EJEMPLO

Especificación de Requerimientos Software.

La presente especificación de requerimientos aparece en el proyecto fin de carrera de Juan Aº Franco Pastor y está desarrollada siguiendo las directrices de IEEE Guide to Software Requeriments Specifications.

1 Introducción.

1.1 Propósito.

El propósito del presente apartado es definir cuales son los requerimientos que debe tener un programa de diseño que exporte el objeto dibujado a distintos formatos.

Esta especificación de requerimientos está destinada a ser leída por los usuarios o cualquier sujeto que tengan interés en saber como funciona el producto.

1.2 Ámbito.

El producto que vamos a describir puede clasificarse como un programa de diseño vectorial de figuras geométricas en dos dimensiones que genera, entre otros formatos, código L^AT_EX.

El presente producto esta restringido a dos dimensiones, y por tanto no se considerará el eje *z* (profundidad) en la representación de figuras geométricas. Además, dichas figuras deben ser tratadas como objetos, obteniendo todos los beneficios de dicha representación. Es decir, el producto debe ser capaz de trasladar, eliminar, repetir o cambiar el color de una figura representada en la pantalla como una acción. Otra característica que debe tener el producto es que sea WYSIWYGWYSIWYG (What You See Is What You Get) en todos los formatos que sea capaz de exportar. Lo cual significa que lo que veamos en la pantalla del ordenador, es lo que vamos a obtener en la exportación que realicemos para un formato dado. Como consecuencia de lo expuesto anteriormente, se deduce que el programa debe tener distintos modos de funcionamiento según el formato que deseamos obtener (por las restricciones propias del formato).

Dado que el programa funcionará bajo *Windows*, deseamos que se comporte según los conceptos *Windows* de portabilidad, y funcionamiento. El documento System Application Architecture (SAA) en su apartado Common User Access (CUA) [2] de IBM recoge una serie de estándares para los programas que se ejecutan en un entorno de ventanas. Para obtener el beneficio de un menor tiempo de aprendizaje del producto por parte del usuario (ya que todos los programas *Windows* que siguen el estándar IBM funcionan del mismo modo) el producto deberá cumplir la especificación CUA.

Los formatos a los que debe ser capaz de exportar el dibujo una vez generado son:

- L^AT_EX [3]. De este modo tenemos la seguridad de una portabilidad del dibujo a muchos sistemas operativos.
- *Windows Metafile* [4]. Este es el formato que MICROSOFT ha incluido como modelo de metafichero en su sistema *Windows*. Un ***metafichero*** es un conjunto de primitivas gráficas que representan una figura. Una ***primitiva*** es una expresión que identifica una acción sobre el dispositivo gráfico. Este conjunto de primitivas esta expresado de un modo vectorial, por lo tanto, la representación de los objetos es independiente del sistema en que se dibuja. Consecuencia de ello, una buena opción de portabilidad entre programas *Windows* es incluir este formato. Este formato no debe ser confundido con el formato Aldus Placeable Metafile que muchos programas *Windows* incorporan con el mismo nombre y que en realidad son el mismo formato pero con una cabecera adicional.
- Mapa de Bits [5]. Este formato se permitirá a través del portapapeles de *Windows*, para así poder usar los dibujos generados en todas las aplicaciones *Windows* que incluyan la opción de pegado (como editores y procesadores de texto,

hojas de cálculo, ...). El ***portapapeles*** es una zona de memoria global a todas las aplicaciones que en *Window*s se usa para intercambiar información entre las mismas.

La precisión del diseño en este programa debe estar situada en centésimas de milímetro, lo cual no significa que pueda trabajar con dicha precisión con un ordenador cualquiera (siempre que soporte *Window*s). En principio, es requisito necesario para poder acotar la misma cuando deseamos exportar a formatos con menor precisión.

Por último, hay que destacar algunos aspectos de *Window*s y el presente programa:

- El programa debe poder imprimir directamente sobre la impresora, trazador gráfico o cualquier otro dispositivo de impresión.
- Si desea mayor precisión que la conseguida con un ratón normal, puede adquirir dispositivos que admitan mayor resolución (y el programa lo admitirá).
- El mismo razonamiento puede aplicarse a la pantalla. A mayor tamaño de la misma y mayor resolución, mejores serán los resultados visuales obtenidos (usando un dispositivo de entrada con una resolución semejante o superior a la de la pantalla).

