PROGRAMACION MODULAR EN EL MICROMUNDO "ESPACIO-3D": EMPLEO DE ALGUNAS UTILIDADES.

RICARDO LUENGO GONZALEZ (*) (+) (&).
MERCEDES MENDOZA GARCIA (&).
LUIS MARQUEZ ZURITA (&).
CIPRIANO SANCHEZ PESQUERO(&).
TEODORO GONZALEZ BRAVO (*).

RESUMEN

En este artículo mostramos un micromundo que llamaremos "ESPACIO-3D" para el Object-LOGO (sobre un ordenador Macintosh) implementado por nuestro equipo de trabajo. Se hace un estudio de la programación modular, justificando la conveniencia de utilizarla en nuestro micromundo y se ofrecen un conjunto de utilidades para facilitar la programación modular en el espacio tridimensional. A la vez pretendemos ilustrar con ejemplos el manejo del micromundo, el empleo de las utilidades y la integración de todas las herramientas anteriores en gráficos tridimensionales construidos a partir de la idea de "sintonicidad corporal".

En el lenguaje Logo se dispone habitualmente de un subconjunto de primitivas muy potente llamado "micromundo de la tortuga" que permite realizar cualquier dibujo en el plano. En los últimos años numerosos compañeros interesados en LOGO han visto la necesidad de evadirse del plano y poder describir y realizar formas tridimensionales. (Reggini, 1985).

Nuestro equipo de trabajo también abordó este tema (Luengo y otros, 1987) con un trabajo en el que se realizaba una implementación de un micromundo tridimensional a partir de la versión (en dos dimensiones) LOGO -1

(*) Departamento de Didáctica de las Ciencias Experimentales y de las Matemáticas de la Universidad de Extremadura.

(+)ICE de la Universidad de Extremadura.

(&) Grupo BETA.

para APPLE-lle, traducida por Paquita Cortada y suministrada por MICPE S.A. En dicho trabajo se exponía el problema de la representación de formas tridimensionales sobre una pantalla plana, fundamentando a partir de una proyección cónica, el establecimiento de ejes propios y las ecuaciones de cambio para realizar los movimientos espaciales. A partir de estas ecuaciones se implementaba el micromundo "ESPACIO-3D" y se ofrecían algunos ejemplos de aplicación.

Los problemas ya se indicaban al describir el funcionamiento del micromundo y radicaban en la poca memoria disponible. Esta circunstancia impedía añadir nuevas primitivas que hubieran sido muy útiles pero que, al agotar la memoria, imposibilitarían al usuario la realización de cualquier trabajo en el ESPACIO-3D.

Actualmente se ha popularizado el uso de los ordenadores de la gama Macintosh y hay diversos LOGOs para estos ordenadores. Quizá han influido decisivamente varios factores: por una parte las características francamente competitivas del sistema (tanto en Hardware como en Software) y, por otra, los precios ventajosos para el campo de la enseñanza y en concreto para la Universidad debido al consorcio Universitario con APPLE. En todo caso hoy podemos disponer de equipos y programas, que superan ampliamente a sus predecesores y ofrecen nuevas posibilidades para trabajar en nuestro campo.

Uno de los LOGOs de los que actualmente se puede disponer en España para Macintosh, en el Object-LOGO. Este intérprete dobla en número de primitivas y posibilidades a su antecesor el LOGO-II para APPLE-lle. Es un nuevo dialecto de LOGO diseñado para facilitar la "programación orientada al objeto". En Object-LOGO un objeto es una entidad que tiene sus propias versiones de ciertas funciones y variables. Uno se puede comunicar con un objeto, llamándolo para que ejecute algunas expresiones habituales de LOGO. Cada objeto tendrá un estado y un comportamiento. Uno de esos "objetos" es la tortuga (determinada por su estado) a la que podemos asignar tareas por medio de expresiones escritas en LOGO (que regirán su comportamiento).

En este artículo pretendemos, fundamentándonos en nuestro trabajo anterior y remitiéndonos a él en los aspectos teóricos, implementar "ESPACIO-3D" para el Object-LOGO y dotarlo de un conjunto de utilidades que faciliten la programación modular en el espacio tridimensional. A la vez pretendemos ilustrar con ejemplos el manejo del micromundo, el empleo de las utilidades y la integración de todas las herramientas anteriores en gráficos tridimensionales constuidos a partir de la idea de "sintonicidad corporal". (Papert, 1981). Así para la realización de una figura espacial en "ESPACIO-3D" no será necesario conocer las leyes de la representación en un sistema formal, si-

no solo los movimientos que debería realizar un sujeto para describirlo con su propio cuerpo.

En el apartado 2, se describen las condiciones de contorno que han determinado la implementación de "ESPACIO-3D" sobre esta versión de LOGO, la estructura, y el listado de procedimientos que constituyen las primitivas del micromundo junto con un comentario pormenorizado de las acciones que realizan.

El apartado 3 contiene un enfoque sobre la programación en el "ESPACIO-3D" basado en las ideas actuales sobre programación estructurada y un conjunto de ejemplos que pueden ser representativos del "estilo" de programación con el que sintonizamos.

Los apartados 4 y 5 contienen respectivamente el conjunto de utilidades programada y ejemplos de cómo utilizarlos en nuestro micromundo. Por último, en el apartado 6 incluimos las referencias.

2. MICROMUNDO "ESPACIO-3D".

2.1. *Condiciones de contorno.*

Se ha implementado el micromundo en un ordenador Macintosh ED con 512, de memoria RAM y unidad de discos interna de 800K. El LOGO utilizado es de la casa Coral Software: Object-LOGO por Gary L. Drescher (versión de 1985).

Para utilizar este LOGO hay que conocer las técnicas básicas de uso de Macintosh: Hacer "clic", "doble click", presionar, arrastrar, manejo de ventanas, seleccionar y la edición de texto. Además deben conocerse: el uso del "Finder" (para manejar documentos, carpetas, aplicaciones, etc) y la gestión de discos.

Una vez arrancado el sistema, por medio de cualquier disco de arranque, puede introducirse el intérprete. El "disco de arranque" puede a su vez contener al intérprete y aún deja bastante memoria libre al usuario para almacenar sus propios procedimientos. En nuestro caso hemos incluido en el mismo disco un sistema que ocupa 267 k. configurado con un Finder, el System, portapapeles y el documento ImageWriter para la Comunicación con la impresora. Además hemos incluido el documento MacsBug, bien conocido de los programadores de "Mac", que apaga la pantalla cuando el ordenador deja de funcionar unos minutos.

La aplicación Object-LOGO ocupa en disco 283 k, por tanto el conjunto (sistema + aplicación) tiene 557 k (incluyendo el formateado) con lo que nos

queda libre en el disco una capacidad de almacenamieno de 243 k. Esta capacidad es muy suficiente para nuestros propósitos pero de todas formas, si se dispone de una segunda unidad de discos podemos dedicar una al sistema y a la aplicación dejando la segunda íntegramente para el almacenamiento de datos. Otra opción, si se dispone de un Macintosh Plus (con 1 Mb. de memoria interna) sería establecer un "RAM-DISK" en el que podría incluirse el sistema, dejando la unidad de disco para la aplicación y los documentos generados en Object-LOGO.

La aplicación Object-LOGO:

Procederemos ahora a describir la aplicación en los aspectos que la pueden diferenciar de versiones anteriores de LOGO a fin de situar al lector ante las nuevas posibilidades que presenta Object-LOGO.

Una vez abierta la aplicación por el procedimiento habitual (doble click en el icono, o abrir en el menú de archivo una vez seleccionado) entramos en el menú principal que contiene cinco grupos de comandos:

- El primero es la típica manzana de APPLE que contiene los comandos de manejo de escritorio (información acerca de la aplicación, el apuntador, block de notas, calculadora, panel de control, el selector y el teclado).
- El segundo grupo corresponde a los comandos de archivo; contiene los necesarios para abrir un nuevo fichero, cerrarlo, editarlo, grabarlo en la unidad de disco, etc. También contiene los comandos de impresión y la opción de salida.
- El tercer grupo (Edición) tiene los comandos de edición típicos de Macintosh: cortar, copiar, pegar, borrar, deshacer y seleccionar todo. Además aquí se incluyen las opciones de cambios de tipos y mostrar el portapapeles.
- El siguiente grupo incluye comandos propios de LOGO. Unos son relativos al manejo de "espacio de trabajo" (cargar o guardar) y otros relativos al control en la ejecución (stop, pausa, continuar, etc).
- El último grupo de comandos del menú principal (windows) permite el manejo de las ventanas. Todas las ventanas tienen los aditamentos típicos para su manejo: cuadro de cierre, barras de título, cuadro de ampliación, barras de desplazamiento y cuadro de forma/tamaño. Siempre habrá una ventana activa, de las muchas que puede haber en pantalla. Para activar una ventana se puede proceder de dos maneras: Haciendo "click" sobre cualquier punto de ella o bien escogiendo en el menú principal (windows) la ventana deseada.

Con la apertura de Object-LOGO aparece la primera ventana: El "listener". Se trata de una ventana de texto, mediante la cual nos comunicamos directamente con LOGO a través de sus primitivas.

Al teclear en el listener una orden de gráficos se activa una nueva ventana (la de gráficos) y se ejecutan en ella las órdenes cursadas, sin perjucio de que sigamos escribiendo en el listener que sigue abierto en pantalla.

Otra ventana nos muestra el Editor. Se activa mediante la primitiva EDIT seguida de comillas y el nombre del procedimiento que se desea editar. Su manejo es similar al de otros LOGOs de APPLE, por lo que no nos extendemos en su descripción.

