UNIVERSITAT POLITÈCNICA DE CATALUNYA

APORTACIÓN AL ESTUDIO DEL SOFTWARE NECESARIO PARA LA INFORMATIZACIÓN DE LOS MÉTODOS DE APRENDIZAJE DE LAS TÉCNICAS DE EXPRESIÓN GRÁFICA, Y SU POSTERIOR IMPLEMENTACIÓN

Autor: Miquel Castillo i Ballarda
Director: Jordi Mestres i Sardà

1988

DISEÑO ASISTIDO POR ORDENADOR (CAD).

Introducción.

Aunque no es el motivo básico de la tesis, no deja de ser interesante reseñar las caracteristicas básicas que poseen los paquetes comerciales de CAD, para asi poder ver su peculiar forma de acometer la representación bidimensional del espacio. La visualización de los objetos es una parte inherente del diseño técnico y el dibujo. Las técnicas de dibujo, las convenciones en el diseño y las disciplinas de ingeniería han quedado incorporadas como una forma universal de comunicación visual, de manera que se comunica a la vez el contenido y la ejecución de la idea.

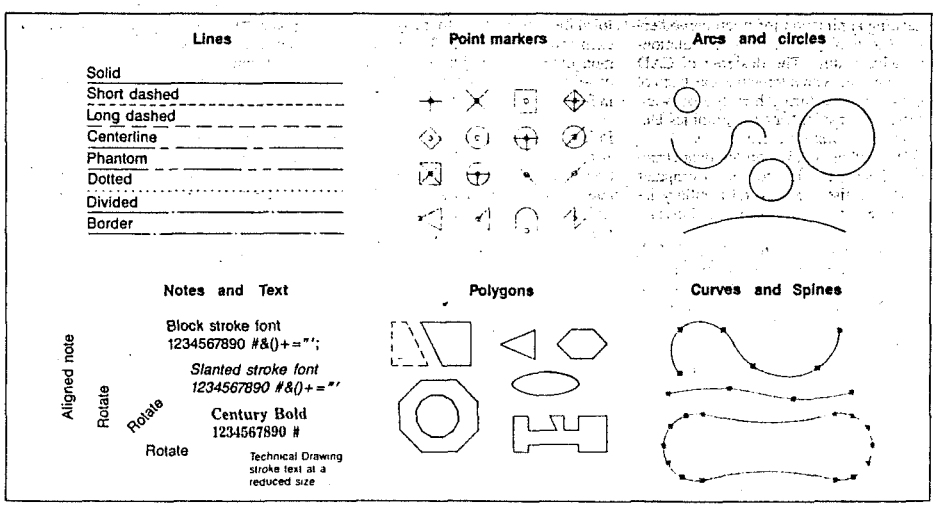
Los sistemas de diseño/dibujo ayudado por ordenador empiezan a ser cada vez más accesibles a los diseñadores, ingenieros, arquitectos, y dibujantes a medida que el coste del hardware disminuye, motivo por el cual no es inteligente quedar al margen de su conocimiento. El software de CAD es una de las aplicaciones más solicitadas de la Microinformática debido a su naturaleza de cálculo intensivo. El diseñador de programas de CAD tiene ante si un interesante espectro de opciones para estructurar, guardar y manipular, por software, los elementos gráficos que componen las imágenes de CAD. En esta introducción se pretende dar una idea de como se enfoca la estructuración de los datos en el CAD con microordenador.

Aproximación básica al CAD.

Hay dos formas de aproximación al diseño y funcionamiento de los sistemas de CAD: el diseño o modelado tridimensional y el dibujo de "esbozo".

La primera de las dos formas crea un modelo análogo del objeto deseado o sea procesa y almacena los objetos, como datos geomètricos, los atributos y sus relaciones. Esta aproximación puede ser usada tanto para visualizar un objeto del mundo-real, como para los tradicionales dibujos de producción. La segunda opción, la del esbozo, crea y manipula primitivas básicas gráficas. El esbozo y dibujo manual de lineas y arcos en papel es reemplazado por operaciones similares en una pantalla de rayos catódicos. Es en este punto donde existe una mayor diferencia entre los sistemas profesionales y los comerciales. Los licenciados en Informática, dirigen sus esfuerzos a la primera de los dos opciones comentadas anteriormente, mientras que los programas comerciales dedicados al gran público apoyan todo el peso de su trabajo en las primitivas gráficas, siendo de esta forma una ayuda a la croquización o esbozo; con la ventaja sobre la realización en papel de que cuando se da por bueno un croquis, èste ya queda en forma válida, pudiendo ser acotado, sacado por impresora o plotter, o enviado a una máquina que lo transforme en impulsos para ser conectado a una cadena de montaje con lo que realizariamos el ciclo CAD/CAM completo.

El dibujo ayudado por ordenador guarda la información gráfica como una colección de imágenes, articulos u objetos. El tèrmino primitivas gráficas es también usado comunmente para designar los elementos básicos que componen una imágen. Cada imágen corresponde a una estructura de primitivas dato que contiene la mayor parte de la información necesária para describir cada imágen individual. Las primitivas básicas más comunes, que se encuentran en cualquier CAD de 2D, son las de la figura.



(1)

Un programa de CAD reune en si mismo un complicado conjunto de dibujos, y la capacidad de operar con ellos, de forma que podemos discernir entre varios programas midiendo estas dos variables. Es por eso que el primer tema a comprobar es la lista de primitivas que se nos ofrecen, las más habituales son:

Puntos.

Lineas Rectas.

Arcos y Circulos.

Rectángulos.

Elipses.

Poligonos Regulares.

Lineas Paralelas.

Curvas definidas por puntos.

Posibilidad de alterar los gruesos de lineas y curvas.

Si queremos introducir las entidades en el ordenador con precisión, el programa debe darnos la posibilidad de entrar las coordenadas de los puntos ya sea por teclado o por digitalizador, asi como debe ser posible, también, el escoger las unidades y la escala a que se va a trabajar. La mayoria de los programas tienen dos opciones de trabajo llamadas GRID y SNAP (2), que nos ayudan a trabajar con la precisión que deseamos. GRID nos permite materializar en pantalla una malla, sólo cuadrada en el caso de AutoCAD y con valores cualesquiera en CADKEY y PC-Draft, a la que podemos referirnos como

la ayuda que nos puede significar el trabajar sobre un papel cuadriculado. Si además queremos referirnos únicamente a los puntos de intersección de la malla creada podemos usar SNAP, con lo que conseguiremos que el cursor se mueva exclusivamente a saltos discretos coincidentes con el GRID que previamente hemos definido. Como utilidad más sofisticada tenemos OBJECT SNAP, también conocida por su nombre abreviado OSNAP, que nos permite localizar puntos que se encuentren en posiciones determinadas por los elementos que en aquel momento se encuentran en pantalla; ejemplo: punto intersección de dos rectas, centro de una circunferencia, punto medio de un segmento, punto extremo de un segmento, etc.