1.3 Definiciones, Acrónimos y abreviaturas.

API. Término con el que se conoce un documento asociado a un producto de desarrollo de aplicaciones en el que se explica detalladamente el mismo (Interfaz del Programador de Aplicaciones).
CUA. Documento desarrollado por IBM que normaliza el comportamiento de una aplicación de ventanas [2.
Buffer. Una área de memoria utilizada para almacenamiento temporal de datos o imágenes.
Clase Abstracta. Una clase de C++ que se toma como clase base para derivar otras clase. No se puede crear un objeto o instancia de una clase abstracta.
Color de fondo. El color sobre el que se dibujan los gráficos. Por defecto es el blanco para una ventana. Cuando dibujamos un rectángulo con color de fondo verde sobre una ventana blanca, parece como si hubiésemos rellenado el mismo con dicho color verde.
Color del lápiz. Color que se aplica para dibujar las líneas de los gráficos.
Hipertexto. Documento que posee texto, sonido, gráficos y animación para explicar un tema. Normalmente se definen hiper-enlaces para acceder a temas relacionados con el tema que se lee y también se pueden realizar búsquedas.
Hiper-enlace. Palabra o zona de un dibujo que al tocarla hace que saltemos a otro tema dentro de un hipertexto.
Icono. Una imagen de bits miniatura que normalmente realiza una acción al ser seleccionada.
Librería de enlace dinámico. Una librería de una aplicación en la que el enlace con la misma se produce en tiempo de ejecución.
Metafichero. Es un conjunto de primitivas gráficas que representan una figura.
Paleta. Es una tabla cuyo contenido son colores. Existen dos paletas normalmente, la paleta hardware y una paleta lógica definida por una aplicación *Window*s para su uso propio.
Pixel. Es el elemento más pequeño que se puede identificar sobre la pantalla. En los sistemas RGB, un pixel es triada compuesta por un punto rojo, otro verde y otro azul.
Portapapeles. Es una zona de memoria global a todas las aplicaciones *Window*s se usa para intercambiar información entre las mismas.
Primitiva gráfica. Es una expresión que identifica una acción sobre el dispositivo gráfico.
Raster. Zona de memoria asociada a la pantalla.
Ventana filial. Ventana que se halla en el interior de otra ventana, estando su tamaño relacionado con el tamaño de la ventana contenedora.

1.4 Referencias.

Para la redacción de este texto se han tenido en cuenta los siguientes documentos:

[1] IEEE Std 830- IEEE Guide to Software Requeriments Specifications. IEEE Standards Board. 345 Eas 47 th Street. New York, NY 10017, USA. 1984.
[2] Lee Adams. Programación Avanzada de Gráficos en C para Windows. McGraw-Hill/ Interamericana de España, S.S. 1993. Pág.24.
[3] Leslie Lamport. LaTeX: A Document Preparation System. Addison-Wesley Publishing Company, Inc. 1986.
[4] Charles Petzold. Programación en Windows. Ediciones Anaya Multimedia, S.A. 1992. Pág: 624.
[5] Lee Adams. Programación Avanzada de Gráficos en C para Windows. McGraw-Hill/ Interamericana de España, S.S. 1993. Pág.205.
[6] David F.Rogers. Procedural Elements For Computer Graphics. McGraw-Hill, Inc. 1976. Pág: 9.
[7] El API de Windows se explica en muchos libros, pero una relación completa del mismo puede encontrarse en el archivo Windows.h de paquete Borland C++.
[8] Manual de Windows. Tulip Computers. P.O. Box 3337, 5203 DH's-Hertogenbosch. 1990.

2. Descripción General.

A continuación vamos a ver los factores que afectan al producto y a sus requerimientos.

2.1. Perspectiva del Producto.

El presente producto debe ser un programa *Windows* puro, y como tal, debe poseer las características comunes a todos ellos. Antes que nada, *Windows* 3.1 es un entorno. Aunque pueda resultar paradójico, *Windows* ejecuta los programas a la vez que los programas ejecutan a *Windows*. El paradigma de paso de mensajes hace que *Windows* y las aplicaciones cooperen como compañeros en tiempo de ejecución. Esta dependencia mútua aparece explicitada en la figura 2-1.

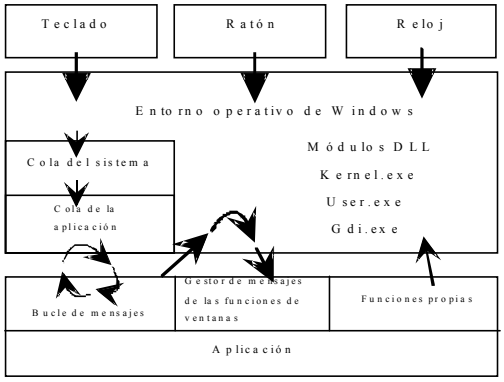


Figura 2-1

Los programas de *Windows* se interrelacionan con *Windows* a través de un proceso que se llama de ***enlace dinámico***. Cuando un programa *Windows* realiza una llamada a una función *Windows*, se realiza en tiempo de ejecución una llamada lejana a ***la librería de enlace dinámico*** (la cual se enlaza con la aplicación en ese momento, no en tiempo de compilación como ocurre normalmente). De hecho, el propio *Windows* está construido principalmente por tres librerías de enlace dinámico llamadas Kernel (responsable de la administración de memoria, carga, ejecución y planificación de tareas), User (interfaz de usuario y administrador de ventanas) y Gdi (gráficos).

Otra característica de *Windows* es que realmente es un entorno orientado a objetos. Esto se hace evidente al observar la principal característica que todas las aplicaciones poseen: la existencia del ***objeto ventana.*** Todos los cuadros de diálogo, barras de desplazamiento, menús, campos de texto, etc.. son objetos *Windows*. Cuando redimensionamos una ventana en la pantalla, *Windows* manda un mensaje al programa indicándole el nuevo tamaño de la misma. El programa entonces reajusta el contenido para reflejar el nuevo tamaño. Esta característica es muy importante, ya que amplia el concepto tradicional según el cual las aplicaciones de usuario realizaban llamadas al sistema operativo unidireccionalmente. Esto es fundamental para la arquitectura orientada a objetos de *Windows*. Cada ventana que crea un programa tiene asociado un procedimiento de ventana. Este procedimiento de ventana es una función que podría estar dentro del mismo programa o en una librería de enlace

dinámico. *Windows* manda un mensaje a una ventana por medio de una llamada al procedimiento de ventana. El procedimiento de ventana ejecuta algunos procesos dependiendo del mensaje y después devuelve el control a *Windows*. Hablando con más propiedad, una ventana se crea siempre en base a una *clase de ventana.* La clase ventana identifica el procedimiento de ventana que procesa los mensajes de la misma. Cuando empieza la ejecución de un programa en *Windows*, este crea una ***cola de mensajes*** para el programa. El programa incluye un pequeño proceso denominado ***bucle de mensajes***, para recoger estos mensajes de la cola y despacharlos al procedimiento de ventana adecuado. Otros mensajes se mandan directamente al procedimiento de ventana sin pasar por la cola de mensajes para obtener un menor tiempo de respuesta en la gestión de los mismos.