También podemos abrir directamente un fichero almacenado en el disco mediante otra ventana. Para ello usamos el comando EDITFILE del menú de Achivo. (Por ejemplo en ese fichero podríamos tener almacenado el trabajo realizado en una sesión anterior).

Si hemos ido abriendo todas esas ventanas ahora tendremos en el menú principal, en la opción windows, comandos que corresponderán uno a cada ventana abierta en pantalla. Por medio de ellos podemos determinar, según las necesidades del momento, cual debe ser la ventana activa.

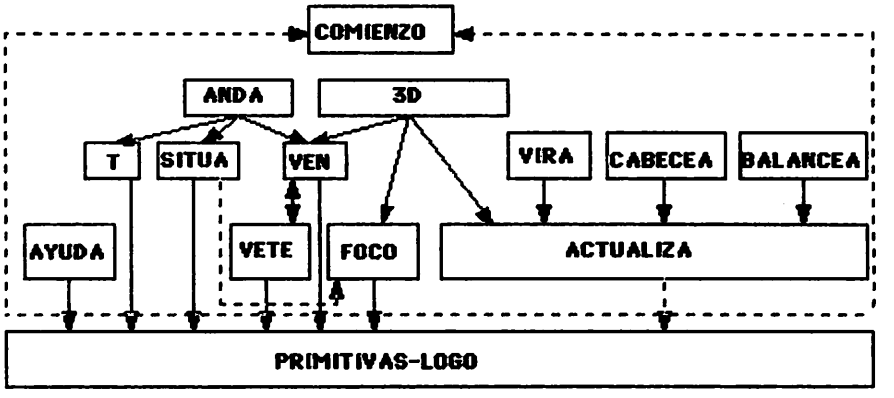
Todas las ventanas de texto permiten trasvase de información por medio del "portapapeles" y se pueden usar todas las técnicas de edición típicas de Macintosh.

No ocurre esto con la ventana de gráficos que no obedece órdenes más que del listener. Por ello los gráficos no se editan (o imprimen) a través del menú principal, lo cual no es problema ya que se pueden obtener por impresora a través de un "volcado de pantalla".

Salvo estas peculiaridades (casi todas son ventajas) este LOGO se puede usar fácilmente por un conocedor de versiones anteriores de APPLE como LOGO-II sobre APPLE-IIe.

2.2 *Implementación de "ESPACIO-3D".*

El micromundo programado consta de un conjunto de procedimientos LOGO cuya estructura se representa en el siguiente diagrama:



ESQUEMA 1

Una vez cargado el micromundo mediante LOAD del menú principal LOGO pasan al espacio de trabajo todos los procedimientos. El procedimiento COMIENZO tiene la misión de "esconderlos" bajo el espacio de trabajo, inicializar el sistema y limpiar la pantalla. Cuando todas esas tareas se han efectuado aparece el mensaje de saludo "BIENVENIDO A LOGO TRIDIMENSIONAL 3D.RL" y ya se pueden usar las nuevas primitivas.

El procedimiento 3D es el encargado de establecer las condiciones iniciales del sistema, llamando a la pantalla gráfica y efectuando un primer borrado. Además hace aparecer la tortuga en el centro geométrico de la ventana de gráficos teniendo la pluma en posición DOWN (baja) dispuesta para dejar rastro en los desplazamientos.

La tortuga tridimensional está representada por un pequeño rombo y señala únicamente el lugar en que se encuentra. A primera vista no proporciona información sobre su dirección y sentido, pero este problema se resolverá en el apartado 4 en el que implementamos las utilidades relativas a la orientación.

Los procedimientos VIRA, CABECEA y BALANCEA permiten efectuar los tres giros básicos en el espacio: (Luengo y otros, 1987)

a) BALANCEA: Eje de giro es el eje longitudinal de la tortuga.

b) CABECEA: Eje de giro es el eje transversal (perpendicular al anterior).

c) VIRA: Eje de giro es perpendicular al plano formado por los anteriores.

Cada uno de ellos calcula los nuevos cosenos directores, una vez efectuado el giro y actualiza los valores de las variables globales que los almacenan quedando la tortuga "virtualmente enfocada" en la nueva dirección deseada.

El procedimiento ANDA, que se ocupa de los desplazamientos, junto con los anteriores permite efectuar cualquier movimiento en el espacio tridimensional. Para la denominación de las primitivas anteriores hemos seguido la "traducción castellana normalizada de LOGO" (Asociación LOGO, 1985).

Para subir y bajar la pluma, no hemos implementado nuevas primitivas pues sirven igualmente las correspondientes del plano.

ANDA incluye en su definición a los procedimientos SITUA, VEN y T. SITUA admite como entradas las coordenadas especiales de un punto, calcula la proyección cónica, teniendo en cuenta el "foco" (establecido por el procedimiento 3D en principio o por el propio usuario mediante el procedimiento FOCO) situa a la tortuga en el punto proyección calculado sobre la pantalla.

VEN hace aparecer la tortuga tridimensional, dibujada por el procedimiento T. Basta modificar este último si se quiere otra forma para la tortuga. VETE hace desaparecer la tortuga. Ambas órdenes (VEN y VETE) pueden ser usadas en cualquier momento, tanto a modo directo como en el modo programado.

ACTUALIZA efectúa el cambio de los valores de los cosenos directores después de haber efectuado un giro o cuando se inicializa el sistema.

FOCO es el procedimiento que establece el "foco" en la proyección cónica y puede ser usado en todo momento (por ejemplo para observar un mismo objeto desde distintos puntos de vista). Más adelante pondremos ejemplos de cómo es de gran utilidad para observar los efectos de "gran angular" o "teleobjetivo", o simplemente para movernos alrededor de un objeto tridimensional.

El foco se mantiene mientras no inicialicemos el sistema con 3D o no lo cambiemos nosotros mismos.

El conjunto se completa con AYUDA, procedimiento independiente, que nos informa en el listener de los comandos básicos del micromundo.

A continuación incluimos un listado completo de los procedimientos implementados:

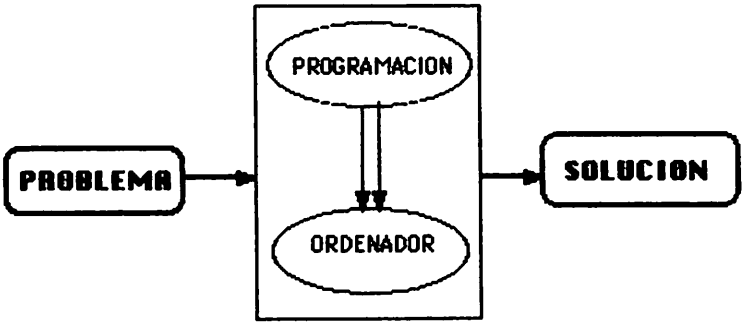
```
TO COMIENZO
BURYALL ERALL 3D
CLEARTEXT
PR [BIENVENIDO A LOGO TRIDIMENSIONAL 3D.RL]
RECYCLE
END

TO VEN
LOCAL "P MAKE "P BL LIST PENMODE "||
MAKE "VEN "TRUE PD T
RUN BL LIST WORD "PEN FIRST :P "||
END

TO CABECEA :BETA
LOCAL "C LOCAL "S MAKE "C COS :BETA MAKE "S SIN :BETA
MAKE "G11 (:C11 * :C - :C31 * :S)
MAKE "G12 (:C12 * :C - :C32 * :S)
MAKE "G13 (:C13 * :C - :C33 * :S)
MAKE "G31 (:C31 * :C + :C11 * :S)
MAKE "G32 (:C32 * :C + :C12 * :S)
MAKE "G33 (:C33 * :C + :C13 * :S)
ACTUALIZA
END

TO T
RT 45
REPEAT 4 [FD 7 LT 90]
LT 45
END

TO VIRA :BETA
LOCAL "C LOCAL "S MAKE "C COS :BETA MAKE "S SIN :BETA
MAKE "G11 (:C11 * :C + :C21 * :S)
MAKE "G12 (:C12 * :C + :C22 * :S)
MAKE "G13 (:C13 * :C + :C23 * :S)
MAKE "G21 (:C21 * :C - :C11 * :S)
MAKE "G22 (:C22 * :C - :C12 * :S)
MAKE "G23 (:C23 * :C - :C13 * :S)
ACTUALIZA
END

TO ACTUALIZA
MAKE "C11 :G11 MAKE "C12 :G12 MAKE "C13 :G13
MAKE "C21 :G21 MAKE "C22 :G22 MAKE "C23 :G23
MAKE "C31 :G31 MAKE "C32 :G32 MAKE "C33 :G33
END

TO FOCO :M1 :M2 :M3
MAKE "MX :M1 MAKE "MY :M2 MAKE "D :M3
END

TO SITUA :X :Y :Z
SETPOS SE (:MX + ((:X - :MX) * :D) / (:D - :Z)) !
          !(:MY + ((:Y - :MY) * :D) / (:D - :Z))
END
```

```
TO AYUDA
PR []
PR [PRIMITIVAS 'ESPACIO.3D.RL']
PR [_____]
PR [] PR [3D - - - - > BORRA TODO, ESTABLECE EL FOCO INICIAL,]
PR [.....INICIALIZA VARIABLES Y HACE APARECER LA TORTUGA !
    !TRIDIMENSIONAL.]
PR [] PR [VEN - - - - > HACE VISIBLE LA TORTUGA]
PR [] PR [VETE - - - - > HACE INVISIBLE LA TORTUGA]
PR [] PR [FOCO - - - - > SITUA EL FOCO EN CUALQUIER PUNTO DEL !
          !ESPACIO <X,Y,Z>]
PR [] PR [ANDA - - - - > CAMINA TANTOS PASOS COMO INDICA SU ENTRADA]
PR [] PR [BALANCEA - - - - > GIRO DE EJE LONGITUDINAL A LA TORTUGA]
PR [] PR [CABECEA - - - - > GIRO DE EJE TRANSVERSO ( PERPENDICULAR !
          ! AL ANTERIOR>]
PR [] PR [VIRA - - - - > GIRO DE EJE PERPENDICULAR AL PLANO FORMADO !
          ! POR LOS ANTERIORES]
END

TO VETE
LOCAL "P MAKE "P BL LIST PENMODE "||
MAKE "VEN "FALSE PE T
RUN BL LIST WORD "PEN FIRST :P "||
END

TO 3D
HT CS PD FOCO 200 200 600
MAKE "G11 0 MAKE "G12 1 MAKE "G13 0
MAKE "G21 -1 MAKE "G22 0 MAKE "G23 0
MAKE "G31 0 MAKE "G32 0 MAKE "G33 1
ACTUALIZA
MAKE "X 0 MAKE "Y 0 MAKE "Z 0
VEN
END

TO BALANCEA :BETA
LOCAL "C LOCAL "S MAKE "C COS :BETA MAKE "S SIN :BETA
MAKE "G21 (:C21 * :C + :C31 * :S)
MAKE "G22 (:C22 * :C + :C32 * :S)
MAKE "G23 (:C23 * :C + :C33 * :S)
MAKE "G31 (:C31 * :C - :C21 * :S)
MAKE "G32 (:C32 * :C - :C22 * :S)
MAKE "G33 (:C33 * :C - :C23 * :S)
ACTUALIZA
END

TO ANDA :L
LOCAL "P MAKE "P BL LIST PENMODE "||
IF :VEN = "TRUE [PE T RUN BL LIST WORD "PEN FIRST :P "]
MAKE "X (:X + :L * :C11)
MAKE "Y (:Y + :L * :C12)
MAKE "Z (:Z + :L * :C13)
SITUA :X :Y :Z
IF :VEN = "TRUE [PD T RUN BL LIST WORD "PEN FIRST :P "]
END
```

