova desarrollaron un entorno matemático llamado GEOMLAND que es un programa desarrollado en LOGO, en la que pueden estudiarse propiedades de diferentes modelos de construcciones geométricas complejas.[11]

- En Grecia, en la Escuela de Filosofía de la Universidad de Atenas, se han desarrollado micromudos de LOGO - en el marco de proyecto YDEES sostenido por la Comunidad Europea - para manejar tablas (tipo planilla de cálculo) , micromundos geográficos,

- En América del Sur está aumentando mucho su uso en Brasil donde compiten TrendLogo (versión en portugués de Logo Gráfico) y MegaLogo (versión en portugués de Comenius Logo) y en Costa Rica donde el programa de Informática Educativa está en etapa de expansión utilizando la versión de LOGO llamada Micromundos.

- En Melbourne (Australia) en la Univ. La Trobe, los profesores estudian LOGO como herramienta matemática y para aumentar el enfoque de enseñanza-aprendizaje colaborativo[12].

- En Moscú (Rusia), en el Instituto de Nuevas Tecnologías de Educación se ha utilizado desde hace varios años con alumnos de 14 a 16 años para simulaciones de física (campos electrostáticos, cinemática), en álgebra vectorial, para el estudio de fractales, etc.

- En Hungría desde 1995 con el nuevo curriculum se recomienda el uso de Logo, y el gobierno compró una licencia a la Univ. de Comenius, para uso de todas las escuelas húngaras.

- En Israel en la Univ. de Bar-Ilan el Prof. Uzi Armon del Depto. de Matemáticas y Ciencias de la Computación trabaja en algoritmos de ecuaciones intrínsecas e integración gráfica con Logo [13].

(1) Editado por MIT Press - 1987- Massachusetts - EEUU
(2) Desafio a la Mente - Seymour Papert - Ediciones Galápago - Bs. As.
(3) La Computadora en la Escuela - Gérard Bossuet - Ed. Paidós - Bs. As.
(4) La computadora como herramienta.
(5) La computadora como simuladora.
(6) Logo como ventana hacia la mente - Jose A. Valente - Univ. de Campinas - Brasil
(7) Apéndice 1 - Computación en el Colegio: Logo: espejo de la mente - Antueno E. -Naveira A. y Thompson H. - Edit. Informática Educativa (Fundaustral) 1983
(8) Computadoras y aprendizaje matemático - Contribución de Celia Hoyles y Richard Noss - Informatics in the Secondary School - Today and Tomorrow (pág. 109) - Edited by Sensova, Azalov y Muirhead - UNESCO.
(9) Tomado de "De la tortuga a la inteligencia artificial" - Luis Rodríguez Roselló - Vector Ediciones - Barcelona - España
(10) Una nueva versión Latinoamericana de Logo - E. Antueno - I. Luppi - C. Grant -Memorias de la 6ta Conferencia Europea sobre Logo (ver 11)
(10a) Memorias de la 6ta Conferencia Europea sobre Logo - Learning and exploring with Logo - Pág. 80 y subsiguientes. - Budapest - Hungría - Recopiladora: M. Turcsanyi-Szabó
(11) Informatics in the Secondary School - Today and Tomorrow (pág. 99) - Edited by Sensova, Azalov y Muirhead - UNESCO.
(12) Increasing Confidence in Mathematics Ability throught Logo and Collaborative Learning - Dr, A. Jones - email: T.Jones@latrobe.edu.au
(13) Journal of Math. Educ. in Sciences and Technology 22(4) pp. 577-584, 1991.

CAPITULO 1

PROGRAMEMOS NUESTROS PROPIOS DIBUJOS ANIMADOS

§ 1.1 LA ANIMACION

¿Qué es la animación? Si recurrimos al diccionario veremos que es "la acción y efecto de dar vida a algo".

Un ejemplo de animación lo tenemos en los dibujos animados que se logran a partir de un dibujo y de pequeñas variaciones del mismo que al verlas en rápida sucesión dan la ilusión de movimiento y de "tener vida".

La sensación de movimiento del cine profesional se logra con la proyección de 24 fotogramas por cada segundo y mediante el fenómeno de persistencia de las imágenes en la retina del ojo humano.

De los distintos recursos que tiene el lenguaje LOGO, gráficos, animación, uso de texto y sonido, siempre se ha destacado la primera, o sea la capacidad de dibujar de la tortuga. Este hecho se ve reflejado en todos los libros extranjeros y en los pocos nacionales que se han editado en los últimos años.

La explicación de esta falencia en el material procedente del extranjero, probablemente se deba a lo que podríamos llamar una determinada "cultura LOGO", que pese a haberse iniciado en EE.UU. en el Instituto Tecnológico de Massachusetts con la excelente versión para Texas Instruments (1979), la cual poseía muchas posibilidades de animación (32 formas móviles), quedó trunca al imponerse en las escuelas norteamericanas las computadoras Apple II y su versión de LOGO de Terrapin (1981) que sólo tenía una tortuga en forma de triángulo y no tenía animación hasta bastantes años después, con la aparición del "Sprite LOGO" (1) que contaba con 30 formas móviles.

En cambio, en nuestro país, la mayoría de las versiones LOGO que se han difundido tienen recursos de animación como puede verse en el cuadro de más abajo.

Históricamente el TI LOGO y el MSX LOGO lograron la mejor animación para computadoras de tipo hogareño (home computers). Luego con la llegada de las PC la animación perdió posibilidades ya que las primeras tenían un chip dedicado al movimiento y las PC no. Para PCs no hubo versiones de LOGO con velocidad y eventos (2) hasta que apareció el LOGO GRAFICO, primero para DOS y luego para Windows, que tiene muy potenciados estos recursos.

(1) Sprite: en inglés significa duende y se refiere a pequeñas porciones de pantalla que es posible definir en forma y color, para luego moverlas. También se las denomina tortugas múltiples, actores o agentes móviles, y su existencia es la que posibilita el recurso de animación.

(2) La versión Logo Writer © utiliza la palabra evento para denominar a lo que suele llamarse macro en computación. es decir una combinación de teclas que desencadena una acción o procedimiento. No deben confundirse con los eventos tratados en este capítulo.

Versión	*Equipo*	*Actores * Y tamaño*	*Velocidad*	*Eventos*	*Intercambio de bitmaps*	*Colores en cada Forma*
LOGO GRAFICO 4	PC	20 de 125x125	SI	13	I-E PCX-BMP	256
LOGO GRAFICO 2	PC	8 de 99x77	SI	13	I-E PCX	16
MICROMUNDOS	PC	16 de 40x40 **	NO***	-	I-E PCX	256
TI-LOGO II	TI-99	32 de 16x16	NO	NO	NO	1
WIN LOGO	PC	32 de 16x16	SI	NO	I PCX	1
LOGO WRITER	PC	4 de 16x16	NO	NO	NO	1
DREAN LOGO	Com64	8 de 16x16	NO	NO	NO	1
HAL LOGO	Com64	8 de 16x16	NO	NO	NO	1
COASIN LOGO	Com64	8 de 16x16	NO	NO	NO	1
EPI- LOGO	Spectrum	sin actores	-	-	NO	-
IBM PC LOGO	PC	sin actores	-	-	-	-

* Número de formas móviles (actores) y tamaño máximo dado en pixels.
** Ampliándola hasta 4 veces las originales.
*** Pero se puede "construir".
I-E Importa y exporta bitmaps.
I - Sólo importa bitmaps

El cuadro de más arriba ha sido ordenado de acuerdo a la capacidad de animación de cada una, la que - a nuestro criterio - está determinada por el número y tamaño de agentes móviles, la posibilidad de darles velocidad, la programación de eventos(3), la posibilidad de intercambiar bimaps(4) que sirven de escenografía y la facilidad de acceso a esos recursos.

(3) Eventos: son condiciones que se indican quedando latentes hasta que se cumplen y entonces desencadenan una acción programada. Usada en los videojuegos.

Papert en su famoso libro Desafío a la Mente los denomina CUANDO DUENDE (Cap. 4, pág. 129 en la versión castellana), aunque aún no había versiones de Logo con eventos.

(4) Decorados: son pantallas realizadas con un graficador, scaneadas o digitalizadas.

§1.2 RECURSOS DE ANIMACION VERSIONES LOGO WRITER Y LOGO GRAFICO

§1.2.1 VERSION LOGO GRAFICO

En la versión LOGO GRAFICO se cuenta con 60 actores(*) que pueden dibujar o disfrazarse con 180 formas(*) (figuras) distintas a elección, con 16 o 256 colores cada una.

Inicialmente sólo la tortuga es visible. Los actores se diferencian de la tortuga en que pueden tomar formas ya diseñadas o creadas por el usuario.

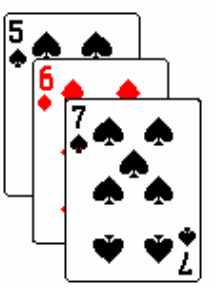
Los actores están numerados del 1 al 60 y se activan -o se los llama- con las primitivas ***ACTIVAR*** o ***DECIR*** seguida del número de actor que queremos activar. Todas las primitivas pueden escribirse acentuadas o no, ó indistintamente con letras minúsculas o mayúsculas. Por ej.:

```
ACTIVAR 3
```

De ahora en adelante todas las órdenes que demos serán recibidas y ejecutadas por ese actor, hasta que activemos otro.

Los actores se ubican en distintos planos, por lo que los que poseen números más altos al superponerse con otros ocultan a los que poseen un número más bajo y son ocultados por los que tienen números más altos (al revés de Logo Writer). En la figura de la derecha se ven con 3 naipes el ordenamiento para los actores de Logo Gráfico.

Para disfrazar un actor con una forma prefijada, debemos utilizar la orden ***FFORMA*** que significa, y debe leerse como, "Fijar FORMA".



(*) Varían según la versión de Logo Gráfico

Por ejemplo:

```
FFORMA 15
```

Debemos elegir un número de forma para poder ver un actor. Si a un actor no le decimos con qué forma debe disfrazarse lo hará con en número de forma por defecto (si es el 1, con la forma 1, así hasta el 3, el resto tomará 1 la forma 1) si no hubiera formas cargadas en memoria dará un mensaje de error.

Al ingresar al Logo se cargan automáticamente las formas tomadas del archivo NORMAL (puede ser .FRM o .FRC, según la cantidad de colores con que esté configurados el Windows); que tiene también grillas libres para dibujar nuevas formas de distintos tamaños. El diseño de nuevas formas es muy fácil, ya que se realiza con el Mouse, eligiendo previamente los colores para dibujar.

NOTA PARA EL DOCENTE: Es aconsejable que los alumnos trabajen primero con las figuras existentes, las coloren y las muevan, antes de ponerse a diseñar nuevas formas.

§1.2.2 VERSION LOGO WRITER©

En la versión LOGO-WRITER© para PC compatibles, se cuenta con 4 tortugas que pueden disfrazarse con 90 formas (figuras) distintas.

Inicialmente sólo la número 0 se encuentra visible, mientras que las otras 3 están escondidas (invisibles) y en los vértices imaginarios de un cuadrado de 40 pasos de lado.

Las tortugas están numeradas del 0 al 3 y para dirigirnos a una de ellas lo hacemos con la orden DILE seguida del número deseado, es decir que si queremos "decirle" algo a la tortuga 3, escribimos (con letras minúsculas o mayúsculas):

```
DILE 3
```

De ahora en adelante todas las órdenes que demos serán "escuchadas" y ejecutadas por esa tortuga, hasta que le hablemos a otra tortuga (con una nueva orden DILE).

Para disfrazar una tortuga con una forma, debemos elegir un número del 1 al 90. Las primeras de ellas vienen dibujadas, y el resto están libres para ser diseñadas por nosotros.

Las tortugas se ubican en distintos planos, por lo que las de números más bajas tapan a las que poseen un número más alto. Esto podemos explicarlo como si estuvieran ordenadas como un mazo de naipes. La número 0 tapa a la 1, la 1 tapa a la 2 y la 2 a la 3.

Para disfrazar una tortuga con una forma prefijada, debemos utilizar la orden "ffig" que debe leerse como, "fijar figura". Debemos elegir un número del 0 al 90, por ejemplo:

```
FFIG 26
```

disfraza a la tortuga activa con forma de camión, ya que la figura 26 corresponde a ese disfraz.

§1.3 PROGRAMA DE PRESENTACION

Siempre que queramos hacer aparecer un actor disfrazado en un lugar debemos hacerlo en un procedimiento, que cumpla exclusivamente con ese fin. Esto se logra "hablándole" al actor (LOGO GRAFICO) o a la tortuga deseada (orden 1), ordenándole que se disfrace con la figura elegida (orden 2), que se coloque en un lugar fijado por sus coordenadas cartesianas (orden 4) y que finalmente aparezca (orden 6). La orden 5 es optativa y estipula el color con que aparece (no existe en vers. LOGO GRAFICO).

PARA LOGO GRAFICO

```
para camion
activar 1              (1)
fforma  6              (2)
sp                     (3)
fx (-70) fy 30         (4)
visible                (6)
fin
```

4 Puede reemplazarse por FPOS [-70 30]
6 Puede usarse también MT

NOTA: A veces suele ocurrir que si la forma es muy grande y la elección del lugar está próximo al borde aparece el mensaje de FIGURA FUERA DE LA PANTALLA,. Esto ocurre ya que el punto de referencia es el centro de la forma, y si los bordes de la misma sobrepasan los extremos de la pantalla se produce este mensaje. Esto no ocurrirá si estamos en el modo ***LIBRE*** en vez de ***JAULA*** (que es el modo por omisión de los actores).
En modo ***JAULA*** los actores no pueden salir de la pantalla, en modo ***LIBRE*** los acotres salen de la pantalla, ya que el entorno o mundo de la tortuga es mayor que en tamaño del monitor (en pasos de tortuga). Otra forma de trabajo es poner al Logo Gráfico en modo ***VUELTA,*** este modo de trabajo es similar al LogoWriter, ya que el actor sale por un lado de la pantalla y retorna por el otro. (Para más información remítase a la Ayuda de Logo Gráfico).
En muchos caso puede convenir - para evitar el inconveniente mencionado - ubicar al actor antes de fijarle la forma, es decir intercambiar las órdenes (2) por la (4). También es importante que el tamaño de la grilla no sea excesivamente grande con respecto al tamaño real del actor dibujado. En las versiones 4.4 en adelante se puede usar la opción de menú - RECORTAR para eliminar las partes de la cuadrícula que sobren.

PARA LOGO WRITER:

```
para camion
dile 1             (1)
ffig 26            (2)
sp                 (3)
fx -70 fy 30       (4)
fcolor 2           (5)
mt                 (6)
fin
```

(1) se dirige a la tortuga 1
(2) se "disfraza" con la figura que corresponda.
(3) permite ubicar inicialmente a una tortuga en cualquier lugar de la pantalla sin dejar rastro.
(4) este par de órdenes la ubican en el lugar elegido al indicar sus coordenadas, y puede reemplazarse por FPOS [-70 30]
(5) le fija el color 2.
(6) permite verla, ya que todas salvo la número 0 están invisibles.

DESAFIO

¿Te animas a colocar 4 actores disfrazados a tu gusto en distintos lugares de la pantalla? Deberán utilizarse 4 programas individuales.

Nota: El maestro puede sugerir otros lugares distintos de los elegidos para observar si los alumnos comprenden el sentido de los signos de las "x" e "y", y para entrenarse en la estimación de las medidas de la pantalla. Se supone que este tema ya lo han trabajado con la geometría de la tortuga. Puede marcar las pantallas de los monitores con una fibra para pizarra blanca haciendo 4 ó 5 cruces para que los alumnos coloquen allí los actores.

DESAFIO

¿Podrías ubicar la Luna en la parte superior de la pantalla y un gato maullando en la inferior. Sugerencia: Dibujar con la tortuga el piso o un contorno simple de edificios. Usar los actores 1 y 2 para la Luna y el gato.

§1.4 ACTIVIDADES POSIBLES

Es conveniente iniciarse en la tarea de animación manejando un solo actor, para luego, al dominar las técnicas necesarias de desplazamiento y de movimiento en su lugar (cambio de forma) podamos hacerlo con varios de ellos.

Podemos ensayar de hacer una enumeración de actividades que es posible encarar, aunque no tratemos todas éstas en este capítulo.

A) Trabajando con un actor solo.

a1- Desplazarlo en línea recta (§1.6).
a2- Cambiarle de disfraz sin desplazarse (§1.7 y §1.12).
a3- Cambiarle de disfraz y desplazarlo en línea recta (§1.8).
a4- Desplazarlo con movimientos no rectilíneos(§1.9, §1.10 y §1.11).
a5- Cambiarle de disfraz y moverlo no rectilíneamente (§1.13).

B) Trabajando con más de un actor:

b1- 2 o más actores con desplazamientos simultáneos(§1.14).
b2- Personaje formado por dos actores unidos

Este artificio no resulta necesario en la versión LOGO GRAFICO ya que las formas son mucho más grandes, pero en otras versiones la única manera de obtener formas grandes y de más de un color en la misma forma es utilizar varios actores, uno para cada parte del cuerpo. Por ejemplo, un actor para la cabeza, otro para el tronco y brazos, y otro para las piernas.

- Presentar un personaje formado por 2 partes (2 actores).
- Desplazar el personaje.
- Cambiar de disfraz a una de sus partes sin desplazamiento.
- Cambiar de disfraz a una de sus partes con desplazamiento.
- Cambiar de disfraz a sus 2 partes sin desplazamiento.
- Cambiar de disfraz a sus 2 partes con desplazamiento.

C) Otras situaciones:

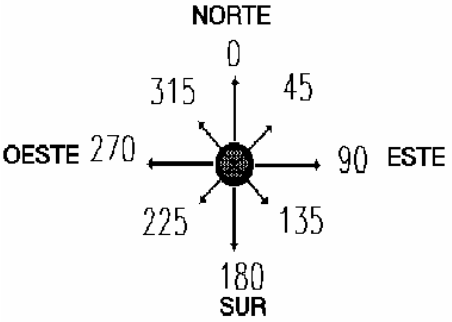
- Dividir una forma en varias.
- Ocultar progresivamente a una forma.

§ 1.5 UNA PRIMITIVA PARA FIJAR EL RUMBO

Los rumbos se definen como los puntos cardinales, y se estipula que el que mira hacia el Norte (hacia arriba de la pantalla) es el rumbo 0 (cero). A partir de esa dirección los otros serán los que se aprecia en el dibujo.

A diferencia de los giros a derecha e izquierda, al fijar un rumbo no necesitamos tener en cuenta hacia adonde está apuntando originalmente la tortuga sino sólo hacia dónde queremos que apunte. Por eso se dice que la primitiva ***FRUMBO*** es absoluta mientras que DERECHA e IZQUIERDA son relativas.

Es de hacer notar que cuando un actor disfrazado cambia de orientación (rumbo) su forma permanece inalterable y sin cambio, ya que el rumbo es inherente al actor pero no al disfraz. En las versiones 4.3 en adelante de LOGO GRAFICO es posible asignar varias formas a un actor con lo que al cambiar el rumbo éste podría parecer que rota (ver manual).



§ 1.6 MOVIENDO EL CAMION

Si queremos que el actor (o la tortuga en LogoWriter) disfrazado se mueva desde un lugar a otro prefijado -por ejemplo distante 200 pasos- podemos hacerlo de dos maneras: La primera, que llamaremos "paso a paso" consiste en moverlo de a un paso repetidas veces, para lo que hacemos el siguiente procedimiento:

VERSION LOGO GRAFICO

```
para camion
activar 1                  (7)
fforma 2                   (8)
frumbo 90                  (9)
repetir 20 [adelante 1]    (10)
fin
```

(7) o su sinónimo DECIR 1, activa al actor 1.
(8) Puedes buscar una forma de actor que se pueda utilizar para el ejemplo, si no la hubiera dibújala. Puedes encontrar camiones en el archivo MOVILES.FRM.
(8) Fijamos el rumbo del actor
(9) Admite REPITE como variante.

VERSION LOGO WRITER©

```
para andar
dile 1
frumbo 90
repite 200 [adelante 1]
fin
```

DESAFIO

¿Por qué no lo movemos con adelante 200? ¿Cuál es la diferencia?

EXPLICACION

(7) si trabajamos con un solo actor, esta orden no sería necesaria, pero conviene acostumbrarse a hacerlo pues al usar varios actores debemos activar al que queremos decirle algo.

(8) La apuntamos hacia la derecha de la pantalla. Ver nota al pie (**)

(9) El movimiento de 200 pasos aparece como algo continuo.

DESAFIO

¿Cómo lo moverás más rápido la misma distancia? ¿Y más despacio? Ayuda: puede intercalarse una orden de **ESPERAR 5**.

§ 1.7 HACIENDO EJERCICIOS

Si un actor alterna dos disfraces que representen posiciones distintas de un mismo personaje se producirá una sensación de movimiento similar al que se logra al proyectar las imágenes de una película (ver § 1.1)

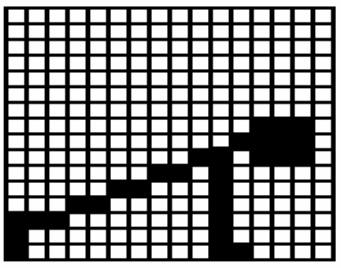
Podemos ensayar este efecto - por ejemplo - con un monigote que hace flexiones de brazos.

Disfracemos, por ejemplo, al actor 2 de gimnasta en posición de hacer flexiones en LOGO GRAFICO.

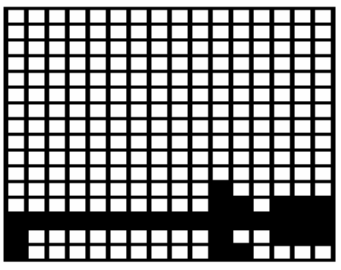
```
para gimnasta
activar 2
fforma 10
sp
centro
mt
fin
```

Y para que cambie de disfraz cada medio segundo:

```
para flexión      (10)
activar 2
fforma 11
esperar 50        (11)
fforma 10
esperar 50
fin
```



FORMA 10



FORMA 11

Y para lograr, por ejemplo, 20 flexiones:

```
para flexiones
repetir 20 [flexión]
fin
```

(10) Cuidado con los acentos pues, no es lo mismo el procedimiento "flexión" que "flexion".
(11) EN LOGO GRAFICO, como ya dijimos, 100 unidades de tiempo equivalen a 1 segundo, por lo que "espera 50" corresponde a medio segundo.

Estas pausas luego de cada cambio de disfraz son necesarias siempre para darle el ritmo de movimiento adecuado.

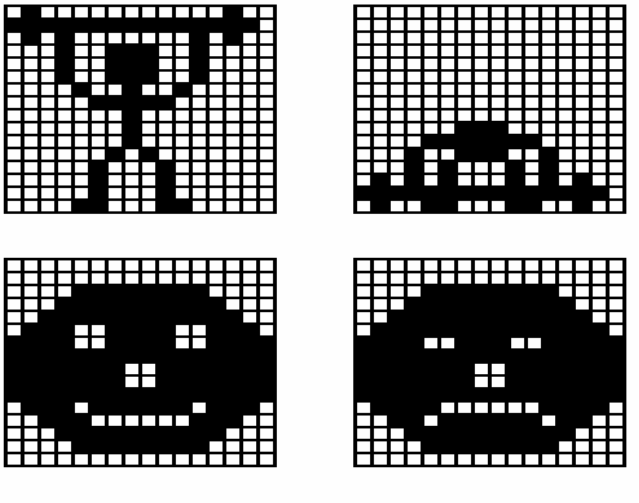
A continuación y a modo de ejemplo se detallan distintas situaciones con sus pausas aproximadas, calculadas por alumnos.

SITUACION	*PAUSA*
HELICE DE HELICOPTERO GIRANDO	5 a 10
HOMBRE CORRIENDO	15
HOMBRE CAMINANDO	15 a 30
RANA SALTANDO	45
FLEXIONES	50
ESQUIADOR COMPITIENDO	60
ESQUIADOR PASEANDO	85

PROPUESTA

Dibuja algunas formas o utiliza las que están en las formas provistas y desarrolla procedimientos:

a) Haciendo gimnasia
b) Levantando pesas
c) Cara triste y sonriente.
d) PacMan gesticulando
e) Guiñando un ojo
f) Cara sacando la lengua
g) Humanoide saludando
h) Personaje moviendo un pie



§ 1.8 LOS PRIMEROS PASOS

Si queremos que nuestros personajes empiecen a caminar debemos, además de alternar sus disfraces, desplazarlos en la dirección y sentido deseados.

La magnitud del desplazamiento debe ser acorde con el tamaño del personaje y de la velocidad que le queramos imprimir. Por ejemplo, un hombrecito como el que podemos diseñar

en una retícula de 90 x 30 (ver figura más abajo) pixels(*) desplazamientos sean mayores a 12 unidades.

(*) Pixel es la denominación de cada uno de los puntos que componen la imagen en una pantalla. Viene del inglés PICTURE CELS que significa "celdas de imagen".

Nota para docentes: Es conveniente que para dibujar estas retículas el maestro o profesor haga dramatizar en cámara lenta esta acción de dar un paso a un alumno o hacerlo él mismo, para hacer notar que la retícula con las piernas juntas no corresponde a un hombre quieto sino al momento en que sus piernas se cruzan. Esta sucesión de sólo 2 posiciones es una simplificación de un movimiento que para lograrlo bien necesitaría entre 12 y 24 posiciones distintas para un paso tipo que dura 1 segundo (una imagen por cada 24avo de segundo, según lo visto en 1.1) Este cálculo es teórico, en la práctica -como las formas son pequeñas, y esto influye en el resultado visual- es raro que se hagan más de 3 ó 4 formas para una secuencia dada.

Una vez diseñadas las dos retículas necesarias (hacerlo por ejemplo en las formas 40 y 41) un procedimiento para lograr un desplazamiento podría ser:

VERSION LOGO GRAFICO

```
para camina
fforma 40
esperar 30
fforma 41
esperar 30
adelante 15
fin
```

VERSION LOGOWRITER

```
PARA CAMINA
FFIG 45
ESPERA 5
FFIG 46
ESPERA 5
ADELANTE 10
FIN
```

Se sugiere al lector compararlo con el procedimiento flexion del 1.7.