En cuanto a la máquina a usar en el programa, puede ser cualquiera, que soporte *Windows* 3.1 o superior. MICROSOFT se ha comprometido a poder instalar el sistema (*Windows NT*) en todos los modelos de máquina existentes, desde PC's hasta ordenadores RISC con varios procesadores, estando asegurado en la actualidad el uso por parte de las aplicaciones (y en concreto de este producto) de un segundo microprocesador. Además, se asegura que la aplicación será portable de un sistema a otro en caso de necesitar mayores prestaciones (velocidad, etc..) sin costes adicionales. Según MICROSOFT, como mucho, habrá que recompilar el programa, por ejemplo para acceder a propiedades como que el programa trabaje con 32 bits.

Como consecuencia de todo lo dicho, se deduce que el programa será independiente de cualquier dispositivo que esté conectado al ordenador. Esto significa que si se dispone de una tableta digitalizadora en vez de un ratón, un trazador en vez de una impresora y un monitor de alta resolución de gran tamaño, tendremos una estación de diseño aceptable, adaptándose el programa a las nuevas características sin tener que modificar ni siquiera la instalación del producto.

2.2. Funciones del Producto.

Las funciones que debe realizar el producto la podemos clasificar en varios bloques:

a) Almacenamiento del objeto dibujado.

- Almacenar en memoria el dibujo realizado como unión de primitivas.
- Almacenar y recuperar el objeto dibujado en un metafichero con un formato definido para el producto.
- Exportar el objeto dibujado como mapa de bits a través del portapapeles.
- Exportar el objeto dibujado como metafichero con formato *Windows*.
- Comenzar otro objeto eliminando de la memoria el que ya existía.
- Exportar el objeto dibujado con formato $\LaTeX$.

b) Herramientas de ayuda en la gestión del diseño.

- Imprimir el objeto dibujado sobre cualquier dispositivo gráfico, como trazadores o impresoras.
- Poder configurar el dispositivo de impresión sin salir del programa.
- Incorporar unas reglas en centímetros que permitan el perfecto ajuste de los dibujos al tamaño deseado por el usuario.
- Activar o desactivar una rejilla de fondo para que puedan situarse mejor los objetos dibujados.

- Adaptar las características del formato de exportación al funcionamiento del programa: Cambiar menús, fuentes admitidas, colores, tipos de herramientas así como sus modos de funcionamiento y todo cuanto afecte al formato en el que se esté trabajando.

c) Herramientas de diseño en dos dimensiones (2D).

- Dibujo a mano alzada.
- Dibujo de rectángulos.
- Dibujo de elipses.
- Dibujo de óvalos.
- Dibujo de líneas rectas.
- Escribir textos.
- Seleccionar la fuente y tamaño del texto.
- Cambiar los colores del lápiz y fondo de un objeto.
- Dibujo de vectores.
- Cambiar el grosor de las líneas.
- Rellenado de zonas cerradas.

d) Mantenimiento de las formas geométricas diseñadas.

- Seleccionar un conjunto de objetos para realizar operaciones con ellos.
- Copiar al portapepeles un objeto o conjunto de objetos seleccionados.
- Pegar tantas veces como se desee el objeto u objetos que tenemos en el portapapeles.
- Eliminar los objetos seleccionados.
- Poder realizar la traslación del objeto seleccionado a otras coordenadas.
- Cambiar el color y/o grosor de una primitiva geométrica simple seleccionada.
- Deshacer la última operación realizada, volviendo al estado anterior.

e) Ayuda mediante un archivo hipermedia con formato *Windows* .

- Definir un índice de ayuda de entrada al archivo.
- Explicar el menú de la aplicación.
- Clarificar algunos usos de funcionamiento que estén confusos.
- Explicar como se debe usar la ayuda.

2.3. Características del Usuario.

A continuación vamos a ver qué tipo de usuarios van a usar el producto y como afectan estos a las funciones que debe realizar el producto.

En primer lugar, aparecen los usuarios *Window*s que pueden usar el producto como un pequeño programa de diseño. Estos usuarios esperan un comportamiento CUA de la aplicación, y por este motivo se respetará dicha norma. Otras características que esperan los usuarios es poder imprimir el dibujo, configurar la impresora que poseen, o copiar el dibujo al portapapeles para usarlo con otras aplicaciones. Otro modelo de usuarios son los que poseen estaciones de CAD, que, aunque el producto funciona perfectamente sobre dichos sistemas, se supone usarán otros productos más avanzados. De todos modos, se debe poner una resolución elevada para poder usar dichos sistemas. Un siguiente nivel de usuarios son los que precisan de un programa de diseño que genere el formato *Windows Metafile* y por este

motivo se incluirá este formato de exportación. El siguiente nivel son los usuarios L^AT_EX, que impondrán en el conjunto de herramientas soportadas por el producto restricciones importantes. Los usuarios que realizarán el mantenimiento del producto desean que la configuración, instalación o reinstalación del mismo sea lo más simple posible. En concreto, un usuario L^AT_EX puede tener fuentes adicionales a las que posee otro usuario L^AT_EX, por tanto, deben poder elegir las fuentes que van a usar en dicho modo. Si deseamos realizar reinstalaciones por cambio de máquina, hay que tener en cuenta el concepto de independencia de dispositivos detallado anteriormente. Otros usuarios a destacar son los distribuidores de aplicaciones que para acceder a mercados internacionales precisan de la característica de independencia del idioma.