3. PROGRAMACION MODULAR EN EL ESPACIO.3D.

3.1. *Conceptos básicos.*

Comencemos diciendo que, en principio, podemos considerar la programación como un camino que nos conduce desde un problema que tengamos planteado a su solución; se llega a ésta con el concurso de una máquina y a través de la ejecución de un programa. (Ver esquema 2).



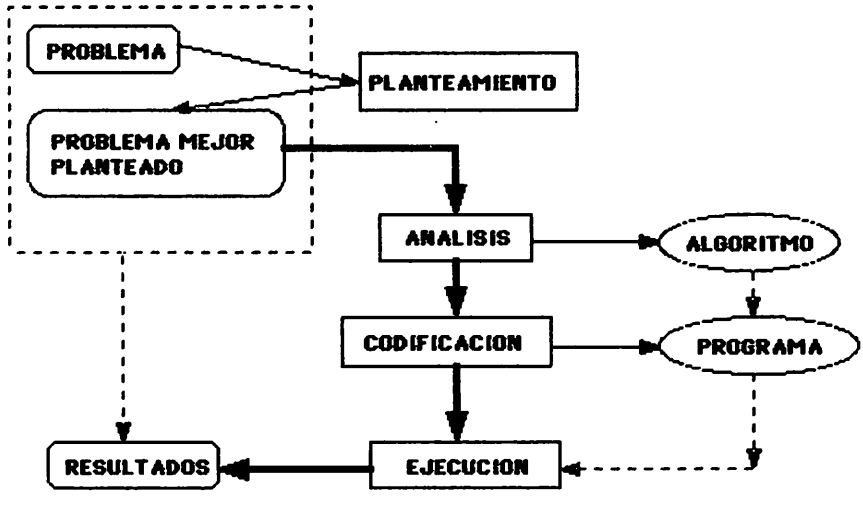
ESQUEMA 2

Generalmente la reflexión inicial sobre el problema, nos puede llevar a un replanteamiento y a una mayor comprensión del mismo. (Ver esquema 3).

Después de este proceso (planteamiento) podemos llegar a un problema mejor planteado, con el que comenzar el análisis. El análisis es la parte más delicada del proceso y la más importante y desemboca en la concrección de un conjunto de acciones secuenciadas (algoritmo) que resuelven el problema planteado. Es prácticamente independiente del lenguaje y del ordenador que vayamos a utilizar, aunque la representación del algoritmo sea más conveniente hacerla con determinados diagramas según el lenguaje que se vaya a emplear. (Por ejemplo "organigramas" para lenguajes no estructurados y "diagramas de árbol" para los lenguajes estructurados).

El siguiente proceso es el de Codificación que consiste en expresar el algoritmo en un lenguaje de programación concreto (Programa). Por último llegamos a la Ejecución en la que interviene directamente el ordenador que ejecuta los pasos especificados en el programa y nos produce unos resultados. Es evidente que esta descripción se ha hecho a grandes rasgos y no se han detallado

algunas partes del proceso (como por ejemplo la depuración de errores de la que hablaremos más adelante), pero por ahora es suficiente para nuestros propósitos.



ESQUEMA 3

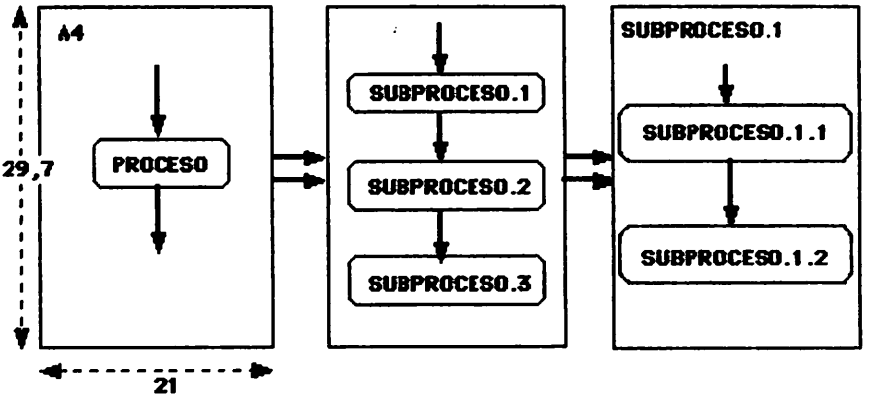
Un programa debería estar escrito con claridad, ser veloz y utilizar la mínima memoria posible; sin embargo, en la mayoría de los casos estas cualidades son antagónicas y hay que buscar una solución de compromiso. En todo caso en los programas educativos (en los que no siempre se va a tratar un volumen grande de datos) antepondríamos sobre todo la claridad. En este sentido hemos de señalar que la mayoría de los programadores no han tenido esa preocupación y es frecuente ver en las publicaciones programas construidos con una estructura tan enrevesada que es casi imposible que alguien los entienda excepto el autor (a veces intencionadamente para protegerse de copias ilegales, etc.)

Programación estructurada. Surge como una reacción a la situación anterior y está ligada a la idea de "programación sin GOTO". (Esta instrucción tan empleada es, a juicio de muchos autores de reconocido prestigio, la culpable de la estrutura ininteligible de muchos programas).

En esencia se trata de descomponer un proceso en subprocesos y con una marcha descendente (del más completo al más simple) escribir los programas utilizando ciertas estructuras de base y sólo ellas. Por ejemplo: IF THEN EL-

SE (SI... ENTONCES... SINO), DO UNTIL (HACER HASTA QUE...), DO WHILE (HACER MIENTRAS QUE...). Se ha demostado que se puede escribir cualquier programa utilizando estas estructruras únicamente.

Otra reacción a la costumbre de representar en un organigrama los programas completos plagados de temibles "GOTO" (formato "sábana") fue la aparición del método "A4" que sintoniza de alguna manera con los principios de la programación estructurada; consiste en escribir los programas con un método descendente obligándose a no utilizar nunca un papel mayor del formato "A4" (21 × 29,7 cms) aceptando un procedimiento de "bloques" (o "cajas negras") que se detallan en diagramas separados. (Ver esquema 4).



ESQUEMA 4

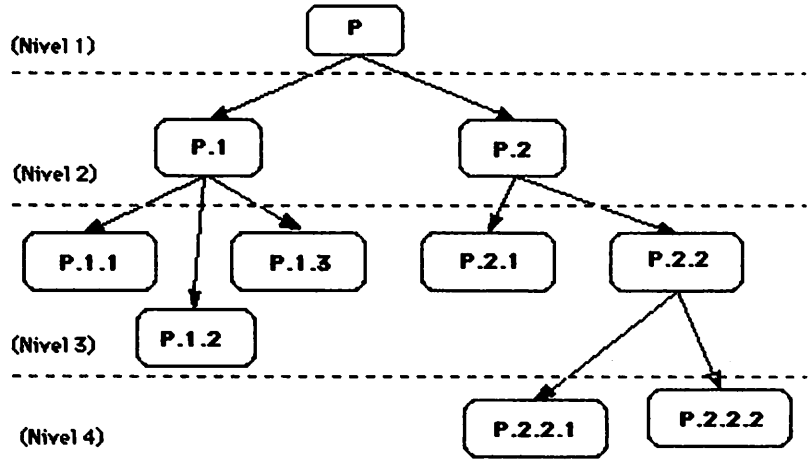
El teorema de Bohm y Jacopini afirma que bastan tres estructuras básicas para todos los tratamientos posibles en el momento de escribir un programa: La secuencia, la alternativa y la repetición. La secuencia consiste en un conjunto de órdenes que deben ejecutarse en orden establecido. La alternativa es una estructura básica en la que el control de ejecución ordena ejecutar un conjunto de órdenes, o bien otro conjunto de órdenes diferentes (o ninguna) según se cumpla o no cierta condición. Por último en la repetición se trata de efectuar un "bucle" (o "iteración") mientras se cumpla una condición preestablecida.