Existen otras ayudas de dibujo como pueden ser: la materialización de unos ejes, en 2D, en pantalla -con la precisión y unidades que deseemos-, tener constantemente en la linea de diálogo del programa las coordenadas que nos dan la posición del cursor, exigir que el movimiento que realice éste sea exclusivamente horizontal o vertical, etc. Especial interés tienen dos de las más usuales utilidades de un paquete de CAD: ZOOM y PAN. ZOOM nos sirve para aumentar o disminuir las escalas de los objetos que se encuentran en pantalla, con lo que podremos ver y trabajar con más o menos detalle, según nuestras conveniencias; mientras que PAN nos permite cambiar de posición una porción del dibujo, sin variarle la escala.

Puede darse el caso de que hayamos dibujado un cuerpo a base de aristas vivas y nos convenga re-

dondearlas, para ello usaremos la utilidad FILLET, la cual, previa consulta de cual debe ser el radio del arco que una los tramos rectos, nos resolverá el problema. Si lo que queremos es hacer un chaflán, entendiendo por tal exactamente lo mismo que en el caso anterior pero sustituyendo el arco por una linea recta, podemos recurrir a la utilidad CHAMFER.

Mención aparte merece el uso de los LAYERS o niveles, que sustituyen a las hojas de acetato que podemos añadir a una hoja de papel de forma que podamos programar, por ejemplo, una presentación de unos dibujos por fases. Si se tratára de un dibujo de interès puramente geométrico, tendría cierta lógica, simplificando, colocar en un layer los datos, en otro las operaciones, y finalmente en un tercero los resultados. La consulta de cada layer por separado, o de varios superpuestos justifica su utilidad práctica. Respecto al uso de los layers, es interesante destacar que pueden llegar a complicar mucho el volcado de un dibujo (loading), pues, como es lógico, si damos la orden de cargarlo con todos los layers visibles, esta carga se realizará ordenadamente layer por layer lo que hace la operación lenta. Hasta aqui el comentario entra dentro de lo previsible, hemos consumido tiempo porque hemos pedido varios layers, ahora bien, ocurre que aunque cargemos un dibujo con varios layers desactivados el programa los lee igualmente invirtiendo el mismo tiempo que si realmente los volcara en pantalla. Consecuencia de lo anterior es, que aun que tengamos los layers en OFF es necesária la existencia de otra orden más res-

trictiva de forma que no se pierda tiempo leyendo algo que no va a ser representado en pantalla. AutoCAD lo resuelve con la orden FROZEN sobre los layers que previamente están en OFF; FROZEN "congela" este layer de forma que el programa lo ignora al cargar el dibujo en que se encuentra. Como es lógico, cuando queramos volver a ver este layer antes de pulsar ON, es necesario descongelarlo. Aunque en todo el comentario anterior he usado la expresión "volcado en pantalla", lo mismo se aplicaría en un paso a plotter de un dibujo, se pueden, pues, hacer visibles o invisibles los layers según nuestros intereses. Igual comentario se puede hacer con los textos que se añaden a los dibujos o las acotaciones, algunos paquetes los envian directamente a layers de texto, de forma que podamos hacerlos visibles o no mediante el mismo criterio comentado anteriormente.

Otra de las utilidades que, de forma más o menos compleja, o según otros, más o menos estructurada, nos dan prácticamente todos los paquetes de CAD, es la posibilidad de guardar un dibujo y despues volcarlo sobre otro en una posición determinada. Dependerá de la bondad del programa, la sencillez de esta manipulación y las operaciones que se puedan hacer con estos dibujos, usualmente llamados BLOQUES, tales como variación de escala, rotaciones, etc. Normalmente lo que se pide al definir el bloque es un punto de referencia que será el que lo identificará más adelante. Cuando lo queramos insertar en otro dibujo, se nos pedirá que escogamos sobre este último cuál será el punto del bloque a insertar.

Finalmente, queda el tema de la programación de macros. En la reseña de Glenn Hart en (2), se hacia el comentario de que varios paquetes de CAD daban la opción de ser linkados con lenguajes de alto nivel como Fortran, Pascal o C, mientras que sólo AutoCAD tenia un lenguaje interno, AutoLISP. En breve tiempo eso se ha convertido en usual y PC-Draft usa tambien un lenguaje interno similar a Pascal, sin serlo, y CADKEY el CADL parecido al BASIC. Los tres paquetes dan la opción de manipular la información mediante programas de tratamiento de datos como dBase II, dBase III o programas integrados como Symphony. También es de interés tener en cuenta el grado de dificultad que conlleva el aprender estos lenguajes adicionales; en el caso de AutoLISP, el manual remite inmediatamente a cualquier texto que trate el tema del LISP, con lo que da pie a las criticas de los comentaristas especializados, los cuales por un lado aplauden la idea de usar un lenguaje apoyado por una considerable bibliografia, mientras que no aprueban que haya sido justamente LISP el elegido, debido a su complejidad. Tanto CADKEY como PC-Draft optan por incluir en un pequeño apartado las nociones que creen necesárias para trabajar con sus macros. No obstante, es mucho más riguroso PC-Draft, pues le dedica un pequeño tomo especifico, mientras que CADKEY trata el tema bastante de pasada.

TRATAMIENTO DE DATOS.

En este apartado pretendo comentar, someramente, como se estructuran y manipulan las principales primitivas gráficas.

Puntos y señalizaciones.

Un punto es, conceptualmente, merecedor de una localización especifica por el valor de unas coordenadas. La especificación de un punto en solitário no es a menudo necesária en una imágen acabada, pero nos puede servir como apoyatura en un dibujo para juntar notas y referenciar localizaciones. Muchos sistemas permiten especificar códigos gráficos que hacen que un punto pueda ser dibujado con marcas o formas que pueden ser usados como dimensionadores (flechas, puntos, cajas, cortes), lineas de flujo, cruces y simbolos de ruptura, etc. La geometria de estos simbolos de referência aparece en pantalla, pero normalmente no puede ser creada manualmente por el usuario. Algo parecido a lo que se preconizaba en cuanto a signos convencionales en HYPATIA.

Lineas.

Las lineas son almacenadas como los valores de las dos coordenadas de los puntos inicial y final de cada una de ellas. El color, el estilo de linea, anchura y atributos de nivel son usualmente asignables. Otra forma de almacenar una linea es como dos vectores que señalan la situación de las coordenadas de principio y final. Dado que la ma-

yoria de los dibujos consisten en conectar segmentos del mismo estilo de tipo de linea, los atributos que las diferencian, a las lineas, son la situación de sus puntos original y final, el uso de los vectores parece la forma mas eficiente de determinarlas. De esta forma, los valores de las coordenadas sólo es necesário almacenarlos una sola vez, refiriéndonos siempre unicamente a los vectores. Si usamos como vectores números enteros (long-integers) usados en coordenadas en doble precisión y todas las lineas están conectadas por lo menos en un sitio, el uso de vectores requiere el 25% menos de espacio en dos y tres coordenadas.