2.4. Obligaciones Generales.

Otro aspecto que influye en los requerimientos es la eficiencia. Dado que el programa funciona bajo *Windows*, posee la característica de portabilidad. Pero esta portabilidad se paga con una menor velocidad. Los sistemas de diseño que actúan directamente sobre el ***raster*** (zona de memoria asociada a la pantalla) [6] tardan menos en realizar funciones como dibujar formas geométricas sobre la pantalla. Si consideramos por ejemplo un programa que recibe como entrada los movimientos entrada de piezas a través de una cinta trasportadora, y el ordenador debe ir dibujando cualquier acción con dicha entrada, entonces posiblemente necesitamos acceder al raster para dibujar y así obtener un menor tiempo de respuesta. Pero por razones de portablidad, consideramos que tal acción es contraproducente para el producto que estamos describiendo.

2.5 Asunciones y Dependencias.

El presente producto depende enteramente de *Windows* y donde llega *Windows* 3.1 y sus futuras versiones (*Chicago*, *Windows NT* ...) debe poder llegar el programa, ya que debe cumplir todas las normas *Windows* expresadas en su API (Interfaz para el Programador de Aplicaciones) [7] y por tanto MICROSOFT nos asegura la migración.

3. Requerimientos Específicos.

3.1. Requerimientos funcionales.

3.1.1. Almacenar en memoria el dibujo realizado como unión de primitivas gráficas.

El presente producto debe ser capaz de almacenar en memoria una representación de cada uno de los objetos expresados en pantalla. Esta función es condición necesaria para que puedan realizarse todos y cada uno de los requerimientos que se expresan en los apartados siguientes.

Esta función recibe como parámetro el objeto que se ha dibujado sobre la pantalla. El proceso consiste en almacenar en memoria las primitivas geométricas dibujadas mediante una representación interna que identifique todas y cada una de las propiedades de cada una de las primitivas dibujadas sobre la pantalla. El presente producto debe usar toda la memoria disponible para dicha tarea. En caso de no tener suficiente memoria para tal fin, deberá avisar al usuario.

3.1.2. Almacenar y recuperar el objeto dibujado en un metafichero con un formato definido para el producto.

Existen dos tendencias de representación en imágenes: las matrices de puntos y los metaarchivos (o metaficheros). En las matrices de puntos, se almacena y se visualiza la configuración real del pixel para una imagen. Este tipo de gráficos es mas rápido de visualizar y es el único método existente de representar imágenes del mundo real tales como fotografias. Los metaarchivos por su parte almacenan la información gráfica como una serie de llamadas al interfaz del dispositivo gráfico. Las ventajas de un metaarchivo frente a un mapa de bits son muchas, entre las que destacan una

mayor independencia del dispositivo de impresión, así como poseer propiedades de modificación y tratamiento de las imágenes sin perdida de información.

El producto deberá obtener un nombre de fichero valido para realizar la lectura o la escritura del metafichero. Una vez obtenido, podrá almacenar el metafichero en el disco (en caso de desear almacenar los dibujos realizados sobre la pantalla). La condición que debe cumplir dicha escritura es que el metafichero debe poseer un formato de fácil lectura por parte de usuarios avanzados mediante cualquier editor de textos.

En caso de que la operación sea una lectura de un metafichero, se procederá a representar las primitivas almacenadas sobre la zona de dibujo, obteniéndose la figura expresada en el interior del mismo y permitiendo cualquier modificación sin pérdida de ninguna propiedad, es decir, los objetos expresados sobre la pantalla se comportarán del mismo modo que si se acabasen de dibujar.

3.1.3. Exportar el objeto dibujado como mapa de bits a través del portapapeles.

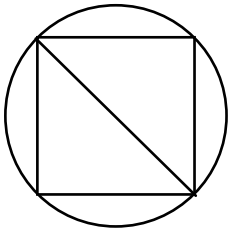
El API de *Windows* posee un conjunto de funciones que son útiles para la programación del portapapeles. El portapapeles es una zona de memoria global a todas las aplicaciones mantenida por *Windows* en tiempo de ejecución. Como todos los programas de *Windows* pueden leer y escribir en el portapapeles, es un método apropiado para compartir imágenes entre todas las aplicaciones.

El proceso consiste en copiar el dibujo representado en la pantalla al portapapeles de *Windows*. La salida producida será una imagen en el portapapeles que será copia del dibujo realizado con el producto. De este modo, puede ser tratada por otras aplicaciones *Windows* como *PaintBrush* o incrustada en un documento *Write*.

3.1.4. Exportar el objeto dibujado como metafichero con formato Windows.

Windows posee un formato de metafichero propio que lo usan muchas aplicaciones como formato de importación y exportación. Un metafichero puede verse como una sucesión de primitivas que representan un dibujo. Un ejemplo de representación puede ser:

Rectángulo(0,0,100,100)
Círculo(50,50,50)
Línea(0,0,100,100)



El presente producto debe poseer como formato de exportación el de los metaficheros de *Windows*. Existen dos formatos de metafichero *Windows*, el formato *Windows* puro y otro formato denominado *Windows placeable metafile*. Se soportará únicamente el primer formato.