3.2. *Programación modular.*

Hay lenguajes que permiten y facilitan este tipo de programación (programación estructurada) porque incorporan estas estructuras en el propio len-

guaje (por ejemplo PASCAL y LOGO). En algunos lenguajes (como los citados) existen unas estructuras llamadas *procedimientos* (PROCEDURES), construidas a partir de las estructuras básicas, que son independientes entre sí pero pueden ser "llamadas para realizar una tarea. Cada procedimiento puede resolver un pequeño problema de los resultantes al subdividir el problema principal.

En esencia se trata de dividir el problema planteado (P) en varios subproblemas (P.1) y (P.2) —por ejemplo— (ver esquema 5).



ESQUEMA 5

A continuación concentrándonos en (P.1) —y olvidando incluso la existencia de (P) y (P.2)— tratar de subdividirlo en varios problemas más sencillos (por ejemplo (P.1.1), (P.1.2) y (P.1.3) y así seguir hasta que (P_{ij}) sean fácilmente abordables (o prácticamente triviales para el programador). Ahora olvidarnos de (P), (P.1) y de toda esa rama del "árbol" que vamos construyendo para concentrarnos única y exclusivamente en (P.2). Procederemos igualmente subdividiendo (P.2) en subproblemas más sencillos y éstos, a su vez, en otros. Así llegaremos a problemas (P_{mn}) fáciles de abordar y que no merezca la pena subdividir hasta completar todo el árbol y terminar este proceso de análisis.

Cuando desarrollamos un Proyecto-LOGO, una vez concluido el ANALISIS, es fácil codificar el algoritmo contenido en el árbol: un procedimiento

puede resolver cada uno de los pequeños problemas que constituyen las "hojas del árbol" (Pij). Ensamblando procedimientos a manera de "módulos" (de ahí el nombre de PROGRAMACION MODULAR) se pueden ir resolviendo los problemas más complejos hasta llegar, en este proceso de SINTESIS, a resolver el problema (P) planteado inicialmente.

De todas formas una vez obtenido el "árbol", que inicialmente creemos resuelve el problema, se puede recorrer de dos maneras diferentes que llevan implícitos dos estilos distintos de programación.

- *Estilo Planificador (TOP-DOWN).*

Se trata de recorrer el árbol de una manera descendente, partiendo de lo general hacia lo particular. (O lo que es lo mismo de lo más global a lo más detallado). La filosofía del método se basa en no complicarse, en principio, en resolver los detalles hasta esbozar las líneas principales de la evolución lógica del problema. Para usar este método hay que tener mucha práctica y dominio del lenguaje, pues no se pueden probar los resultados hasta no haber programado todos los procedimientos.

- *Estilo "Bricolage" (BOTTON-UP).*

En este caso se recorre el árbol al programar de una manera ascendente, comenzando con la resolución de los pequeños problemas que constituyen las "hojas" del árbol y ensamblando los procedimientos (a menera de un puzzle) para constituir superprocedimientos que se apoyan en los anteriores, procediendo así hasta recorrer todas las "ramas" y llegar a resolver el problema planteado (P). La filosofía del método está en comenzar obteniendo éxitos al resolver los pequeños problema abordables, y ensamblar los módulos, trabajando con la seguridad de que las "piezas" con las que construimos ya las hemos probado y funcionan. Este estilo de programar no exige tanto dominio del lenguaje y permite emplear los procedimientos para explorar el "micromundo" en el que se trabaje. A veces se pueden emplear dichos procedimientos en otros contextos muy distintos de los que originaron su programación.

En principio, en las primeras exploraciones de un "micromundo" nos mostramos partidarios de sugerir a nuestros alumnos que actúen con un estilo "bricolage". Hay muchas razones para sostener esta afirmación pero quizás una de las que más nos convencen es que los niños actúan así cuando construyen su propio conocimiento. Marvin Minsky y Seymour Papert vienen elaborando desde hace años una "teoría social de la mente" en la que está patente el principio de modularización. La idea central consiste en considerar que un sistema cualquiera funciona sobre la base de subsistemas relativamente inde-

pendientes entre sí. La génesis del pensamiento respondería tambien a este principio, según J. Piaget; los pensamientos se elaboran de manera modular a partir de la combinación de pequeñas entidades de conocimiento. La mente, más que un inmenso procedimiento de gran complejidad, podría asemejarse a un conjunto numerosísimo de procedimientos simples que interaccionan entre sí. (Papert, S. 1981).

En este sentido sintonizamos con Reggini cuando afirma:

"Trabajando con subprocedimientos es fácil descubrir los inevitables errores que siempre se cometen al abordar programas extensos. Las partes pueden probarse, depurarse y modificarse independientemente. La posibilidad de definición de procedimientos facilita el desarrollo de proyectos basados en diseños modulares" (Reggini, Horacio C. 1982).

En general nos mostramos partidarios, con Cynthia Solomon, de fomentar en cada uno de nuestros alumnos su propio estilo de programación. Para ello es bueno comenzar a explorar en modo "bricolage" pero es bueno también mostrarles otros estilos y otros puntos de vista al enfocar un problema. (Solomon, Cynthia. 1987).

3.3. *Ejemplos de programación modular en el ESPACIO.3D.*

- *Primer ejemplo: La silla.*

Si en el caso del plano es necesario programar de una forma modular en el ESPACIO.3D se hace imprescindible. Comencemos por intentar programar *la silla* que mostramos en la figura 1. Se ha escogido un diseño sencillo cuyas proporciones se pueden poner en función de una única variable (L) que nos puede servir de factor de escala. El ángulo menor (interno) del rombo que forman los laterales del asiento lo fijamos en 75.°

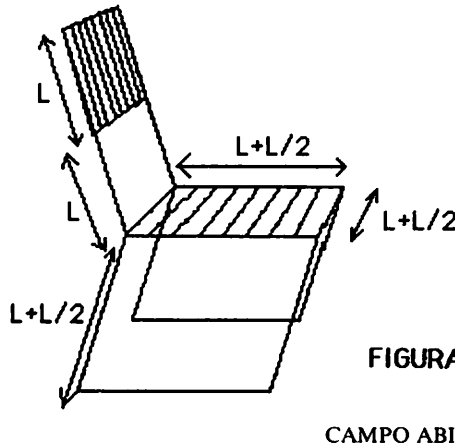
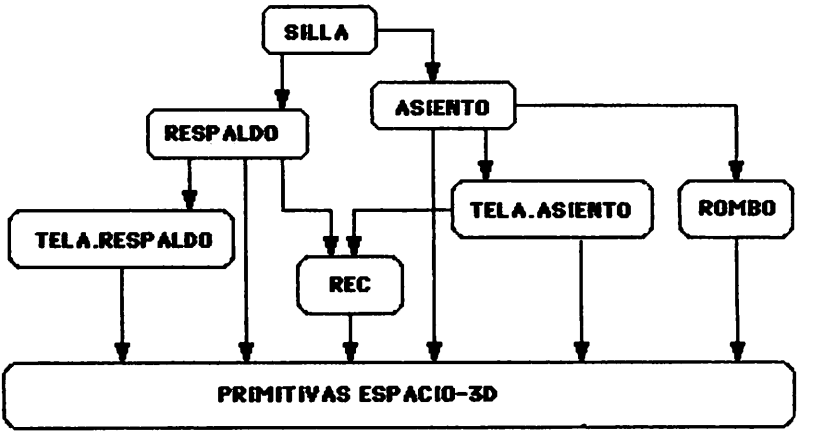


FIGURA .1

Empleando el método de análisis indicado en el apartado anterior el dibujo de la silla se puede descomponer en dos partes bien diferenciadas: respaldo y asiento.

Concentrémonos en la construcción del respaldo; podemos dibujarlo a partir de un procedimiento que dibuje rectángulos (que nos puede servir para otros rectángulos, una vez programado) y también necesitamos otro procedimiento que nos dibuje la tela del respaldo. Estos problemas son ya tan sencillos que no merece la pena descomponerlos más y los procedimientos que los resuelven se pueden construir a partir de las propias primitivas del ESPACIO.3D.

Vamos a construir ahora el asiento. Necesitaremos un procedimiento que dibuje un rombo de 1 + 1/2 de lado y ángulo menor interno de 75°, que nos servirá para las patas, y otro procedimiento que dibuje la tela del asiento. Este último puede servirse del rectángulo antes programado para el respaldo. *El árbol de procedimientos* (esquema 6) nos muestra la estructura del superprocedimiento silla que resuelve el problema planteado.