Las coordenadas, que son palabras de 4 bytes con coma flotante, consumen la misma cantidad de espacio que las de long-integer de 4 bytes, mientras que las coordenadas de enteros de 2 bytes requieren el 33% más de espacio en el caso de 2D y 16% más en las 3D. Sin embargo, la aproximación por vector necesita un contador de referència almacenado con cada coordenada para reconocer cuando una primitiva no gráfica usa esa coordenada.

Arcos y circulos.

Existen varios mètodos de guardar los datos de circulos y arcos. El más común es almacenar los puntos centro, los radios, y los ángulos de inicio y final, o el ángulo de inicio y el ángulo de incremento con una dirección positiva o negativa. Este método del ángulo del incremento hace muy sencilla la identificación de la dirección de un arco. Lógicamente, también tres puntos pueden definir un

arco. Las coordenadas de los puntos inicial y final pueden ser almacenadas junto con un tercer punto del mismo arco. El tercer punto se usa para obtener, calculándolos analiticamente, el centro del arco, radio, y los ángulos inicial y final que necesitamos.

Polígonos.

Tienen los mismos atributos los segmentos de líneas conectados, los polígonos (considerados segmentos de líneas conectados y cerrados), o las series conectadas de puntos o líneas. La diferencia entre determinar vectorialmente o por coordenadas un polígono se encuentra en el número de bits que se usan en cada caso. Por ejemplo, en 2D muchas líneas están simplemente conectadas no multiplemente conectadas con otras líneas. Esta conectividad es una parte inherente de la estructura de los polígonos; asi, el uso de los vectores, en este caso aumenta los requerimientos de almacenamiento.

En los sólidos de 3D, es preferible el sistema de las coordenadas por vectores, dado que tres o más polígonos casi siempre comparten el mismo punto, reduciendo los requerimientos de almacenamiento.

Curvas y Splines.

Entendiendo por curvas las líneas suaves que pasan entre una serie de puntos (tradicionalmente llamados nudos) o cerca de un conjunto de puntos llamados puntos de control. Varias funciones

matemáticas controlan el comportamiento de la curva, siendo los más comunes los B-Splines o las curvas de Bezier.

El control de los nodos es almacenado de una forma similar al esquema usado para almacenar segmentos conectados o poligonos. La función determinada es almacenada como un código que unicamente define el método usado para devolver la curva por la pantalla o por el aparato que sea, impresora, plotter, etc, mientras los coeficientes exactos de la representación funcional del camino de la curva pueden ser almacenados para cada tramo de la curva. El tramo está definido entre dos puntos sucesivos de control o dos nudos.

Anotaciones textuales.

Los caracteres textuales en la mayoria de los sistemas de CAD están normalmente representados como series de pinceladas (strokes) o segmentos lineales. Este método permite al sistema escalar, rotar y mover textos con precisión, mediante procedimientos similares a los usados para manipular cualquiera de las otras primitivas gráficas que hemos visto. La mayoria de los sistemas de plotter incluyen un creador de formas en ROM, y los programas de CAD pueden escoger y direccionar las instrucciones para dibujar el texto de los distintos modelos de plotter. Sin embargo, se prefiere, a menudo, el poder generar texto desde el mismo software del CAD. Pese a ser esta opción más lenta, tiene la ventaja de que se puede generar una más

amplia variedad de textos, en distintos tamaños, formas y estilos.

Simbolos.

Los simbolos son una colecciòn de dibujos, que a menudo incluyen otros simbolos, que se comportan como unidades singulares. Pueden ser guardados, insertados, y transformados ya sea como unidades singulares o como piezas individuales. Los simbolos son generalmente muy usados en todos los sistemas de CAD para representar elementos standard en los dibujos, algo parecido, en su uso, a los glosarios que se insertan cuando y donde nos conviene en los tratamientos de texto. Los simbolos reciben distintos nombres como: bloques, objetos, componentes o partes.

Cada vez que se usa el mismo simbolo en un dibujo, se almacena un referencia de toda la informaciòn del simbolo original, sus componentes y caracteristicas. Cuando los simbolos contienen listas de otros simbolos, se dice que estàn anidados. No es posible que un simbolo se referencie a si mismo. Las partes de un simbolo están generalmente referidas a un sistema local de coordenadas del simbolo, establecida cuando se definió éste. Los simbolos se almacenan como una estrutura de datos de un objeto que lleva consigo la informaciòn de la posiciòn, el escalado, la orientaciòn y toda la informaciòn geomètrica para posicionarlo correctamente y transformar sus coordenadas locales a las coordenadas globales del dibujo dentro del cual es insertado.

Atributos de las primitivas gràficas.

Cada una de las primitivas gràficas es definida con informaciòn geomètrica. Ademàs, cada primitiva tiene unos atributos para describir como han de ser dibujadas o visualizadas. Las lineas y las curvas pueden tener un parámetro para determinar su color, estilo de linea, y anchura de forma; que son usados para su representaciòn por pantalla o su volcado por plotter o impresora.

Las plumillas de dibujo màs usuales trabajan en un solo color, las diferencias entre unas y otras vienen determinadas por los gruesos y tipos de linea. Las màs comunmente usadas en el dibujo mecànico son: lineas de trazos, de puntos, de linea y punto, lineas fantasma, etc (3). Las varias disciplinas de los usuarios de CAD tienen, lògicamente, unas convenciones de forma que una determinada linea de puntos añade informaciòn a un observador experto en el tema -la màs usual puede ser la linea de trazos que simboliza las aristas ocultas en la representaciòn bidimensional de un objeto-.

Estos estilos de linea y propiedades estàn relacionadas con los tamaños finales de las hojas y las convenciones tècnicas, y no siempre casan bien las medidas con las coordenadas, escalas y tamaños. En la mayoria de los casos, las anotaciones simbòlicas no se refieren directamente a las dimensiones del objeto. El tamaño viene determinado por el tamaño del dibujo final, y por el hecho de que debe ser legible (4). Estas incorrecciones son

aceptadas por las normas ISO y ANSI, que especifican muchos aspectos de estilo y tamaño. Es poco probable que un usuario tenga que generarse simbolos especiales, dada la cantidad de ellos que existen en las librerias creadas a tal efecto.

Una atributo adicional o indice que aparece los dibujos en 2D es el código de layer o nivel. Los simbolos son usados para relacionar y organizar la información entre regiones conectadas lógicamente en un simple dibujo plano; los layers son usados para registrar y relacionar información que está fisicamente almacenada en un dibujo de dos dimensiones.

Transformaciones de visualización.

Las transformaciones de visualización son usadas para volcar el modelo fisico de la base de datos en la vista de un dibujo en 2D sobre un tubo de rayos catódicos o un plotter. Existen muchos sistemas que almacenan porciones de dibujo, en ocasiones llamadas vistas. Cada vista tiene asociada una matriz de transformación que nos pasa de las coordenadas globales a las coordenadas de pantalla.

Transformaciones de diseño y modelado.