El proceso comienza validando el nombre y existencia de dicho fichero en el disco. Si el este existe, se debe considerar la operación como una sobre escritura.

3.1.5. Comenzar otro objeto eliminando de la memoria el que ya existía.

La aplicación posee un estado en un momento dado, definido al almacenar en memoria las selecciones realizadas sobre utilidades. Evidentemente, el comenzar un nuevo dibujo, no significa que debemos eliminar todas las selecciones realizadas. Este proceso, simplemente descarga la memoria ocupada por el objeto expresado en la pantalla y limpia la ventana de dibujo.

3.1.6. Imprimir el objeto dibujado sobre cualquier dispositivo gráfico, como plotters o impresoras.

En muchos casos, la utilización de la impresora y del trazador gráfico en *Windows* es sencilla. Los controladores de dispositivos para la impresora y el trazador

facilitados por *Windows*, o por los fabricantes de impresoras y trazadores, proporcionan una amplia gama de dispositivos de salida. El tratamiento que realiza *Windows* sobre estos dispositivos es como si se estuviera dibujando sobre la pantalla, usando las mismas funciones. A veces, este tratamiento puede resultar complicado ya que se necesita soportar la inicialización del dispositivo (formato de página, etc..), u otros elementos de control que complican el proceso.

El producto debe poder emitir por el dispositivo de impresión activo de la máquina una expresión exacta del contenido de la pantalla. Esta característica permite afirmar que el producto se va a comportar como WYSIWYG. Por otro lado, se soportarán todos los dispositivos conectados al sistema como impresoras, trazadores, salidas fax/módem, etc..

3.1.7. Poder configurar el dispositivo de impresión sin salir del programa.

La impresora que emplea para la salida *Windows*, se determina leyendo la sección [*Windows*] del archivo WIN.INI [8]. La sección [devices] de este archivo contiene información sobre las impresoras y los controladores que pueden estar conectados al sistema. De este modo, puede obtenerse qué dispositivo está en estado activo conectado al sistema y cargar de su controlador asociado el proceso de configuración suministrado por *Windows* o el fabricante. Así, pueden obtenerse funciones como imprimir la página vertical u horizontalmente, seleccionar el tamaño de página, la resolución con la que queremos que se realice la impresión, y muchas cosas más que están en función del dispositivo conectado.

El proceso consiste en obtener el nombre del controlador del dispositivo, y a partir del mismo, ejecutar el proceso de configuración del dispositivo suministrado por el fabricante (mediante una librería de enlace dinámico con la extensión .DRV).

3.1.8. Incorporar unas reglas en centímetros que permitan el perfecto ajuste de los dibujos al tamaño deseado por el usuario.

Para poder ajustar el dibujo que se esté realizando sobre la pantalla a la realidad, debemos tener unas reglas (una horizontal y otra vertical) que nos permitan plasmar a escala un dibujo realizado sobre papel. Estas reglas deben poseer la propiedad de ser independientes del dispositivo de visualización, y por tanto adaptarse al modelo de pantalla que estemos usando. Otra propiedad es que deben estar sincronizadas con los movimientos del ratón, indicándonos en todo momento en qué lugar nos hallamos. La posición de las citadas reglas será el pegar lo más cerca posible a la zona de dibujo para un mayor ajuste de los objetos dibujados.

3.1.9. Activar o desactivar una rejilla de fondo para que puedan medirse mejor los objetos dibujados.

Este proceso consiste en dibujar sobre la ventana de dibujo una cuadrícula para poder ajustar con mayor precisión los objetos dibujados. Esta cuadrícula debemos poder activarla o desactivarla según deseemos. Cuando se entra al programa, este deberá ponerla si al salir la última vez se dejó activada.

El proceso consiste en ir imprimiendo cada cierta distancia unas cruces que formarán la cuadrícula dentro de la ventana de dibujo.

3.1.10. Adaptar las características del formato de exportación al funcionamiento del programa.

El presente producto admite tres formatos de exportación: *Windows metafile*, *Bitmap* (a través del portapapeles) y LaTeX. Los dos primeros son propios de *Windows* , en cambio LaTeX es multiplataforma. Evidentemente, para que el producto sea más eficiente a la hora de exportar dibujos sobre un formato, debe adaptarse a dicho modelo. Por tanto se deduce que el producto trabajará en dos modos: el modo *Windows* y el modo LaTeX.

En el modo *Windows*, no debemos poner ninguna restricción, dado que al estar el producto desarrollado bajo este sistema, se adapta perfectamente a las necesidades exigidas.

Veamos las restricciones que debemos poner al producto al trabajar en modo LaTeX (entorno picture). Al lado de cada restricción se ha incluido el apartado en donde se analizan todos los aspectos que influyen en la misma.

- Debemos anular los colores, dado que LaTeX trabaja en color negro fundamentalmente. 3.1.18.
- Con LaTeX, tenemos dos modelos de grosor de línea al dibujar figuras geométricas. 3.1.20.
- No puede dibujar elipses. 3.1.13.
- El tamaño de los círculos está limitado a 40 puntos. 3.1.13.
- El tamaño de los discos (círculos rellenos de negro) también está limitado. 3.1.13.
- La pendiente de las rectas que pueden dibujarse debe poseer unos ciertos ángulos. 3.1.15.
- La pendiente de los vectores debe poseer otros ángulos. 3.1.19.
- Las fuentes que admite LaTeX para escribir texto están limitadas.. 3.1.17.
- La pendiente de las rectas dibujadas a mano alzada debe poseer los mismos ángulos que en el caso de las rectas que hemos comentado anteriormente. 3.1.11.
- Existen objetos en modo *Windows* que no tienen sentido en modo LaTeX, como por ejemplo el relleno de zonas irregulares con color.