ESQUEMA 6

A continuación adjuntamos el *conjunto de procedimientos* que dibujan "silla":

```
TO TELA.RESPALDO :L
LOCAL "T MAKE "T :L + :L / 2
REPEAT 4 [ANDA :L VIRA |-90| ANDA :T / 8 !
! VIRA |-90| ANDA :L VIRA 90 ANDA :T / 8 VIRA 90]
VIRA 90 ANDA :T VIRA -90
END

TO ASIENTO :L
PD VIRA -15 ROMBO :L
ANDA :L VIRA 15 CABECEA 90 TELA.ASIENTO :L
ANDA :L CABECEA -90 VIRA 165 ANDA :L VIRA 180
ROMBO :L ANDA :L VIRA 30
END

TO TELA.ASIENTO :L
REC :L :L
REPEAT 4 [ANDA :L VIRA |-90| ANDA :L / 8 !
! VIRA |-90| ANDA :L VIRA 90 ANDA :L / 8 VIRA 90]
VIRA 90 ANDA :L VIRA -90
END

TO REC :L1 :L2
REPEAT 2 [ANDA :L1 VIRA |-90| ANDA :L2 VIRA |-90|]
END


TO VUELVE.INICIO :K
BALANCEA 90 VIRA 15
CABECEA -90 ANDA :K + :K / 2 CABECEA 90
VIRA 135 ANDA :K + :K / 2 VIRA -165
END
                                        135
TO RESPALDO :L
REC :L * 2 :L + :L / 2 ANDA :L REC :L :L + :L / 2
TELA.RESPALDO :L
VIRA 180 ANDA :L VIRA 180
END

TO SILLA :K
ASIENTO :K + :K / 2
BALANCEA -90 RESPALDO :K
VUELVE.INICIO :K
END

TO ROMBO :L
REPEAT 2 [ANDA :L VIRA |-75| ANDA :L VIRA |-105|]
END
```

Los procedimientos se han programado de forma que sean generales, dentro de las especificaciones del diseño. Se ha procurado que el procedimiento silla sea *transparente*. (Se llaman procedimientos transparentes aquellos que después de su ejecución dejan la tortuga en el mismo estado en que comenzó el procedimiento —misma posición e idéntica orientación—). Para

conseguir que silla sea transparente se ha añadido el procedimiento VUELVE. INICIO que lo único que hace es volver al estado en que se encontraba a la tortuga después de haber hecho el asiento y el respaldo. Este procedimiento no hubiera sido necesario si hubiéramos dispuesto de una primitiva que colocara a la tortuga en cualquier punto del espacio y que implementaremos en el apartado de utilidades. En ese caso hubiera bastado con almacenar al comienzo el estado de la tortuga y enviarla al final a ese mismo estado.

Es interesante observar la *serie de sillas obtenidas por cuatro balanceos* de 90° con los correspondientes desplazamientos para que las figuras no se superpongan.

En la figura 2, se parte de la posición normal (con la tortuga mirando hacia la parte positiva del eje Y, que es la posición con la que partimos al comenzar a trabajar después de actuar el procedimiento 3D. Las sillas b) y d) corresponden a las posiciones de espaldas y de frente (respectivamente), mientra que a) y c) corresponden a sillas situadas en posición lateral derecha e izquierda (respectivamente) según miramos.

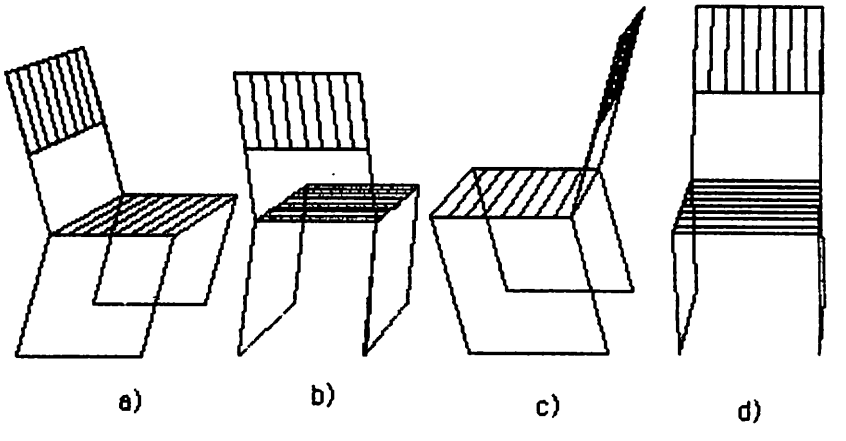


FIGURA 2

En la figura 3, en a) se parte de un escorzo con asiento a la derecha (según miramos) con un balanceo previo de 15 grados; en b) tras el primer balanceo vemos la silla de espaldas; en c) después de un segundo balanceo vemos la silla en escorzo con asiento situado hacia la izquierda y por último en d) vemos la silla de frente. La disminución de tamaño desde a) hasta d) es consecuencia del alejamiento paulatino del foco que se produce al desplazar la tortuga desde la

posición en que comienza a pintar la silla hasta la posición de comienzo del dibujo de la silla siguiente.

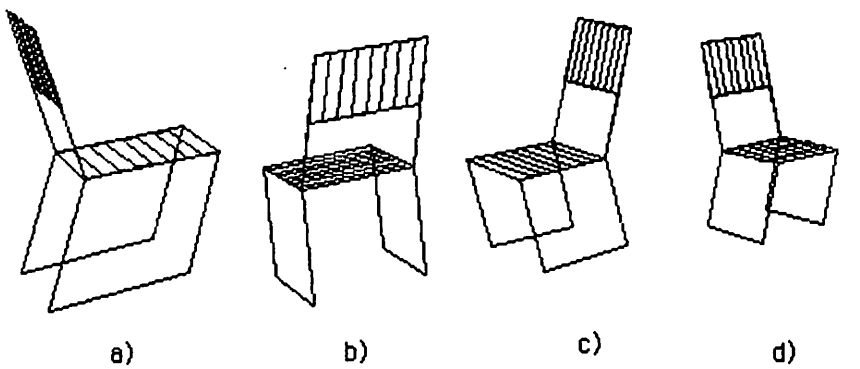


FIGURA 3

Silla tiene además como entrada la variable "K que puede actuar como *factor de escala,* como podemos ver en las figuras 4.a) y 4.b) en las que hemos dibujado varias sillas de distinto tamaño.

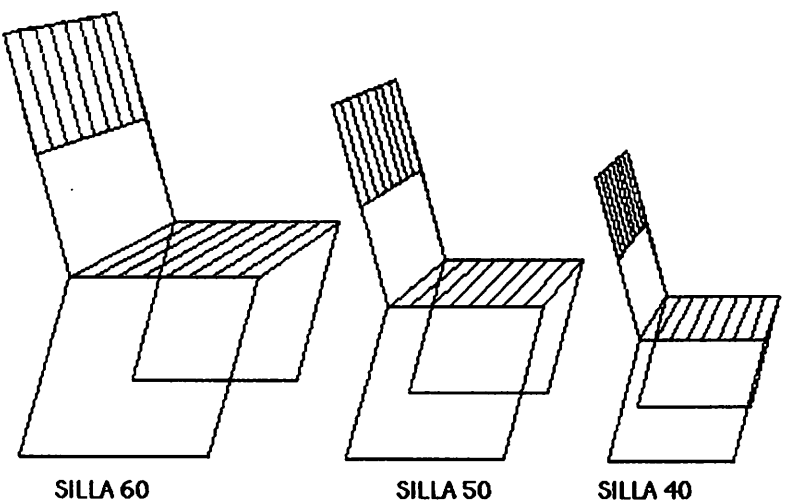


FIGURA 4.a).

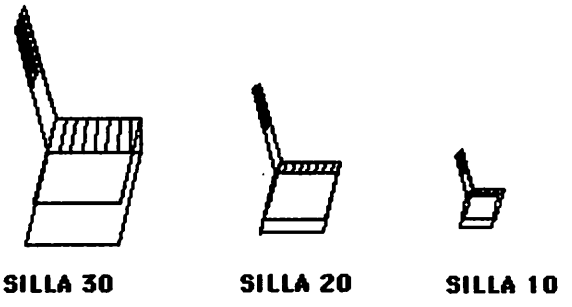


FIGURA 4.b).

Efectos "gran angular" y "teleobjetivo": La posibilidad de poder mover el foco nos da la oportunidad de observar un objeto quieto desde distintos puntos de vista. El foco inicial que se establece por defecto (mediante el procedimiento 3D) está situado a (200 200 600). En la Figura 5 a) mostramos la silla inicial con ese foco. Veamos que ocurre si lo trasladamos a (200 200 200), es decir si nos acercamos la tercera parte de la distancia del foco inicial a la pantalla de TV en la coordenada Z —figura 5 b)—:

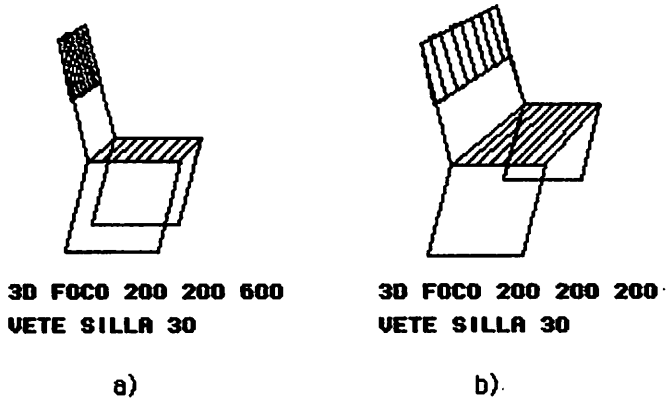
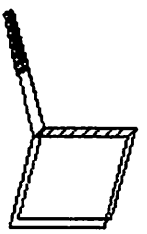


FIGURA 5

Se observa el efecto fotográfico propio del "gran angular", que reforma los objetos: los cercanos aumentándolos y los lejanos disminuyéndolos. Obsérvese la deformación del cuadrado que constituye el asiento y la disminución del rombo de las patas más alejadas respecto del de las más cercanas.