Las transformaciones de modelado están asociadas con el hecho de referenciar, posicionar, hacer espejo, escalar, rotar, ensanchar, mover y copiar dibujos. Como hemos comentado anteriormente, los objetos organizados y almacenados bajo una estructura de datos de objeto (símbolos) están

definidos en su propio sistema de referencia y después transformados o volcados (mapped) en las coordenadas globales del sistema de dibujo dentro del cual están colocados. Los objetos también pueden ser transformados entre diferentes coordenadas locales de forma que se puede pasar informacion entre varios dibujos hechos por el usuario.

Las transformaciones de modelado son similares a las transformaciones de ventana en las 2D. Si las transformaciones de ventana tenian lugar en dos pasos, transformación de coordenadas locales a coordenadas globales, para despues usar las interfaces del aparato dibujador para transformar las coordenadas globales en sus propias coordenadas de dibujo, el primer paso será idéntico en el caso de la transformación de modelado. Todas las imágenes mencionadas anteriormente pueden ser transformadas correctamente excepto los arcos y las circunferéncias. Cuando ensanchamos, escalamos de forma no isotrópica, y movemos arcos y circulos, éstos pasan a formas elipticas. Algunos sistemas añaden una primitiva elipse para trabajar con ella; otros tratan la transformación de arcos y circulos como un objeto especial y usan la transformación de objetos para representar las elipses o arcos de elipse.

Estructuración de los datos de las primitivas gráficas.

La información de los datos de las primitivas gráficas es almacenada de una forma especializada, los sistemas de CAD han empezado a usar muchas téc-

nicas de los sistemas standard de tratamiento de datos, para acceder, actualizar, modificar, y cambiar dichos datos. No obstante, en muchos casos la estructura de datos de CAD debe satisfacer los requerimientos de un interface de usuario altamente interactivo. Las principales diferencias son dictadas por la demanda del hardware por la interactividad de la pantalla o del plotter, y las complejas relaciones geometrico-matemáticas de las mismas primitivas.

Las primitivas gráficas, por definición, contienen información de las coordenadas posicionales. Las coordenadas de situación de un objeto precisan números reales, 4 bytes para los puntos de coma flotante y 8 bytes para los de doble precisión. Las coordenadas de 4 bytes con coma flotante cargan entre 6 y 7 digitos de precisión, y los de doble entre 15 y 16. Los cálculos son extensos y necesitan ser hechos atentamente para evitar errores acumulativos o de computación, especialmente en Topografía, Control Numérico y otras operaciones que llevan asociada la precisión.

Trabajar con enteros geométricamente no es ni correcto ni lógico para la mayor parte de los ingenieros y diseñadores técnicos. Excepto en algunos pocos casos, los sistemas de CAD basados en coordenadas enteras, son considerados cuestionables cuando no sencillamente inaceptables. En general, la precisión de los números reales es la deseada en todas las aplicaciones, incluso cuando no parece ser necesária, como por ejemplo: esquemas, dia-

gramas de flujo, charting (5), y muchos de los dibujos de arquitectura.

Actualmente la mayoria de los sistemas de pantalla y plotters aun necesitan coordenadas enteras. En particular los sistemas de bit-map (punto a punto). Dentro de la gama de las pantallas usadas corrientemente y los plotters grandes, el uso de las coordenadas de 2 bytes (desde -32768 hasta 32767) es suficiente para la resolución de estas máquinas. De hecho, es discutible el añadir más resolución excepto en el caso de máquinas especializadas.

Las coordenadas de imágen CAD pueden ser convertidas a una máquina de coordenadas enteras antes de ser pasadas por pantalla o dibujadas por plotter. Para el programador esto implica un continuo acto de malabarismo quitando velocidad para obtener precisión y escogiendo entre dejar que la aplicación trabaje o acomodarla a las rutinas de las máquinas de salida. La transformación de coordenadas es compleja desde un punto de vista computacional y usualmente se realiza mejor con números reales, como es el caso del clipping -acción de representar sólo una porción del dibujo sin perder la información de la entidad entera-. Convertir las coordenadas de la imágen a enteros simplemente, y pasarlas a la rutina de la máquina de salida es, en general, inadecuado.

Estructuración de los datos de dibujo.

En el listado que se copia a continuación, podemos ver una estructura tipica, en lenguaje C, de algunas primitivas tales como lineas o arcos (6).

```
typedef struct                    /* world  coordinate
point*/
{
        float  x,y;
}wcoord;

typedef struct                      /* origin, scale,
angleref wcoord system */
{
             wcoord origin;        /* x and y origin
coordinates */
          float  xscale,         /* x scaling factor
*/
                 yscale,         /* y scaling factor
*/
                  angle,          /* positive x-axis
rotation (radians) */
                 skew;           /* skew factor for
shear in x direction */
} osa;
typedef struct                      /* rectangular box
coordinates */
{

        float  xmin, ymin, xmax, ymax;
} rbox;

typedef struct                   /* entity attribute
data structure */
             char  layer;           /* entity layer
index */
          char  pen;             /* entity pen type
code */
         union
         {
                 char  mark_type; /* entity type
code */
                 char  line_type; /* entity type
code */
```

```
                char  font_type; /* entity type
code */
                char  poly_type; /* entity type
code */
                char  obj_type;  /* entity type
code */
        }type;
"entity_ads;

                                 /* line entity
typedef struct
data */
"
          entity_ads  attrib;
              wcoord      start;  /* startpoint
coordinate*/
              wcoord        end;    /* endpoint
coordinate */
} line_entity;

                                 /* ptmark entity
typedef   struct
data */
"
          entity_ads   attrib;
              osa                rcs;    /* marker
positioning */
} ptmark_entity;

                                 /* arc and circle
typedef   struct
entity data */
"
          entity_ads   attrib;
              wcoord          center;  /* center
coordinate */
              float        radius;  /* radius of arc
or circle */
              float        alpha;   /* start angle of
arc */
              float          delta;   /* incremental
angle to end */
} arc_entity;

                                 /* note entity
typedef   struct
data */
"
          entity_ads   attrib;
          osa          rcs;     /* note placement
data */
               int           length;  /* note string
length */
```

```
            char          *text;    /* address of
text string */
} note_entity;

typedef   struct                    /* poly entity
data */
{
         entity_ads    attrib;
         ptmark_entity *marker; /* Vertex marker
type */
          unsigned      fill:1;  /* Is it filled
in? */
          unsigned      closed:1;/* Is poly open
or closed? */
            unsigned        spline:1;/* Is poly
spline smoothed? */
          int           n;        /* Vertex count
*/
            wcoord         *pts;     /* array of
vertex coords */
} poly_entity;

typedef   struct                    /* object data
*/
{


         entity_ads    attrib;
            osa               ref;      /* object
placement */
            struct  _entity_node  *entities; /*
pointer to symb header */
             char           *name;    /* address of
label string */
} obj_entity;

typedef   struct  _symb_entity    /*symbol entity
data */
{


          int           id;       /* symbol index
*/
            int               ref_cnt; /* symbol
reference count */
         rbox          extent;   /* symbol extent
(local) */
            struct   _entity_node  *entities /*
symbol entity list */
             char           *name;    /* address of
symbol label */
} symb_entity;
```

```
typedef   struct   _entity_node   /* entity node
data struct */
{

             struct   _entity_node  *forward; /*
next entity node pointer */
             struct    _entity_node *backward;/*
previous entity node pointer */
          int                      entity_type; /*
entity type and vis flags */
        union
{

        ptmark_dat   *point_ptr;  /* pointer to
point data */
        line_dat     *line_ptr;   /* pointer to
line data */
        arc_dat      *arc-prt;    /* pointer to
arc data */
        note_dat     *note_prt;   /* pointer to
note data */
        poly_dat     *poly_prt;   /* pointer to
poly data */
        obj_dat      *obj_prt;    /* pointer to
object data */
        symd_dat     *symb_prt;   /* pointer to
symb data */
        } ptype;
} entity_node;

typedef   struct_dbh            /*  Main  drawing
database header */
{
        struct_dbh   *next /* pointer to   next
database */
             entity_node   *vis;      /* list of
visible drawing entities */
             entity_node   *masked;  /* list  of
blanked drawing entities */
             entity_node    *group;   /* list of
workset drawing entities */
             entity_node    *symbols; /* list of
symbol header entities */
           char            *name;     /* drawing
database name */
             char          status;    /* active-
inactive status flag */
} drawing_dbh
```