Todas las restricciones que vamos a poner al producto son debidas a que las hemos encontrado en información referente al propio LaTeX, o bien , han sido probadas y contrastadas bajo MSDOS sobre un sistema LaTeX (PCTeX). Pero esto no significa que LaTeX no pueda hacer estas cosas, simplemente, significa que con los medios de que disponemos, nos vemos obligados a restringir el producto. Un ejemplo de lo dicho puede ser que en algunas versiones de LaTeX (extensiones **eepic** sobre LaTeX), existen primitivas para dibujar elipses: \ellipse{x-diameter}{y-diameter} y en cambio, no podemos incluirlo como opción dado que, según se documenta en la fuente de información de donde hemos leído esta afirmación, se dice, que ni siquiera en todas las extensiones eepic de LaTeX se da esta posibilidad, realizando algunas de ellas aproximaciones mediante óvalos.

El cambio de modo de trabajo vendrá dado al seleccionar una opción del menú, que deberá quedar permanente aún cuando se salga del programa y se desee entrar de nuevo. De todas las restricciones comentadas anteriormente, la selección de la fuente para el texto es la única que presenta una entrada. Este concepto lo explicamos en el apartado 3.1.17, en donde se detallan todos los aspectos a tener en cuenta.

Mediante estas restricciones, se obtiene un modo de trabajo adecuado al formato que deseamos exportar.

3.1.11. Dibujo de líneas rectas.

Mediante esta opción, el producto nos permite dibujar líneas rectas entre dos puntos sobre la ventana de dibujo. Veamos ahora algunos aspectos a considerar en cuanto al dibujo y representación de una línea. En primer lugar, debe empezarse por situar un punto sobre la zona de dibujo. A partir de éste debemos situar otro punto sobre la pantalla. En *Windows,* y dado que no existe ninguna restricción, unimos ambos puntos mediante una recta. Pero bajo el modo LaTeX, esto cambia bastante, dado que debemos analizar la pendiente de la recta que pasa por esos dos puntos, y modificar la línea para que se ajuste exactamente a una recta válida LaTeX.

Las entradas que recibe este proceso son varias. En primer lugar están los eventos de ratón que se producen para representar el objeto. Estos son varios y pueden clasificarse en: pulsar el botón izquierdo, mover el ratón a una posición final y soltarlo. Además también necesitamos un lápiz de relleno (representado por la selección de su color) y otro para la tinta (representado por el color del lápiz y el grosor seleccionados).

El proceso consiste en situarse sobre un punto inicial sobre la pantalla y pulsar el botón izquierdo del ratón. Acto seguido, y manteniendo pulsado dicho botón, movemos el ratón hasta el punto final que deseamos forme la figura. A medida que movemos el ratón, se va mostrando la primitiva u objeto sobre la pantalla para que nos hagamos una idea sobre su tamaño y representación. De este modo, es fácil ver como va a quedar el objeto representado. Cuando ya creemos que tenemos el objeto como deseábamos, soltamos el ratón, visualizándose la figura en su posición final.

Muchas de las funciones que vamos a detallar se comportan del mismo modo que el descrito para este caso; por este motivo, al expresar el modo de funcionamiento de estas funciones, haremos referencia a dicho comportamiento indicando que "se comportan de mismo modo que el caso de las líneas".

3.1.12. Dibujo de rectángulos.

Esta función nos permite dibujar rectángulos sobre la zona de dibujo. En este caso, no existen restricciones en modo $L^{A}T_{E}X$.

Este proceso se comportará del mismo modo que las líneas, y posee sus mismas entradas. Los eventos de ratón consisten en pulsar el botón izquierdo, mover el ratón a una posición final y soltarlo. Además también necesitamos un lápiz de relleno (representado por la selección de su color) y otro para la tinta (representado por el color del lápiz y el grosor seleccionados).

3.1.13. Dibujo de elipses.

Esta función nos permite dibujar elipses y círculos (como una restricción del caso de la elipse) sobre la zona de dibujo. En modo $L^{A}T_{E}X$, existen algunas restricciones. En primer lugar, no pueden representarse elipses, y por tanto solo pueden representarse círculos. Otra restricción es que el rádio del círculo no puede ser superior a 40 puntos.

Es evidente que tanto las entradas que recibe este proceso como el modo de funcionamiento (expresado por los eventos de ratón) son idénticos al caso de las líneas. La única diferencia estriba en que el objeto representado al dibujar será una elipse.

3.1.14. Dibujo de óvalos.

Esta función nos permite dibujar óvalos sobre la zona de dibujo. En cuanto al modo de funcionamiento, es idéntico al descrito para el caso de las líneas, salvo que el objetos expresado es un óvalo.

3.1.15. Dibujo a mano alzada.

Mediante esta opción, el producto nos permite dibujar a mano alzada sobre la ventana de dibujo. En modo $L^{A}T_{E}X$, existe la restricción de que la pendiente de cada una de las rectas que se van formando entre los puntos consecutivos por los que nos movemos no puede ser cualquiera. Para ello, debemos calcular cuál es la recta cuya pendiente es más próxima.