Este efecto se podría mejorar tratando de que la orden del desplazamiento ***ADELANTE*** 12 la desdoblemos en dos ***ADELANTE*** 6 o inclusive en este caso puede aumentarse a un valor ***ADELANTE*** 8, que los pondremos asociados con cada cambio de figura. Por lo que los programas quedarán así:

VERSION LOGO GRAFICO

```
para camina
fforma 40
esperar 30
adelante 8
fforma 41
esperar 30
adelante 8
fin
```

VERSION LOGOWRITER

```
PARA CAMINA
FFIG 45
ESPERA 5
ADELANTE 8
FFIG 46
ESPERA 5
ADELANTE  8
FIN
```

Y para que nuestro personaje, que podría ser el actor (o la tortuga número 2) , camine un total de 80 pasos tendrá que repetir 10 veces los 8 pasos que da, o sea:

VERSION LOGO GRAFICO

```
para caminar
activar 2
sp
frumbo 90
repetir 10 [camina]
fin
```

VERSION LOGOWRITER

```
PARA CAMINAR
DILE 2
SP
FRUMBO 90
REPITE 10 [CAMINA]
FIN
```

DESAFIO

¿Cómo harías para hacer caminar a un perro? Diseña primero las 2 retículas.

DESAFIO

¿Qué pasa si cambias el valor del tiempo entre cambios de disfraz a la mitad o al doble?

§1.9 CONDUCIENDO POR LAS CALLES... A LOS SALTOS

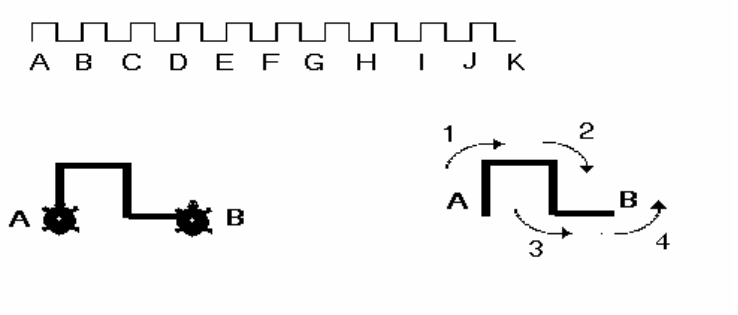
Si queremos producir un desplazamiento no rectilíneo sólo debemos modificar el método ya visto en 1.6.1 pensando en el dibujo que haría la tortuga si estuviera con la pluma.

Por ejemplo, para desplazarse "a los saltos" podemos ensayar el siguiente camino (ver Figura más abajo) que va desde A hasta K, que si lo analizamos descubriremos que lo podemos pensar como el módulo AB que se repite 10 veces.

```
para salto
adelante 4
derecha 90
adelante 4
derecha 90
adelante 4
izquierda 90
adelante 4
izquierda 90
fin

para saltos
frumbo 0          (*)
repetir 10 [salto]
fin
```

(*) Nos aseguramos que salga hacia arriba.

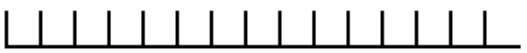
Siempre conviene verificar previamente los módulos del camino graficando con la pluma del actor y luego aplicarla al desplazamiento deseado.



Tendremos que presentar al auto con el procedimiento *auto* y el superprocedimiento que hace todo será entonces:

```
para auto
decir 1
fforma 6
sp fpos [-130 0]
visible
fin

para todo
auto
saltos
fin
```

DESAFIO



¿Te animarías a reproducir otros saltos como estos?

Ayuda: Primero se debe identificar los módulos a repetir.

DESAFIO

Quieres crear otros caminos divertidos y un auto para recorrerlos? Desarrolla los procedimientos necesarios.

§1.10 MOVIENDOSE CIRCULARMENTE

Los trazados y movimientos curvos, particularmente los circulares, nos ofrecen muchas posibilidades, tanto en función de creatividad como en el conocimiento más profundo de la geometría... Para describir una órbita circular necesitamos una sucesión de avances y giros como los del procedimiento siguiente:

```
para orbita
repetir 180 [adelante 1 derecha 2]
fin
```

Nota para docentes: Suponemos que este tema ya fue trabajado con anterioridad en el modo gráfico, así como los conceptos que corresponden a semicircunferencias y cuartos de circunferencias.
Se puede demostrar que la relación entre lo que gira y lo que avanza determina la forma de la trayectoria. O sea que, si giro 2 grados y avanzo 1 paso, o bien, si giro 4 grados y avanzo 2 pasos, o si giro 6 grados y avanzo 3 pasos, como la relación (cociente) entre giro y desplazamiento es en todos estos casos 2/1=4/2=6/3 igual a 2, entonces la forma de la curva (aproximada a arco de circunferencia) será siempre igual.

Distintas velocidades para las órbitas se obtendrán con:

```
repetir 90 [adelante 2 derecha 4]
```
(duplica velocidad)
```
repetir 60 [adelante 3 derecha 6]
```
(triplica velocidad)
```
repetir 45 [adelante 4 derecha 8]
```
(cuadruplica velocidad)

Para que en todos los casos gire una vuelta completa, buscamos de repetir una cantidad de veces adecuadas para que el producto de ese número y el giro resulte igual a 360.

El perímetro se obtendría multiplicando el número de veces que repite por el avance de cada vez, o sea 180 x 1 = 90 x 2 = 60 x 3, etc.

Se pide al lector comprobar que la trayectoria no cambia.

Para disminuir la velocidad con que gira se puede intercalar una orden de esperar 1, esperar 2, esperar 5, etc. dentro de cada orden de repetir, o sea:

```
repetir 180 [adelante 1 derecha 2 esperar 2]
```

§1.11 PASEANDO CON EL CANGURO

Si queremos que un canguro (forma 1 del archivo CANGURO.FRM) se desplace saltando necesitamos un módulo - además del programa de presentación - que realice un movimiento de semicircunferencia, es decir:

```
para salto
frumbo 0
repetir 10 [ad 1 de 18]
fin

para cincosaltos
repetir 5 [salto]
fin
```

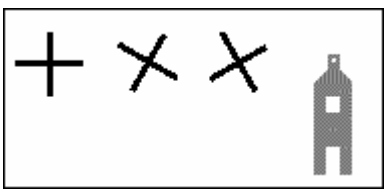
Nota: Puede comprobarse que para que se mueva saltando sobre una misma horizontal, debe agregarse una orden adelante 2 luego de cada salto, con lo que la línea quedaría:
repetir 5 [salto ad 2].

§1.12 ROTACION DE FORMAS

Para que una forma, como por ejemplo las aspas de un molino, roten uniformemente, hacemos por lo menos 3 formas procediendo así:

1. Entramos en el editor de formas
2. Elegimos la opción "Borrar todas". Sin miedo, el archivo NORMAL.FRC, donde están guardadas todas las formas que vienen con el LOGO no se borrarán. En la zona de la izquierda se ven todas las formas con la indicación NO DEFINIDA.
3. Elegimos la forma 1 y hacemos doble clic en "1 NO DEFINIDA" (o bien pulsamos el botón de Editar).
4. Elegimos 34 para el valor horizontal y vertical.
5. Dibujamos una cruz que serán las aspas del molino.
6. Salimos del editor con Archivo - Salir y a la pregunta "LA FORMA FUE MODIFICADA. CONSERVAR LOS CAMBIOS? Elegir Sí. Tener en cuenta que por ahora la computadora conserva esta forma en su memoria, aún no la hemos guardado en el disco.
7. Pulsar el botón COPIAR y luego eligiendo la forma 2 (que dice NO DEFINIDA) pulsamos el botón PEGAR con lo que las aspas se han copiado en la forma 2.
8. Ahora tenemos que girarlas, entonces en el menú Corrimientos y Giros, buscamos Rotar Bloque, y ponemos 30. Salimos como en el punto 6.
9. Vamos a la forma 3 y ponemos PEGAR nuevamente con lo que tenemos otras aspas como las de la forma original (sin rotar). Editamos y giramos ahora 60 grados y salimos.
10. Luego dibujamos el molino en la forma 4. Y no olvidemos guardar en el disco RÍGIDO con el nombre MOLINO.FRC. Si trabajamos con Windows en 16 colores guardaremos con MOLINO.FRM. Puede consultarse en el disquete los archivos MOLINO.LGO y MOLINO.FRM.

En la figura vemos las formas 1 a 4.



Los procedimientos necesarios son:

```
para moli
decir 1
fforma 4
centro
mt
decir 2
fforma 1
fxy 0 14
mt
fin

para gira1
decir 2
fforma 2 esperar 5
fforma 3 esperar 5
fforma 1 esperar 5
fin

para girando
repetir 30 [gira1 ]
fin

para molino
rg ot fcolorf 0
moli
girando
fin
```

Otro ejemplo:

Igual se puede proceder con cualquier forma que rote sobre su eje, como el caso de las viejas peluquerías con sus columnas giratorias, para lo que debemos construir tres formas como las de la figura y alternarlas con un ***ESPERAR*** 5 o 10 entre cada una, como hemos visto recién.

En la figura vemos las formas 1 a 3 del archivo PELUQUE.FRM

§1.13 ANDANDO EN HELICOPTERO

Cuando se trata de una forma que debe trasladarse y cambiar, debemos

```
para heli
decir 1
fxy (-50) (-27)
fforma 1
```

```
mt
frumbo 0
fin

para helice1
decir 1
fforma 1 tono 60 2 ad 2
fforma 2 tono 60 2 ad 2
fforma 3 tono 60 2 ad 2
fforma 2 tono 60 2 ad 2
fin
```

En este procedimiento estamos introduciendo la primitiva ***TONO*** que hace sonar una nota de la frecuencia indicada por el número que la sigue (60) con una duración de 2 centésimos de segundo. Además cada cambio de forma lo movemos con ***ADELANTE*** 2

En las formas 1 , 2 y 3 dibujamos un helicóptero con sus aspas giradas como las que están en el archivo HELICOP.FRM.

```
para volar
repetir 15 [helice1 ]
fin

para helicoptero
rg ot fcf 0
heli
volar
frumbo 90
volar
fin
```

§1.14 MOVIMIENTOS DE 2 ACTORES SIMULTÁNEOS

Para lograrlo se presentan dos actores y luego para moverlos o desplazarlos nos dirigimos a cada uno de ellos alternadamente como se ve en el procedimiento paso.

```
para soldado
decir 1
fforma 35
fx (-70)
fy (-6)
visible
frumbo 90
fin

para paso
decir 1 fforma 35 ad 4      esperar 10
decir 2 fforma 38           esperar 10
```

```
decir 1 fforma 36 ad 4      esperar 10
decir 2 fforma 39           esperar 10
fin
```

Las formas se han tomado del archivo NORMAL.FRM. pero si trabaja con 256 o más colores puede cargar el archivo de formas SOLDADO.FRC (o .FRM) y tomar de allí los personajes.

Al actor 1 lo desplazamos y el actor 2 solamente se mueve en su lugar.

```
para soldado2
decir 2
fforma 38
fx 110
fy (-6)
visible
fin

para accion
decir 1
repetir 20 [paso ]
fin
```

§1.15 EJEMPLO DE UN PROGRAMA DE ANIMACION SECUENCIAL

El programa del archivo SOLDADO.LGO sirve de muestra de cómo trabajar un argumento simple como el siguiente: un soldado se encuentra en un puesto de guardia caminando, hasta que aparece su jefe y le pide que le sirva un mate. El soldado va a buscarlo y vuelve con la pava y el mate. Luego se lo entrega y el jefe toma el mate. El soldado se aleja y se sienta.

Es interesante notar que para caminar se utilizan los procedimientos *caminar* y *caminar3* para hacia la izquierda y *caminar2* para ir hacia la derecha.

Nótese que el *caminar3* sólo difiere del caminar en que lleva el mate en la mano.

El uso de una variable :n (que representa la cantidad de pasos) en los procedimientos antes citados, facilitan la depuración del programa.

```
para caminar3 :n
decir 1 fangulo 180     (*)
repetir :n [paso3 ]
fin
```

(*)Es similar a frumbo 270

En general los procedimientos de este ejemplo hacen buen uso de la modularidad ya que cada uno de ellos representa una acción bien definida. En estos casos es posible crear una nueva escena con facilidad. Podemos cargar y definir los archivos de formas y de procedimientos SOLDADO.FRM y SOLDADO.LGO y luego probar:

```
rg soldado caminar 4
jefe saludo  caminar 2   saludo
jefe saludo  caminar3  2  saludo
```

CAPITULO 2

USO DE VARIABLES

Este Capítulo no trata de ser una explicación sistemática y exhaustiva del tema variables, sino un primer acercamiento al mismo que permitirá al lector su uso en situaciones variadas.

§ 2.1. PROCEDIMIENTOS CON ARGUMENTO

Para crear un procedimiento utilizamos la palabra PARA seguida de un nombre. Por ejemplo:

```
para escalera (*)
repetir 12  [ad  8 de 90 ad 8 iz 90]
fin
```

o bien:

```
para sonidos
repetir 10 [tono 200 20  tono 500 25]   (**)
fin
```

(*) Esta palabra PARA es adoptada como idea de PARA QUE APRENDA LA PALABRA escalera., o PARA ENSEÑARLE LA PALABRA escalera.
(**) TONO toca un sonido de frecuencia = 500 y duración 25 centésimas de seg.

Pero si en el procedimiento sonidos necesitamos ir cambiando alguno o algunos de los valores numéricos de las primitivas que figuran en él, por ejemplo, el número del repetir, resultaría muy tedioso cambiar ese número 10 por 15, luego por 20, y así continuar, generando para cada cambio un nuevo procedimiento. Al cambiar el 10 por 20, el procedimiento quedaría:

```
para sonidos
repetir 15 [tono 200 20  tono 500 25]
fin
```

Para un caso como éste resulta conveniente tener una forma de indicarle al procedimiento que deseo cambiar ese número en el repetir (llamémoslo por ej. veces) por distintos valores. Para eso se escribe el procedimiento así:

```
para sonidos :veces
repetir :veces [ tono 200 20 tono 500 25]
fin
```

siendo :veces el llamado argumento (o variable local) del procedimiento. Para indicarle al procedimiento que veces es una variable le ponemos pegado a la primera letra el signo " : ".

Para que el procedimiento funcione es necesario éste encuentre dentro del mismo en qué lugar debe reemplazarse el valor que le asignamos a :veces; por ello después del repetir escribimos el nombre de la variable (:veces).

Nótese que el nombre (que elegimos nosotros) podría ser cualquiera, por ejemplo :v , :n, etc. Si lo llamáramos :n quedaría:

```
para sonidos :n
repetir :n [ tono 200 20 tono 500 25]
fin
```

IMPORTANTE: Como las variables tienen un nombre cualquiera - al igual que los procedimientos - se indican precediéndolas de dos puntos (pegados a la primer letra) para identificarlas. Luego veremos en el parágrafo 2.3 que hay otro tipo de variables.
También a diferencia de otras versiones de LOGO, Logo Gráfico diferencia mayúsculas y minúsculas en los nombres de las variables .

Finalmente, para ejecutar el procedimiento debo asignarle un valor a la variable, por lo que queda:

```
sonidos 8
sonidos 12
sonidos 3
```

que es la manera de asignarle el valor 8, 12 o 3 a la variable :n (o :veces). Ahora el procedimiento ejecutará las órdenes:

```
repetir 8 [ tono 200 20 tono 500 25]
repetir 12 [ tono 200 20 tono 500 25]
repetir 3 [ tono 200 20 tono 500  25]
```

respectivamente.

De esta manera evitamos tener que repetir un procedimiento muchas veces por la variación de un solo valor.

§ 2.2. PROCEDIMIENTOS CON ARGUMENTO COMPUESTO

El caso recién visto, se denomina con argumento simple, pues hemos necesitado una sola variable, pero puede ocurrir que sea necesario tener más de un valor que sea variable. Como ejemplo, veamos el caso de dibujar una escalera con un procedimiento escalera, como el del parágrafo anterior. Si queremos variar no sólo el número de escalones (12) sino también el 8 que representa el tamaño (alto y ancho) de cada escalón, o sea:

```
para escalera :num :tam
repetir :num [ad :tam de 90 ad :tam iz 90]
fin
```

Para ejecutar este procedimiento debemos escribir después del nombre los dos valores de las variables definidas, en el orden que van y separadas
de un espacio en blanco. Por ejemplo:

```
escalera 4 15                (1)
```

(1) 4 es el número de escalones y 15 el tamaño del escalón

Nota: Es importante no alterar el orden de los valores. Si primero está definida la variable que cambia el número de repetir, cuando ejecutamos el procedimiento debemos poner como primer argumento ese valor.
También es importante no olvidar asignarle a cada variable o argumento definido su valor.

PREGUNTA:

Resultarán escaleras que cubrirán el mismo tramo los casos 1, 2 y 3?
Caso 1: escalera 10 10
Caso 2 : escalera 20 5
Caso 3: escalera 4 25

DESAFIO

Cómo hacemos un procedimiento *escalera2* que haga escaleras con el ancho del escalón distinto del alto y con distintas cantidades de escalones. ¿Cuántas variables usaremos?

DESAFIO

Cómo haríamos para lograr repeticiones de sonido (tono) de diferentes alturas y duraciones? Cuántas variables puede tener el procedimiento sonidos? (ver archivo VARIABLS.LGO).

DESAFIO

Sabiendo que con la primitiva HORA se obtienen la hora minutos y segundos (probarlo) y con ROTULARC seguido de una palabra la tortuga la escribe en la pantalla gráfica, podemos probar de hacer:

```
rotularc hora
```

Como esta primitiva escribe en el sentido en que apunta la tortuga debemos hacerla mirar primero hacia la derecha con la orden de ***FRUMBO*** 90 (fija el rumbo hacia la derecha).

¿Cómo hacer un procedimiento con una variable que haga rotular una cantidad de veces cualquiera un reloj digital? . (respuesta en el archivo VARIABLS.LGO).

Ver también los procedimientos CUADRADO, PALOTES, CIRCUNF, POLI del archivo VARIABLS.LGO

§2.3. VARIABLES GLOBALES

Las variables como las que hemos visto en los ejemplos anteriores adoptan valores válidos sólo para esos procedimientos. Justamente por eso son llamadas "variables locales". Son valederas dentro del procedimiento donde se definen y luego desaparecen, no quedan en memoria.

Existen otras llamadas globales, que resultan muy útiles, una vez creadas permanecen en memoria y pueden ser utilizadas y/o modificadas por cualquier procedimiento.

Estas variables se definen con la primitiva HACER (o su sinónimo ASIGNA) que son las primitivas que permiten definir y asignarle un valor a una variable.

Estos valores pueden ser palabras, números, listas de palabras o números, etc. Las usamos como "pequeños depósitos" en los que se guarda información. En este libro usaremos la primitiva HACER, pero ambas funcionan igual.

Para definir una variable se procede así:

```
hacer "veces 5
```

(Se le asigna a la variable que llamamos "veces el valor 5)

Notar que para definir una variable NO usamos los dos puntos (:) sino que se escribe con comillas (") pegadas a la primera letra, y estas comillas no se cierran. También puede usarse una comilla ('). Esto indica que la variable se llamará veces y su valor será 5.

De esta manera la variable "veces permanece en la memoria y puede ser usada por cualquier procedimiento.

Si deseamos saber el valor o contenido de una variable escribimos:

```
esc :veces
5
```

Aquí va con dos puntos (:) ya que estamos preguntando por su valor o contenido, y no definiéndola con un nombre. Puede ponerse esc valor "veces, ya que VALOR es otra primitiva, pero esta notación es poco usada.

Otro caso sería asignarle a una variable el valor o contenido de otra o de sí misma, por ejemplo:

```
hacer "CARLOS 8
hacer "veces :CARLOS
esc :veces
8
```

Primero creamos una variable llamada "CARLOS y le damos valor 8.
Luego a la variable "veces le damos el valor o contenido de la variable "CARLOS.
Finalmente pedimos ver en pantalla el valor de la variable "veces.

Las variables globales se pueden usar para múltiples propósitos, por ejemplo para guardar información numérica o textual, hacer contadores y acumuladores (ver parágrafos 4 y 5), llevar información de un procedimiento a otro, etc.
Por ejemplo:

```
hacer "vocales [ a e i o u]                    (1)
hacer "notas [ DO RE MI FA SOL LA SI]          (2)
hacer "capitales [ LIMA BRASILIA QUITO SANTIAGO CARACAS];
hacer "colores [ azul verde rojo]
hacer "x 1                                     (3)
hacer "nombre "Roberto                         (4)
```

En los 4 primeros ejemplos estamos guardando información en una lista que quedará en la memoria para consultarla cuando sea necesario. En los dos últimos casos estamos guardando - o asignándole a las variables creadas - un número y una palabra respectivamente.

(1) Está asignándole a la variable :vocales una lista con las 5 vocales.
(2) Está guardando en la variable :notas los nombres de las 7 notas musicales.
(3) Guarda en la variable :x el valor 1.
(4) Guarda en la variable :nombre la palabra "Roberto. Notemos que como es una palabra se precede con comillas (Ver Capítulo 3) En el caso (3) es indistinto poner las comillas o no ponerlas, pues es un número.

§2.4. ¿COMO SE USAN LOS CONTADORES?

Los contadores son variables que sirven para contar la cantidad de veces que ocurre algo, que se ejecuta un procedimiento, que se acierta un número, etc.

Podríamos necesitar un contador que guarde en la memoria una variable, que indique cuántas veces se ejecutó un procedimiento. Para ello será necesario, primero definir la variable en donde guardamos la información. Entonces quedaría:

```
hacer "num 0
```

Le asignamos a la variable de nombre "num el valor 0. Seguidamente haremos un procedimiento sencillo donde veremos como se usa un contador

```
para ganarpuntos
esc [GANASTE UN PUNTO MÁS]      (1)
hacer "num :num + 1             (2)
esc :num                             (3)
fin
```

(1) Se escribe en pantalla de texto el mensaje: GANASTE UN PUNTO MÁS.
(2) Le asignamos a la variable "num el valor de esa misma variable :num más una unidad. De esta manera la variable suma al valor que tenía una unidad.
(3) Escribimos el nuevo valor de la variable "num.

Si no quedó claro probemos en modo directo este otro ejemplo:

```
hacer "puntos 4
esc :puntos
4

hacer "puntos  :puntos + 5
esc :puntos
9
```

EJERCICIOS

Otro uso de las variables podría se el de definir 2 variables con al alto y ancho de un rectángulo y una tercera con el área del mismo:

```
hacer  "ALTO 5
hacer  "ANCHO 6
hacer  "ÁREA   :ALTO * :ANCHO
esc  :ÁREA
30
```

§ 2.5 QUE ES UN ACUMULADOR?

Es muy parecido a un contador, pero en vez de irle sumando 1, le vamos sumando un valor cualquiera, o sea que el acumulador va guardando la suma parcial obtenida en cada momento.

Se usa cuando tenemos procedimientos o primitivas que generan distintos valores que queremos ir sumando, por ejemplo si cada vez que uno gana a un juego se le suman 100 puntos, será:

```
hacer "suma :suma + 100
```

En el archivo ACUMULAD.LGO (en el disquete) hay un ejemplo con el uso de acumuladores en un juego de azar.

CAPITULO 3

INTRODUCCION AL USO DE LISTAS Y PALABRAS

Aparte de las enormes capacidades gráficas del lenguaje LOGO, éste puede usarse con mayor ductilidad que muchos otros lenguajes para el manejo de palabras y de información no numérica, ya que es un lenguaje derivado del famoso LISP (List Processor = procesador de listas) que fue creado, al igual que LOGO, en el Laboratorio de Inteligencia Artificial del Instituto Tecnológico de Massachusetts.

Es posible, por lo tanto, la elaboración de programas de relativa sencillez para incursionar en el uso de la computadora en temas no vinculados a la geometría ni a la matemática.

Resulta entonces muy útil el conocimiento del tema **Listas y Palabras**, ya que es ésta la forma en que accederemos a la mayor potencia que nos brinda el lenguaje LOGO y descubriremos la inmensas posibilidades de su uso en todos los niveles del sistema educativo.

Los elementos que LOGO maneja son llamados "objetos-LOGO", y pueden ser *palabras* o *listas*..

§ 3.1. PALABRAS

Para LOGO, una palabra "**es un conjunto ordenado de caracteres cualesquiera, que no incluyan el espacio en blanco**", el que es usado, como en nuestra lengua, para separar palabras.

Este primer hecho lo diferencia de otros lenguajes que no poseen el reconocimiento automático del "espacio", y una frase cualquiera en otro lenguaje, representa para ellos, una entidad única ("cadena" de caracteres donde los espacios son considerados como un carácter más) y no palabras separadas por espacios.

Esto hace que esos lenguajes sean menos adecuados para la manipulación de lenguaje natural.

Una palabra puede estar integrada no sólo por letras del abecedario sino por cualquier carácter como por ejemplo:

4 6 7 b c f G y % $ # @ ! ? , ñ á ó etc.

Nota: No se admiten como caracteres que formen una palabra los corchetes, paréntesis o símbolos matemáticos.

Cuando, en la definición de "palabra", dijimos que era un conjunto ordenado, es porque para LOGO no es lo mismo la palabra SACO que COSA, pues aunque tengan los mismos caracteres, difieren en su "orden".

Como caso especial debemos considerar los números, que también entran en la categoría de palabras, aunque ya veremos luego que LOGO las puede distinguir fácilmente.

La palabras deben precederse de comillas ("), por ejemplo, para escribir en la pantalla la palabra APRENDIZAJE, ordenamos:

```
escribir  "APRENDIZAJE
```

Las comillas en general - en Logo - no se cierran, pero en esta versión es indistinto cerrarlas o no, o también usar comilla simple.

Existe una palabra muy especial que -al igual que un conjunto vacío- no tiene elementos, y es llamada **palabra vacía**. También puede indicarse

con comillas (") o abrir y cerrar comillas dejando o no un espacio blanco (" "). Probar:

```
escribir  "
```

o bien:

```
escribir  ""
```

§ 3.2. LISTAS

Una lista es un conjunto ordenado de objetos-LOGO, ya sean palabras u otras listas(*). Al escribir una lista se lo hace con sus elementos separados por espacios, y todo encerrado entre dos corchetes.