La presentación por pantalla, la selección de imágenes (llamada picking), la edición, "plotteado", y la comprobación de las rutinas, opera a través de esta estructura. Las imágenes pueden ser añadidas, borradas, modificadas, copiadas, y transformadas usando unas rutinas especiales de tratamiento de datos. Esta estructura de los datos es básicamente el corazón del sistema de dibujo. Los otros algoritmos y las estructuras de datos que conectan al usuario con la interface, el archivo de Entrada/Salida, la calculadora, las utilidades para editar texto, los transformadores de unidades y formatos de pantalla, el cursor de pantalla y el menu, los puntos de entrada de datos, y la mayoria de la creación geométrica, cálculo y algoritmos de transformación que forman un sistema de CAD interactivo.

Resumiendo, se ha pretendido dar una visión general de como es el trabajo mediante el uso de las primitivas, especialmente, hay que reconocerlo, en el área de las 2D. El CAD es capaz de reemplazar muchos de los tradicionales trabajos y dibujos preliminares de los diseñadores y los ingenieros. La actual tecnologia asociada a los ordenadores personales es adecuada para muchos de los trabajos que requerian especializadas workstations -estaciones de trabajo para varios usuarios a la vez-, hace no muchos años. La cada vez mas alta potencia del hardware y la disminución de los precios permite avanzar en la creación de cada vez más sofisticadas herramientas de CAD capaces de ayudar en los más complejos diseños de ingenieria, análisis, y técni-

cas de modelado en tres dimensiones hasta ahora sólo asequibles en sistemas mucho más grandes, como podían ser CATIA, CADAM, etc.

Los paquetes comerciales de CAD. Su relación con el tema de las tres dimensiones.

Como se ha destacado en el párrafo anterior, todo lo dicho tiene una clara aplicación al CAD en 2D, pero, ¿qué ocurre cuando pretendemos resolver también problemas en 3D?. Es interesante comentar al respecto, que el Dr. Pere Brunet en (7) matiza muy claramente que no es lo mismo el dibujo ayudado por ordenador que el verdadero diseño ayudado por ordenador; los programas de CAD, al menos los que ha podido manipular el que suscribe, se adscriben sin grandes distorsiones al primer tipo, digan lo que digan sus creadores. Además no hemos de olvidar que se trata de microordenadores, no de otros segmentos de hardware que pueden permitirse obviar algunas cuestiones.

Antes de continuar, creo necesario especificar que los programas comerciales de CAD a los que he tenido acceso para la realización de esta parte de la tesis son: AutoCAD 2.15, PC-Draft 4.3B y CAD-KEY 2.02. De los tres, el que, de entrada, renuncia al tratamiento de las 3D es PC-Draft, aunque incluye una utilidad de deformación, que nos permite obtener una vista isométrica de una figura a partir de las vistas diédricas de ésta, convenientemente "deformadas" al efecto. AutoCAD tiene un complejo y poco operativo sistema para determinar cuerpos en 3D, a los que posteriormente elimina sus líneas

ocultas. En cambio, CADKEY es, a mi modo de ver, el que resuelve más claramente el problema de las 3D, aunque existen algunas discusiones sobre si la generación de esas 3D no es en realidad un sistema de 2 1/2 D. Volveremos sobre el tema más adelante. No obstante, no se puede obviar el auténtico aluvión publicitario que nos invade prometiendo auténticas 3D últimamente, primera mitad de 1987; a juzgar por dicha información, publicitária repito, AutoCAD ha diseñado auténticas 3D, CADKEY ha entrado con éxito en el campo del modelado, a partir de una figura de alambres, y RoboCad, que era un sistema de poca categoria que trabajaba originariamente sobre Apple II con la interface de iconos de Apple Macintosh, anuncia RoboSolid en donde ofrece incluso operaciones booleanas. Refiriéndonos nuevamente a AutoCAD, su versión 2.6, aun no comercializada, implementa una utilidad que permite que una vez que tenemos un cuerpo acotado, cuando realizamos alguna variación geométrica interactiva sobre él, las cotas sufren las variaciones correspondientes. Lógicamente, estos son temas a seguir.

Retomando el hilo de la pregunta sobre como podemos trabajar en 3D con un programa de CAD, independientemente de que después se comenten algunas particularidades, podemos afirmar que la principal dificultad radica en la creación de esta figura tridimensional; las manipulaciones a que se someta dicha figura tienen una importancia relativamente menor, sin querer minimizar con este comentario la cantidad de estudios que se están realizando al respecto. Lo que más habitualmente encontramos en los paquetes de CAD es el uso de las primitivas

deformadas por el proceso de barrido o sweep, con las dificultades que se han indicado cuando dichas primitivas incluyen arcos y circunferèncias, ya que èstas se transforman en primitivas elipticas.

Abandonando el campo de las primitivas y jugando el papel del Usuario-Puente, comentado a menudo en este estudio, existe el camino de aplicar la Geometria Analitica, Proyectiva y Descriptiva, y algùn lenguaje suficientemente estructurado, para resolver estos problemas, que es lo que se plantea el programa HYPATIA. Si optamos por un camino màs sencillo y ortodoxo, podriamos apoyarnos en algùn texto de prestigio en el tema, como puede ser el de Ammeraal (8), en donde se expone de una forma clara este trabajo, tanto bi como tridimensional, aunque decepcionantemente, y coincidiendo con lo que expresa Brunet en (7), la opciòn que escoge es la de crear el cuerpo de forma que el propio programa diriga al usuario de manera que sòlo se pueda crear un cuerpo correcto. Dicho de otra forma, Ammeraal define los cuerpos a partir de sus vèrtices numerados, exigiendo al usuario un determinado orden de introducciòn de los datos para asi asegurarnos de que las caras, planos al fin, estèn definidas de forma que el vector normal a ellas tenga el sentido conveniente (reconozcamos que con el programa HYPATIA llegamos a parecidas conclusiones cuando nos planteamos cortar un cuerpo por un plano). El modelo que adopta es el de alambres, aunque luego paradòjicamente, como el mismo autor reconoce, propone un nuevo programa para la eliminaciòn de lineas ocultas, de interès puramente pedagògico. El autor justifica el programa escogido debido a que

elige la inteligibilidad frente a la rapidez, no hemos de olvidar que nos encontramos ante un libro básico. El texto propone la utilización de las subrutinas que van apareciendo a lo largo de su desarrollo, para lo que promete su edición en forma de diskettes en breve, con lo que además de su adquisición hará falta el concurso de un compilador de C (9).