Esta acción consiste en situarse sobre un punto inicial sobre la pantalla y pulsar el botón izquierdo del ratón. Acto seguido, y manteniendo pulsado dicho botón, movemos el ratón hasta el punto final que deseamos forme la figura. A medida que nos movemos se van dibujando pequeñas rectas uniendo todos los puntos por los que se pasa, formando el trazo deseado. Cuando ya creemos que tenemos el objeto como deseábamos, soltamos el ratón.

3.1.16. Escribir textos.

Esta función nos permite escribir texto en cualquier lugar de la pantalla. Para ello, hay que considerar que se debe aplicar una *fuente* a dicho texto. Una fuente define los atributos del texto que deseamos escribir, esto es, su color, tamaño, tipo de letra y propiedades como cursiva, negrita, etc...Por defecto, existe una fuente seleccionada cuando entramos al programa, pero puede cambiarse si se desea. Véase el punto 3.1.17. si se desea más información. El texto en sí mismo, no está restringido en modo LaTeX, aunque tinene algunas características especiales, tales como restingir la representación de algunos simbolos. Podemos situar el texto en cualquier punto de la pantalla sin ningún tipo de restricción respecto a la posición.

Si estamos en modo LaTeX, podemos escribir los símbolos especiales del modo en que nos aconseja LaTeX o bien en el modo natural de una máquina de escribir. En este último modo, se ve el texto como va a quedar sobre el papel. Los símbolos que admite el programa son:

á é í ó ú	Á É Í Ó Ú	à è ì ò ù	À È Ì Ò Ù
ä ë ï ö ü	Ä Ë Ï Ö Ü	Ñ ñ	Ç ç

Como entrada el proceso recibe el evento de ratón que se produce al pulsar el botón izquierdo para situar el texto en un punto dado. También necesitamos un lápiz para la tinta (representado por su color y grosor seleccionados). Otra información necesaria es el tipo de fuentes que debemos aplicar al texto.

Primero nos situaremos sobre un punto inicial en la pantalla y pulsaremos el botón izquierdo del ratón. Acto seguido, se abre una ventana donde introducimos el texto deseado. Posteriormente, al terminar la línea, debemos pulsar la tecla de retorno de carro para aceptar la entrada o la tecla de escape para anular la misma.

3.1.17. Seleccionar la fuente y tamaño del texto.

Esta función nos permite seleccionar la fuente que deseamos se aplique a el siguiente texto que se escriba. La configuración de la fuente se almacena en memoria para posteriores usos. En modo *Windows*, existen un conjunto de fuentes con una serie de propiedades. En cambio, en modo LaTeX, el conjunto de las fuentes y sus propiedades son distintas. Para solucionar este problema cuando estamos en modo LaTeX, deberemos realizar una cierta correspondencia de las fuentes LaTeX y sus correspondientes fuentes *Windows*. Como conclusión, se deduce que en función del modo en que estemos, tendremos la posibilidad de seleccionar un conjunto de fuentes u otro y aplicarle un conjunto de propiedades u otro.

Esta función necesita saber el color de la tinta seleccionada (ya que durante la selección se debe sacar una muestra de la fuente con el color seleccionado para hacernos una idea más exacta). Otra información necesaria es el conjunto de fuentes que pueden seleccionarse y la correspondencia que existe en cuanto a tipos y tamaños de las fuentes LaTeX sobre las fuentes *Windows*.

El proceso consiste en detectar el modo de funcionamiento, ya sea LaTeX o *Windows*, y sacar un cuadro de diálogo con todas las características que pueden aplicarse a las fuentes. El usuario puede seleccionar un tipo de fuente del conjunto de fuentes admitidas (según el modo), presentándose una muestra de cómo va a verse el texto sobre dicha fuente. Además, se mostrarán al usuario diferentes tamaños, que podrán ser elegidos para una fuente concreta.

3.1.18. Cambiar los colores del lápiz y fondo seleccionados.

Esta función nos permite seleccionar el color de la tinta de relleno y el color de la tinta del lápiz a aplicar a la siguiente primitiva que se dibuje. La configuración de los colores se almacena en memoria para posteriores usos. Por supuesto y como

hemos dicho antes, esta propiedad de seleccionar los colores no existe en modo LaTeX, por tanto, este proceso no podrá realizarse en dicho modo.

Se necesita una tabla en memoria con los colores que podemos seleccionar. Otra información necesaria es el modo en que estamos trabajando.

Se mostrará una ventana con 48 colores que representan los colores que se puede seleccionar para el lápiz y el relleno al recibirse el evento de ratón sobre uno dado. Los colores seleccionados se almacenan en memoria para posteriores usos.

3.1.19. Dibujo de vectores.

Mediante esta opción, el producto nos permite dibujar un vector entre dos puntos sobre la ventana de dibujo. Veamos ahora algunos aspectos a considerar en cuanto al dibujo y representación. En primer lugar, debe de empezarse por situar un punto sobre la zona de dibujo. A partir de este debemos situar otro punto sobre la pantalla. En *Windows* y dado que no existe ninguna restricción, unimos ambos puntos mediante un vector, pero bajo el modo LaTeX, esto cambia bastante, dado que debemos ver la pendiente del vector, y modificar el mismo para que se ajuste exactamente a un vector válido LaTeX.

Tanto los parámetros de entrada, como el proceso en sí mismo, se comportan del mismo modo que en el caso de dibujar una línea.