(*)NOTA: Para el lector conocedor de la teoría de conjuntos: este último caso sería como conjuntos que tienen dentro a otros conjuntos llamados subconjuntos. Aunque la diferencia con los conjuntos es que aquellos no pueden contener elementos repetidos y en cambio las listas si pueden.

Son ejemplos de listas:

```
a) [ 1 2 3 4 5]
b) [UNA TEORIA DE EDUCACION]
c) [¡ Cierra la puerta !]
d) [[perro negro] [gato malo] [osa blanca]]
e) [[JOSE 8] [ANA  3] [LUIS 4] [JUAN 7]]
```

Para LOGO las tres primeras (a, b y c) son listas de palabras, mientras que las últimas dos (d y e) son listas que contienen otras listas (estas últimas llamadas subsistas). Nótese que el ejemplo c) tiene 5 palabras que incluyen los signos de admiración.

Existe también la lista vacía [ ] que es una lista que no tiene elementos.

Para trabajar con ellas en la computadora podemos tipear:

```
mostrar [1 2 3 4 5]
[1 2 3 4 5]
```

o bien:

```
escribir [1 2 3 4 5]
1 2 3 4 5
```

Veremos que en el segundo caso no se ven los corchetes en la pantalla, mientras que en el primero, sí.

La primitiva ***MOSTRAR*** sirve para mostrar en pantalla los corchetes y distinguir en algunos casos confusos, por ejemplo el caso de una lista de un solo elemento, si el objeto-LOGO que estamos tratando es lista o palabra. También podemos utilizar las primitivas ***LISTA?*** o ***PALABRA?*** que nos devuelven si es una lista o una palabra.

Saber trabajar con Palabras y Listas significa saber como modificarlas, recorrerlas, buscar un elemento en ellas, acortarlas, unirlas a otras, etc., para ello necesitamos conocer algunas primitivas más.

§ 3.3. PRIMERO Y ULTIMO

Si de una palabra o lista queremos tomar solamente el primer elemento usaremos(*):

```
primero [LOGO GRAFICO]      <Enter>
```

y obtendremos en la pantalla:

LOGO

*(*)En primitivas que son **funciones**, o sea que devuelven un valor (por ejemplo: **primero, ultimo, rumbo, ponerprimero,** etc.), o cuando consultamos el valor de una variable (por ejemplo :var) NO es necesario anteponer la primitiva escribir o mostrar, ya que es una característica de LOGO GRAFICO que responde con el valor de la función ejecutada, sin dar ningún mensaje de error.*
*Por lo que en los ejemplos no estarán antepuestas las primitiva **esc o mostrar**.*

en cambio si ordenamos:

```
ultimo [LOGO GRAFICO]  <Enter>
```

obtenemos:

GRAFICO

Probemos que ocurre con:

```
a) pri "TORTUGA                  <Enter>
b) ult 12340                     <Enter>
c) primero [¡ HOLA ! ]           <Enter>
d) ultimo [ [ a b c] [ 1 2 3 ] ] <Enter>
```

Podemos probar con ***primero*** y ***ultimo*** con los casos a, b, c, d, y e del parágrafo anterior.

Nótese que estamos usando los apócopes de primero (pri) y ultimo (ult). Primero y ultimo - al igual que la primitivas que se ven en el parágrafo siguiente son funciones (en Logo Writer son llamadas reporteros)

§ 3.4. MENOSPRIMERO Y MENOSULTIMO

Si queremos eliminar el primero o el último elemento de una palabra o lista usaremos las primitivas ***menosprimero*** y ***menosultimo***. Se abrevian con ***mp*** y ***mu***.

Por ejemplo con:

```
menosprimero [ 1 2 3]       <Enter>
```

se obtiene:

2

Como dijimos antes, si deseamos visualizar los corchetes podemos usar la primitiva ***mostrar*** en vez de ***escribir***, entonces escribamos:

```
mostrar menosprimero [1 2 3]          <Enter>
```

se obtiene:

[2 3]

Se pueden usar su abreviaturas que son ***mp*** y ***mu*** respectivamente.

Se sugiere probar o anticipar que ocurrirá con:

```
a) mu  "CALA
b) mp "CALA
c) mostrar mp mp [ a e i o u ]
d) mu mp [EL ERROR ES FUENTE DE APRENDIZAJE]
e) mp mp mp [ a e i o u]
f) mp mp mu [ [ a 1] [ b 2] [ c 3] ]
g) mp mp [ 1 2]
h) mp 123 (*)
i) mp mu 123 (*)
```

(*) Los valores numéricos pueden o no llevar antepuestas las comillas.

Para obtener el segundo elemento de una lista o palabra podemos proceder así:

```
primero mp [A E I O U ]          <Enter>
E
```

pues el primero de [E I O U] es la letra "E".

Nótese que para analizar como están actuando estas funciones conviene verlo de derecha a izquierda, partiendo de la lista [A E I O U]. La función que está más cerca a la izquierda de la lista es ***mp*** entonces se aplica a la lista [A E I O U] y se obtiene [E I O U], luego actúa la función ***primero*** que extrae la letra A de la lista ya que es la primera.

Nota para maestros:

Las primitivas **MENOSPRIMERO** y **MENOSULTIMO** es recomendable explicarlas a alumnos (edades entre 11 y 12 años) con la analogía de un tubo de pelotas de tenis de distintos colores o hacer un tubo de papel con bolitas de distinto color en el que para sacar tomar la segunda bolita es necesario sacar la primera antes, o para sacar la última es necesario empezar desde atrás, o si deseamos sacar la anteúltima debemos sacar primeramente la última y así explicar como vamos sacando pelotas o bolitas de un tubo, que será la lista de palabras o elementos a explicar.

DESAFIO

¿Aplicando las primitivas vistas, ***primero, ultimo, mp y mu***, te animas a que la computadora imprima en la pantalla los elementos subrayados?

a) EL GATO BLANCO ES MUY BUENO
b) abcdefghi
c) PENSAR SIGNIFICA MANEJAR PALABRAS

§ 3.5 RECORRIENDO UNA PALABRA O LISTA

Probemos que ocurre si tipeo:

```
primero primero [ERASE UN AVE SOLITARIA]
```

Obtendré una

E

ya que:

```
primero [ERASE UN AVE SOLITARIA]
```

devuelve...

ERASE

y como se le aplica otra vez el operador ***primero***, ahora actúa sobre la palabra ERASE, y nos devuelve la letra E.

Para entender mejor cómo LOGO ejecuta nuestra orden, especialmente en los casos donde éstas se aplican en forma reiterada (*), como en el ejemplo anterior, conviene "leerla" de derecha a izquierda. Supongamos el caso:

```
pri ult mu [LOS HERMANOS SEAN UNIDOS]
<-- <-- <--
III  II  I
```

S

Esta operación se ejecuta en 3 pasos:

I) se aplica ***mu*** a la lista y queda:

```
[ LOS HERMANOS SEAN ]
```

II) se aplica ***ultimo (ult)*** a la lista obtenida, y este operador entrega la palabra:

SEAN

III) Finalmente se aplica ***primero (pri)*** a la palabra SEAN y lo que se entrega a ***escribe*** es la letra S, que se imprime en la pantalla.

S

(*) En matemática una función como ésta es llamada "función compuesta" o "función de función".

§ 3.6. UNA PRIMITIVA MAS

La primitiva ***ítem*** nos permite tomar un elemento cualquiera de una lista o palabra:

```
item 4 "AVIONES        <Enter>
```

obtenemos:

O

Y si ordenamos:

```
item 7 [a b c ch d e f g h i j k]          <Enter>
```

vemos en la pantalla:

f

O también:

```
item 3 [ FRANCIA PARIS INGLATERRA LONDRES PERU LIMA];
```

obtenemos:

INGLATERRA

§ 3.7. USO DE LA PRIMITIVA "PALABRA"

Cuando deseamos unir dos caracteres o palabras lo hacemos así:

```
palabra "ALFA "BETO              <Enter>
```

nos da:

ALFABETO

También se puede hacer con números

```
palabra "23 "00
```

nos da:

2300

Si usamos variables globales (Ver el capítulo correspondiente) podríamos ejemplificarlo así:

```
hacer "pal1 "GENTIL        (6)
hacer "pal2 "HOMBRE
```

(6) Asigna el valor GENTIL a la variable :pal1. para más aclaraciones ver cap. 2.

```
hacer "unidas palabra :pal1 :pal2
esc :unidas
GENTILHOMBRE
```

Como se expresó más arriba , también se puede usar esta primitiva para "unir" dos caracteres o números:

```
palabra 9  0          <Enter>
```

nos da:

```
90
```

Para más de dos palabras a unir se utilizan tantas primitivas ***palabra*** (o su abreviatura ***pal***) como sean necesarias o bien el operador ***&&*** .

```
pal pal "META "BI "SULFITO  <Enter>
METABISULFITO
```

Puede emplearse también:

```
"META && "BI && "SULFITO
```

§ 3.8 DOS PRIMITIVAS MUY UTILES: MIEMBRO? y NUMMIEMBRO

La primera de ellas verifica si un elemento está presente en una lista o palabra. Por ejemplo si escribimos (en modo directo):

```
miembro? "Z "CALABAZA
```

obtenemos:

VERDAD

DESAFIO

Probar y analizar con las siguientes casos:

```
A. miembro? 3  243
B. miembro? "E "CALABAZA
C. miembro? "a  "CALABAZA
D. miembro? "LONDRES [ FRANCIA PARIS INGLATERRA LONDRES PERU
LIMA];
```

```
E. miembro? 15 [ 5 10 15 20 25 30]
F. miembro? 2 [ 5 10 15 20 25 30]
G. miembro? "PARI [FRANCIA PARIS INGLATERRA LONDRES PERU LIMA];
H. hacer "vocales "aeiou  esc miembro? "e  :vocales
```

Verificar que sólo dan VERDAD los casos A, D, E y H.

Inventar otros 5 que den VERDAD y 5 que den FALSO.

La primitiva ***NUMMIEMBRO*** suele usarse como complemento de ***MIEMBRO?*** ya que si el objeto buscado pertenece a la lista o palabra, podemos saber que posición ocupa.
Por ejemplo:

```
nummiembro "Z "CALABAZA
```

obtenemos:

7

DESAFIO:

Probar y analizar con las siguientes casos:

```
A. nummiembro 3 243
B. nummiembro "E "CALABAZA
C. nummiembro "a "CALABAZA
D. nummiembro "LONDRES [ FRANCIA PARIS INGLATERRA LONDRES PERU
LIMA];
E. nummiembro 15 [ 5 10 15 20 25 30]
F. nummiembro 2 [ 5 10 15 20 25 30]
G. nummiembro "PARIS [ FRANCIA PARIS INGLATERRA LONDRES PERU
LIMA];
H. hacer "vocales "aeiou esc nummiembro "e :vocales
```

A: 3 , D: 4 , E: 3, H: 2, y el resto da 0 que quiere decir que no pertenece a la lista.

Generalmente esta segunda primitiva se usa en combinación con ***MIEMBRO?*** , así:

```
si miembro? "L "MURCIELAGO hacer "posicion nummiembro "L
"MURCIELAGO;
```

También es muy útil combinarla con ***ITEM***.

Veamos un ejemplo concreto.

Si en una lista de países y capitales buscamos la capital de un país determinado podemos hacer así:

```
hacer "paiscapi [ FRANCIA PARIS PERU LIMA ..];

para capitalde :pais
esc item nummiembro + 1 :pais :paiscapi
fin
```

se usaría así:

```
capitalde "INGLATERRA
```

o

```
capitalde "PERU
```

Nota: En realidad este procedimiento debería llamarse SIGUIENTE pues busca al siguiente elemento de una lista. Con ese nombre y con algunas mejoras consta en el archivo LISTAS.LGO y en SIGUIENT.LGO.

§ 3.9 PARA CONSTRUIR UN ESPEJO

Podemos programar la inversa de una palabra o lista con la primitiva ***INVERTIR***

```
invertir "DESAFIO
```

OIFASED

```
invertir "NEUQUEN
```

NEUQUEN

Si al derecho y al revés son iguales (capicúa) la palabra se llama "palíndromo".

¿Cómo podemos programarlo para que si escribimos:

```
espalindromo "SACAS
```

responda ES PALINDROMO o NO ES PALINDROMO (ver archivo ESPALIN.LGO)

§ 3.10 USO DE LA PRIMITIVA "FRASE"

Como hemos visto, utilizamos la primitiva ***PALABRA*** para unir dos o más palabras. Por ejemplo:

```
esc palabra "ca "sa
casa
```

Para poder unir una lista con otra, o para ir guardando datos (palabras) dentro de una lista usaremos la primitiva ***FRASE***.

La primitiva ***FRASE*** se puede abreviar ***FR*** y al igual que con la primitiva ***PALABRA*** se podrán usar dos o más si queremos unir más de dos listas o palabras.

Veamos cómo se unen una lista con una palabra:

```
mostrar frase [JUAN PEDRO] "DIAZ
[JUAN PEDRO DIAZ]

mostrar frase "La [noche fría]
[La noche fría]
```

En el primer ejemplo, el resultado es equivalente a usar la primitiva ***PONERULTIMO***, en el segundo ejemplo equivale a la primitiva ***PONERPRIMERO***.

Para el caso de manejar listas almacenadas dentro de variables veremos algunos ejemplos:

```
hacer "fra1 [La noche está ]
hacer "fra2 [fría y hermosa]

mostrar frase :fra1 :fra2
[La noche está fría y hermosa]
```

Siempre une el segundo argumento al final del primero; o sea que pone el contenido de la segunda variable al final de la primera, si cambiamos el ejemplo nos queda:

```
mostrar frase :fra2 :fra1
[fría y hermosa La noche está]
```

Podemos unir dentro de una variable que contiene una lista, otra lista.

```
hacer "fra1 fr :fra1 :fra2
esc :fra1

La noche está fría y hermosa
```

Hemos unido dentro de la variable "fra1 el contenido de la misma, más el contenido de la variable "fra2.

§3.11 USANDO EL PROCEDIMIENTO DESORDENAR

Podría haber en LOGO muchísimas más primitivas pero como es posible construir procedimientos cada usuario puede construirlos en la medida de sus necesidades y conocimientos del lenguaje.

Se han incluido en el disquete que acompaña este material una cantidad de procedimientos o funciones para manejar listas en el subdirectorio LISTAS. Pueden cargarse con el menú ARCHIVO - CARGAR Y DEFINIR... que hace que no aparezcan en el editor de texto pero sí quedarán cargados en la memoria como "útiles"(*)

Uno muy útil es la función DESORDENAR ya que entrega una palabra o lista con sus elementos cambiados de lugar lo que nos permite usarlo para adivinanzas o anagramas.

De esta manera, si cargo el archivo DESORDEN.LGO puedo usarlo así:

```
desordenar "adivinar  (probarlo)
```

o bien

```
desordenar 123456
```

(los números pueden usarse con o sin comillas)

(Estos procedimientos están en el archivo ADIVINA.LGO).

(*) En la versión de LOGO WRITER existe la primitiva CARGAUTILES que funciona de manera parecida. En la versión de LOGO GRAFICO 4.4 se pueden ver los procedimiento cargados de esta manera en memoria con la opción de menú ***EJECUTAR - LISTAR PROCEDIMIENTOS...*** *o el botón identificado con* ***para...*** *de la Barra de Herramientas. Entrando con cualquiera de estas opciones accedemos a una ventana que nos indica con (M) los procedimientos o funciones cargadas directamente en la memoria sin poderse acceder a ellos en el editor de textos.*

CAPITULO 4

ALGUNAS CUENTAS

Veremos ahora cómo se usan algunas funciones matemáticas como por ejemplo:

```
ESCRIBIR SUMA 2 3
da 5              igual que ESCRIBIR 2 + 3
```

Nota: En muchas versiones de Logo - salvo la versión DR. LOGO de Digital Research - se adoptó la modalidad de que las funciones debían ser precedidas de un procedimiento u orden como ESCRIBIR. Si por ejemplo escribía:

3 * 5 se producía el mensaje **NO SE QUE HACER CON 15**

pero en esta versión (al igual que con DR LOGO) se prefiere que por defecto si no se indica que hacer con una función se toma que debe salir por pantalla, por lo que en general podría omitirse la orden *ESCRIBIR* precediendo la función.

```
DIF 8 2
da 6              es análogo a   8 - 2
PROD 3 4
da 12             es análogo a   3 * 4
DIVISION 5 4
da 1.25           es análogo a   5 / 4
```

Existe también ***COCIENTE*** que nos devuelve la parte entera de la división (despreciando los decimales que pudiera haber), por ejemplo:

```
COCIENTE 5 4
 da 1        y no 1.25 como daría DIVISION 5 4
```

es decir, devuelve la parte entera del resultado de la división.

También están ***RC*** (raíz cuadrada) y ***POT*** (potencia) que se usan:

```
RC 16             da 4
POT 3 2           da 9
```

ya que se escribe primero la base y luego el exponente. Entonces la raíz cúbica de 27 se calcula con:

```
POT 27 (1/3)
que da 3.
```

ya que el exponente es 1/3. También puede usarse el "acento circunflejo" que representa el operador de la potencia, así es que puede ponerse:

```
27 ^ (1/3)
```

§ 4.1 ALGUNAS FUNCIONES MATEMATICAS

Como Logo es un lenguaje extensible, podemos crear funciones matemáticas de acuerdo a nuestra necesidades y Logo las interpreta como si fueran primitivas.

Podemos usar el conocimiento de listas - que se explicarán en el Capítulo 9 - para realizar procedimientos o funciones que nos permitan sumar una lista de n números, por ejemplo con la función (que está en el disquete) ***sumar :lista.***

Podemos sumar una lista de números positivos o negativos, por ejemplo

```
sumar [ 1 2 3 4 5 6 7 8 9]
45
sumar [ 3 -2 4 -2.5]
2.5
```

Otras funciones que hemos creado y que figuran en el disquete son:

promedio
máximo

Para hacer funciones que nos resulten útiles se debe tener en cuenta que para que devuelvan (o entreguen un resultado) debe usarse la primitiva ***RESPUESTA*** o ***RESP***. Debiendo seguir las normas para la realización de funciones con Logo Gráfico, donde el procedimiento debe comenzar con "func".

Por ejemplo :

```
FUNC CUADRADO :N
RESP :N * :N
FIN

CUADRADO 4
16

FUNC  INVERSO :N
SI :N = 0 RESP [ NO SE PUEDE DIVIDIR POR CERO]
RESP 1/:N
FIN

INVERSO 8
0.125

FUNC MEDIA :X :Y
RESP (:X + :Y) /2
FIN

MEDIA 5 6
5.5
```

A veces hay órdenes que no admiten valores con decimales como es el caso, por ejemplo de ***FCOLOR, TONO, FTRAZO, REPETIR,*** etc.

Entonces suele ser útil conocer una forma de despreciar los decimales tomando sólo la parte entera, para ello usamos la primitiva ***ENTERO*** :NUM.

Por ejemplo si queremos que a la nota musical DO de frecuencia 262 le sigan las notas de dos escalas completas, se verifica que cada frecuencia es igual a la anterior por el factor 1.05946 - que es en realidad la raíz 12ava de 2 . Con este dato podríamos hacer un procedimiento recursivo como este:

```
para escala :frec
si :frec > 523 parar   (*)
tono entero :frec 20
esperar 40
escala :frec * 1.05946
fin
```

(*) Para al llegar al DO siguiente

Se usaría así:

```
escala 262
```

§ 4.2 EJEMPLOS CON OPERACIONES MATEMATICAS

Podemos generar algunos procedimientos muy sencillos en los cuales ingresando el dato resolvemos problemas matemáticos.

Para la programación de estos procedimientos, hemos procedido de dos formas:

- Una, generando un procedimiento (para) donde le asignamos el valor de la operación matemática a una variable, y luego con las primitivas ***ESCRIBIR*** y ***FRASE*** formamos una oración con la respuesta del resultado. Para ello los procedimientos constan de uno o dos argumentos (variables) de entrada donde escribimos los datos.

- La otra, creando funciones, (ver capítulo 9) que directamente nos devuelvan el resultado de la operación; éstas también tienen uno o dos argumentos de entrada. La ventaja de usar funciones es que en éstas se pueden asignar a una variable o como dato para un procedimiento.

En la parte B del manual figura, en el capítulo de funciones matemáticas, los niveles o jerarquías para las funciones prefijas e infijas.

Veremos ejemplos de ambas formas y al final cómo utilizar una función como dato.

```
para perimetro :d
hacer "perim pi * :d
esc fr [El perímetro de la circunferencia es igual a: ] :perim
fin

func peri :d
resp pi * :d
fin
```

Donde **PI** es una primitiva que equivale a 3.1415926

```
para superf :r
hacer "sup pi * POT  :r  2
esc fr [La superficie de la circunferencia es igual a: ] :sup
fin

func sup :r
resp pi * :r ^ 2
fin

para PITAGORAS  :lmenor :lmayor;
hacer "hipo rc (:lmenor ^ 2  + :lmayor ^ 2);
esc fr [La hipotenusa del triángulo rectángulo es: ] :hipo;
fin;

func pitag :lmenor :lmayor
resp rc (:lmenor ^ 2  + :lmayor ^ 2)
fin

para perimetro_for :lado :cant;
hacer "perif  :lado * :cant;
esc fr [El perímetro de la figura es igual a:] :perif;
fin;

func peri_for :lado :cant
resp  :lado * :cant
fin

para polígono :perim :apotema;
hacer "pol (:perim * :apotema) / 2;
esc fr [La superficie de un polígono regular es: ] :pol;
fin;

func poli :perim :apotema
resp (:perim * :apotema) / 2
fin

para porcentaje :porc :valor;
hacer "porcen  (:valor * :porc) / 100;
escs fr "El :porc escs fr [porciento de] :valor esc fr [es]
:porcen;
fin;

func porcen :valor :porc
resp (:valor * :porc) / 100
fin
```

Como vemos, tenemos en primer término el procedimiento, el segundo es la función; la distinguimos por cómo es el encabezado (una es ***PARA*** y la otra es ***FUNC***).

Por ejemplo:

El procedimiento ***polígono*** nos da como respuesta:

```
La superficie de un polígono regular es: ...
```

Esta oración es fija para ese procedimiento, pero nosotros con la función podemos calcular la superficie del polígono asignándole el resultado a cualquier oración o procedimiento.

```
poli 4 5
10

esc fr [ la superficie es:] poli 4 5
la superficie es: 10
```

*Nota: La primitiva **frase** une la lista [la superficie es:] con el número que entrega como resultado la función **poli**, haciendo una sola lista.*

Otro ejemplo:

```
PITAGORAS 3 4
```

La hipotenusa del triángulo rectángulo es: 5

```
pitag 3 4
5
```

Si en un ejemplo de geometría debemos hacer caminar a la tortuga el perímetro de una circunferencia de radio 35 hacemos

```
peri 35
109.955743
```

Este número es difícil de interpretar y decir, así que lo vamos a redondear

```
entero peri 35
109
```

Entonces le diremos a la tortuga:

```
ad entero peri 35
```

Aquí puede no ser necesario el redondeo, ya que la primitiva ***ADELANTE*** permite números decimales.

§ 4.3 Pidiendo la hora

Logo Gráfico dispone de funciones para saber la fecha y la hora, así es que si escribo:

```
fecha
```

responde (por ejemplo) con 3 números en forma de lista:

```
[ 25 5 97]
```

que representan en orden el día, el mes y el año.

Igualmente si escribo:

```
hora
nos daría:   [ 9  22 15]
```

que representan la hora, los minutos y los segundos en ese momento.

Si queremos crear una función que nos de solamente los segundos procederemos así:

```
func seg
resp ultimo hora
fin
```

Para obtener sólo la fecha de ese día hacemos:

```
func dia
resp primero fecha
fin
```

También podemos necesitar una función que nos devuelva los minutos que pasaron en ese día, entonces hacemos:

```
func min
resp 60*(primero hora) + item 2 hora
fin
```

DESAFIO

¿Cómo hacer que nos devuelva la cantidad de segundos que pasaron del día?

§ 4.4 CAMBIANDO LA PRESICION DE UNA OPERACION

Con la función ***FORMATO*** logramos elegir la cantidad de decimales con que queremos trabajar o expresar un resultado, por ejemplo, nótese la diferencia entre:

```
ESC FORMATO   5/3  2
1.67
ESC FORMATO   5/3  3
1.667
ESC FORMATO   5/3  4
1.6667
```

§ 4.5 TAMBIEN PODEMOS TRABAJAR CON PARES ORDENADOS

Por ejemplo puedo sumar dos pares ordenados (o vectores), como lo son las posiciones en un plano definidas por sus coordenadas.

```
[ 3 4] + [ 5 6]
```

```
[ 8 10]
```

También si multiplicamos un número por un par ordenado obtenemos:

```
3 * [ 20 30]
[60 90]
```

CAPITULO 5

LOGO Y LA PROBABILIDAD

Que un aparato o maquinaria haga alguna tarea automáticamente a esta altura del siglo XX y casi en el XXI, no asombra a nadie. Pero entre los innumerables y modernos inventos disponibles en la actualidad - como automóviles, lavarropas, procesadoras, etc., no hay ninguno que haga tareas al azar.

El azar es un factor interviniente en muchos juegos como ruleta, dados, lotería quiniela, cartas, etc.

En la computadora se ha incluido esta posibilidad que siempre asombra con sus resultados, inclusive en los procedimientos más simples, ya que le da cierta similitud con el comportamiento de los seres vivos. Es decir, parece que hiciera lo que ella quiere. Sabemos que no es así, pero no obstante siempre intriga -y hasta apasiona- la incertidumbre del resultado a obtener.

Son innumerables las aplicaciones que se pueden concebir, veamos algunas.

§ 5.1 LA PRIMITIVA AZAR

La primitiva ***AZAR*** se utiliza para provocar un sorteo entre N números pertenecientes a un intervalo entre 0 y N -1, siendo N el número que acompaña a ***AZAR***.