Ciñéndonos al problema de las 3D en los paquetes que estudiamos, sólo merece atención el desarrollado por CADKEY. La solución adoptada es la de trabajar por planos, de forma que nos permite dibujar cualquier entidad gráfica en cada uno de los infinitos planos en que se puede dividir el espacio. Este sistema de trabajo se materializa mediante el cursor, al que vemos siempre en pantalla moviéndose bidimensionalmente, pero al que podemos asignar la tercera dimensión de "profundidad" a voluntad. La impresión del valor de esta tercera dimensión se encuentra siempre en pantalla, por lo que el usuario tiene la máxima información posible, por un lado las coordenadas bidimensionales "móviles" en todo instante sobre la pantalla, y, convenientemente destacada, la coordenada fija de la profundidad. Originariamente, CADKEY provee 8 vistas numeradas de cualquier dibujo que hagamos, dando la posibilidad de crear nuevas vistas seleccionando de una forma determinada tres puntos sobre la figura dada, o mediante una serie de giros no muy afortunadamente documentados en el manual. Sólo las vistas 7 y 8 tienen características "axonométricas", mientras que las otras 6 son las tres vistas diédricas típicas en doble proyección

por delante y por detràs. En cada una de las seis vistas bidimensionales podemos alterar el valor de la "profundidad" que en unos casos serà el perteneciente a la coordenada x, en otra a la y o la z. La coordenada que variamos es siempre la que no se encuentra en el plano de la pantalla sino perpendicularmente a ella. Es obvio que el trabajo con este tipo de coordenadas debe realizarse lo màs ordenamente posible, rehuyendo su manipulaciòn en las vistas tridimensionales 7 y 8.

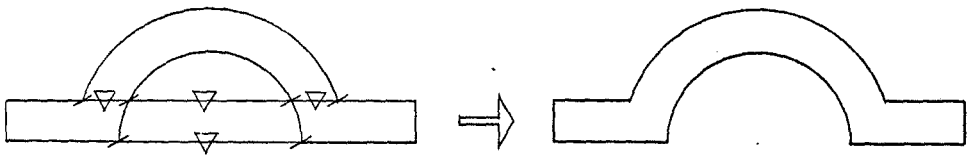
Mediante este juego de la tercera coordenada, y con el concurso de la funciòn TRANSFORMACION nos serà posible duplicar total o parcialmente un dibujo creado en 2D. Podemos de esta forma obtener una copia manteniendo asimismo el original, o una copia desapareciendo èste. En el ùltimo caso si damos como valor del desplazamiento magnitudes medidas en la direcciòn perpendicular al plano de nuestro dibujo, obtendremos traslaciones "espaciales" ya que lo que haremos es modificar el plano en que se encuentra la nueva figura. Finalmente, existe la opciòn UNIR que nos permite que las dos posiciones de la transformaciòn queden enlazadas por segmentos que unan vèrtices homòlogos, tenemos de esta forma creadas las aristas de los cuerpos volumètricos que deseàbamos. El mètodo, de hecho, guarda mucha similitud con el procedimiento de sweep asociado normalmente a las 2 1/2 D, o estructura de alambres. Cuando se trata de obtener, mediante este sistema, cilindros la soluciòn que obtenemos debe apoyarse en ciertos convencionalismos. Como ya hemos comentado en otra parte, las circunferèncias son representadas como poligonos de

gran número de lados, de forma que ópticamente sean aceptables; si aplicamos el proceso de generación de cilindros automáticamente igual que hemos comentado anteriormente, nos debería aparecer un prisma con gran número de generatrices-aristas. No es así. El resultado óptimo sería aquel que nos diera en cada vista el contorno aparente de cada cilindro, CADKEY no lo da, sino que la respuesta gráfica que obtendremos es la de dos círculos (que en las vistas "espaciales" veremos como elipses, lo que es un gran logro) y una generatriz arbitrária que une dos puntos de esos círculos. Si queremos dibujar "a mano alzada" -considerando la proyección bidimensional del espacio justamente esto, una figura plana-, posteriormente el contorno aparente, nos ocurre algo parecido a lo que comentaremos más adelante en la eliminación de líneas ocultas: obtendremos dos líneas que serán contorno aparente en una vista, mientras que serán líneas cualesquiera en otra vista; sólo si nos interesa presentar con corrección una vista vale la pena el trabajo a realizar.

Existen, como es lógico, algunos trucos que permiten resolver las dificultades comentadas de una forma que puede parecer razonable sino no somos excesivamente rigurosos al juzgarlos. Mediante la opción PUNTO, se crean unos puntos distribuidos simétricamente sobre la circunferencia que al desplazarse nos dará el cilindro. Dichos puntos, una vez ha tenido lugar la transformación, darán lugar a generatrices del cilindro. Por la forma en que han sido generadas dichas generatrices, lo seguirán siendo sea cual sea la vista que escogamos. Si cae-

mos en la cuenta que, de hecho, sòlo nos interesarán el alzado y el perfil del cilindro, escogiendo cuatro puntos tenemos resuelto, aceptablemente, nuestro problema del contorno aparente. Posteriormente podemos eliminar los puntos auxiliares con una sencilla orden.

En honor a la verdad, y pese a lo dicho hasta aquí, hay que decir, no obstante, que una vez creado el cuerpo tridimensional, y mediante la manipulación del valor de la tercera coordenada, trabajamos realmente en tres coordenadas, que es el especial mèrito que tiene CADKEY. Trabajar en 3D quiere decir poder ejecutar, por ejemplo, la utilidad FILLET -redondeo de aristas-, señalando lados contiguos de una cara de un "sòlido". La eliminación de lineas ocultas por la misma naturaleza del programa como manipulador de estructuras de alambre, carece de sentido. No obstante, es posible modificar las lineas que forman un cuerpo de la misma forma que lo haríamos con làpiz, papel y goma; en el procedimiento tradicional procederíamos a borrar parcialmente las lineas para que nos diera la sensación de que una arista queda oculta por una cara. En CADKEY, y también en PC-Draft, existe la posibilidad de ROMPER las lineas, realizando previamente la operación de subdividirlas en varias partes, de forma que su imagen siga siendo continua. Al realizar la acción de "romper" una arista, ya se nos pedirá con respecto a que entidad la rompemos, lo que nos servirá posteriormente para delimitar las partes que deberán ser borradas.