El efecto de "teleobjetivo" se puede observar en la figura 6 a) y b). En el primer caso se ha producido un alejamiento de tres veces la distancia del foco inicial a la pantalla. No se observan deformaciones (únicamente los efectos de perspectiva cónica). En el segundo caso además de alejarnos nos hemos elevado por la parte positiva del eje de las Y, y podemos observar el objeto "a vista de pájaro".



```
3D FOCO 200 200 1800
VETE SILLA 30
```

a

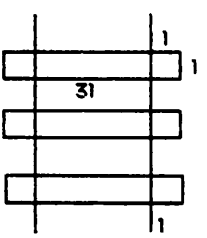
```
3D FOCO 400 2000 800 VETE
BALANCEA.-15 SILLA 20
```

b

FIGURA 6

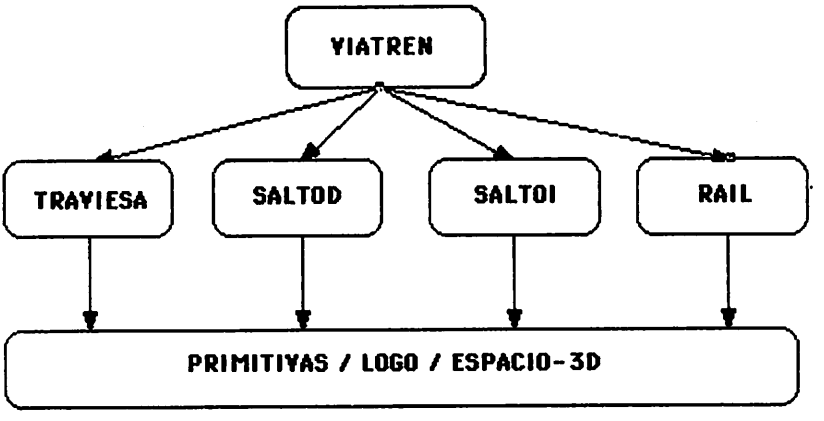
- *Segundo ejemplo: La vía de tren.*

En este segundo caso se ha escogido un figura muy sencilla, pero que puede ilustar con mucha efectividad el efecto propio de la perspectiva cónica.



Para que el diseño resulte sencillo hemos escogido como referencia de tamaño el ancho de la traviesa (1). Este parámeto va a actuar como factor de escala. El largo de la misma será de 5 veces el ancho.

El árbol de procedimientos de VIATREN se muestra en el esquema 7. El procedimiento fundamental es el llamado TRAVIESA, que no es más que un rectángulo programado en el espacio con las dimensiones acordes con el diseño y con "l" actuando como factor de escala. Los procedimientos SALTOI y SALTOD permiten efectuar un salto en escuadra, con la misma escala y a la izquierda y derecha rspectivamente. El procedimiento RAIL dibuja un rail, adecuado al diseño y con el mismo factor de escala que los anteriores. El superprocedimiento VIATREN se ha proyectado con dos entradas: una de ellas es el factor de escala y otra nos permite determinar el número de traviesas que se deseen.



ESQUEMA 7

El superprocedimiento VIATREN repite un número de veces igual al de traviesas un salto en escuadra hacia la izquierda seguido del dibujo de la propia traviesa y un salto hacia la derecha; con esto a la vez va construyendo uno de los railes con todas las traviesas. Por último dibuja el otro rail, calculando su longitud en función del número de peldaños y hace regresar la tortuga al punto de partida para conseguir que el procedimiento sea transparente. Es evidente que habría otras muchas maneras de resolver el problema, y puede que haya varias interesantes. Por esto debemos de fomentar distintos enfoques entre nuestros alumnos y hacerles ver varias posibilidades muy distintas y que todas ellas resuelven el problema.

Adjuntamos a continuación el conjunto de procedimientos que resuelven el proyecto "VIATREN" en 3D:

```
TO VIATREN :N :L
REPEAT :N [SALTOI :L TRAVIESA :L SALTOD :L]
RAIL :N :L
END

TO TRAVIESA :L
REPEAT 2 [ANDA :L VIRA [-90] ANDA :L * 5 VIRA [-90]]
END

TO RAIL :N :L
ANDA :L VIRA -90 PU ANDA :L * 3 VIRA -90 PD ANDA (:L * :N * 2) + :L
VIRA -90 PU ANDA :L * 3 VIRA -90 PD
END

TO SALTOD :L
PD VIRA -90 ANDA :L VIRA 90 ANDA :L
END

TO SALTOI :L
PD ANDA :L VIRA 90 ANDA :L VIRA -90
END
```

En este caso se echa en falta una primitiva del espacio que nos informara del lugar en el que se encuentra la tortuga, para almacenarlo (en las correspondientes variables) y poder volver a ese punto una vez terminada una tarea. Un caso particular, pero de mucha utilidad, sería un procedimiento que nos permitiera volver al centro geométrico de espacio-3D.

Sería interesante en ciertos casos conocer la orientación de la tortuga; en esta caso se ha salvado la situación procurando programar con procedimientos transparentes y teniendo en cuenta que el problema planteado se encuentra en un plano, aunque ese plano pueda colocarse en diversas posiciones espaciales, o manteniéndolo fijo seamos nosotros los que observemos desde distintos puntos de vista variando el foco.

Vamos a observar distintas vías dibujadas con el procedimiento VIATREN:

Comencemos por dibujar una vía de tren con treinta traviesas y una longitud de 25 unidades para "l", con el foco en la posición inicial (por defecto) que nos proporciona el micromundo ESPACIO-3D (200 200 600) (figura.7).

Podemos variar ahora nuestro punto de vista situándonos en (50 150 600), con lo que nos habremos puesto casi enfrente de la vía; no hemos variado sin embargo la distancia del foco a la pantalla. Observaremos, como corresponde a la perspectiva cónica, la ilusión de convergencia de los railes que en el espacio son paralelos (figura 8).

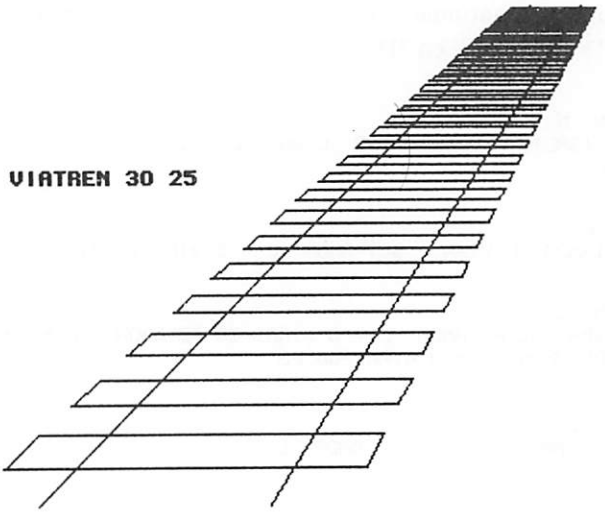


FIGURA.7

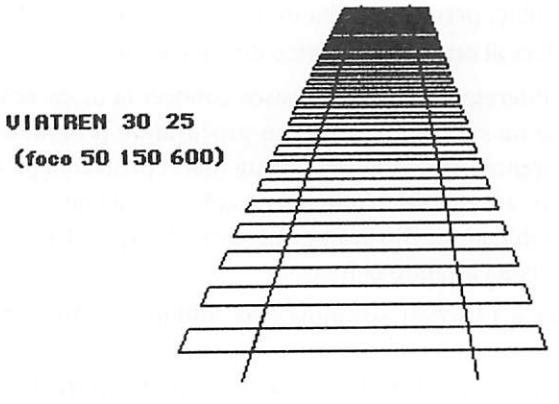


FIGURA.8

Situemos ahora el foco en (-30 300 600), con lo cual, manteniendo la distancia del foco al plano de la pantalla, nos hemos elevado sobre la vía (al aumentar la coordenada Y) con lo que vemos la vía a "vista de pájaro" (Figura.9).

VIATREN 18 25
(FOCO -30 300 600)

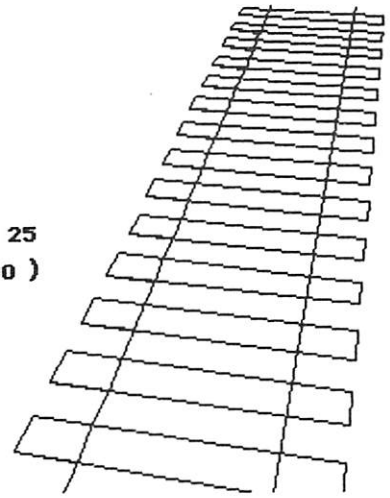


FIGURA.9

Por último hemos creído interesante presentar el caso de bajar drásticamente el foco (casi hasta el plano que contiene a la vía) y situarnos de frente para observar la vista "rasante" de la figura 10.

VIATREN 28 40
(FOCO 20 50 700)

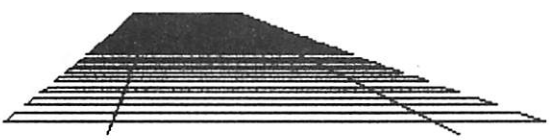


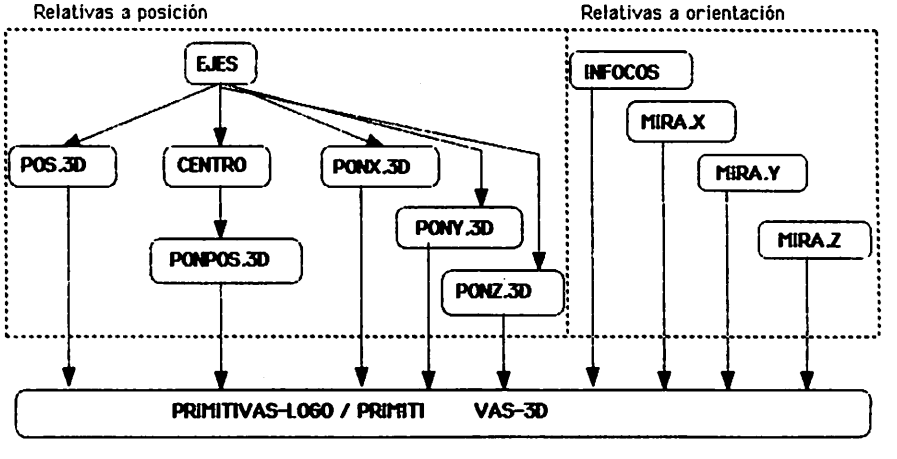
FIGURA.10

4. CONJUNTO DE UTILIDADES: "U-3D"

A lo largo de los apartados anteriores hemos puesto de manifiesto la necesidad de disponer de un conjunto de procedimientos auxiliares homólogos a otros frecuentemente utilizados en el micromundo gráfico bidimensional de LOGO que facilitarán la programación en "ESPACIO-3D".