Por ejemplo:

```
AZAR 10
```
sortea entre 10 números (de 0 a 9)
```
AZAR 2
```
sortea entre 2 números (0 y 1)
```
AZAR 100
```
sortea entre 100 nros. (de 0 a 99)

Para probar que esto es cierto podemos probar con :

```
repetir 20 [esc azar 100]
```

o para que aparezcan en la misma línea

```
repetir 30 [escribirs azar 100 ]
```

si queremos separar los dígitos sorteados por un guión ponemos:

```
repetir 30 [escs azar 100 escs "-]
```

Probémoslo con un par de gráficos de tortuga:

```
adelante azar 100
```

que producirá un avance de la tortuga entre 0 y 99 pasos, o bien :

```
repetir 100 [ ad 40 at 40 de azar 31]
```

que producirá un giro entre 0 y 30 a un conjunto de rayos - dispuestos radialmente desde el centro- de 40 pasos de longitud.

Si se quiere sortear números desde un valor que no sea cero debe sumársele este extremo inferior del sorteo, así por ejemplo:

```
FCOLOR 3 + AZAR 10
```

pondrá un color sorteado entre 3 y 12, ya qe ***azar 10*** sortea entre 0 y 9 y al sumárseles 3 el resultado quedará entre 3 y 12 como se dijo.

De esta manera es posible lograr sorteos de números entre cualquier par de números extremos A y B con:

```
A + azar (B - A + 1)
```

siendo A= el extremo inferior y B= el extremo superior del conjunto de números posibles. En el ejemplo este valor máximo B es 10 pues 12 - 3 + 1 = 10.

Por ejemplo para sortear entre 30 y 40 usaremos:

```
30 + AZAR (40 - 30 + 1)
```

o sea

```
30 + AZAR 11
```

Se aconseja ayudarse para entender es concepto, hacer un procedimiento:

```
PARA NUM
ESC 30 + AZAR 11
FIN
```

y verificar este razonamiento con REPETIR 20 [NUM] viendo los números que salen sorteados.

§ 5.2 USANDO EL AZAR

El azar se puede usar de muchas formas interesantes en la parte gráfica.

IDEA 1

Probemos el siguiente gráfico usando colores al azar

```
repetir 100 [fcolor azar 16 ad 90 de 123]
```

que sortea 16 colores y hace un trazo de cada color

IDEA 2:

Con el procedimiento ***circulo*** lograremos un interesante círculo puntillista de radio 100. ¿Por qué?

```
para circulo
fcf 0             () Pone color de fondo negro.
pentera           () Pone pantalla gráfica entera
ot sp
fgrosor 2
repetir 2000 [puntito]
fin

para puntito
fcolor 1 + azar 16     () Fija color del trazo entre 1 y 15
fpos [ 0 0]
frumbo azar 360        () A interpretar por el lector...
ad azar 40             () Idem anterior
punto
fin
```

Se consiguen variantes interesantes trabajando con distintos grosores (probar con ***FGROSOR 4*** y ***FGROSOR 6***.).

§ 5.3 SIMULANDO UN DADO

Si queremos programar un juego para tratar de adivinar un número sorteado por un imaginario dado, podemos empezar así:

```
esc 1 + azar 6
```

con esta orden solamente sorteamos un número entre 1 y 6 y lo escribimos, pero debemos modificarlo ya que no debemos escribirlo sino guardarlo en una variable que podríamos llamar ***"dado*** usando la primitiva ***hacer,*** (este programa está en el disquete con el nombre DADO.LGO)

```
para dado
hacer "dado 1 + azar 6      (1)
apostar
dado                        (2)
fin

para apostar;
esc [A QUE NUMERO LE APUESTA? (Pulsar 1 a 6)];
hacer "num leercar     ;                  (3)
si :dado = :num esc [GANASTE! ] sino escribirs [LO LAMENTO!]; (4)
escribirs [SALIO EL...] esc :dado;
fin;
```

(1) Guarda en la variable :dado el número sorteado.
(2) Convierte el programa en recursivo, es decir se llama a sí mismo
(3) Esta orden registra que tecla se ha pulsado y la guarda en una variable llamada :num.
(4) Verifica si el numero pulsado :num es igual que el sorteado y guardado en :dado.

§ 5.4. ALGO DE PROBABILIDADES Y ESTADISTICA

Sabemos que la probabilidad de que ocurra un hecho se expresa como el cociente entre el número de casos favorables sobre los posibles.

En el caso de un dado esta probabilidad es 1/6 ya que los casos posibles son 6 (del 1 al 6) y hay sólo un caso favorable (que es cuando sale el número esperado).

Vamos a simular esta situación. Por ejemplo si con el procedimiento **unsorteo** sorteamos y escribimos el número que salió**.**

```
para unsorteo
hacer "num 1 + azar 6
escs fr "_ :num            (1)
fin
```

(1) Esta línea es para separar los números con guiones.

Con **sorteos** (procedimiento con variable) sorteamos una cantidad de veces (:cant) y queremos contar cuántas veces sale - por ejemplo - el número 5, hacemos así:

```
para sorteos :cant
hacer "veces 0
repetir :cant  [unsorteo si :num = 5 hacer "veces :veces + 1]
linea
esc frase [REPITENCIAS DEL 5 ... ]  :veces
esc frase [TOTAL DE TIROS: ] :cant
esc frase [RELACION ACIERTOS/TOTAL] :veces/:cant
fin
```

Hemos agregado una línea que calcula el cociente entre la cantidad de veces que sale el número y la cantidad total. Este cociente debería acercarse al valor de la probabilidad de acertar - en este caso es 1/6 = 0.166666 periódico).

Podemos probar ordenando varias veces (con una no sirve).

```
sorteos 10
sorteos 100
sorteos 500
```

y ver como se acerca cada vez más al valor esperado.

EJERCICIO:

Supongamos que queremos poner 500 puntos al azar en la pantalla (como estrellas) diseminadas en la zona comprendida desde el centro de la pantalla hasta x = 100 y hasta y = 100 (o sea una zona cuadrada de lado 100 a la derecha y arriba del centro de la pantalla).

Queremos contar cuántos puntos estarán a menos de 40 pasos de tortuga del centro.

Primero hacemos un procedimiento para poner un puntito en un sitio al azar.

```
para puntito
fx azar 101          (1)
fy azar 101          (1)
```

```
si (distancia [0 0]) < 40 hacer "n :n + 1 fc 21 sino fc 12    (2)
punto                (3)
fin
```

(1) En las dos primeras líneas se coloca la tortuga con coordenadas al azar de 0 a 100.
*(2) Si la distancia al centro [0 0] es menor que 40 pasos de tortuga para contabilizar ese punto aumentamos en 1 el contador :**n** y para que se vea mejor le ponemos color 21 (blanco), si está más lejos le ponemos otro color, por ejemplo el 12 (lila).*
*(3) Las primitiva **punto** sirve para poner un punto del grosor de trazo actual (fijado en ese momento) en el lugar donde está la tortuga. El grosor del trazo de la tortuga al iniciar el programa es 2.*

Finalmente elaboramos el superprocedimiento (*) *cercanos* colocándole valor cero a la variable ***:n*** antes de empezar a sortear los puntos.

*(*) Se le dice **superprocedimiento** a un procedimiento que "llama" o se vale de otro u otros procedimientos, que son - en relación al primero - **subprocedimientos**, como son los subconjuntos a un conjunto. En una jerarquía más compleja puede haber otros subprocedimientos dentro de los subprocedimientos.*

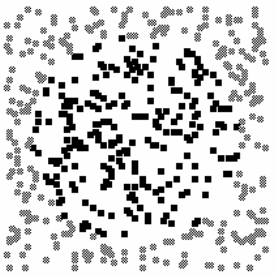
```
para cercanos
bp fcf 0 ot sp
fgrosor 2
hacer "n 0
repetir 500 [puntito]
esc [PUNTOS TOTALES = 500]
escs [PUNTOS CERCANOS =]
esc :n
fin
```

En este procedimiento le ponemos grosor 2 a los puntos y hacemos saltar a la tortuga de punto en punto. Como los puntos son muchos nos conviene ocultar a la tortuga ya que así grafica más rápido.

También imprimimos un mensaje que dice la cantidad total de puntos y cuántos están más cerca de 40 pasos del centro de la pantalla.

DESAFIO:

Cómo harías para que ahora los 500 puntos fueran más grandes (ver primitiva ***FGROSOR***) y estuvieran en una zona diseminados en la zona comprendida desde el centro de la pantalla 50 pasos para arriba y 50 para abajo, 50 para la derecha y 50 para la izquierda.
(o sea una zona cuadrada de lado 100 pero con el centro de la pantalla centrado, como la que se ve en la figura).



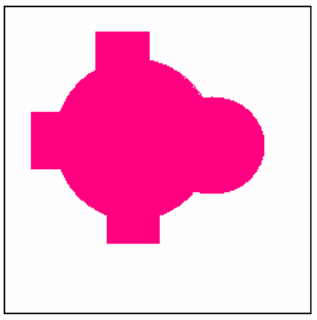
Solución en archivo CIRCULO.LGO en el disquete.

§ 5.4 .1 LA PROBABILIDAD NOS AYUDA A MEDIR AREAS

Les propongo una forma novedosa de medir áreas de figuras irregulares cerradas, por ejemplo la de la figura.

Necesitamos un cuadrado o rectángulo de área conocida que lo abarque (como vemos en la figura). Si trasladamos este dibujo - por ejemplo importándolo como bitmap - y lo colocamos en

el 1er cuadrante. Supongamos que el dibujo queda encerrado en un cuadrado (como figura de referencia de área conocida) de 100 x 100.



El método se basa en "tirar" 1000 puntos al azar en el cuadrado de referencia. Cuanto más grande es la figura desconocida, más probabilidad de que los puntos la "toquen". Si la forma ocupara todo el cuadrado el 100% de los puntos caería adentro, si la forma ocupara sola la mitad de la superficie solamente el 50% de los puntos caería adentro... y así sucesivamente.

O sea que la cantidad de puntos que caen dentro de la forma es proporcional al área a calcular!

Para saber cuando un punto cae dentro de la forma debe estar pintado (por ejemplo con color 2). En este caso podemos usar el procedimiento *puntito* (ya visto) con una pequeña modificación:

```
para puntito
fx azar 101            (1)
fy azar 101            (1)
si colordebajo = 2  hacer "n :n + 1   (2)
punto
fin
```

Si la figura no quedara encerrada en la zona 0 < x< 100 e 0 < y <100 solamente debemos aumentar uno o ambos de estos valores, por ejemplo en (1) quedaría :

```
fx azar 200
```

Luego ejecutamos el procedimiento MIDEAREA con una cantidad de repeticiones grande:

```
para MIDEAREA  :area_referencia;
bp fcf 0 ot sp
hacer "n 0
repetir 5000 [puntito]
esc frase [AREA DE LA FIGURA = ](:n  * :area_referencia )/ 5000;
fin;
```

Entonces si el cuadrado tiene 100x100 = 10.000 de área, ponemos:

```
MIDEAREA 10000
```

y nos dará el área buscada aproximada.

En el disquete puede encontrase el archivo CALCAREA.LGO y una forma en CALCAREA.PCX.

(*) Este procedimiento es válido para cálculos aproximados de cualquier figura regular o irregular.Puede comprobarse con polígonos regulares.

Incluimos también el programa ROMANO.LGO(**) que simula un bolillero con 10 bolillas: 6 de un color y 4 de otro. En este programa se compara la probabilidad teórica de salida de una bolilla de color azul con la simulación computacional práctica de la misma situación.

(**) Este programa fue desarrollado por Esteban Romano cursando Programación IV en el Instituto terciario de la Escuela Técnica Alemana Moreno.

CAPITULO 6

TRABAJANDO CON FORMAS Y DECORADOS

Vamos a explicar cómo trabajar con decorados, cómo poder utilizar el portapapeles, cómo copiar formas de decorados y llevarlos a las retículas, y algunos trucos de captura de pantallas y transformación de decorados.

§ 6.1 DECORADOS

§ 6.1.1 CARGA DE DECORADOS

Para cargar un decorado del disco se debe tener predeterminado en "configuración" el directorio a donde Logo buscará el archivo, o bien junto con la instrucción se le debe dar la ruta completa (path).

La instrucción para cargar decorados es:

```
cargardec "golf
```

o bien, con la ruta incluida:

```
cargardec "c:\logow\golf
```

Existen dos formatos en los que trabaja Logo con los decorados, uno es PCX y el otro es BMP. Esto se determina en la configuración inicial de Logo, en el menú **Archivo - Configuración**. Si en la configuración se determinó que trabajaremos en formato PCX, Logo irá a leer un archivo ***GOLF.PCX***, en cambio si estaba configurada para leer BMP buscará el archivo ***GOLF.BMP.***

Si no sabemos cuál es la configuración inicial o el programa deberá ejecutarse en otra computadora lo mejor será que le explicitemos la extensión al tratar de cargar el decorado, por ejemplo:

```
cargardec "golf.pcx
```

La ventaja de trabajar en formato PCX es que suelen ser archivos más chicos y ocupan menos lugar en el disco, y si trabajamos en 256 colores o más esa diferencia es notable.

Por otro lado, en general los archivos BMP se cargan un poco más rápido que los PCX.

NOTA:
Es muy importante aclarar que a diferencia de otros Logos, según la configuración de LOGO GRAFICO y del Windows donde se programe y ejecute un programa logo, la ruta de búsqueda de los archivos de formas, gráficos, decorados, y eventualmente, sonido y video puede variar de una máquina a otra.
Para que no tengan problemas en ejecutar un programa hecho en LOGO GRAFICO es importante ponerle la extensión a todos los archivos auxiliares que utilicen, y que, dentro del programa NO poner rutas (path).
*Una forma segura de que el programa encuentre todos los archivos es usando la primitiva **FDIR** que cambia y define donde LOGO GRAFICO buscará todos los archivos auxiliares. Sin importar la configuración inicial.*

§ 6.1.2 CARGA DE BITMAPS EN MEMORIA

En Logo Gráfico - para versiones 4.4 o posteriores - existe un recurso interesante que se puede utilizar, por ejemplo, si queremos intercambiar rápidamente dos bitmaps en pantalla.

Primero se cargan temporalmente en un lugar de memoria (llamado registro) con una primitiva llamada ***CARGARDECMEM*** (CARGAR DECorado en MEMoria), y luego al invocarlos se mostrarán mucho más rápidamente que con la orden ***CARGARDEC***, pues se ahorra el tiempo de acceso al disco rígido.

La instrucción para cargar el decorado ***golf.pcx*** en memoria (registro) es:

```
cargardecmem "golf.pcx 1
```

donde 1 indica el registro interno donde se guarda. Luego para mostrarlo usaremos:

```
cargardec 1
```

O sea que para cargar 2 decorados en memoria - por ejemplo ***ciudad.pcx*** y ***ciudad2.pcx*** - y luego alternarlos, podríamos hacer algo así:

```
para dec_en_mem
cargardecmem "ciudad.pcx   1
cargardecmem "ciudad2.pcx  2
fin

para ilumina
cargardec 2
fin

para apaga
cargardec 1
fin

para titila
repetir 10 [ ilumina  esperar 100 apaga esperar 100 ]
fin
```

Pueden probarlos buscando el archivo CIUDAD.LGO en el disquete.

También en el archivo ESTATUA.LGO del disquete se muestra el uso de este recurso para la carga consecutiva de 3 bitmaps distintos cargados en 3 registros distintos de memoria. La versión 4.4 permite usar hasta 10 registros.

Un inconveniente que puede traer el abuso de este recurso es ocupar demasiada memoria dinámica, que podría provocar el agotamiento de la misma y hacer que el sistema funcione más lento, por lo cual debe manejarse con cuidado.

Para eliminar los decorados cargados en memoria con esta primitiva se puede emplear la instrucción ***RG*** o ***BOTODO***.

§ 6.1.3 RECORTANDO DECORADOS

Tenemos la posibilidad de recortar una porción del decorado y llevarla al Editor de Formas. En el menú de Edición existen dos opciones que son:

Marcar y Copiar Bitmap.

Al activar esta opción convierte la flecha del "mouse" en una cruz. Pulsando sobre el vértice superior izquierdo del rectángulo que queremos recortar, arrastrando el "mouse" hasta el vértice opuesto (inferior derecho) y soltando el botón del "mouse", esta porción de pantalla pasa al portapapeles de Windows. Desde allí la podemos modificar con cualquier programa graficador de Windows (como el Paintbrush, Paint, etc) o sino entrando al Editor de Formas de Logo, elegimos una forma no definida y con el botón Pegar quedará grabada como una forma nueva. Si el tamaño de la porción elegida excede las dimensiones máximas de la retícula se recortará.

Otra posibilidad es utilizar esa porción de pantalla para pegarla en otra parte de la misma, o en otra que podríamos cargar luego.

Marcar y Pegar Bitmap

Al activar esta opción, la flecha del "mouse" se convierte en una cruz. De esa manera podemos señalar el lugar donde queremos pegar la porción de pantalla que tenemos almacenada en el portapapeles.

Así como podemos recortar una porción de pantalla desde Logo al portapapeles, podemos hacer la inversa, o sea desde el graficador de Windows recortar una porción cualquiera de un dibujo y a través del portapapeles llevarla a la pantalla del Logo o al Editor de Formas.

§ 6.1.4 ESTAMPANDO FORMAS

Logo tiene, además de la orden ***ESTAMPAR***, una instrucción con la cual podemos estampar formas en un lugar cualquiera de la pantalla indicándole el número de forma y la coordenada en X e Y, sin importar donde esté el actor activo. La primera (estampar) deja como si fuera un sello con la forma que tiene en ese momento el actor que está activado, mientras que con:

```
estamparforma 2 [100 35]
```

se estamparía la forma en el punto de coordenadas x =100 e y =35, sin importar dónde está en ese momento el actor activo. Un ejemplo de estampar varias formas en la pantalla gráfica sería:

```
estamparforma 2 [0 20]
estamparforma 15 [30 40]
estamparforma 21 [-30 10]
```

Para estampar 10 formas iguales a la número 1 a la misma altura y = 50 y en distintas abscisas (x) se puede hacer:

```
repetir 10 [estamparforma 1 lista azar 100 50]
```

DESAFIOS

¿Podrías, usando la primitivas ***ESTAMPARFORMA*** y ***AZAR***, estampar 33 flores sobre la misma horizontal cada 8 pasos de tortuga?

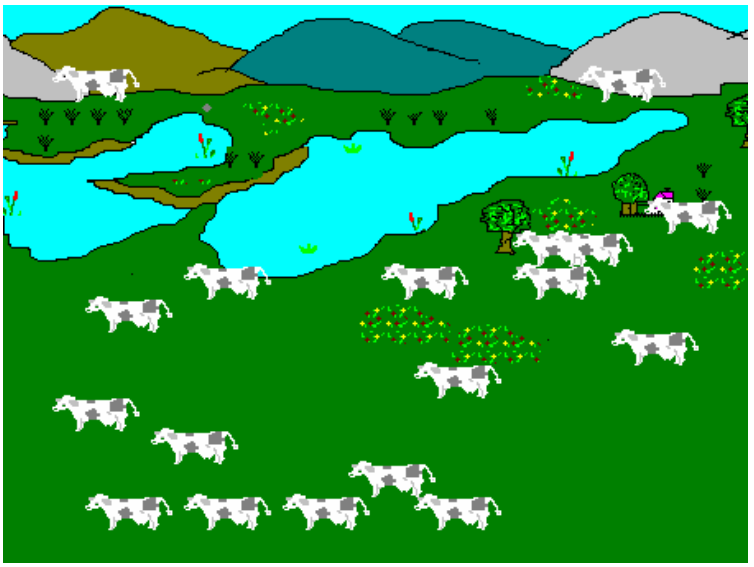
¿Y estampar 100 árboles desde y =-100 hasta y=24 y para x entre x=-150 y x=149 ?

Para consultar las soluciones buscar en los ejemplos VOLAR.LGO y SOLDADO.LGO que se entregan con el programa LOGO GRAFICO (procedimientos PARA FLORES y PARA ARBOLES)

§ 6.1.5 ESTAMPANDO PEQUEÑOS DECORADOS

Otra posibilidad muy cómoda y útil, es la de poder estampar un decorado en cualquier lugar de la pantalla, pero éstos deberían ser de un tamaño reducido.

La gran ventaja de estos recursos es que podemos cargar distintos decorados y ponerlos en diferentes lugares de la pantalla. Suponiendo que tenemos un pequeño decorado en un archivo llamado MUESTRA.PCX, para estamparlo en un lugar determinado de la pantalla la instrucción que se emplea es:



```
estamparforma "muestra.pcx [-100 90]
```

la coordenada indica el vértice superior izquierdo del decorado a estampar, no así cuando estampamos formas, pues en ese caso la coordenada indica el centro de la forma a estampar.

En la figura que ilustra esta posibilidad hemos estampado 17 pequeños decorados con forma de vaca sobre otro cargado previamente con el paisaje de fondo.

§ 6.1.6 ESTAMPANDO DESDE EL PORTAPAPELES

También existe la posibilidad de estampar en un punto cualquiera de la pantalla el contenido del portapapeles. Al igual que con un archivo la coordenada indica el vértice superior izquierdo

```
estamparforma 0 [-40 30]
```

donde el número cero le indica a Logo que es un bitmap o forma que está en el portapapeles.

Es decir que previamente se lo ha copiado desde Logo o desde otro programa bajo Windows.

§ 6.1.7 ESTAMPANDO ARCHIVOS ALMACENADOS EN MEMORIA

Como hemos visto en 6.1.2. podemos estampar un archivo o decorado cargado en memoria con la primitiva ***CARGARDECMEM***.

Cuando un bitmap se ha almacenado en el registro de memoria 1, con la instrucción:

```
estamparforma "#1 [-40 30]
```

donde #1 indica el registro de memoria donde se halla el archivo o decorado.

El símbolo # (llamado numeral) debe preceder al número del registro donde se cargó el bitmap.

También este recurso puede emplearse en el efecto conocido como "scroll" que consiste en que un bitmap de gran tamaño se deslice en sentido vertical u horizontal. (Ver archivo SCROLL.LGO). Si el bitmap fuera pequeño es más conveniente para moverlo copiarlo en una forma y luego moverlo con un actor (ver los capítulos de animación).

Es posible encontrarle utilidad en infinidad de situaciones sólo limitadas por la imaginación, como bajar y subir un telón (ver archivo TELON.LGO), transiciones entre una escena y otra, borrados de pantalla de derecha a izquierda, etc.

§ 6.1.5 GUARDAMOS LOS DECORADOS

Logo permite guardar los bitmaps realizados con actores, estampados de formas y decorados cargados previamente.

Con la primitiva ***GUARDARDEC*** (GUARDAR DECorado) Logo guarda los trazos y dibujos hechos con los actores, la paleta de colores, y los actores presentes en el momento de guardar el decorado. El formato en que lo guardará será el especificado en la configuración inicial del Logo (PCX o BMP), a no ser que se le indique la extensión y en ese caso este formato prevalecerá sobre el de la configuración.

Pueden usarse indistintamente mayúsculas o minúsculas.

```
guardardec "CASITAS
```

o bien indicando el formato BMP:

```
guardardec "CASITAS.BMP
```

El directorio donde se guardará también es el que consta en la configuración (ver el menú **Archivo-Configuración**) a menos que se le indique la ruta (path).

```
guardardec "C:\PROGRAMAS\CASITAS.BMP
```

§ 6.2 FORMAS

Vamos a ver las posibilidades de trabajo con formas, las distintas primitivas que nos permiten disfrazar actores y darles movimiento.

§ 6.2.1 CARGANDO Y GUARDANDO FORMAS

Debido a las distintas configuraciones de Windows, con respecto al uso de colores (16, 256, TrueColor, HighColor, etc.), pero fundamentalmente con 16 y 256 las formas son guardadas y/o leídas en distinto formato, es decir, para 16 colores la extensión del archivo es .FRM y para 256 o más colores la extensión del archivo es .FRC. La diferencia radica en la forma de guardar cada archivo, para 16 colores lo hace de una manera y para 256 de otra.

Cuando trabajamos en 256 colores podemos leer archivos FRM , pero no a la inversa. Por consiguiente si trabajamos en 16 colores no podemos leer archivos FRC. A partir de la versión 4.4 si se trabaja con 256 colores se puede guardar con 16 colores (adaptando la paleta aproximadamente) con el solo hecho de colocarle la extensión FRM. Esto se ha hecho así con el propósito de poder llevar un trabajo a un equipo con 16 colores, aunque el mismo se haya realizado en uno con 256 colores, siempre que se haya tomado la precaución de guardar las formas con extensión FRM.

También en una configuración de 65000 o 16 millones de colores las formas se guardan con 256 colores como máximo (o sea con extensión .FRC). Al existir distintos formatos según la configuración, resulta conveniente, colocar la extensión con que se crearon, en el programa LOGO que las utilizará, por ejemplo:

```
para inicio
rg bt fcolorf 4
cargarformas "pajaros.frm
...... etc.....
```

De esta forma si este programa se trata de ejecutar en otro equipo con 256 colores el programa buscará a PAJAROS.FRM y no PAJAROS.FRC que no existe. Si hubiéramos omitido la extensión el programa daría mensaje de error.

§ 6.2.2. TRABAJANDO CON FORMAS

La mayoría de las instrucciones relacionadas al uso de formas se explican en distintos capítulos de este libro. Ahora explicaremos algunas características que tiene la primitiva ***FFORMA*** (Fijar FORMA).

Cuando la instrucción está seguida por un número, significa que se le asigna al actor activo la forma definida con ese número. Pero existe la posibilidad de asignarle al actor activo, una lista de dos números: el primero indica el número de forma que tomará el actor cuando su rumbo sea 0 (o sea apuntando hacia arriba) y el segundo número indica la cantidad de formas que usaremos para que rote 360 grados.