Como es lògico, las lineas borradas en una vista quedan de esta misma forma en cada una de las otras vistas, no dando en ninguna de estas el efecto deseado en la que se ha realizado el rompimiento, como ocurria en el caso del contorno aparente de los cilindros.

Para dar la opciòn de poder presentar varias vistas con lineas ocultas, CADKEY ha previsto un trabajoso sistema de almacenamiento de la informaciòn, veàmoslo: cuando guardamos un dibujo de forma normal, de èste quedan almacenadas, como minimo, las 8 vistas de que hemos hablado anteriormente, existe, no obstante, la posibilidad de guardar la informaciòn de una sola vista. Si estructuramos nuestro trabajo de forma que eliminamos todas las lineas ocultas total o parcialmente, y grabamos la vista obtenida, para, posteriormente, eliminar todas las manipulaciones realizadas (todas las lineas borradas son razonablemente recuperables), con una orden prevista a tal efecto; tendremos nuevamente todas las vistas del objeto con todas sus lineas y

además una vista, ya almacenada, con las lineas ocultadas convenientemente. Cambiando de vista y reanudando las operaciones, podemos llegar a tener una serie de vistas correctas desde el punto de vista de las lineas ocultas, que posteriormente pueden ser invocadas, como bloques, y colocadas en un dibujo en la posición que nos convenga. Repito que el proceso es prolijo, pero no se puede negar que es una solución a tener en cuenta.

Es conveniente destacar algunas caracteristicas notables que diferencian CADKEY de otros sistemas de CAD habituales. Estudios realizados al efecto, les demostraron a sus creadores que no era lógico que los menús de pantalla estuvieran a la derecha, dado que leemos de izquierda a derecha, parece más habitual que dichos menús, en forma de columna, se encuentren a la izquierda. De hecho el comentario anterior nos lleva a otra de las caracteristicas diferenciadoras de este paquete: mientras que Auto-CAD hace gala de que sea la que sea la fase del trabajo en que nos encontremos, siempre es posible llamar cualquier orden entre las ofertadas, y es cierto, en CADKEY no es posible, veamos por que. El diálogo usuario-ordenador se realiza mediante frases, de las que se van guardando las palabras que las componen en la parte superior de la pantalla, que deben componerse con sucesivas llamadas a la columna-menú, que va variando su contenido, a medida que vamos seleccionando opciones. Sólo cuando la frase formada tiene coherencia "sintáctica" para el software, el diálogo cambia de forma, desplazándose a la parte inferior de la pantalla, permitiéndosenos dar los valores puntuales que nos

interesan. Un ejemplo aclaratorio de esta forma de trabajar, podria ser el camino que hemos de seguir para dibujar un arco por tres puntos:

1.- DIBUJAR. Seleccionado sobre la primera columna, en donde se encuentran todas las ordenes generales, como GUARDAR, SALIR, FIJAR COORDENADAS, etc.

2.- ARCO. Seleccionado sobre una columna distinta de la anterior, que ha aparecido al ser invocado DIBUJAR. Otras posibilidades de dibujos son: LINEAS, RECTANGULOS, CIRCULOS, etc.

3.- TRES PUNTOS. Seleccionado sobre un nuevo menú, por un procedimiento similar al seguido en el paso 1-2. Existen otras opciones como: DOS PUNTOS Y UNA

DIRECCION, CENTRO RADIO Y ARCO, etc.

4.- COORDENADAS PRIMER PUNTO. Esta orden ya aparece en la linea horizontal inferior, pues la instrucción creada anteriormente ya tiene coherencia sintáctica; en la parte superior permanece impreso hasta que hayamos dado todos los puntos y cambiemos de orden, la frase: DIBUJAR ARCO TRES PUNTOS. A la vez que aparece la posibilidad de entrar las coordenadas, en la columna-menú, según vemos en 5.

5.- POR CURSOR. Existen distintas formas de entrar un punto, concretamente nueve, por lo que es necesario escoger una. Por cursor es la más interactiva,

la he usado como ejemplo. Coordinada con una malla correcta puede tener suficiente precisiòn.

6.- COORDENADA SEGUNDO PUNTO. Una vez almacenadas las coordenadas del primer punto, se nos piden las del segundo en la misma linea horizontal inferior.

7.- POR TECLADO. He escogido sobre la columna-menú la opciòn de entrar las coordenadas (en 2D) por teclado. Nos serán pedidas en la linea horizontal inferior con la notaciòn: X= e Y= o Xv= e Yv=, segùn las coordenadas sean GLOBALES o LOCALES, màs adelante se comentarà la diferencia entre ambas.

8.- COORDENADA TERCER PUNTO. Información del último punto.

9.- COMO EXTREMO DE ... Dentro de la opción POR CURSOR, también existe la posibilidad de usar OSNAP, como PUNTO MEDIO DE, INTERSECCION DE, etc.

El seleccionar una opción sobre la columna del menú se puede hacer con el ratón que soporta CADKEY o, teniendo en cuenta que el programa ha sido diseñado para trabajar con ordenadores compatibles, pulsando alguna de las diez teclas de función. Por ello nunca hay más de 10 opciones en el menú. Si nos acostumbramos a seleccionar sobre las teclas de función con la mano izquierda, vemos que el programa resulta ergonómico ya que aumentamos su velocidad y la soltura de trabajo, usando la mano derecha exclusivamente para la manipulación del ratón.

Merece un breve comentário la posibilidad de trabajar con coordenadas Globales o Locales. Debido a la continua posibilidad de movernos por las ocho vistas usuales, es posible que cuando queramos introducir coordenadas por teclado no recordemos cuál es el eje X o el Z en dicha vista, dado que nada

excepto el manual nos indica los ejes que se consideran en la vista de la que sabemos por pantalla su número. Las coordenadas Locales nos aseguran que si introducimos, por ejemplo: Xv=5, Yv=8, Zv=0, el valor 5 se adjudicará al eje que se encuentra en la posición horizontal en esa vista, sea cual sea, el valor 8 será el que tomará sobre el eje vertical correspondiente, y no se alterará el valor sobre el eje perpendicular al plano de la pantalla. Las coordenadas Globales implican el conocer correctamente, y en cada momento, que ejes están en juego. Para ello he construido, de forma sencilla, unos ejes "numerados" con las letras de los ejes correspondientes, de forma que en cualquier vista sabemos a que atenernos. Los ejes antes comentados se guardan como dibujo y pueden ser invocados al empezar cualquier sesión de trabajo. En esta opción X=5, Y=8, Z=0, no tienen más significado que el obvio.

Como conclusión, y aceptando que la creación de las 3D se realice mediante la técnica de swepping, CADKEY 2.02 es la respuesta más correcta con la que he podido trabajar dedicándole el tiempo de aprendizaje que he considerado necesário. No obstante, en un mercado tan cambiante como el que estamos comentando, he tenido ocasión de cambiar impresiones con usuarios de otros sistemas de CAD. El mismo CADKEY en su versión 2.11, trabaja con Splines y tiene implementado un sistema de definir helices muy competente. Es de destacar la resolución satisfactória de un viejo problema de los paquetes de CAD, como es el trazar la perpendicular a una recta por un punto de ésta. Quizás por desidia de los

programadores o por un afán de simplificación, el caso de la perpendicular de un punto a una recta, quedaba englobado de forma tácita dentro de la problemática distancia punto-recta. Como es obvio cuando el punto se encuentra sobre la recta, la respuesta de que su distancia es nula es correcta, pero implica la imposibilidad de realizar la perpendicular si era esta nuestra pregunta. Como he dicho, la nueva versión de CADKEY, ya no incurre en este error.