En nuestro caso, con Object-LOGO, desaparecerán los problemas de memoria puestos de manifiesto cuando se utiliza LOGO I o LOGO II sobre APLLE-lle (Luengo, R y otros, 1987); en Macintosh hay suficiente memoria RAM como para tener a la vez el micromundo y el conjunto de utilidades. Además las cargas adicionales de procedimientos no borran la pantalla de gráficos, lo que constituye una indudable mejora.

El conjunto de utilidades, que por brevedad desde ahora llamaremos "U-3D", consta de procedimientos relativos a la manipulación de la posición y otros que actúan sobre la orientación de nuestro elemento móvil. El esquema.8 muestra la estructura de este conjunto de utilidades:



ESQUEMA.8

Relativos a la posición hemos implementado el procedimiento informativo POS.3D que nos indica las coordenadas actuales de la tortuga (X Y Z); en cuanto a la orientación INFOCOS nos proporciona la matriz que contiene los cosenos directores que determinan la orientación de los ejes propios de la tortuga.

• EJES ——→ Muestra los ejes cartesianos con origen el centro del espacio 3D (ejes fijos). No modifican el estado de la tortuga.

El resto de las órdenes, que constituyen el conjunto de utilidades, sirven para cambiar el estado de la tortuga en el espacio:

• PONPOS.3D ——→ Coloca la tortuga en la posición del espacio determinada por una lista que contiene las coordenadas cartesianas (X Y Z).

• PONX.3D, PONY.3D, PONZ.3D ——→ Coloca la tortuga en la coordenada especificada (x, y o z respectivamente) manteniendo constante el valor de las otras dos.

• CENTRO ——→ Coloca la tortuga en el Origen (Centro) de los ejes fijos.

Las órdenes precedentes no actúan sobre la pluma, por lo que dejará "rastro" o no según las condiciones anteriores; tampoco sirve si la tortuga es visible o no.

En cuanto a las utilidades que provocan cambio en la orientación de la tortuga hemos implementado tres órdenes básicas: (ver figuras 11 y 12).

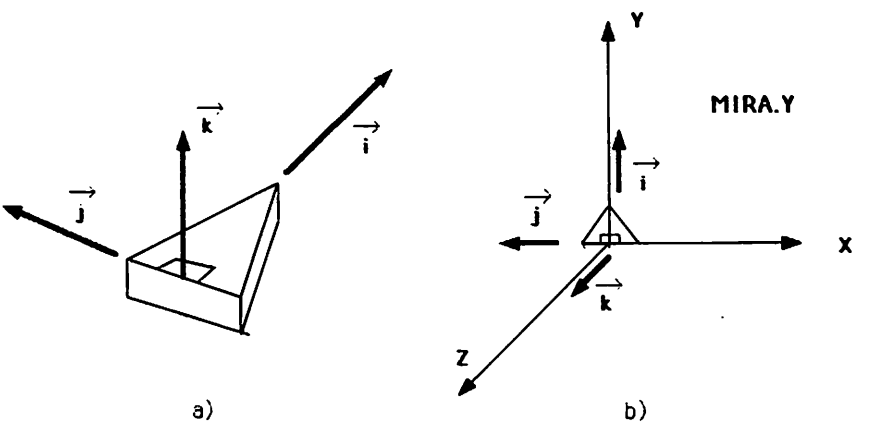


FIGURA.11

Tomemos como ejes móviles solidarios con la tortuga los indicados en la figura.11 a). Con esa disposición de ejes se han calculado la ecuaciones que rigen al micromundo ESPACIO-3D (ver Luengo y otros, 1987). A orientación por defecto que tiene la tortuga al comenzar una sesión es la indicada en la figura 11 b): En ella el vector unitario *i* va en la dirección y sentido positivo del eje de las Y, y el vector unitario *k* va en la dirección y sentido positivo del eje de las Z. Construiremos un procedimiento (MIRA.Y) que permita orientar a

la tortuga en esa disposición de ejes en cualquier momento de nuestra sesión de trabajo.

Análogamente se han construido otros dos procedimientos que se llaman MIRA.X y MIRA.Z con las disposiciones relativas de los ejes móviles respecto de los fijos que se pueden observar en las figuras 12 a) y b). Con el conjunto de los tres procedimientos tenemos la posibilidad de orientar a la tortuga en las tres direcciones básicas de los ejes de referencia.

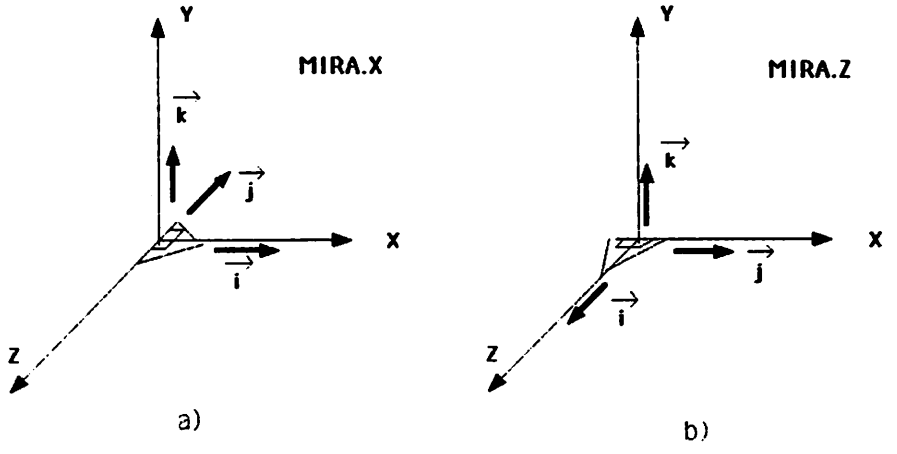


FIGURA.12

A continuación incluimos un listado de los procedimientos implementados que constituyen "U-3D":

a) Relativos a la posición:

```
TO PONZ.3D .N
LOCAL "P MAKE "P BL LIST PENMODE "||
IF :VEN = "TRUE [PE T ( RUN BL LIST WORD "PEN FIRST :P " )]
MAKE "Z :N
SITUA :X :Y :Z
IF :VEN = "TRUE [PD T ( RUN BL LIST WORD "PEN FIRST :P " )]
END

TO PONY.3D :N
LOCAL "P MAKE "P BL LIST PENMODE "||
IF :VEN = "TRUE [PE T ( RUN BL LIST WORD "PEN FIRST :P " )]
MAKE "Y :N
SITUA :X :Y :Z
IF :VEN = "TRUE [PD T ( RUN BL LIST WORD "PEN FIRST :P " )]
END

TO POS.3D
OP (LIST :X :Y :Z)
END
```

```
TO PONPOS.3D :LIST
LOCAL "P MAKE "P BL LIST PENMODE "||
IF :VEN = "TRUE [PE T PD ( RUN BL LIST WORD "PEN FIRST :P " )]
MAKE "X FIRST :LIST
MAKE "Y FIRST BF :LIST
MAKE "Z LAST :LIST
SITUA :X :Y :Z
IF :VEN = "TRUE [PD T ( RUN BL LIST WORD "PEN FIRST :P " )]
END

TO CENTRO
PONPOS.3D [0 0 0]
END

TO EJES
LOCAL "P MAKE "P BL LIST PENMODE "||
LOCAL "LUGAR MAKE "LUGAR POS.3D
LOCAL "TORTUGA MAKE "TORTUGA :VEN
IF :VEN = "TRUE [VETE]
PU CENTRO PD
PONX.3D 200 CENTRO
PONY.3D 200 CENTRO
PONZ.3D 200 PU
PONPOS.3D :LUGAR
RUN BL LIST WORD "PEN FIRST :P "||
IF :TORTUGA = "TRUE [VEN]
END

TO PONX.3D :N
LOCAL "P MAKE "P BL LIST PENMODE "||
IF :VEN = "TRUE [PE T ( RUN BL LIST WORD "PEN FIRST :P " )]
MAKE "X :N
SITUA :X :Y :Z
IF :VEN = "TRUE [PD T ( RUN BL LIST WORD "PEN FIRST :P " )]
END
```

b) Relativos a la orientación

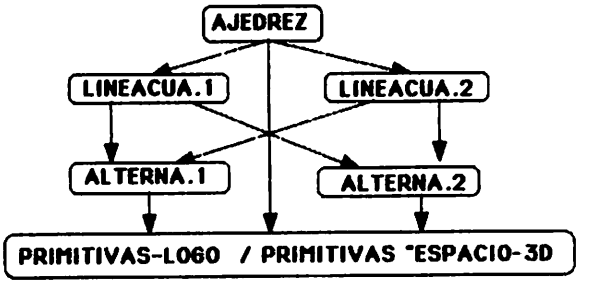
```
TO MIRA.X
MAKE "G11 1 MAKE "G12 0 MAKE "G13 0
MAKE "G21 0 MAKE "G22 0 MAKE "G23 -1
MAKE "G31 0 MAKE "G32 1 MAKE "G33 0
ACTUALIZA
END

TO MIRA.Y
MAKE "G11 0 MAKE "G12 1 MAKE "G13 0
MAKE "G21 -1 MAKE "G22 0 MAKE "G23 0
MAKE "G31 0 MAKE "G32 0 MAKE "G33 1
ACTUALIZA
END

TO MIRA.Z
MAKE "G11 0 MAKE "G12 0 MAKE "G13 1
MAKE "G21 1 MAKE "G22 0 MAKE "G23 0
MAKE "G31 0 MAKE "G32 1 MAKE "G33 0
ACTUALIZA
END

TO INFOCOS
PR (LIST "[ :C11 :C12 :C13 "])
PR (LIST "[ :C21 :C22 :C23 "])
PR (LIST "[ :C31 :C32 :C33 "])
END
```

5. EJEMPLOS.

Vamos a ver unos ejemplos de utilización de "U-3D". Para ello vamos a tomar una figura que se preste adecuadamente a la observación de los efectos tridimensionales: Un tablero de ajedrez.