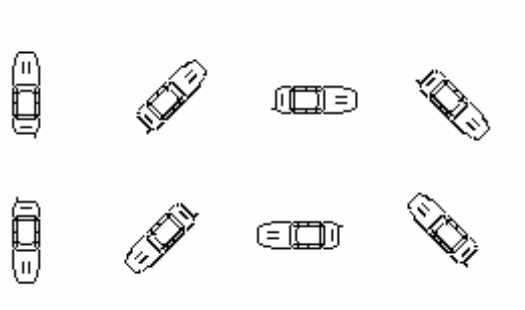
Un conjunto de instrucciones adecuadas podrían ser:

```
activar 1
cp mt
frumbo 0
fforma [10 8]
```

¿Cómo determina LOGO qué forma tendrá el actor?

El **actor 1** tomará la **forma 10** para el rumbo 0. Si el rumbo es 45, tomará la forma siguiente (la 11), con rumbo 90 tomará la forma 12, y así sucesivamente hasta completar los 360 grados. Podemos comprobarlo cambiando en rumbo al actor.

Si el rumbo del actor toma un valor intermedio, (por ejemplo: rumbo 40) éste fijará la forma más cercana al rumbo que tiene, en este caso la forma correspondiente al rumbo 45 (forma 11).



Esta función de la primitiva ***FFORMA*** es particularmente útil ya que, sabemos que los disfraces de los actores no giran junto con el actor, y por lo tanto evitamos tener que programar qué forma deberá tomar según el rumbo que tenga.

Dependerá de la cantidad de formas que estén indicadas en la lista para determinar entre qué rangos del rumbo van a girar las formas.

Es fundamental que las formas estén creadas sucesivamente a partir de la forma inicial, en el ejemplo desde la 10 a la 17, y rotando en el sentido de las agujas del reloj.

Para darle a una forma distintos rumbos, existe en el Editor de Formas la opción de menú **Corrimientos y Giros - Rotar**, en la cual al pulsar en **Rotar** aparece una ventana de diálogo donde podemos ingresar el ángulo que queremos que gire la figura. Es muy importante que la retícula donde está la forma sea cuadrada y esté en el centro de la retícula, esto es para que el efecto de giro de realice en forma pareja.

Si la forma a girar no fuese cuadrada, existe también una opción de menú **Tamaño - Hacerla Cuadrada** que modifica la retícula y la hace cuadrada, de esta manera podemos girar la forma sin inconvenientes.

CONSEJOS:

Para girar varias veces una forma, copiar la forma inicial con el botón Copiar, luego Pegar la forma en la retícula siguiente, editarla y rotarla; para la siguiente forma volver a tomar la forma inicial (no la recientemente rotada) y girarla; de esta manera no se perderá definición en la forma, ya que debido a las sucesivas rotaciones la forma se va deformando.

Es también importante que la forma se halle en el centro de la retícula, cuando utilizamos la opción **Hacerla Cuadrada**, Logo genera una retícula cuadrada, pero no mueve la forma, sino que ésta queda en la posición inicial. Lo que se debe hacer es ir corriéndola hasta ponerla en el centro de la retícula. Para versiones 4.4 en adelante al hacerla cuadrada la centra automáticamente.

Veremos un ejemplo sencillo; si tenemos la forma inicial en 10 y hemos rotado la forma en sucesivas 7 cuadrículas, 45 grados cada una, tendremos:

```
para giros
activar 1
frumbo 0
fforma [10 8]
mt
cp repetir 8 [ad 50 esperar 100 de 45]
fin
```

Veremos que el actor al cambiar el rumbo cambiará de forma y siempre apuntará en la dirección que avanza cuando construye el octógono. Para esto debemos tomar alguna forma y Editarla para que cada forma esté girada 45 grados. (Ver la figura con los 8 autos girados).

CAPITULO 7

ANIMACION SIMULTANEA

OTRA FORMA DE HACER ANIMACION

En este capítulo trataremos de explicar con ejemplos sencillos el uso de primitivas en animación simultánea, y cómo poder generar programas más complejos a través de los ejemplos aquí citados.

Sólo se nombran las primitivas más usadas en animación y las primitivas para el control de eventos en animación simultánea, pudiendo encontrarse la totalidad de éstas en la Ayuda de Primitivas del programa.

Nota Importante:

La mayoría de las formas citadas en los ejemplos sencillos corresponden al archivo NORMAL.FRM, pero si el usuario no posee dicho archivo (que acompaña al software LOGO GRAFICO), deberá dibujar las formas en el Editor de Formas o buscar en el archivo NORMAL.FRM la que más se parezca al ejemplo.

§ 7.1 INICIANDO UN MOVIMIENTO

Al igual que en la animación consecutiva primero se debe presentar a un actor con un procedimiento a tal efecto, por ejemplo:

```
para camión
decir 1          (1)
fforma 6         (2)
fx (-140)        (3)
fy 60            (4)
visible          (5)
fin
```

(1) Llama al Actor 1 (puede ponerse ACTIVAR 1)
(2) Le Fija la FORMA número 6 que es un camión, en caso de no existir la podrá hacer usted mismo
(3) Fija la X a la izquierda de la pantalla
(4) Fija la Y en la zona superior de la pantalla
(5) Recién ahora se visualiza pues antes estaba oculto o invisible (puede usarse la primitiva MT que se usa también con la tortuga)

Para lograr que se mueva debemos hacer un procedimiento que le determine la dirección ***(FRUMBO)*** del movimiento y la velocidad, por ejemplo:

```
para salir
decir 1
frumbo 90
fvel 20
actuar
fin
```

La primitiva ***ACTUAR*** desencadenará el movimiento programado con las características fijadas; en este caso sólo se le ha fijado la velocidad, que por defecto, es constante. Cuando el actor llegue al extremo derecho de la pantalla - como está en modo ***JAULA*** - rebota invirtiendo el sentido del movimiento.

Esto podemos verlo si ejecutamos los procedimientos ***camion*** y ***salir***. Para detener la animación simultánea debemos pulsar la tecla ESC o pulsando el botón Pare de las barras de herramientas.

La velocidad que toma el actor depende de la velocidad del microprocesador y podía ajustarse con FRETARDO; ahora, después de la versión 4.3 se fija por defecto con un valor calculado internamente según la velocidad de la computadora.

Si quisiéramos que no se mueva indefinidamente sino sólo durante un corto tiempo podemos usar la orden ***MOVERSE*** seguida del tiempo que deseamos que lo haga, p. ej. 2,5 segundos. En ese caso el procedimiento quedará:

```
para salir2
decir 1
frumbo 90
fvel 20
moverse 250
fin
```

El número que acompaña a MOVERSE está en centésimas de segundo, o sea que 100 son 1 segundo, 200 son 2 segundos y así sucesivamente.

Aquí veremos que si ejecutamos:

```
camion  salir2          <ENTER>
```

se detendrá luego de los 2,5 segundos saliendo sola de la animación simultánea.

DESAFIO

¿Con ***FVEL*** 40 cuánto le corresponde al argumento de ***MOVERSE*** para recorrer aproximadamente el mismo camino?.

§ 7.2 DOS VEHICULOS

Si queremos que se muevan 2 o más actores, debemos hacer la presentación del segundo como hicimos con el camión. Tomaremos la figura 8 que representa un auto y lo haremos desplazarse de derecha a izquierda hasta chocar con el camión.

Suponiendo que estamos con la pantalla mixta, para ver la forma del automóvil vamos al Menú de Formas, con ALT + F o haciendo clic sobre la palabra **Formas** de la Barra de Menú o sobre el botón correspondiente de la Barra de Herramientas. Luego, de ese menú elegimos **Editar Formas** haciendo clic con el mouse u oprimimos las teclas ALT + E.

Ubicando la barra iluminada sobre la forma indicada veremos la forma de un auto que mira hacia la derecha.

Haciendo clic con el mouse en el botón **Editar** entramos al **Editor de Formas.** Como queremos que se dirija hacia la izquierda elegiremos en el menú **Simetrías - Horizontal**, que

le producirá una transformación simétrica horizontal (o sea, con eje Vertical- y el auto quedará mirando hacia la izquierda. Saldremos del editor aceptando el cambio de forma.

Si no deseamos realizar el cambio al salir del editor de formas NO aceptamos los cambios y la forma quedará sin cambiar. Nótese que aceptar el cambio de forma no implica que esa modificación quede guardada en el disco, sino en la memoria. Antes de terminar la sesión de trabajo deberá guardarse el archivo de formas en el disco.

Nota: En caso de no coincidir los números de formas, busque alguna forma equivalente en el archivo NORMAL.FRM provisto en el disquete del lenguaje.

Ahora ubicaremos el automóvil a la misma altura que el camión - Y = 57, 3 pixels más abajo para que las ruedas queden al mismo nivel, ya que las coordenadas se miden respecto del centro de la forma y el camión es más grande - con un procedimiento:

```
para auto
decir 2
fforma 8
fx 130
fy 57
visible
fin
```

El procedimiento ***salir3*** será ahora:

```
para salir3
decir 1
frumbo 90
fvel 20
decir 2
frumbo 270
fvel 20
moverse 200
fin
```

Este valor de 200 (equivalentes a 2 segundos) lo elegimos bastante arbitrariamente ya que lo iremos ajustando con el método de ensayo y error, para que los vehículos se detengan al encontrase. Este orden indica a todos los actores que tengan velocidad asignada que se empiecen a mover con las condiciones estipuladas.

Nota: La velocidad en tiempo real en la animación simultánea disminuye a medida que hay más actores en escena.

Para reunir los procedimientos en uno solo se usa el superprocedimiento ***escena***

```
para escena
auto
camión
salir3
fin
```

§ 7.3 PISANDO EL ACELERADOR

Otro interesante recurso de la animación simultánea es la posibilidad de acelerar los móviles, pudiendo inclusive simular muy fácilmente situaciones de movimiento como las que ocurren realmente en la Tierra, es decir con la atracción de la gravedad.

Acelerar un móvil significa cambiarle paulatina y constantemente su velocidad.

Un cuerpo que se mueve con velocidad constante - es decir que la velocidad no cambia - tiene aceleración nula (o sea cero).

Primero veremos una aceleración en el sentido de un movimiento horizontal.

Las órdenes para acelerar un móvil son ***FACELX*** y ***FACELY*** que se leen Fijar ACELeración en X y Fijar ACELeración en Y. Un móvil como el que define el procedimiento ***camion*** puede acelerarse si combinamos dicho procedimiento con otro que le dé valor a ***FACELX***, por ejemplo:

```
para acelerar
camion
frumbo 90
fvel 0
facelx 0.3
moverse 250
fin
```

De esta manera veremos acelerar al camión durante 2,5 seg. aproximadamente, aumentando su velocidad desde VEL= 0 hasta VEL=36 aproximadamente.

También puede ocurrir que nuestro camión rebote con el extremo derecho de la pantalla. Si no queremos que eso ocurra podemos darle la orden LIBRE que extiende el campo de movimiento hasta 3 veces para cada costado. En este caso si el actor sobrepasa el campo posible (X mayor que 480) aparecerá el mensaje: ***ACTOR (número de actor) FUERA DE SUS LIMITES.*** *La orden contraria a LIBRE es JAULA.*

Si en cambio le asignamos una aceleración contraria al movimiento, partiendo de una velocidad de 35 tendremos:

```
para frenar
camion
frumbo 90
fvel 35
facelx (-0.25)
moverse  250
fin
```

Los valores negativos se colocan siempre entre paréntesis.

Nota: En este caso puede ocurrir que luego de frenarse el camión salga para atrás. Eso ocurrirá si la aceleración que se opone al movimiento actúa durante bastante tiempo (probar con un argumento para MOVERSE de 300, 350 , etc.) o la aceleración toma un valor más negativo - o sea negativo y con mayor valor absoluto - como -0.5 ó -0.8. Un movimiento como éste se llama retardado o desacelerado.

§ 7.4 PATEANDO UNA PELOTA

Si la aceleración la colocamos en el eje Y (vertical) y colocamos un objeto pequeño - como la forma de una pelota.

```
para pelota
decir 1
fx (-130)
fy 60
```

```
fforma 16
visible
fin

para patear
jaula
decir 1
frumbo 90
fvel 15
facely (-0.5)
actuar
fin
```

Coloquémonos en la pantalla mixta y demos la orden ***pelota*** pulsar <Enter>, ***patear*** pulsar <Enter>, y veremos la pelota saltar rebotando contra los bordes de la pantalla y el piso (representado por la barra que separa la pantalla de texto con la gráfica). Para interrumpir la animación debemos pulsar la tecla ESC o pulsando el botón PARE de las barras de herramientas.

DESAFIO

Probemos colocar en la penúltima línea de ***patear*** la orden ***FFRICCION*** 2, luego cambiemos por ***FFRICCION*** 4 y también probemos ***FFRICCION*** 8. Podemos investigar ¿Con qué valor no llega al extremo derecho?

La primitiva FFRICION se lee Fijar FRICCION y significa que la simulación de movimiento experimentará - como ocurre en la realidad - una pérdida de velocidad debido al rozamiento entre dos superficies o entre un objeto y el aire (en física se habla de pérdida de energía cinética).

DESAFIO

¿Qué orden debemos colocar para que cada vez que toca el piso se escuche un sonido corto? (ayudarse con el ejemplo de 2.3 o consultar el manual)

Un ejemplo del uso de fricción es el siguiente programa, el cual está incluido en el disquete provisto con el nombre de CIUDAD.LGO

```
para inicio
bp ot
cargardec "ciudad.pcx
cargarformas "ciudad.frc
cam
cart
andar
subir
fin
para cam
decir 1 mt fforma 29 sp
fx (-130) fy (-18)
frumbo 90 fvel 0
fin
para cart
decir 2 mt fforma 39 sp
fx (-140) fy (-14)
```

```
frumbo 90 fvel 0
fin

para andar
decir 1 fvel 11 ffriccion 1
decir 2 fvel 11 ffriccion 1
moverse 400
pedir [1 2] [fvel 0]
fin

para subir
decir 2 frumbo 0 ffriccion 0
fvel 8
moverse 140
fin
```

§ 7.5 USO DE LAS PRIMITIVAS PEDIR Y TODOS

Vamos a ver ahora varias primitivas que nos serán muy útiles a la hora de mover varios actores a la vez.

Cuando quiero dirigirme a varios o todos los actores simultáneamente lo puedo hacer con

```
PEDIR [1 2 3] [Lista de órdenes o procedimientos]
```

Existe la opción de dirigir nuestras órdenes a todos los actores al mismo tiempo y es:

```
PEDIR TODOS [Lista de órdenes o procedimientos]
```

Nota: En esta versión no se puede usar la combinación de órdenes ACTIVAR TODOS como se usaba en MSX LOGO. Por otro lado la orden PEDIR sirve para que cuando nos dirigimos a un actor A y queremos dirigirnos momentáneamente a otro actor B, luego automáticamente seguimos hablándole al actor A sin necesidad de escribir ACTIVAR A.

La primitiva TODOS es una función que equivale a la lista [1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20]. En realidad si el LOGO GRAFICO está ejecutándose en un equipo Pentium o más rápido el total de actores en versiones posteriores a la 4.4 es de 60, entonces la función TODOS no representa la totalidad de los mismos.

§ 7.6 EVENTOS

El recurso más poderoso e interesante de la animación simultánea son los eventos. Un evento es una condición que se programa, y que cuando ésta se cumple, una orden, lista de órdenes o procedimiento programados se ejecutará.

§ 7.6.1 CONTROLANDO EL CHOQUE

Si queremos que cuando los actores se toquen se detengan:

```
DECIR 1 CUANDOCHOCA 2  [ INTERRUMPIR ]
```

La orden ***INTERRUMPIR*** detiene el modo de animación simultánea.

O bien cuando el camión toque el lado izquierdo de la pantalla que invierta la forma.

```
DECIR 2 CUANDOIZQ [ FFORMA 7 ]
```

En este caso la forma 7 deberá haber sido creada utilizando el recurso de simetría para dar vuelta el camión.

Para programar los eventos existen las primitivas CUANDO que son 13 (*) y que podemos consultarlas en la Ayuda de Primitivas por Grupo (Animación Secuencial y Simultánea).

(*) La lista completa de eventos es: CUANDOABAJO, CUANDOARRIBA, CUANDODER, CUANDOIZQ, CUANDOBOTON, CUANDOACTOR, CUANDOBLOQUE, CUANDOCHOCA, CUANDOPASEN, CUANDOPISACOLOR, CUANDOTECLA, CUANDOX y CUANDOY.

En el caso que estamos viendo podemos modificar el procedimiento ***salir3*** quedando así:

```
para salir4
decir 1
frumbo 90
fvel 20
decir 2
frumbo 270
fvel 20
decir 1
cuandochoca 2  [ interrumpir ]
actuar
fin
```

Nos dirigimos al actor 1
Y le programamos que cuando choque con el actor 2 interrumpa la animación simultánea.
Tanto los movimientos como los eventos no funcionan si no damos la orden ACTUAR.

Como opcional si queremos que produzca un sonido al chocar se puede agregar dentro de los corchetes y antes de interrumpir la orden:

```
TONO 300 20
```

con lo que la línea quedará:

```
CUANDOCHOCA 2 [TONO 300 20 INTERRUMPIR]
```

Para terminar esta sesión de trabajo guardamos los procedimientos del Editor con ALT + A y pulsamos la letra G o picando con el mouse en **Archivos - Guardar...**. Colocamos un nombre, por ejemplo CHOQUE, en la ventana de diálogo de Guardar Archivos, le determinamos el directorio deseado y picamos en Aceptar o pulsamos el Enter.

§ 7.6.2 Evento CUANDOPASEN

Este evento permite ejecutar una lista cada un determinado período de tiempo. Es ideal para repetir algo dentro de una animación simultánea. Siempre es necesario haber ejecutado la orden ***ACTUAR*** o ***MOVERSE***.

Por ejemplo si deseamos que un actor cambie de rumbo cada 1/2 segundo (50 centésimas de segundo) girando 60 grados y haciendo un sonido por cada giro...

```
CUANDOPASEN 50 [ DERECHA 60 TONO 1000 20]
```

Podemos probarlo con este procedimiento.

```
para hexago
decir 6
mt
fvel 20
cuandopasen 50 [ derecha 60 tono 1000 20]
actuar
fin
```

También puede usarse para que realice algo por una sola vez desactivándose el evento luego, por ejemplo en:

```
para unavez
decir  5
visible
fvel 15
cuandopasen 150 [estampar desactivar]
actuar
fin
```

Probar la diferencia si no se pone ***DESACTIVAR***.

Esta primitiva desactiva el evento luego de cumplir por primera vez la orden, por lo que si ponemos las órdenes invertidas [***DESACTIVAR ESTAMPAR***] no altera el efecto.

§ 7.6.3 Evento CUANDOACTOR

Este evento es similar a ***CUANDOPASEN*** pero dirigido a un determinado actor, por ejemplo si hay dos actores (1 y 2) moviéndose en la pantalla y ordenamos:

```
decir 1
cuandoactor 50 [estampar]
decir 2
cuandoactor 25 [estampar]
```

las órdenes de estampar quedarán dirigidas a cada uno de ellos repitiéndose cada 50 y 25 centésimas de segundo para los actores 1 y 2 respectivamente.

*Se podrán listar los eventos a través del menú **Ejecución - Listar Eventos...** donde aparecerán qué eventos tiene asignados cada actor.*

§ 7.6.4 Evento CUANDOTECLA

Este evento permite ejecutar una lista cuando se toca una tecla cualquiera. Siempre es necesario haber ejecutado la orden ***ACTUAR*** o ***MOVERSE***.

Si damos velocidad a un móvil con:

```
decir 1
```

```
fforma 12 visible
fvel 20
```

Y luego programamos el evento

```
CUANDOTECLA [ DERECHA 90]
```

si ahora entramos en el modo de animación simultánea con

```
actuar
```

podremos verificar que cada vez que tocamos una tecla se cumple que el actor 1 girará 90 grados.

La orden ***CUANDOTECLA*** puede usarse en combinación con la función ***LEERCAR*** con lo que puede verificarse qué tecla ha sido oprimida.

*Ante todo debemos saber que LEERCAR o LC que debe leerse **Leer Carácter** comprueba si una tecla se ha pulsado y verifica cuál es esa tecla.*

Por ejemplo:

```
para comandar;
decir 10 ;
visible;
frumbo 0;
cuandotecla [si leercar = 0 entonces sp sino cp];
actuar;
fin;
```

Nota: La palabra entonces se puede obviar sin que cambie el programa, se usa para mejorar la comprensión del procedimiento. Las palabras si...entonces ...sino se han colocado en minúsculas solamente para resaltar la sintaxis, pero pueden usarse en mayúsculas con igual resultado.

Complicando un poco el procedimiento:

```
para comandar2;
decir 6 ;
visible;
frumbo 0;
cuandotecla [hacer "tecla leercar  si :tecla = 0 entonces sp sino
fc :tecla];
actuar;
fin;
```

con lo que logramos cambiar el color del trazo con cada tecla de números que pulsemos. El procedimiento se cortará por error si pulsamos alguna tecla con letras, ya que la primitiva ***FCOLOR*** no aceptará una letra por argumento, apareciendo el mensaje:

No se sabe qué quiere decir...

§ 7.6.4.1 USAR LAS FLECHAS PARA COMANDAR UNA NAVE

Para programar que con las teclas de dirección (flechas) que un actor en animación simultánea cambie de rumbo podemos hacer un programa de presentación ***nave*** y otro ***dirigirnave*** utilizando la primitiva ***CUANDOTECLA***, como los que siguen:

```
para nave
decir 1  fforma 12
visible
sp centro
frumbo 90
fin
```

Verificar que la forma 12 sea una nave, sino diseñarla

```
para dirigirnave
fvel 20
cuandotecla [ mando]
fin

para mando
hacer "tecla  lc
si :tecla = "flechaarriba frumbo 0    (1)
si :tecla = "flechader frumbo 90
si :tecla = "flechaabajo frumbo 180
si :tecla = "flechaizq frumbo 270
fin
```

(1) La primitiva LEERCAR devuelve los nombres de las teclas direccionales (al igual que las de función, escape, enter, tab, home, insert, end, delete, pagedown, pageup van en minúsculas). Estas palabras deben usarse con minúsculas.

```
para inicio
nave
dirigirnave
actuar
fin
```

IDEA: UNA EXPLOSION

Mostraremos a continuación un ejemplo de uso de la programación de las teclas para animación simultánea.

Nos proponemos hacer un programa que simule una explosión de 20 partículas (o formas) iguales que luego desaparezcan al llegar cerca del los bordes de la pantalla.

Para ello cargaremos las formas EXPLOSIO.FRM del disquete. Se encuentra allí en la Forma 1 una imagen de una pequeña piedra.

Podríamos programarla simplemente usando la primitiva ***moverse*** que hace que luego de un tiempo medido en centésimas de segundo se interrumpa la animación simultánea.

```
para explosio1
bp libre
```

```
cargarformas "explosio
fcf 0
pentera
pedir todos [fforma 1]
pedir todos [centro mt frumbo quien * 18 ]
pedir todos [fvel 100]
moverse 150                          (1)
pedir todos [desaparecer ]           (2)
fin
```

Este programa está en el disco como EXPLOSIO.LGO

(1) Entra en modo animación simultánea durante 150 centésimas de segundos.
NOTA: Este efecto es similar - pero más simple - que usar ***actuar*** *y luego con* ***cuandopasen 150 [pedir todos [desaparecer interrumpir]*** *(verificarlo)*
(2) Luego de 150 unidades de tiempo los actores desaparecen.

```
para explosio2;
bp libre
cargarformas "explosio
fcf 0
pentera
pedir todos [fforma 1]
pedir todos [centro mt frumbo quien * 18 ]
cartel [PULSE UNA TECLA] [ 0 90] 5   (3)
cuandotecla [ salir]                 (4)
actuar
fin
```

(3) Podemos usar un cartel en la parte superior de la pantalla para indicar que debe pulsarse una tecla para comenzar el movimiento.
(4) Llama al procedimiento salir que les asigna velocidad muy parecida a la orden (2) del procedimiento ***explosio1.***

```
para salir
pedir todos [fvel 100 ]
cuandopasen 150 [pedir todos [desaparecer interrumpir]]
fin
```

§ 7.6.5 Evento CUANDOPISACOLOR

Vamos a ver un ejemplo del uso de la primitiva ***CUANDOPISACOLOR***. Dibujaremos una figura cuadrada girada 45 grados con la tortuga con color verde (color 2) y la rellenamos con relleno pleno (1) con el mismo color 2.

```
para cuadri
bp
sp
fpos [40 0]
fc 2
cp frumbo 315
frelleno 1 2
repetir 4 [ad 50 de 90]
rellenar
fin
```

Además lanzamos una pelota (buscamos la forma de una pelota) desde el centro con velocidad 40. Además le programamos que al pisar el color 2 gire 180 grados con la primitiva ***CUANDOPISACOLOR***.

```
para pelota
decir 6 fforma 16 mt
fvel 40 frumbo 45
cuandopisacolor 2 [de 180]
actuar
fin
```

Finalmente probamos el efecto ordenando:

cuadri pelota <Enter>

Veremos la pelota rebotando con los bordes del cuadrilátero.

DESAFIO

Programar una circunferencia grande con borde de grosor 5 y lanzar desde el centro una pelota que rebote dentro.