Capitulo aparte merece RoboSolid, que puede generar cuerpos sólidos por swepping tanto por traslación como por rotación, ofreciendo la posibilidad de la eliminación inmediata de lineas ocultas. Añadámos a eso la posibilidad de realizar operaciones booleanas, con lo que resolvemos los casos de agujeros, secciones, prolongaciones, etc. En el capitulo negativo hemos de destacar:

a) el uso de iconos, inspirado claramente en el Apple Macintosh, que al no estar soportado por un microprocesador de la familia 68000 como aquel, da como resultado una grave ralentización de la respuesta.

b) RoboSolid debe ser soportado por otro programa, RoboCad, para obtener salidas gráficas por plotter o impresora. Dado que independientemente de la figura que tengamos representada, estamos trabajando con una representación bidimensional de ésta, para hacer manipulaciones sobre dicha representación, como pueden ser rayados o las salidas gráficas antes comentadas, hay que recurrir a Robo-

Cad. El problema radica en que RoboCad es un sistema de menor precisión, que sólo trabaja en 2D. Al hablar de precisión, me refiero a que mientras que en RoboSolid podemos entrar los puntos mediante coordenadas por teclado, en RoboCad no es posible debiendo determinar puntos sobre mallas predefinidas por el usuario, con la consiguiente pérdida de precisión.

Intercambio de información gráfica.

En 1979, y bajo el patrocinio del National Bureau Of Standards, se formó un grupo de trabajo, formado básicamente por voluntarios, que se propuso resolver los problemas que se plantean cuando queremos trasladar un dibujo creado por un paquete de CAD a otro paquete, ya que en esa época la opción usada era la del traslado directo. Una solución es crear un standard de gráficos como es el GKS, y otra hacer que el standard sea el sistema de traslado. En esta segunda via IGES (Initial Graphics Exchange Specification) fue la respuesta.

Historicamente existen tres versiones de IGES: la 1.0 fue publicada en 1981 como parte del Standard ANSI, la 2.0 en 1982 y la 3.0 en 1986. Nuevamente, nos topamos con la parte popularmente comercial de la Microinformática ya que independientemente de estos trabajos, AutoCAD crea su propio sistema DXF (Intergraph también crea otro, menos popular, SIF). Pero mientras que IGES es un archivo "neutral", DXF depende exclusivamente de las funciones que tenga implementadas la versión de AutoCAD con la que estemos trabajando; ello nos lleva a

graves problemas, por ejemplo con dibujos que hayan sido creados mediante splines, dado que no existe esta función en dicho paquete.

A consecuencia de la gran competencia entre distintos paquetes de CAD, ha resurgido IGES y se ha convertido, de hecho, en el sistema más seguro de intercambio de información gráfica, ello no obsta para que aun se le pueda augurar bastante vida a DXF, debido a formar parte del paquete más popular del mercado.

Desde nuestro punto de vista, el puramente educativo, puede resultar interesante pasar la información de un paquete de CAD a otro, para ver distintas soluciones a problemas puntuales. Un ejemplo muy obvio de esta manipulación sería enviar un cuerpo creado con CADKEY a RoboSolid para realizar con él operaciones booleanas, y a continuación volver a CADKEY para aprovechar su principal característica de la facilidad de crear vistas en 3D.

Notas y Referencias.

(1) El dibujo es una copia del que ilustra el interesante artículo de Larry Pfortmiller, "Data Structures In CAD Software", publicado en Byte en Junio de 1987.

(2) Dado que AutoCAD es el sistema standard de CAD, muchas de las utilidades de los distintos programas cuando quieren ser descritas en el argot con una sola palabra, irremediablemente se cae en la incorrección de usar justamente la que asigna para

tal utilidad AutoCAD. Yo incurriré a menudo en esta práctica, como por otra parte lo hace Glenn Hart, en su estudio sobre los distintos sistemas de CAD comercial en el número de Marzo de 1986 de PC Magazine.

(3) Para ver, desde el punto de vista del usuario, los distintos tipos de línea que ofrece AutoCAD, se pueden consultar las páginas 226 a 228 de "Inside AutoCAD" de Daniel Racker y Harbert Rice, que no es más que un manual de dicho paquete racionalizado de forma atractiva y coherente.

(4) Esta afirmación no es todo lo general que puede parecer, pues, por confirmación própia, PC-Draft no se toma estas molestias, con lo que su velocidad de respuesta mejora ostensiblemente. Mientras AutoCAD, pongamos por caso, invierte bastante tiempo para escribir todas las letras del cajetin, de forma que sean legibles, PC-Draft los sustituye por signos convencionales que pueden ser sencillamente lineas de puntos.

(5) Nuevamente, una sola palabra inglesa encierra significados amplios, en este caso, charting se refiere a la utilidad de presentar estadisticas de forma atrayente de datos introducidos en forma de tanto por ciento. Entre los chartings más conocidos podemos destacar los de pastel.

(6) Listado aparecido en Byte, Junio de 1987, articulo citado anteriormente. Sin pretender comentar el listado con profundidad, el articulo tampoco lo hace, si que puede resultar interesante observar

la forma ordenada de estructuración de los ejes coordenados, los arcos y circunferencias e incluso los textos asociados a los dibujos. Al copiarlo, no he traducido los comentarios dada la sorprendente síntesis del idioma inglés aplicado a estos menesteres.

(7) Sistemas CAD/CAM/CAE. Diseño y Fabricación por Computador. Marcombo, 1986. Varios Autores.

(8) Programming Principles In Computer Graphics por Leendert Ammeraal. John Wiley & Sons, 1986. Es un interesante, y sorprendentemente reducido, texto que nos introduce en el trabajo en 3D, apoyándose en el lenguaje C, del que incluye, al final del libro, una breve introducción.

(9) El sistema propuesto es similar a la utilización de librerías de subrutinas preparadas a tal efecto, como pueden ser, en el área gráfica, las comercializadas por Borland o Filtrex. Posteriormente, Mayo de 1987, el mismo autor ha publicado "Computer Graphics For The IBM PC", en donde aborda más claramente el problema de diseñar un CAD no comercial al que llama DIG (Drawing with Interactive Graphics) y acentua con sus dos diskettes, uno para el primer libro y uno para éste, el carácter de Toolbox Gráfico para ayuda de programadores. De hecho, el propio autor reconoce en un interesante prólogo al segundo texto, que encontró, junto con varios usuarios, serias dificultades en implementar sus diskettes de segmento alto (high level graphics) en ordenadores personales; por lo que de-

cidió crear unas aplicaciones especificas para el IBM PC (low level graphics).