Es evidente que hay muchas maneras de programar el tablero de ajedrez. Nosotros hemos comenzado, siguiendo una programación modular, por los procedimientos ALTERNA.1 y ALTERNA.2 que dibujan una línea de trazos alternos con cuatro trazos correspondientes a los cuadros negros y de longitud la misma que el tablero. La primera comienza con el trazo y la segunda comienza sin trazo. Los dos procedimientos se implementan a partir de las primitivas de ESPACIO-3D y de las primitivas-LOGO.

A continuación hemos construido LINEACUA.1 y LINEACUA.2 que realizan una línea de 8 cuadros alternos (blancos y negros) cada una, empezando LINEACUA.1 por cuadro negro y LINEACUA.2 por cuadro blanco. Tanto una como otra se apoyan en los procedimientos anteriores con la estructura que puede observarse en el esquema 9. Por último AJEDREZ realiza el tablero completo.



ESQUEMA.9

A continuación mostramos el conjunto de procedimientos "AJEDREZ-3D":

```
TO AJEDREZ :L
LOCAL "LUGAR MAKE "LUGAR POS.3D
REPEAT 4 [LINEACUA.1 :L LINEACUA.2 :L]
PU PONPOS.3D :LUGAR PD
END

TO ALTERNA.1 :L
REPEAT 4 [PD ANDA :L PU ANDA :L]
END

TO ALTERNA.2 :L
REPEAT 4 [PU ANDA :L PD ANDA :L]
END
```

```
TO LINEACUA.2 :L
REPEAT :L / 2 [ALTERNA.2 :L VIRA [-90] ANDA 1]
VIRA [-90] ALTERNA.1 :L VIRA 90 ANDA 1 VIRA 90]
END
TO LINEACUA.1 :L
REPEAT :L / 2 [ALTERNA.1 :L VIRA [-90] ANDA 1]
VIRA [-90] ALTERNA.2 :L VIRA 90 ANDA 1 VIRA 90]
END
```

Comencemos la ejecución del procedimiento AJEDREZ, trazando los ejes cartesianos y a continuación un ajedrez que se verá de frente a nosotros (Ver Figura 13.a) contenido en el plano (X,Y).

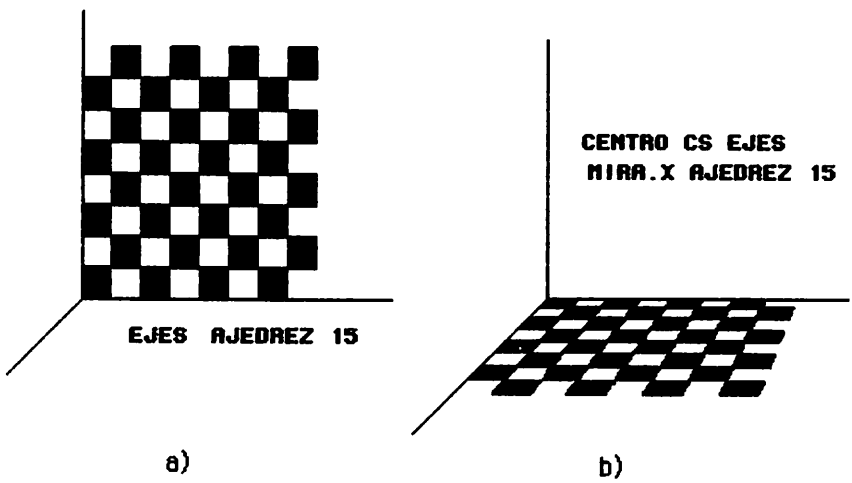
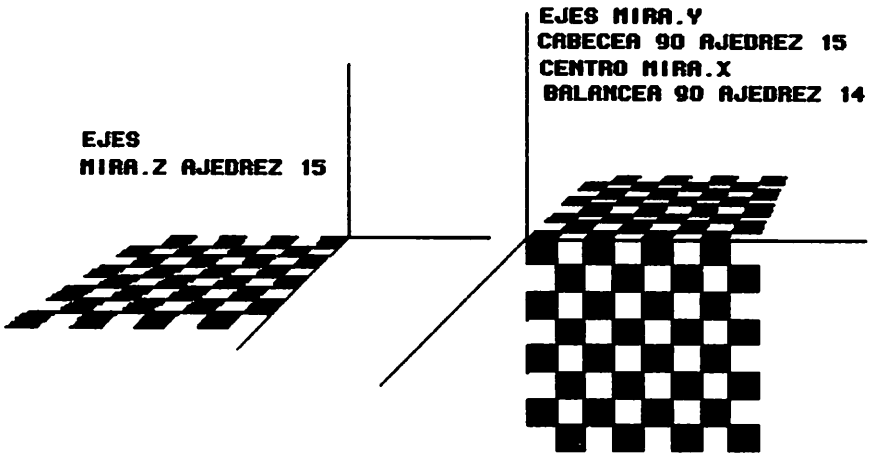


FIGURA 13

A partir de MIRA.X y MIRA.Y podemos fácilmente situarlo en cualquier plano. Por ejemplo en la figura 13 b) lo hemos situado en el plano horizontal (X,Z) región positiva de las X y de las Z, mientras que en la figura 14 a) está en el plano horizontal pero ahora en la región negativa de las X y positiva de las Z.

Estando AJEDREZ programado como procedimiento transparente y gracias a las utilidades U-3D se puede manipular con gran facilidad. En la figura 14.b) podemos ver dos tableros contenidos en dos planos que forman un ángulo recto.



a) b)

FIGURA.14

Por último, y para terminar estos ejemplos, vamos a simular un cubo cuyas caras sean tableros de ajedrez. Hemos dibujado los ejes (ver figura 15) y empleado casi todos los recursos disponibles en U-3D. Además hemos variado el foco para poder ver con comodidad las tres caras del cubo.

```
FOCO 400 200 600
CENTRO CS EJES PU
PONPOS.3D [96 0 96]
MIRA.X VIRA 180
PD AJEDREZ 12
MIRA.Y VIRA 180
AJEDREZ 12 BALANCEA -90
VIRA 90 AJEDREZ 12
```

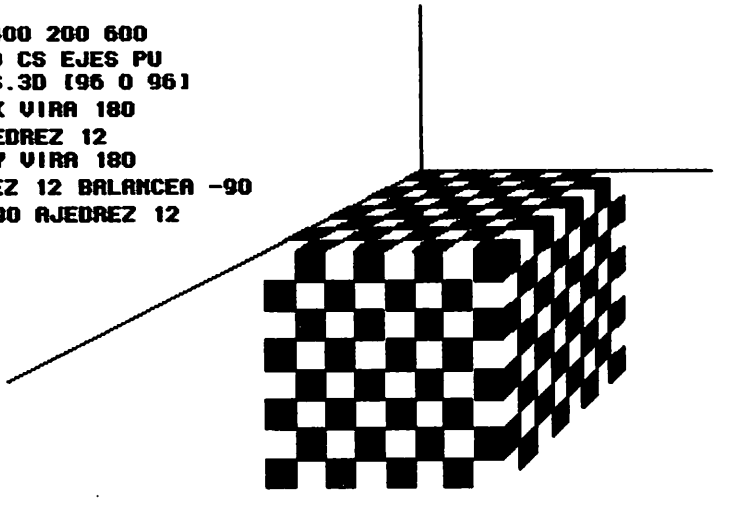


FIGURA 15

6. REFERENCIAS

(1) ASOCIACION NACIONAL ESPAÑOLA DE LOGO, 1985: *"Traducción normalizada de LOGO"*. Boletín LOGO y EDUCACION N.º 2.

(2) LUENGO GONZALEZ, RICARDO y OTROS, 1987: *"Implementación de un micromundo LOGO tridimensional a partir de una versión de LOGO en dos dimensiones"*. Campo Abierto n.º 4, ED. Univesitaria de Magisterio de Badajoz (Universidad de Extremadura.)

(3) PAPERT, SEYMOUR, 1981: *"Desafío a la mente"* ED Galápago. Buenos Aires (Argentina).

(4) REGGINI, HORACIO C., 1985: *"Ideas y formas"*. ED Galápago. Buenos Aires (Argentina).

(5) REGGINI, HORACIO C., 1982: *"Alas para la mente"* ED Galápago. Buenos Aires (Argentina).

(6) SOLOMON, CYNTHIA, 1987: *"Entornos de aprendizaje con ordenadores"* ED Paidós/M.E.C. Barcelona.