§ 7.6.6 Evento CUANDOBLOQUE

Vamos a dibujar dos bloques con la primitiva ***DEFBLOQUE*** que se usa así:

```
DEFBLOQUE :NumDeBloque [X e Y de los vértices sup. derechos]
:Ancho :Alto
```

El parámetro o variable **:NumDeBloque** es un número correlativo que debemos empezar con 1 y seguir con 2, 3, etc. hasta 10 para cada bloque que queramos definir. En este ejemplo usaremos 2 solamente.

```
para bloques
bp ot pentera
frumbo 90 fgrosor 1
defbloque 1 [20 70] 30 50                (1)
sp fpos [20 70]
frelleno 1 6                             (2)
cp repetir 2 [ad 30 de 90 ad 50 de 90]   (2)
rellenar                                 (2)
defbloque 2 [-60 70] 30 50
sp fpos [-60 70]
frelleno 1 13
cp repetir 2 [ad 30 de 90 ad 50 de 90]
rellenar
fin
```

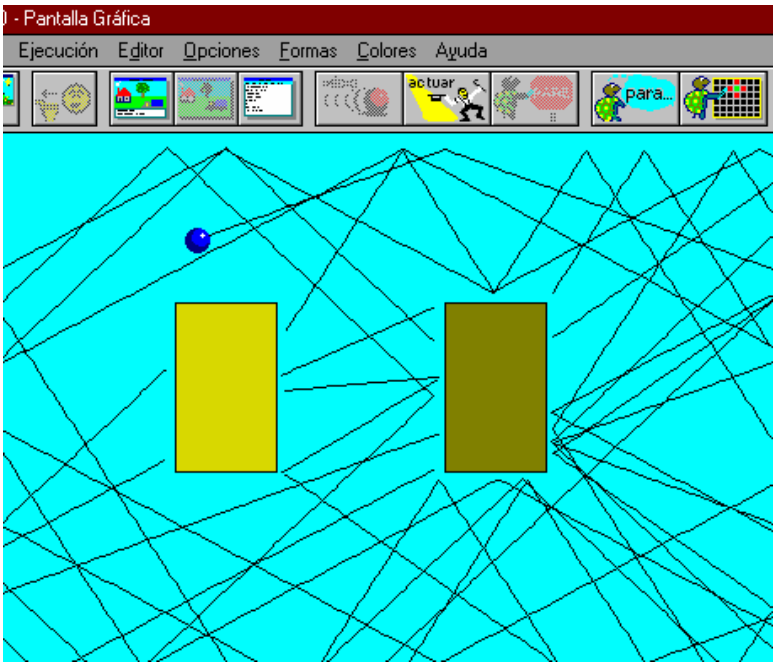
En (1) definimos el bloque con un tamaño de 30 de ancho por 50 de alto, los bloques se definen en forma lógica y no gráfica, por lo tanto en (2) debemos dibujar un rectángulo en el mismo lugar y del mismo tamaño que el bloque lógico. Esto es igual para los dos bloques.

Programamos una pelotita que al chocar en el rectángulo (bloque) 1 rebota y activa la pluma ***(CP)*** y que al chocar con el otro rectángulo se queda sin pluma ***(SP)***.

```
para pelota
decir 1
fforma 16          (*)
fpos [0 0]
mt fc 0
fgrosor 1
frumbo 45 + azar 270
fvel 20
cuandobloque 1 [rebotar cp]
cuandobloque 2 [frumbo azar 360 rebotar sp]
actuar
fin
```

(*) Deberemos dibujar una pelota en la forma 16.

```
para inicio
bloques
pelota
fin
```



DESAFIO

¿Cómo le agregarías un sonido corto y agudo al chocar en el bloque 1 y largo y grave al chocar con el 2?

DESAFIO

Hacer variantes con 3 y 4 bloques. Que cambien de color en cada bloque.

Existen cuatro primitivas que tiene incidencia directa con los bloques, que son: **ladoarriba?**, **ladoabajo?**, **ladoder?**, **ladoizq?** que sirven para programar eventos más específicos con respecto a donde toca el actor un bloque, y poder ejecutar un procedimiento diferente, según en qué lado del bloque chocó el actor.

§ 7.6.7 Eventos CUANDODER, CUANDOIZQ, CUANDOARRIBA y CUANDOABAJO

El poder controlar el movimiento de los actores cuando chocan con los bordes del monitor en animación simultánea, resuelve muchos problemas de programación cuando queremos que se ejecute un procedimiento en ese instante.

Para eso contamos con cuatro primitivas que nos permiten controlar independientemente a un actor cuando choca con algún borde de la pantalla gráfica.

CUANDOARRIBA: Cuando el actor choca con la parte superior de la pantalla gráfica.

CUANDOABAJO: Si estamos con la pantalla completa o entera se produce el evento al chocar el actor con la parte inferior del monitor, si en cambio estamos en pantalla mixta con la barra de separación entre la pantalla gráfica y la de texto, el evento se produce al chocar el actor con ésta.

CUANDOIZQ: Cuando el actor choca la parte izquierda de la pantalla gráfica.

CUANDODER: Cuando el actor choca con la parte derecha de la pantalla gráfica.

Ahora veremos algunos ejemplos de programación usando estos eventos.

```
para frenar
decir 1
fforma 5
frumbo 90
mt
sp fvel 20
cuandoder [interrumpir]
actuar
fin
```

Cuando el actor choque con el borde derecho se interrumpe la animación. Si lo modificamos un poquito para lograr otro efecto... (primero haremos un programa de presentación del actor para simplificar los ejemplos).

```
para presen
decir 1
sp fpos [0 0]
fforma 7
frumbo 0
fvel 20
mt
fin

para cambio
presen
```

```
cuandoarriba [frumbo 120]
cuandoizq [interrumpir]
actuar
fin
```

DESAFIO

¿Te animás a predecir que sucederá?

DESAFIO

¿Cómo harías para que cuando choca con el lado inferior de la pantalla haga un sonido?

DESAFIO

¿Y si quisiera que cambie de forma cuando choca con el lado derecho?

DESAFIO

Este es bien difícil. Programa todos los ejemplos en un solo procedimiento, además que cuando choque con un lado tenga pluma, en otro se la saque, en otro le cambie el color, y por último que puedas interrumpir la animación cuando pulses una tecla. ¿Te animás?

§ 7.7 USO DE LA PRIMITIVA QUIEN

Por ejemplo, si queremos hacer un baile de 20 flores, con el Editor de Formas diseñamos una flor en la cuadrícula 1, y luego programamos:

```
para baile
bp
pentera                                    (1)
pedir todos [fforma 1 frumbo quien * 18]   (2)
pedir todos [visible]                      (3)
pedir todos [fvel 20]
actuar
fin
```

(1) Se ve la pantalla entera.
(2) Le da a cada actor un rumbo distinto, 18 grados girado del anterior.
(3) Todos se hacen visibles.

*La primitiva QUIEN **entrega** el número de actor al que se dirige o actor activo.*
O sea que:

*el actor 1 recibe la orden frumbo 1 * 18 = frumbo 18*
*el actor 2 recibe la orden frumbo 2 * 18 = frumbo 36*
*el actor 3 recibe la orden frumbo 3 * 18 = frumbo 54*
*el actor 4 recibe la orden frumbo 4 * 18 = frumbo 72*

Queda al lector deducir los rumbos para los actores 5, 6, 7, ... y 19.
*Y el actor 20 recibe la orden frumbo 20 * 18 = frumbo 360*
Este rumbo coincide con el de 0 grados, es decir apunta hacia arriba.
Es así que los 20 actores quedan dirigidos radialmente (desde el centro hacia afuera) con rumbo diferentes en 18 grados entre cada uno.

Idea

Probar de intercalar una línea que diga:

```
CUANDOTECLA [PEDIR TODOS [ESTAMPAR]]
```

antes de ***ACTUAR***. Luego pulsar una tecla mientras se ejecuta el programa.
Probar de mantener una tecla oprimida mientras las flores se desplazan.

Idea

Probar cambiando la línea de la idea anterior por:

```
PEDIR TODOS [ CP FGROSOR 3 FCOLOR QUIEN]
```

Idea

¿Y si agregamos, además de la línea anterior la que sigue?

```
CUANDOTECLA [PEDIR TODOS [FC COLOR + 1]]
```

§ 7.7.1 OTRA APLICACION USANDO QUIEN Y AZAR

Una buena aplicación para trabajar con varios actores sería programar una carrera de veleros.

Intentémoslo...

Primero dibujaremos en el editor de formas 3 veleros en las formas 1, 2 y 3 con los números 1, 2 y 3 dibujados en correspondencia con el número de actor.

Luego ubicaremos los tres a la izquierda de la pantalla con:

```
para ubicarveleros
pedir [1 2 3 ][ ubicarcadauno ]
fin

para ubicarcadauno
fforma quien
fx (-130)
fy quien * 25
visible
frumbo 90
fvel 6
fin
```

Nótese que con fy quien * 25 logramos que los tres veleros estén en coordenadas y distintas.
Si hubiéramos puesto ***fy quien*** estarían los 3 actores en las coordenadas
y = 1
y = 2
y = 3 , o sea, casi superpuestos,

pero con esta orden estarán en y = 25, y = 50 e y - 75 respectivamente.

Sugerimos probar este procedimiento y cambiar el quien * 25 por quien * 10 , quien * 40 , etc. y ver los resultados.

Ahora trataremos de que tengan distintas velocidades al azar, por ejemplo entre 3 y 5.

Esto se lograría con:

```
para cambiarvel
pedir [1 2 3 ] [ fvel 3 + azar 3]
fin
```

Además sería mucho más interesante que la velocidad de cada velero fuera cambiando a medida que transcurre la cerrera, ya que sino no sería muy divertido pues desde que salen sabríamos quien va a ganar.

Esto se logra con:

```
cuandopasen 100 [cambiarvel]
```

que cambiará la velocidad cada 1 segundo (recordar que ***CUANDOPASEN*** se acompaña del tiempo en centésimas de segundo por lo que 100 es 1 segundo.

No conviene que varíen mucho ya que por ejemplo con ***fvel*** 3 + ***azar*** 10 los cambios serían muy abruptos.

También podríamos hacer que los veleros vayan cada vez más rápido con:

```
fvel vel + azar 3
```

en vez de:

```
fvel 3 + azar 3
```

En este caso le estamos diciendo que se fije qué velocidad tiene y le sume el resultado que nos da el azar (entre 0 y 2).

Hasta ahora el programa ***inicio*** estaría así:

```
para inicio
rg
fcf 1                        () ponemos fondo azul
ubicarveleros
cuandopasen 100 [cambiarvel]
fin
```

Nos falta ahora lograr que se termine la carrera al llegar el primero:

```
pedir [1 2 3] [cuandoder [interrumpir]]
```

Que vigila a los tres interrumpiendo la animación simultánea al llegar uno de ellos al borde derecho de la pantalla.

Notemos que si no ponemos ***pedir*** [1 2 3] sólo vigilará a uno de ellos, el que esté activo en ese momento.

Finalmente sería interesante que el programa nos diga quien ganó y ponga un mensaje en la pantalla.

Esto lo logramos con:

```
para llegada
cuandoder [bt esc quien interrumpir]
fin
```

O usando la primitiva ***CARTEL*** (ver ayuda de esa primitiva)

```
para llegada;
cuandoder [cartel fr [GANO EL ] quien [0 50] 3 interrumpir];
fin;
```

El programa ***inicio*** quedaría aproximadamente así:

```
para inicio
cargarformas "carrera
rg bt fcf 1
ubicarveleros
cuandopasen 100 [cambiarvel]
pedir [ 1 2 3 ] [llegada]
actuar
fin
```

El programa entero con el archivo de forma puede verse en el disquete bajo el nombre de CARRERA.LGO y las formas CARRERA.FRM

§ 7.7.2. CUANDO 20 DIBUJAN A LA VEZ
Muchas manos en un plato...

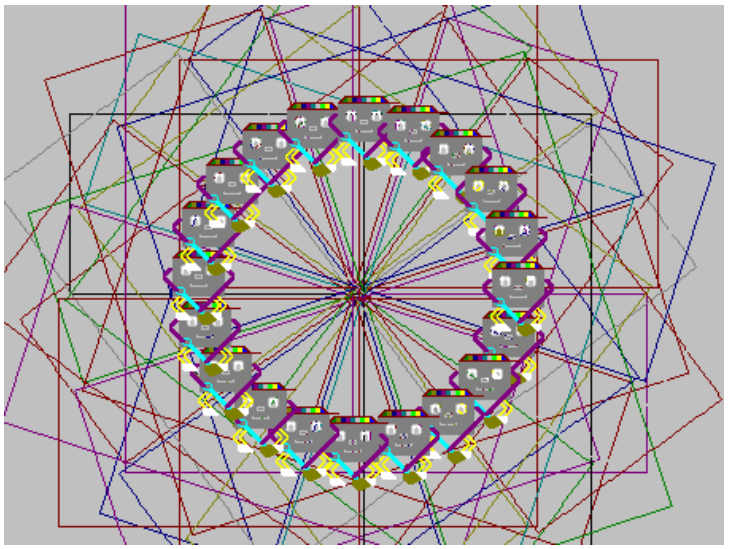
Ahora haremos una coreografía para 20 actores que les haga salir desde el centro radialmente con la misma velocidad como la explosión anterior, pero cada vez que toquemos una tecla empezarán a girar hacia la derecha simulando un especie de baile.

(Este caso está programado para la versión 4.0 o superior y está en el disco con el nombre COREOGRA.LGO).

```
para baile
bp
fcolorf 14
libre                                   (1)
pentera
pedir todos [unobaila]                  (2)
cuandotecla [pedir todos[derecha 90]]   (3)
actuar
fin

para unobaila
```

```
fforma 10 mt
frumbo quien * 18
fgrosor 1
cp fcolor azar 16
fvel 10
fin
```

(1) Permite que los actores salgan de la pantalla (ver manual).
*(2) La instrucción **pedir todos** activa momentáneamente a todos los actores y ejecuta la lista que le sigue.*
(3) Al tocar una tecla cualquiera giran hacia la derecha 90 grados. Precisamente debemos pulsar una tecla cada vez que los actores se estén por salir de la pantalla para que sigan bailando.



Si reemplazamos la línea (3) de ***para baile*** por:

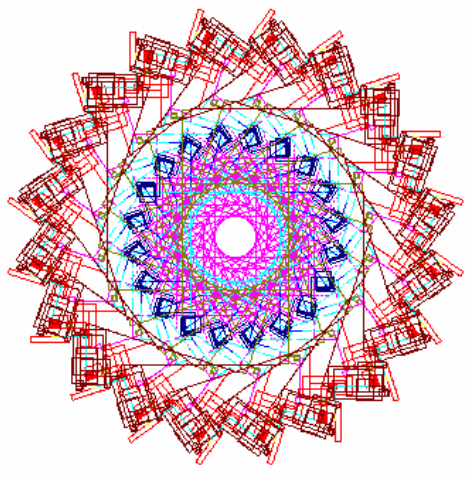
```
cuandotecla [quetecla]
```

y programamos los procedimientos ***quetecla*** y ***arrancar*** como está más abajo, este grupo de procedimientos permitirá crear con las teclas una serie de coreografías que inclusive se pueden graficar (interesantes diseños como los que se ven en la figura) si pulsamos la tecla "1" , que le activa el trazado.

```
para quetecla;
hacer "tecla lc
si :tecla = 1 pedir todos [cp ];
si :tecla = 2 pedir todos [sp ];
si :tecla = 3 pedir todos [de 180];
si :tecla = 4 pedir todos [de 90];
si :tecla = "c hacer "col azar 1+ azar 16 pedir todos [fc :col];
si :tecla = "espacio pedir todos [frumbo quien * 18 de 90];
si :tecla = "flechader pedir todos [frumbo 90];
si :tecla = "flechaizq pedir todos [frumbo 270];
si :tecla = "flechaabajo pedir todos [frumbo 180];
si :tecla = "flechaarriba pedir todos [frumbo 0];
si :tecla = "enter pedir todos [arrancar];
fin;

para arrancar
```

```
sp
centro
fforma 1
frumbo quien * 18
mt
fvel 10
fin
```

Se logrará buenos diseños controlando los actores que no se vayan de la pantalla con la barra espaciadora, la "3"y "4"que hacen girar a los actores. Si pulsamos cada tanto la tecla "C" irán cambiando de color los trazos lográndose interesantes motivos.



§ 7.8 FUNCIONES de ANIMACION y GRAFICAS

Son *funciones* - también llamados *operaciones* o *reporteros* - las primitivas que sirven para interrogar a Logo, o sea, si quiero saber en determinado momento qué forma está llevando el actor 4, tipeo:

```
DECIR 4
FORMA
```

a lo que Logo *responde* imprimiendo en pantalla el número de forma que tiene.

Lo mismo puedo hacer si quiero saber el ***RUMBO*** o la ***VEL***ocidad que tiene un actor.

Las funciones *reportan* (1) o responden con un valor, por lo que pueden ser utilizadas como argumento de un procedimiento (comando).

Por ejemplo en ***FFORMA FORMA*** + 1 que hace que un actor tome la forma cuyo número sea igual al que tenía antes más 1, ***FFORMA*** es un procedimiento o comando, en cambio ***FORMA*** es una función u operación.

(1) Nota del autor: En las distintas versiones de Logo los autores de los manuales no se han puesto de acuerdo en qué palabra usar en castellano, es así que se usan como sinónimos Reporta, Devuelve, Entrega, Responde. Ultimamente se ha usado mucho REPORTA y REPORTERO que no es muy de mi agrado ya que es un anglicismo proveniente de Report= Informe. Tradicionalmente se usó con más criterio la palabra Responde y la primitiva RESPUESTA como la utilizada en esta versión (ver manual).

DESAFIO

De igual manera pueden imaginarse ejemplos como los que siguen.

¿Qué hace cada uno ellos?

```
fcolor color + 3
fx coorx + 40
fy coory - 30
fangulo angulo + 90
frumbo rumbo + 180
fcolor quien
fvel vel + 5
fx azar 100
```

Para más detalles del tema puede consultarse el Manual de Logo Gráfico.

§ 7.8.1 FUNCIONES GRAFICAS Y DE ANIMACION

ANGULO	RUMBO	GROSOR
COORX	COORY	COLORFONDO
DISTANCIA	FORMA	TRAZO
POS	VEL	COLOR
QUIEN	VELX	COLORDEBAJO
VELY	HACIA	

Idea

Probar un procedimiento que utilice la orden ***FCOLOR COLOR*** + 1 dentro de un evento ***CUANDOPASEN*** para que un actor vaya cambiando de color cada décima de segundo pintando toda la pantalla. Queda mejor con un grosor 3 ó 4.

§ 7.9 MOVIMIENTO Y ROTACION DE FORMAS

Un interesante recurso para usar en animación simultánea es la posibilidad de que el actor o actores puedan cambiar de forma durante el movimiento. De este modo se puede simular que el actor, además de que se está moviendo dentro de la pantalla, muestre una acción determinada.

En el siguiente ejemplo hemos simplificado para su explicación un ejemplo de programación, que está incluido en los discos de instalación del LOGO GRAFICO, sobre cómo se pueden cambiar las formas durante la animación simultánea.

Para lograr esto utilizamos una primitiva llamada ***ROTARFORMA***, que irá cambiando las formas de los actores entre la 1 y la 5 cada 7 centésimas de segundo. Ver Ayuda de Primitivas.

```
PARA CONDICION
fforma 1 sp
fxy 180 + 2 * quien 40 + 2 * quien
mt
fvel 8 + azar 7
frumbo 240 + azar 60
flimites [-200 190] 420 153
rotarforma 1 4 7
FIN
```

```
PARA FONDO
rg  ot libre
fcf 7
pentera
sp fpos [-160 35]
frumbo 90
cp fc 15
repetir 2 [ad 320 iz 90 ad 85 iz 90]
sp ad 40 iz 90 ad 40
frelleno 1 15
pintar
at 80 frelleno 1 7 pintar
FIN

PARA ARBOLES;
repetir 100 [estamparforma 9 + azar 3 lista (-150 + azar 300)
(-110 + azar 135)];
FIN;

PARA FLORES;
repetir 350 [estamparforma 12 lista (-160 + azar 320) (-100 +
azar 140)];
FIN;
PARA VOLAR
pedir todos [CONDICION]
FIN

PARA inicio
BP OT
quitarbarrah
cargarformas "VOLAR.FRM
FONDO
ARBOLES
FLORES
VOLAR
cuandotecla [interrumpir]
actuar
rg
FIN
```

Luego de definir el ejemplo, escribimos

```
inicio       <ENTER>
```

Los pájaros "volarán" dentro de la pantalla batiendo sus alas, en distintas direcciones y a diferentes velocidades.

DESAFIO

¿Te animás a dibujar en 3 o 4 retículas sucesivas el movimiento de un actor que se desplace por la pantalla saltando?

Condición: El dibujo en las retículas deben estar en forma sucesiva al movimiento que van a simular. Ver explicación de la primitiva ROTARFORMA.

§ 7.10 USO DE LA PRIMITIVA "CARTEL" EN ANIMACION SECUENCIAL Y SIMULTANEA

La primitiva cartel se usa cuando tenemos actores en pantalla en una animación simultánea.

Cuando es necesario poner un texto en la pantalla gráfica con el uso de actores usamos esta primitiva que puede incluirse dentro de cualquiera de los dos tipos de animación.

Actualmente, los actores también pueden utilizar la instrucción ***ROTULAR*** y ***ROTULARC*** dentro de la animación, pero sólo secuencial no en la simultánea.
La podemos utilizar como respuesta de un procedimiento o dentro de un CUANDO...

Para darle el tamaño, color, y forma de letra ver descripción de primitivas de la Ayuda o la Descripción del lenguaje del manual de LOGO GRAFICO.

Ahora veremos algunos ejemplos de uso en las distintas animaciones y algunos trucos de como escribir y borrar un cartel sin tener que borrar toda la pantalla gráfica o inicializarla.

```
para mensa;
fcolor 1;
estamparforma 5 [100 40];
cartel [esto es una forma estampada] [100 20] 7;
fin;
```

Esta es una forma sencilla de colocar un cartel debajo de una figura estampada.

Ahora veremos como usar la primitiva en un CUANDO...

```
para presen
bp ot pentera
fcolor 1
fcolorletras 1
decir 3
fforma 4
sp fpos [-90 -60]
fvel 20
frumbo azar 360
mt
fin

para poncartel;
fanchotipo 1;
cuandoactor 100 [cartel [aquí estoy] pos 7];
jaula;
moverse 1000;
fin;
```

Luego de definirlos tipeamos en modo comando:

```
presen poncartel      <ENTER>
```

Veremos que el actor se moverá con un rumbo fijado al azar con ***FRUMBO*** , y con la primitiva cuandoactor cada 100 centésimas de segundo aprox. pondrá un cartel en su actual posición, dada con la primitiva ***POS*** ya que él es el actor activo, que dirá Aquí estoy.

Esto sucederá durante el tiempo que dure la animación dado con la primitiva ***MOVERSE*** que es de 10 seg. aproximadamente y luego se detendrá.

Veremos un ejemplo más complejo, donde no sólo usamos la primitiva ***CARTEL*** dentro de un evento, sino que combinamos varios de éstos para crear un efecto interesante en la aparición de carteles y un truco para borrar un cartel sin detener la animación.

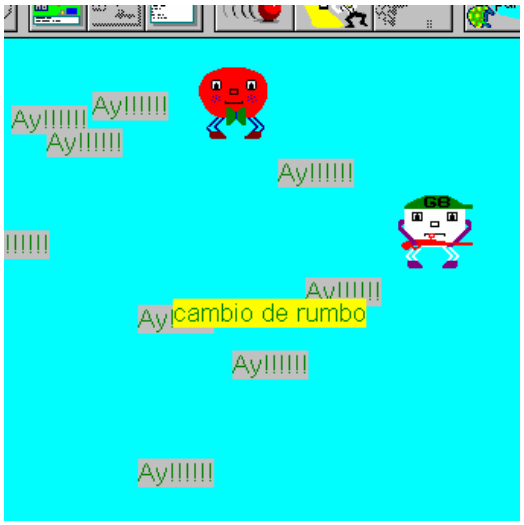
```
para actor1
decir 1
sp fpos [0 -90]
fforma 2 mt
cuandochoca 2 [cartel [Ay!] pos 14 rebotar]    (3)
fvel 35
frumbo azar 360
fin

para actor2;
decir 2;
sp fpos [-90 0];
fforma 3 mt;
frumbo azar 360;
fvel 35;
cuandoabajo [cartel [cambio de rumbo] [0 0] 15 frumbo rumbo +
50];    (2)
cuandoarriba [cartel [· · · · · · · · · ·] [0 30] 20];      (4)
fin;

para inicio
bp ot pentera
fcolorf 17
fcolorletras      (1)
fanchotipo 1      (1)
actor1
actor2
actuar
fin
```

Una vez definidos estos procedimientos, en modo comando tipeamos (con minúscula)

```
inicio            <ENTER>
```

Pasamos a explicar el ejemplo

Primero en el procedimiento ***inicio*** definimos el color de las letras dándole la instrucción ***FCOLORLETRAS 2*** y el tamaño de letra (1).



Luego le asignamos la forma, posición, eventos, etc. a cada actor interviniente (procedimientos ***actor1*** y ***actor2***).

En (2) le decimos al actor 2 que cuando choque con la parte inferior de la pantalla ponga un cartel en el centro que diga "Cambio de rumbo" y le 50 sumamos al rumbo actual.

En la instrucción (3) le asignamos al actor 1 el evento ***CUANDOCHOCA*** con el actor 2 que ponga un cartel en la posición del choque que diga Ay!!!!.

Ahora explicamos un truco de para borrar un cartel. En la orden (4) le decimos al actor 2 que cuando choque con la parte superior de la pantalla ponga un cartel con los espacios forzados, (con CTRL + B), con la misma cantidad de caracteres necesarios para tapar el cartel inicial, pero con el color de fondo del cartel igual al color de fondo de la pantalla o del lugar donde se sobreimprimen los carteles. De esta forma ponemos un cartel del color del fondo sobre el cartel anteriormente escrito, tapándolo.

También se puede escribir el mismo cartel pero con las letras y el color de fondo del cartel igual al de la pantalla o del lugar donde se sobreimprimen los carteles. Esto se logra con la primitiva ***FCOLORLETRAS***.

Existen muchas posibilidades de crear mensajes en animación con esta primitiva. Trata de usarlos en tus programas!.

§ 7.11 LIMITES PARA LOS ACTORES EN ANIMACION

Una de las primitivas interesantes de LOGO es la posibilidad de determinar para cada actor un límite o porción de la pantalla en la cual se puede mover.

Este límite puede estar dentro o fuera del área visible del monitor (no olvidemos que, aunque no lo veamos, el tamaño del espacio donde se mueven los actores es mayor que toda la pantalla visible del monitor). De esta manera podemos darle al actor un límite mayor que el monitor o menor que éste, dando la posibilidad de encerrar en una especie de "corralito" al actor en un sector del monitor a determinar.

La primitiva para esto se llama ***FLIMITES*** y consta de varios argumentos que se detallan el la Ayuda de Primitivas o en la Descripción del lenguaje del manual.

Esta orden puede darse a cada actor por separado, es decir que le podemos asignar un límite distinto a cada actor.

Veremos algunos ejemplos...

```
para limit
decir 3
sp fpos [-60 60]                (1)
fforma 3 mt
flimites [-130 100] 120 100     (2)
fvel 20
frumbo 38
actuar
fin
```

En este ejemplo posicionamos un actor en el sector superior izquierdo (1) y en la instrucción (2) le definimos los límites en los cuales se moverá el actor, tomando esos límites como si fueran los propios bordes del monitor, teniendo validez los eventos relacionados con los bordes del mismo.

Se debe tener en cuenta si damos la orden ***LIBRE*** dentro del programa, ya que puede anular la primitiva ***FLIMITES*** para el actor seleccionado.

§ 7.12 VEMOS AL ACTOR SOLO EN UN SECTOR DE LA PANTALLA

Otra primitiva muy interesante da la posibilidad de ver al actor sólo en una porción de la pantalla, pero permitiendo que éste se mueva por toda la misma. Es como hacer una ventana en el monitor y poder ver pasar un actor dentro de la animación simultánea.

Esta primitiva se llama ***FAREAVISIBLE***, que como su nombre lo indica nos fija un área del monitor donde sólo podrá verse el actor, ya sea en animación simultánea o secuencial. Esta primitiva tiene varios argumentos los cuales se detallan en la Ayuda de Primitivas o en el manual de Descripción del lenguaje.

Cuando el actor sale de esa ventana se oculta, pero no deja de interactuar con otros actores o de cumplir con los eventos que tenga asignados, como por ejemplo, rebotar en los bordes de la pantalla gráfica (si está en modo JAULA) o cualquier otro tipo de evento.

Esta orden se le puede asignar a cada actor por separado, fijándole un área visible distinta a cada actor.

¿Qué efectos se logran?. Los veremos en algunos divertidos ejemplos...

Ejemplo: 1

```
para presen
decir 9
fforma 33
sp fpos [-90 -90] mt
frumbo 47
fvel 35
fin
```

```
para juego1
bp ot pentera fcf 0
presen
fareavisible [-80 80] 150 150        (1)
actuar
fin
```

Nota: La forma 33 es la de un satélite. Si no la tiene dibuje una forma similar

El primer procedimiento llamado ***presen*** lo utilizaremos en todos los ejemplos que describiremos, de esta manera haremos una sola vez el programa de presentación del actor.

El procedimiento ***juego1*** nos muestra sencillamente cómo funciona la primitiva fareavisible (1) dándole al actor el área de la pantalla donde lo veremos pasar, y cuando este salga de ella desaparecerá sin dejar de moverse por el monitor, rebotará en el borde del mismo y volverá a pasar por la ventana, así hasta cortarlo con la tecla ESC o el botón Pare de las barras de herramientas.

Ejemplo: 2

```
para juego2
bp ot pentera fcf 0
presen
fareavisible [-80 80] 150 150
cuandoarriba [fforma 1 + azar 30]         (1)
actuar
fin
```

Aquí le hemos agregado un evento (1) tal que cuando choque con la parte superior del monitor el actor cambiará de forma eligiéndola al azar. Esto producirá que, cuando el actor choque arriba y vuelva a pasar lo hará con otra forma. Como este efecto, se podrá realizar cualquier otro que definamos con algún evento.

Ejemplo: 3

```
para juego3
bp ot pentera
fcf 1                          (1)
fc 2                           (2)
frumbo 90                      (3)
frelleno 1 0                   (4)
sp fpos [-80 80]               (5)
cp repetir 4 [ad 150 de 90]    (6)
rellenar                       (7)
presen
fareavisible [-80 80] 150 150
cuandoarriba [fforma 1 + azar 50]
cuandoabajo [tono 400 3]       (8)
actuar
fin
```

En este ejemplo se complicaron más las cosas, ya que no sólo le fijamos un área visible, sino que le dibujamos una ventana.

Los pasos son los siguientes.

En (1) definimos un color de fondo de la pantalla distinto al color de la ventana que utilizaremos.

Las instrucciones que van desde (2) hasta (7) están dirigidas a que la tortuga gráfica dibuje y rellene un cuadrado de color negro (distinto al color de fondo de la pantalla) con un borde de color verde. Este cuadrado coincide exactamente con las coordenadas y tamaño del área visible, lo que producirá el efecto de como si hubiéramos abierto una ventana en el monitor y viéramos pasar al actor (imagínenlo si cargamos un decorado con una ventana dibujada y creamos un área visible sobre ésta !!!! Se animan al desafío?).

En (8) le asignamos un evento adicional para que cuando choque abajo produzca un sonido.

CONSEJOS:

Traten de que el color del área visible sea opuesto al color del actor, para que se pueda ver con claridad.

En el ejemplo 3, el cuadrado pintado de negro se hizo con la tortuga gráfica, antes de presentar los actores, ya que luego no se puede utilizar. Con los actores también se puede realizar el cuadrado, pero se debe utilizar la primitiva ***PINTAR*** (ver como se usa en la Ayuda de Primitivas o en la Descripción del Lenguaje en el manual o en el archivo LENGUAJE.WRI que está incluido en el disquete del programa).

El decorado del desafío se puede hacer en el PaintBrush o cualquier graficador de paleta que guarde archivos BMP o PCX.

DESAFIO

Traten de combinar los ejemplos y de incluir un decorado en el programa. Dibujen sus propias formas para los actores, y como broche de oro incluyan la primitiva ***ROTARFORMA*** en el o los actores que utilicen.

CAPITULO 8

CONVERSANDO CON LOGO

Con Logo podemos programar en forma muy sencilla diálogos con la computadora. Esto lo haremos con la utilización de primitivas comunes, que servirán para otros usos o que ya han sido usadas en capítulos anteriores.

A través de sencillos procedimientos podemos lograr que, ante una pregunta del usuario, el programa produzca una respuesta preparada por el programador. Esta tendrá una coherencia dentro de ciertos límites que le dará el programa.

Veremos cómo empezar este diálogo con Logo

Escribimos:

```
esc "LOGO
LOGO
```

Donde ***ESC*** es una primitiva que escribe en la pantalla de texto o mixta una palabra o lista sin las comillas o sin los corchetes.

Si queremos escribir una oración con espacios debemos encerrarla entre corchetes

```
esc [Trabajando con Logo]
Trabajando con Logo
```

Existen dos alternativas para escribir en la pantalla de textos o mixta, una es ***ESC*** que luego de ejecutarla salta al renglón siguiente. Ejemplo:

```
para probar
esc [PRIMERO]
esc [SEGUNDO]
fin
```

Y que al ejecutar ***probar*** quedará:

```
PRIMERO
SEGUNDO
```

La otra es ***ESCS*** que es similar a la anterior pero -al usarla repetidas veces como en el ejemplo de más abajo- no deja una línea entre ambas impresiones. Por ejemplo:

```
para unir
escs "gentil
escs "hombre
fin
```

Y al ejecutar **unir** aparecerá:

gentilhombre

Veremos unos ejemplo sencillos de programas para dialogar con Logo.

```
para conversar
escs [ Yo soy Logo Gráfico. ]
esc [ Tú, cómo te llamas? ]
fin
```

Ejecutamos el procedimiento definido:

Yo soy Logo Gráfico. Tú, cómo te llamas?

Si contestamos con nuestro nombre Logo lo tomará como un comando y dará un mensaje de error. Para eso debemos preparar al programa para que pueda continuar con el diálogo a través de una primitiva llamada ***LEERLINEA*** .

Esta primitiva lo que hace es esperar que ingresemos con el teclado una palabra o lista (oración), que se le puede asignar como valor a una variable o dar como dato a otras primitivas, luego pulsamos Enter. Esta primitiva funciona en pantalla de textos o mixta. Si no estuviera la pantalla mixta activada el uso de la primitiva ***esc*** la activa.

Para ver cómo sería el programa le agregaremos al programa anterior.

```
para conversar
escs [ Yo soy Logo Gráfico. ]
esc [ Tú, cómo te llamas? ]
hacer "nombre ll                        (1)
linea                                   (2)
esc frase [Me gusta tu nombre, ] :nombre  (3)
fin
```

(1) Le asignamos el contenido de leerlinea a la variable :nombre
(2) Dejamos una línea en blanco
(3) A través de la primitiva ***FRASE*** *, al escribir en la pantalla, unimos la oración o lista con el contenido de la variable :nombre*

Si ejecutamos el procedimiento nos resulta así:

Yo soy Logo Gráfico. Tú, cómo te llamas?

Aquí se queda esperando que nosotros ingresemos el dato, y escribimos:

```
José              Enter
```

Nos responde:

Me gusta tu nombre, José

La primitiva ***LEERLINEA*** es muy utilizada para el ingreso de cualquier tipo de datos dentro de un procedimiento. Cada vez que deseamos entrar un valor numérico de más de un carácter, una palabra o lista utilizamos esta primitiva, que siempre devuelve el contenido en forma de lista no importa si ingresamos una lista o una palabra (para más datos ver diferencias entre lista y palabra en el capítulo 2).

Es importante tener en cuenta este dato, ya que puede complicar el manejo del contenido cuando utilizamos las primitivas ***PRIMERO, ULTIMO, MENOSPRIMERO,*** etc.

Otra posibilidad que brinda Logo Gráfico es una primitiva llamada ***ENTRARDATOS***, que abre una ventana de diálogo donde no sólo nos pide que ingresemos el dato, sino que dentro de ella podemos ponerle un título y un texto donde explica o nos dice qué dato debemos ingresar. Veremos un ejemplo sencillo del uso de esta primitiva.

```
esc entrardatos "Nombre [Escribe aquí tu nombre];
```

Luego de ingresar el dato y pulsar Enter o picar con el mouse en Aceptar escribirá el dato ingresado.

Si reemplazamos ***LEERLINEA*** con esta primitiva, en el procedimiento anterior quedará:

```
para conversar2;
hacer "nombre entrardatos [ Yo soy Logo Gráfico] [Vos, cómo te
llamas? (ingresa aquí tu nombre)];
linea;
esc frase [Me gusta tu nombre] :nombre;
fin;
```

Haga la prueba e invente más formas de uso con las primitivas ***LEERLINEA*** y ***ENTRARDATOS***, si tiene alguna duda busque ayuda en Ayuda de Primitivas del programa Logo Gráfico

Podemos ir aumentando el diálogo con la máquina agregando y complicando un poco más los procedimientos.

```
para dialogo;
esc [Qué edad tienes?];
hacer "edad primero ll;
linea;
esc frase [Cuándo cumples] :edad + 1; (1)
hacer "fecha ll;
linea;
esc [Cuántos hermanos tienes?];
hacer "num primero ll;                (2)
linea;
si :num > 2 esc [Qué familia numerosa !!] sino esc [Qué poquitos
son...];                              (3)
fin;
```

(1) Nos pregunta cuándo cumplimos la edad que tenemos más uno, o sea la fecha de nuestro cumpleaños.
(2) debemos ingresar el número de hermanos que tenemos; como verán para analizar y/o escribir lo ingresado con leerlinea tenemos que pedir el primer elemento de la lista, aunque hallamos escrito una palabra sola.
(3) aquí compara el número de hermanos y si éste es mayor que dos escribe "Que familia numerosa !!; y si es menor "Que poquitos son...

Es importante saber que estos sencillos procedimientos no contemplan la posibilidad de prevenir el mal ingreso de datos, por ejemplo si cuando nos pregunta la edad ingresamos una palabra o letra, nunca podrá hacer la suma de la variable más uno, por lo que nos dará un mensaje de error. Cuando se programan diálogos más avanzados en programas más complejos, se deben tener en cuenta todas las posibilidades sobre qué tipo de datos se pide y cómo vamos a manejarlos dentro del procedimiento.

Ahora veremos un ejemplo más complejo y muy interesante para tomar como base en un programa más extenso. Está incluido en el disquete con el nombre PSICOANA.LGO

```
para psicoanalisis;
bt modotexto;
esc [ Buenos días, dígame su nombre y apellido];
hacer "nombre ll;
linea;
esc fr [Cuál es su problema?] ult :nombre;
escs [>] hacer "problema ll;
linea;
analisis;
fin;

para analisis;
linea;
esc [Hace mucho que le sucede esto?];
hacer "resp pri ll;
linea;
si :resp = "si esc [Háblеme de la primera vez que le ocurrió
esto] hacer "relato ll edipo
  sino esc [Puede relacionarlo con su infancia?] hacer "relato
pri ll si :relato = "no reprime sino edipo;
fin;
```

Tal vez éste sea el procedimiento más complejo, ya que realiza dos condicionales (si) en forma sucesiva; primero compara la respuesta a la pregunta [Hace mucho que...], si es afirmativa ejecuta una acción, si es negativa vuelve a interrogar y realiza una comparación de la respuesta.

```
para edipo;
linea;
esc [Continúe asociando libremente, trataremos de llegar hasta el
final];
linea;
hacer "relato ll;
linea;
saludo;
fin;

para reprime;
linea;
esc [Parece que hay cosas que le cuesta recordar, háblеme de lo
que se le ocurra];
linea;
hacer "relato ll;
linea;
saludo;
fin;

para saludo;
```

```
linea;
esc [Lo lamento, se terminó el tiempo, son cien pesos, nos vemos
la próxima semana.];
linea;
fin;
```

Como está expuesto esto es una forma sencilla de diálogo donde las respuestas son muy limitadas y no se tienen en cuenta formas alternativas de responder a una misma pregunta, pero partiendo de esta base podrá programar infinidad de tipos de diálogo para jugar con sus alumnos o amigos.

También podrá obtener información realizando una pequeña encuesta dialogada con sus alumnos, guardando en distintas variables las respuestas dadas a cada pregunta.

Dentro del disquete que se entrega con el manual existe un programa llamado ELISA.LGO donde se muestra un ejemplo un poco más complejo que los anteriores, donde el autor utiliza varios recursos propios de Logo para la realización de diálogos. Es interesante ejecutarlo para ver hasta dónde llegan las posibilidades de conversar con Logo.

CAPITULO 9

TRABAJANDO CON LISTAS

§ 9.1. TRABAJANDO CON LISTAS

Para trabajar con listas es muy útil el uso de la recursión. Es así que si quisiéramos deletrear una palabra, deberíamos ir recorriéndola, acortándola con ***mp*** y escribiendo la primera letra de cada lista así obtenida.

Veamos más detalladamente esto con la palabra MESA. Usaremos la primitiva ***escribirs*** (escribir seguido) para que las escriba todas en el mismo renglón.

```
escribirs primero "MESA
escribirs primero mp "MESA
escribirs primero mp mp "MESA
escribirs primero mp mp mp "MESA
```

¡ Parece que hemos encontrado una forma complicada de escribir una palabra tan simple !

Vamos a buscar una forma más sencilla y que podamos aplicar a cualquier palabra mediante el uso de la recursión con variable.

```
para deletrea :palabra
si :palabra = "parar            (3)
escribirs primero :palabra      (1)
deletrea mp :palabra            (2)
fin
```

(1) Escribe la primer letra de lo que contiene la variable :palabra, con lo que va recorriendo la palabra.
*(2) Llama al procedimiento **deletrear** y acorta la palabra quitándole la primera letra.*
*(3) Verifica si terminó de recorrer toda la palabra, cosa que se verifica si la palabra quedó **vacia** (lo que se indica con ") y en ese caso corta la recursión. También se puede escribir:*

```
si vacia? :palabra  parar
```

Una forma mejorada de este procedimiento (se encuentra en el archivo TRABLIST.LGO), sería:

```
para deletrear :pal
si vacia? :pal parar        (4)
escribirs primero :pal
escribirs car 32            (5)
deletrear mp :pal
fin
```

(4) Es una forma mejor de comprobar si una palabra está vacía. Puede usarse también para listas.
*(5) Deja un espacio en blanco entre cada letra, ya que el **car** 32 representa la barra espaciadora.*

NOTA: A cada tecla le corresponde un número (del 0 al 255) en la codificación ASCII (American Standard Code for Information Interchange = Códigos de Norma Americana para Intercambio de Información). Puede probarse con: esc car 33... esc car 34... hasta... esc car 255. Se debe tener presente que el código ASCII dependerá del tipo de font de Windows que se emplee.

De igual manera podríamos hacer un procedimiento que encolumne una lista de palabras o las letras de una palabra (incluye números), es decir un objeto-LOGO, en general.

```
para encolumnar :objeto
si vacia? :objeto parar
esc primero :objeto
encolumnar mp :objeto
fin
```

Este procedimiento se encuentra en el archivo TRABLIST.LGO del disquete.

Así si ordenamos:

```
encolumnar [LOGO ES UN LENGUAJE MODULAR];
```

obtenemos:

LOGO
ES
UN
LENGUAJE
MODULAR

o bien:

```
encolumnar "ELEFANTE
```

E
L
E
F
A
N
T
E

DESAFIO

¿ Cómo harías un procedimiento BUSCARZ que recorra la palabra, deletreándola y que cuando encuentre una "z" lo haga notar escribiendo "¡Encontré una z !" ?

DESAFIO

¿ Te animarías a hacer un procedimiento ACORTAR que vaya acortando una lista o palabra imprimiéndola encolumnada hasta que quede vacía ?

DESAFIO

¿ Y un procedimiento que le vaya quitando la primera y la última letra cada vez ?

DESAFIO

¿ Cómo hacer un procedimiento SINVOCALES :pal que reemplace todas las vocales de una palabra por guiones? ¿Y todas las consonantes?

Ayuda: La primitiva ***miembro?*** devuelve si un elemento pertenece o no a una palabra o lista (es una función). Por ejemplo ***miembro?*** "a "casa es **VERDAD** en cambio ***miembro?*** "b "casa es **FALSO.**

Entonces se puede usar una expresión como:

```
si miembro? "a "casa escribirs (tal cosa)
```

DESAFIO

Cómo hacer un procedimiento MUCHASUES (muchas úes) que reemplace cada una de las vocales de cualquier palabra con la letra U.

NOTA: Las soluciones de los Desafíos 4 y 5 están en el archivo TRABLIST.LGO.

§ 9.2. CONJUGANDO UN VERBO

Ahora vamos a tratar de enseñarle como conjugar un verbo regular a partir de cualquier infinitivo que le demos.

Lo primero que haremos, será separar la raíz, para lo cual usaremos:

```
para conjugar :verbo
hacer "raiz mu mu :verbo        (7)
...     ...  ...
```

(7) Le quitamos las dos últimas letras, por ejemplo, el verbo "cantar", quedará "cant" que es la raíz.

Recordemos que ***hacer*** crea una variable global (Ver Capítulo sobre variables) Guardaremos las 6 terminaciones en una variable que llamamos :***terminaciones***

```
hacer "terminaciones [o as a amos áis an]
```

que crea una variable llamada :terminaciones a la que se le asigna el valor de la lista [o as a amos áis an].

Las comillas precediendo al nombre de la variable sólo se usan al definirla con ***hacer***, luego al usarla se la llama precediéndola de los dos puntos (:).

De igual manera se le puede asignar a una variable un número, con:

```
hacer "x 50
```

define :x que vale 50.

También:

```
hacer "saludo "HOLA
```

define una variable :saludo que es la palabra "HOLA.

Luego se piden sus valores así:

```
esc :x            <Enter>
50
```

o

```
esc :saludo       <Enter>
HOLA
```

Continuando con la conjugación, ahora a la raíz debemos agregarle las terminaciones correspondientes a cada pronombre personal. Lo hacemos con la primitiva ***PALABRA***, por ejemplo, para el verbo amar es: ***PALABRA*** :raiz "o nos devuelve "amo", y para anteponer cada pronombre usamos ***frase (fr)***

```
para conjugar :verbo
hacer "raiz mu mu :verbo
esc [PRESENTE]
esc [ ]
esc fr "yo palabra :raiz "o
esc fr "tu palabra :raiz "as
esc fr "él palabra :raiz "a
esc fr "nosotros palabra :raiz "amos
esc fr "vosotros palabra :raiz "áis
esc fr "ellos palabra :raiz "an
fin
```

DESAFIO

Si se hicieran 2 listas con los pronombres y las terminaciones, se podría hacer un procedimiento que las recorra y vaya armando las conjugaciones. ¿ Puedes hacerlo?

DESAFIO

¿ Cómo hacer para que la computadora reconozca si es de la primera, segunda o tercera conjugación ?

DESAFIO

¿ En caso de solucionar el Desafío anterior, cómo harías para seleccionar las terminaciones de cada una de las distintas conjugaciones?

En el disco provisto con este libro hay 25 archivos separados con una o más extensiones, que pueden cargarse por separado. Por ejemplo para hacer un promedio, escribimos:

```
cargar "promedio
```

los cuales serán definidos directamente en memoria, si queremos ver como están programados debemos cargarlos desde el Editor de Textos.

Los procedimientos son promedio y sumar, que actúan juntos (sumar es un subprocedimiento de promedio).

Si queremos cargar todos las extensiones de listas juntas, ponemos:

```
cargar "listas
```

que es el archivo que contiene a todos los procedimientos que se mencionan, y al igual que en el caso anterior, si queremos verlos hay que cargarlos en el Editor de Textos.

En general para distinguir cuándo una primitiva actúa sobre una palabra o una lista se ha usado la siguiente nomenclatura:

infinitivo	para listas
tercera persona	para palabras

Nota: Los archivos .LGO a los que se hacen mención, están por separado.

CAPITULO 10

USO DE BOTONES Y CONTROLES

§ 10.1 PONIENDO BOTONES

La posibilidad de crear botones en la pantalla gráfica es una de las grandes novedades que presenta la versión Windows de LOGO GRAFICO.

Esta posibilidad es una herramienta muy útil y dinámica para facilitar el control y manejo del entorno LOGO a través de la programación de botones en la pantalla gráfica. Tenemos dos tipos de botones, los cuales tienen distintas características de funcionamiento, dependiendo de qué queremos controlar o manejar, y dentro de qué tipo de animación, secuencial o simultánea.

§ 10.1.1 BOTONES EN MODO COMANDO

Estos botones se crean a través de la primitiva ***FBOTON*** y su apariencia es similar a los utilizados en el entorno Windows. Sólo funcionan en modo comando o directo, es decir, que no responden cuando se los "pica" con el mouse si se está ejecutando una recursión (animación secuencial) o dentro de animación simultánea.

Con ellos podemos crear una pantalla inicial donde el usuario pueda, picando con el mouse, elegir una opción o programa para su inmediata ejecución.

La primitiva ***FBOTON*** tiene varios argumentos (variables) los cuales son explicados en la Ayuda de Primitivas o en la Descripción del lenguaje del manual.

Daremos un ejemplo de cómo se programa.

Ejemplo: 1

```
para bot_circ
bp ot pentera
sp fpos [-100 90]                          (1)
fboton [circunferencia] "circunf 48 17     (2)
fin

para circunf                               (3)
fc 2
fgrosor 3
sp fpos [0 30 ]                            (4)
cp repetir 90 [ad 1 de 4]
fin
```

Para poner un botón en la pantalla debemos seguir algunos pasos:
En (1) posicionamos la tortuga gráfica, pues donde esté la tortuga ahí se colocará el botón, eso quiere decir que la posición del botón está dada por la ubicación actual de la tortuga en la pantalla. NO funciona en animación (con la posición de actores.
Luego le damos en (2) la orden FBOTON con 2 nombres y 2 números.
*1. Primero figura el nombre del botón, o sea la palabra o texto que aparece sobre el botón. En nuestro caso como se dibujará una circunferencia le ponemos **[circunferencia]**, que puede ir entre corchetes o precedida de comillas (si es una palabra).*

*2. Luego seguirá la instrucción o procedimiento que deberá ejecutar, en nuestro caso **circunf.** En vez del nombre del procedimiento, se podría poner una lista de órdenes (entre corchetes) o también podríamos invocar algún procedimiento con actores..*
3. Luego irán las medidas del botón, primero el ancho y luego el alto del mismo.
*En (3) creamos el procedimiento **circunf** que ejecutará la tortuga cuando piquemos el botón con el mouse en modo directo.*
(4) Es importante, ya que posicionamos nuevamente la tortuga donde queremos que se dibuje la circunferencia.

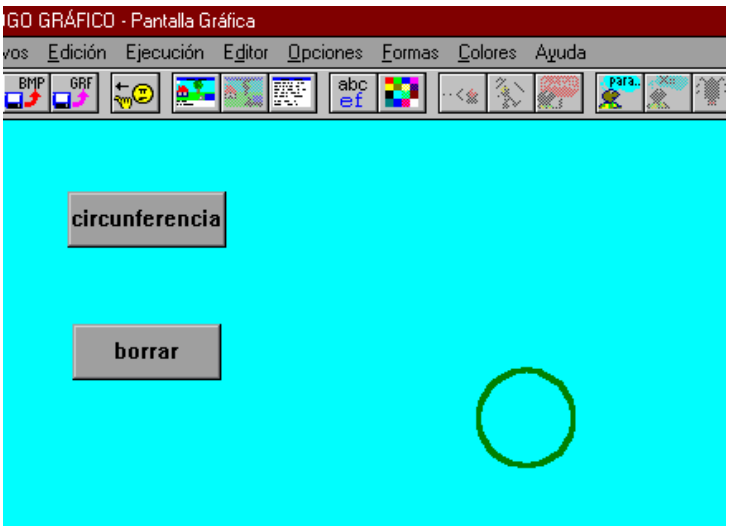
MUY IMPORTANTE:

En la versión actual no se pueden crear botones con actores en escena, es decir que si tenemos que colocar botones debemos hacerlo antes que pongamos algún actor, o sea después de un ***bp*** o ***rg.***

Ejemplo 2:

```
para botones
bp ot pentera
bot_borrar              (1)
bot_circunf             (2)
fin

para bot_circunf
sp fpos [-100 90]
fboton [circunferencia] "circunf 48 17
fin

para bot_borrar
sp fpos [-100 50]
fboton "borrar [lp] 45 17
fin
```

Con estos dos últimos procedimientos, podemos crear dos botones, uno que dibuje la circunferencia, el otro que la borre. Necesitamos también el procedimiento **circunf** del ejemplo anterior. Y para trabajar con más modularidad creamos el super-procedimiento **botones** que junta a todos.



Lo explicamos:

En el procedimiento **bot_borrar** limpiamos la pantalla borrando la circunferencia dibujada.

En el procedimiento **botones** primero creamos el botón de borrar (1), luego en (2) creamos el botón con el cual, al hacer clic en él se dibujará la circunferencia (el orden de creación no altera el resultado, es decir que podríamos ejecutar el procedimiento **bot_borrar** luego del procedimiento **bot_circunf**).

§ 10.1.2 BOTONES ESPECIALES PARA USAR EN ANIMACION SIMULTANEA

Para el uso de botones en la animación simultánea se creará otro tipo de botón (llamados botones especiales o botones-Logo), esto es, con la primitiva ***FBOTONLOGO***, el cual es diferente al anterior por los siguientes motivos:

- La forma del botón la creamos en el editor de formas, o sea que el botón tiene la apariencia de un disfraz de actor. El botón **no es** un actor pero le podemos dar la forma de éstos.

- Se deben editar (crear) dos formas para cada botón. En una retícula se diseña el botón sin oprimir -o sea el que se ve cuando no está accionado- y en otra retícula el botón oprimido, el que se ve sólo en el momento de presionarlo. Esto significa que nosotros al oprimir un botón-Logo éste se transformará entre una forma y la otra, y al soltarlo volverá a su forma inicial.
- Y fundamentalmente el botón-Logo **funciona tanto en modo comando como en animación, ya sea secuencial o simultánea.** Lo que nos brinda la posibilidad de controlar cualquier procedimiento en animación.

Esta última diferencia es de mucha utilidad para generar un entorno de trabajo donde el usuario puede programar distintos efectos dentro de la animación y manejarlos a través de botones en la pantalla.

Recordamos que como con el tipo de botón anterior la posición de éste estará fijada por la posición de la tortuga gráfica, por lo que hay que ubicar los botones **antes de entrar en animación** (o sea antes de usar la orden ***ACTIVAR*** o ***DECIR***) o luego de un ***BP*** o ***RG***,.

Daremos algunos ejemplos de sencillos programas con botones. Las formas usadas son del archivo BOT.FRM de los discos de instalación. Si no las tiene, dibuje sus propios botones. Fíjese que para dar apariencia de botón suelto los bordes izquierdo y superior son claros (iluminados) y los otros oscuros, y en el oprimido al revés (ver figura).

```
para presen
decir 5
fforma 14
sp fpos [90 40]
frumbo 200
fvel 35
visible
fin

para bot
sp fpos [-90 90]
fbotonlogo [actuar] 4 5          (1)
fpos [-90 70]
fbotonlogo [interrumpir] 8 9     (2)
fin
```

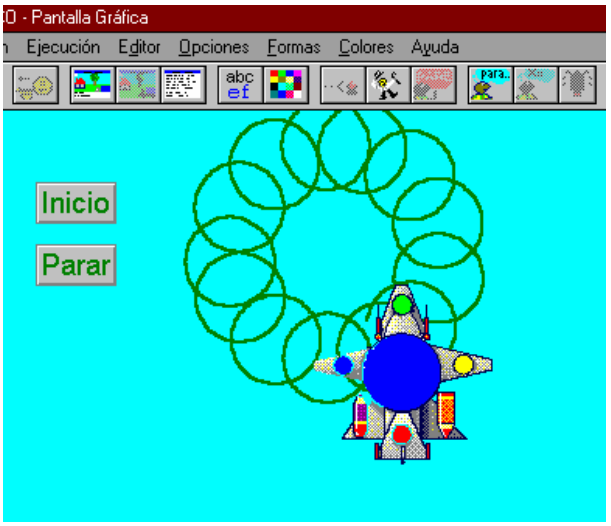
```
para inicio
bp ot fcf 0
bot
presen
fin
```

En estos tres procedimientos lo que hacemos es: en le procedimiento present ubicamos al actor en pantalla con todas sus características. En bot ponemos los botones Logo en la posición deseada (Notar que lo hacemos con la tortuga gráfica) y les asignamos las acciones deseadas. En (1) le damos la orden ***ACTUAR*** *seguida de los números de las formas que debe intercambiar el botón sin oprimir y al oprimirse en (2) le damos la orden* ***INTERRUMPIR*** *y las 2 formas necesarias.*
Luego en el procedimiento ***inicio*** *primero ponemos los botones, después ponemos al actor en pantalla y luego: ¡A probar!*

§ 10.1.3 USO DE BOTONES-LOGO EN PROGRAMACION SECUENCIAL

Ahora veremos un ejemplo con programación secuencial, para mostrar cómo podemos controlar (en este caso, parar) un procedimiento recursivo y no de animación simultánea con los botones-Logo.

```
para recursivo
decir 7
repetir 20 [ad 2 esperar 5]
de 250
cp
repetir 90 [ad 1 de 4]
sp
recursivo
fin

para presen2
decir 7
fc 2
fgrosor 2
sp fpos [0 50]
fforma 6
visible
fin

para bot2
sp fpos [-90 90]
fbotonlogo [recursivo] 1 2
fpos [-90 70]
fbotonlogo [parar] 3 4
fin

para todo
bp ot pentera
fcf 17
bot2
presen2
fin
```



Ejecutemos el procedimiento ***todo*** y luego probemos de pulsar alternadamente los dos botones controlando de esta forma la acción (usamos el archivo de formas BOTONPRG.FRC).

Al probar de trabajar con ellos se verá qué útiles y cómodos son los botones en un entorno de trabajo donde el usuario debe controlar el comportamiento de los actores en la pantalla. También los podemos utilizar como una forma de darles órdenes a los actores o cambiarles las condiciones o eventos.

ALGUNOS CONSEJOS

- Dentro de la forma del botón le podemos dibujar el nombre o leyenda, también se lo podemos poner debajo o arriba del botón con la primitiva ***CARTEL***.

- Es importante que las formas de los botones sean del mismo tamaño, ya que debido a la programación interna del mismo se superponen las formas, lo que implica que, si tenemos formas de distintos tamaños éstas quedarán superpuestas y se verán ambas, la oprimida y la sin oprimir obteniendo un resultado indeseado. También resulta conveniente no usar demasiado el color negro (cero) que en realidad es transparente.

- Cuando colocamos varios botones en pantalla, puede resultar confuso identificar cuál es cada uno y qué acción realizará. para ello se puede hacer clic con el botón derecho del mouse sobre cada uno de ellos, y tendremos la información buscada.

§ 10.2 TRABAJAMOS CON LAS BARRAS DE DESLIZAMIENTO O CONTROLES

Podemos crear dentro de la pantalla gráfica barras de deslizamiento, como las que tienen las ventanas de Windows. Estas barras, al igual que los ***FBOTON***, sólo funcionan en modo comando y los podemos utilizar para controlar y/o asignarle un valor a una variable con el movimiento del cursor de la barra o picando en las flechas de los extremos de éstas.

Las primitivas para crear y leer una barra son tres, ***FCONTROL***, ***POSCONTROL*** y ***FPOSCONTROL***. El uso de estas tres primitivas lo veremos a continuación aplicándolas en la programación de un ejemplo que permitirá variar la base y altura de un rectángulo por medio de las barras de deslizamiento.

```
para inicio
rg fcf 17
ftipo "arial fanchotipo 1
fcolorletras 20
controles
botones
alt
base
posinicial
fin

para controles
sp fpos  [ 15 -30]              (1)
fcontrol "H "base 128 0 100     (2)
```

```
fposcontrol 1 50                (3)
sp fpos [ -55 40]
fcontrol "V "alt 123 0 100      (4)
fposcontrol 2 50
fin
```

En este procedimiento se posiciona la tortuga en el centro donde se creará la barra (1) con ***fcontrol*** *(2) definimos la barra, cuyos parámetros son: "****H*** *que indica que es horizontal, "****base*** *es el nombre del procedimiento que se ejecuta al accionar la barra o modificar su estado,* ***128*** *es el largo que tiene la barra, expresado en pasos de tortuga y* ***0 100*** *determina los 2 valores extremos (máximo y mínimo) de la base. De similar manera definimos la barra vertical en (4).*
La instrucción ***fposcontrol 1 50*** *(3) determina que para el control número 1 (el primero creado) el valor inicial de la variable* ***:b*** *será 50.*

```
para botones
sp fpos [-100 -4]
fboton [Dibuja] "rect  50 15
sp fpos [ -100 20]
fboton [Borrar Todo] [lp]  50 15
fin
```

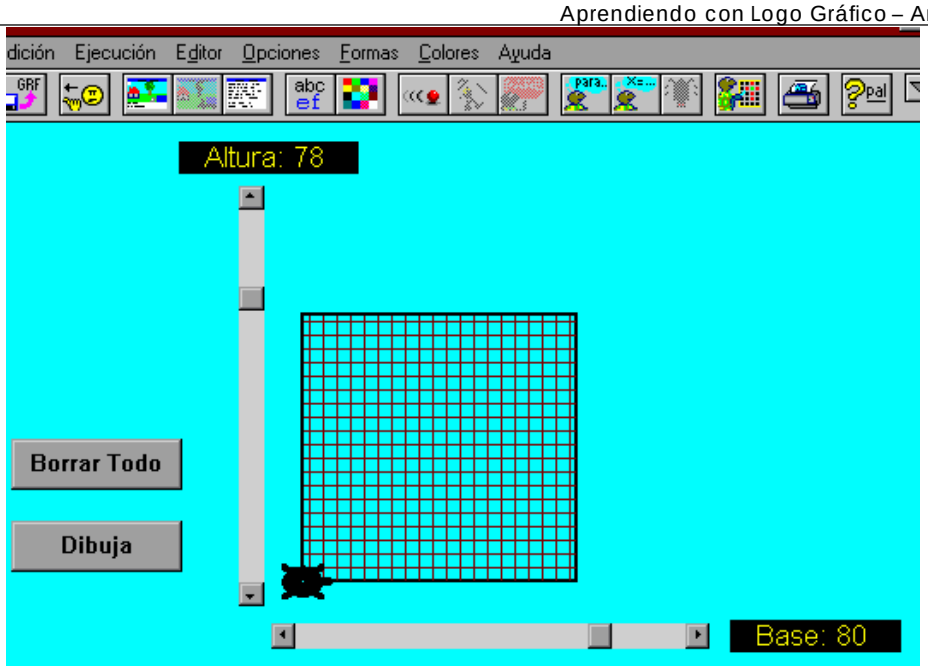
Se crean dos botones, uno para ejecutar el programa ***rect*** *que realiza el rectángulo y otro para empezar de nuevo.*

```
para posinicial
sp mt fpos [-40 -14] frumbo 90 cp
fin

para rect
frelleno 1 + azar 9 1 + azar 15
repetir 2 [ad :b iz 90 ad :h iz 90]
rellenar
fin
```

En ***posinicial*** *determinamos la posición inicial de la tortuga y el procedimiento* ***rect*** *que hará un rectángulo cuando se presione un botón. El tamaño del rectángulo quedará determinado por las variables :b y :h que se varían con las barras de deslizamiento en (1).*

```
para base
hacer "b poscontrol 1                    (1)
cartel fr fr [···Base:] :b [···] [110 -30] 0
fin

para alt
hacer "h poscontrol 2
cartel fr fr [···Altura:] :h [···] [-50 110] 0
fin
```

(1) Con esta línea se leerá el valor del "Control 1" dado por el cursor de la barra de control horizontal y se le asigna ese valor a la variable :b.
(1b) Se imprime un cartel con el valor leído.
El procedimiento ***alt*** *realiza igual operación con la barra vertical.*



§ 10.2 EXPERIMENTANDO CON EL SATELITE

En este programa -desarrollado por uno de los autores- se utilizan los distintos botones, las barras, áreas visibles, animación simultánea y para los que poseen plaqueta de sonido el uso de un archivo WAV. El objetivo del programa es realizar una simulación de las posibles trayectorias de un satélite que se acerca a un planeta que lo atrae. Debemos variar la velocidad inicial con la barra de deslizamiento (control) y lanzarlo con "Actuar", de modo de lograr una trayectoria elíptica permanente.

```
para control
sp fpos [-145 50]
fcontrol "V "posvel 100 10 30    (1)
ftipo "arial fanchotipo 1
fcolorletras 1
fposcontrol 1 20                 (2)
posvel                           (3)
ventana                          (4)
fin

para posvel
cartel [··Vel.:]||(poscontrol 1)||[··] [-105 100] 14  (4a)
hacer "vl poscontrol 1
fin
```

(*) Las barras || equivalen a la primitiva FRASE, o sea que la línea equivale a:

```
cartel fr fr [..Vel.:] (poscontrol 1) [..] [-85 100] 14
```

Ubicamos una barra de control vertical, definimos sus parámetros en (1).
En (2) determinamos la posición inicial del cursor, en (3) con la primitiva cartel escribimos el valor inicial de la barra de control y en (4) dibujamos la "ventana" por donde veremos pasar al satélite.

```
para satel
decir 2 fforma 16 frumbo 90          (5)
sp centro mt                         (5)
repetir 50 [estamparforma 22 lista ((-60) + azar 180) ((-30) +
azar 130)]                           (6)
decir 1 fforma 1
sp fpos [30 50] mt frumbo 90         (7)
fin
```

En satel, en (5) presentamos al satélite, en (6) estampamos las estrellas entre x=-160 y x=120 alrededor del mundo que será ubicado en (7). La línea (6) va toda en el mismo renglón.

```
para movi
decir 2 libre
gravitatorio [30 50] 300        (8)
flimites [-300 300] 600 600 (9)
cuandoizq [inic interrumpir]    (10)
cuandoder [inic interrumpir]
cuandoabajo [inic interrumpir]
cuandoarriba [inic interrumpir]
fareavisible [-60 130] 200 160  (11)
fvel :vl
tocarsiempre  1                 (12)
actuar
fin
```

*En este procedimiento le asignamos todos los parámetros necesarios al actor 2 (satélite) para su control y función. En (8) utilizamos la primitiva **GRAVITATORIO** la cual produce sobre el satélite una simulación de una atracción de tipo gravitatoria con centro en el punto indicado (ver más detalles en la Ayuda de Primitivas).*
En (9) determinamos que el satélite tenga un área de movimiento limitada dentro del entorno Logo, ya que si no controlamos esto el satélite en algún momento puede irse de la ventana y producirá un error de programa al llegar fuera de los límites permitidos.
En (10) le asignamos eventos para que cuando choque con los límites permitidos pare la animación vuelva a su punto de origen.
En (11) determinamos el área visible por donde veremos pasar al satélite.
Con (12) hacemos que comience a sonar el archivo WAV ininterrumpidamente hasta darle una instrucción de parada (pararsonido).

```
para inic
decir 2 centro frumbo 90
facelx 0 facely 0 fvel 0
pararsonido
fin
```

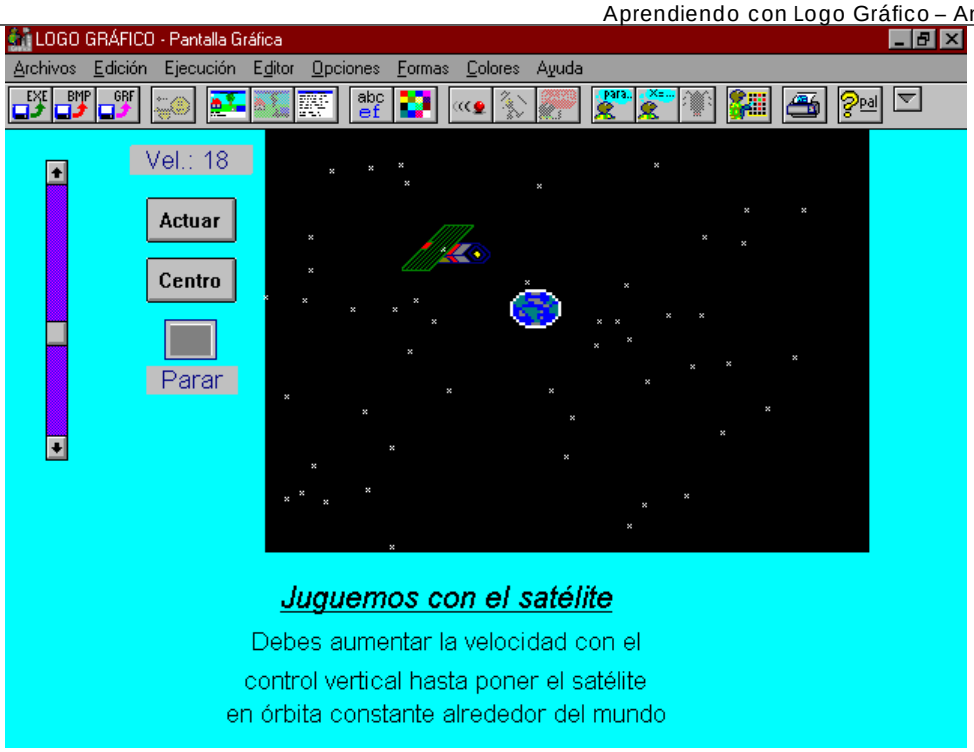
Aquí reasignamos los valores de aceleración y velocidad iniciales cada vez que para, y va al punto de partida, además de apagar el sonido en caso de que se produzca un evento.

```
para inicio
bp ot quitarbarrah
pentera libre
cargarformas 'satelite
fcf 17
si haysonido? cargarsonido 'satelite 1 (13)
hacer "vl 0
control
botones
carteles
satel
fin
```

Este es el procedimiento de inicio del programa, lo único a comentar es que en (13) se interroga al sistema si posee plaqueta de sonido, si es "verdad cargará el archivo satelite.wav.

Quedan dentro del programa tres procedimientos más que son *ventana, botones y carteles*, con los cuales ubicamos y definimos las funciones de los botones, dibujamos un cuadrado negro que simulará la ventana por donde veremos pasar el satélite y rotulamos un breve texto explicativo del juego, que veremos a continuación.

```
para ventana
sp fpos [-60 -30] frumbo 0
frelleno 1 0
cp repetir 2 [ad 160 de 90 ad 200 de 90]
rellenar
fin

para botones
sp fpos [-85 80]
fboton 'Actuar 'movi 30 15
sp fpos [-85 60]
fboton 'Centro 'inic 30 15
sp fpos [-85 40]
fbotonlogo [pararsonido interrumpir] 3 2
fcolorletras 1
cartel [··Parar··] [-85 27] 14
fin

para carteles
ftipo 'arial
festiloletras [cursivo subrayado]
fcolorletras 0
fanchotipo 1.3
decir 12 fforma 1 ot sp frumbo 90
fpos [0 -46]
rotularc [Juguemos con el satélite]
festiloletras [normal]
fanchotipo 1
cartel [ Debes aumentar la velocidad con el] [0 -60 ] 17
cartel [ control vertical hasta poner el satélite] [0 -73] 17
cartel [ en órbita alrededor del mundo] [0 -84] 17
fcolorletras 1
fanchotipo 1
fin
```


LOGO GRÁFICO - Pantalla Gráfica
Archivos Edición Ejecución Editor Opciones Formas Colores Ayuda
Vel.: 18
Actuar
Centro
Parar
Juguemos con el satélite
Debes aumentar la velocidad con el
control vertical hasta poner el satélite
en órbita constante alrededor del mundo

CAPITULO 11

PUERTOS Y MANDOS

Trataremos de explicar brevemente como se usan las distintas primitivas de LOGO GRAFICO para manejar los puertos paralelos, series, y una novedad, con respecto a otros Logos, LOGO GRAFICO tiene la posibilidad de controlar y programar el Joystick que se conecta a la PC.

El uso de Joystick en Logo posibilita la creación de juegos de video y realización de software donde se pone a prueba la habilidad, velocidad y destreza.

Dentro de la programación de animación secuencial o simultánea, el poder controlar el comportamiento de la tortuga o los actores es una poderosa herramienta.

También el control de los puertos serie y paralelo, da a LOGO GRAFICO la posibilidad de comunicación con el exterior de la computadora, y desarrollar programas de control de interfaces para la aplicación de robótica en educación.

Tal vez les resulte extraña estas instrucciones a aquellas personas que no están familiarizadas con el control de los puertos, pero es importante para aquellas que sí tienen interés en la posibilidad de comunicación a través de Logo con el mundo exterior.

§11.1 PUERTO SERIE

A diferencia de LogoWriter el seteo de los puertos serie (COM1 y COM2) se realiza desde dentro de LOGO GRAFICO, con las primitivas ***ABRIRCANAL*** y ***CERRARCANAL***.
Cabe aclarar que en LOGO GRAFICO llamamos canal a los puertos serie.

```
abrircanal 2 1200 0 8 1
```

con esta orden seteamos el puerto serie 2 (COM2) a 1200 baudios de velocidad, paridad 0, ocho bits, bit de parada.

Con ***CERRARCANAL*** cerramos el canal de comunicación, a la vez que eliminamos cualquier dato o carácter pendiente que se halle en el bus. Simplemente dando esta instrucción.

```
cerrarcanal
```

Las primitivas para leer o escribir el canal abierto con ***ABRIRCANAL*** son:

```
leercanal
```

Devuelve el carácter que se halle en el canal abierto, en decimal.

```
esccanal 30
```

escribe en el canal abierto un dato en decimal.

Algo útil para programadores más avanzados es la primitiva ***CARPENDIENTE?*** que devuelve con VERDAD o FALSO si existe un carácter aun no leído en el canal abierto.

§11.2 PUERTO PARALELO

Estas primitivas se asemejan mucho a las primitivas de manejo de puertos de LogoWriter. Tanto en su función como en su sintaxis.

Con estas primitivas podemos leer y escribir cualquier dirección de memoria baja de la PC en un rango de 0 a 64 Kb.

Esta especialmente diseñada para los puertos paralelos, pero para aquellos acostumbrados a usarlas con los puertos serie, también son útiles.

```
escpuerto 888 25
```

Con esta instrucción estoy escribiendo en el puerto paralelo LPT1 (888) el dato 25 en decimal, ya que la dirección de memoria asignada generalmente para el LPT1 es 888 en decimal o 3F8 en hexadecimal.

IMPORTANTE

No debe ser usada esta primitiva por aquellas personas que no tiene un conocimiento exacto de los puertos, ya que un error en la dirección donde escribimos un dato, podemos hacer que LOGO GRAFICO e inclusive Windows se bloquee.

Para leer un dato del puerto solo escribimos:

```
leerpuerto 888
```

donde 888 es la dirección del puerto de donde deseamos leer un dato.

La mayoría de las interfaces de robótica educativa que no funcionan con una determinada versión de Logo, utilizan estos puerto y estas primitivas para su control y manejo.

§11.3 MANEJO DE JOYSTICK

El manejo y programación del Joystick es más sencillo y accesible para los programadores o usuarios de Logo, ya que no necesitan una interface de control específica, sólo necesitan poder conectar un Joystick a su PC (es importante que su Windows o DOS reconozca el puerto o entrada de Joystick).

Con un Joystick podemos guiar un actor, mover la tortuga, controlar a través de los botones de mando un evento en animación simultánea, etc.
Existen seis primitivas para el control del Joystick, sólo citaré las usadas en el ejemplo dado a continuación.

EXIXTEMANDO? Con esta primitiva podemos saber si LOGO GRAFICO reconoce la presencia del Joystick conectado a nuestra PC.

AJUSTARMANDO Pone a cero la posición de la palanca del Joystick. Es importante ya que toma la posición [0 0] el la posición actual de la palanca. Sirve para setear a cero cuando la palanca está en reposo o no está accionada.

POSMANDO Devuelve la posición (como con la primitiva ***POS***) en forma de lista, con X e Y.

El primer valor de la lista indica los movimientos de la palanca hacia la derecha o hacia la izquierda, según devuelva un número negativo o positivo.

El segundo de la lista indica un movimiento de la palanca hacia adelante o hacia atrás, también según sea positivo o negativo.

BOTONMANDO? Devuelve VERDAD o FALSO si a sido accionado alguno de los dos botones que generalmente tienen los Joystick de PC.

Nota: cada una de las primitivas tiene un argumento que le indica con cual de los dos mandos o Joystick estamos trabajando o van dirigidas las órdenes, ya que con LOGO GRAFICO podemos trabajar con dos Joystick.

El ejemplo que se da es para usar en animación secuencial, con el uso de una recursión sencilla. Aquí movemos la tortuga por la pantalla, con o sin pluma según le demos la instrucción a través de los botones del Joystick.

```
para inicio
bp mt pentera fcolor 2 sp
ajustarmando 1    (1)
movi
fin

para movi
si (pri posmando 1) > 20 de 5         (2)
si (pri posmando 1) < (-20) iz 5      (2)
si (ult posmando 1)  > 20 ad 2        (2)
si (ult posmando 1) < (-20) ad 2      (2)
si botonmando? 1 1 cp                 (3)
si botonmando? 1 2 sp
si tecla? parar                       (4)
movi
fin
```

1- Es importante poner a cero la posición inicial del Joystick antes de comenzar a usarlo en el control de un programa.

2- Aquí preguntamos por el primero o último valor de la lista que nos devuelve la primitiva posmando, en base al valor dado le asignamos a la tortuga el rumbo correspondiente a la dirección en que está inclinada la palanca, y la hacemos avanzar dos pasos.

3- preguntamos si está accionado en botón 1 del mando 1 (recuerden que podemos trabajar con dos Joystick), si es así, le ponemos pluma a la tortuga. En la línea siguiente hacemos lo mismo, pero preguntamos por el botón 2 del mando 1, y si está accionado le sacamos la pluma a la tortuga.

4- Le damos una instrucción de control para cortar la recursión, esto es, si oprimimos una tecla del teclado paramos la recursión.

Nota: Preguntamos si el valor es mayor a... para darle cierto margen de control al programa, ya que hay Joystick muy sensibles que pueden devolver valores al más mínimo movimiento. Ese valor se deberá determinar según el comportamiento de cada uno.

Esto es un muy sencillo ejemplo de como podemos usar el Joystick para jugar o hacer programas con Logo. Es importante leer las Ayudas del programa para cada una de las primitivas mencionadas, y también para conocer las otras que no han sido usadas en este ejemplo.