3.1.20. Cambiar el grosor de las líneas.

Esta función nos permite seleccionar el grosor de las líneas a aplicar a la siguiente primitiva que se dibuje. La configuración del grosor se almacena en memoria para posteriores usos. En modo LaTeX, solo existen dos modelos de grosor, y por tanto esta restricción se verá plasmada en el producto.

Se necesita una tabla en memoria con los tamaños que podemos seleccionar. Otra información necesaria es el modo en que estamos trabajando, dado que si estamos en modo LaTeX, no deberán poder hacerse estas funciones.

Mostraremos una ventana con los tamaños y al pulsar el ratón sobre la misma se pondrá una marca sobre el tamaño seleccionado. Este se almacenará para posteriores usos.

3.1.21. Rellenado de zonas cerradas.

Esta función nos permite rellenar una zona cerrada irregular con el lápiz de relleno seleccionado. Dado que en modo LaTeX no existe el color, esto no podrá hacerse. En LaTeX, existe un objeto denominado disco que es un círculo relleno de color negro. Este es el único objeto relleno que se va a permitir. En cuanto al relleno, debemos decir que existen dos modo de relleno. El primero de ellos, se usa cuando deseamos rellenar un objeto regular para el que existe una herramienta que lo dibuja y a la que podemos decirle que su color de fondo es uno dado. El otro modo de relleno se usa cuando deseamos rellenar una zona irregular.

Para tal tarea existe una herramienta que una vez seleccionada, nos permite rellenar dichas zonas irregulares. Esta herramienta recibe como entrada un evento de ratón seleccionando un punto de la pantalla (interior a una zona irregular). Además también necesitamos un lápiz de relleno (representado por la selección de su color).

Seleccionaremos un punto de la pantalla como origen del relleno. Este relleno se hace con la tinta de relleno seleccionada. El proceso va rellenado de color los puntos contiguos al punto original mientras no choque con las paredes que definen la zona irregular. Es importante tener la zona completamente cerrada, dado que podemos llegar a rellenar toda la pantalla si existe una obertura.

3.1.22. Seleccionar un conjunto de objetos para realizar operaciones con ellos.

Esta función nos permite seleccionar un conjunto de objetos dibujados sobre la pantalla para realizar a partir de la misma operaciones que serán detalladas en otros puntos. Esta opción no está restringida en modo L^AT_EX.

Para cada uno de los objetos que dibujamos sobre la pantalla, tenemos una primitiva asociada que representa dicho objeto. Deberá existir un mecanismo que relacione cada una de las primitivas con una región de la pantalla, Este mecanismo lo denominaremos ***diana***. Al situarnos con el ratón sobre un objeto de la pantalla y pulsar el mismo, se analiza si la posición en que nos hallamos incide en el interior de su diana. En caso afirmativo, dicho objeto queda seleccionado.

Para comenzar una selección se usará el botón izquierdo del ratón sobre un objeto. Una vez seleccionado uno cualquiera, se usará el botón derecho para seleccionar los restantes. Se pueden seleccionar tantos objetos como se deseen repitiendo la pulsación del botón derecho. La selección se pierde cuando seleccionamos otra herramienta distinta del proceso de selección.

La salida que se obtiene es un conjunto de primitivas seleccionadas sobre la pantalla. Con estas primitivas seleccionadas, pueden realizarse varias acciones que veremos posteriormente.

3.1.23. Copiar al portapapeles un objeto o conjunto de objetos seleccionados.

El portapapeles es una zona de memoria donde almacenamos información para compartirla con otras aplicaciones o con la propia aplicación. Dado que la información que deseamos compartir con otras aplicaciones es normalmente un mapa de bits, y que la información que deseamos compartir con la propia aplicación son objetos, o sea primitivas, se deduce que el comportamiento del portapapeles será distinto si poseemos objetos seleccionados o no.

Las entradas que recibe este proceso pueden ser de dos modos: si no hay una selección de objetos realizada, significa que deseamos copiar la ventana de dibujo al portapapeles como mapa de bits. Si en cambio, existe una selección de objetos, se entiende que se desea copiar dicho conjunto de objetos al portapapeles para realizar posteriores operaciones de pegado de objetos.

Tanto en el caso de querer copiar la pantalla como mapa de bits, como si deseamos copiar los objetos seleccionados al portapapeles, el proceso consiste en seleccionar la opción de copia. La salida que se consigue con este proceso es almacenar las primitivas en el portapapeles para posteriores usos.

3.1.24. Pegar tantas veces como se desee el objeto u objetos que tenemos en el portapapeles.

La función que vamos a detallar es la complementaria de la acción explicada anteriormente. Tampoco posee ninguna restricción en modo L^AT_EX.

Este proceso consiste en pegar sobre la ventana de dibujo el mismo conjunto de objetos que tenemos seleccionados, con todas sus propiedades (posición, color, tamaño, etc..). Además, sobre la zona de dibujo aparecerá una copia exacta del conjunto de objetos que se hallan en el portapapeles. Normalmente esta acción de pegado se hace justo detrás de la copia, y por tanto tenemos seleccionado el citado conjunto de objetos; por tanto es fácil realizar operaciones como desplazar el conjunto de objetos a otra posición, obteniendo una copia del objeto sobre otra zona de la pantalla.

3.1.25. Eliminar los objetos seleccionados.

Esta acción se realiza después de haber seleccionado un conjunto de objetos estando en cualquiera de los modos de trabajo existentes. No se halla restringida en modo L^AT_EX.

El proceso consiste en eliminar al pulsar la tecla <Supr> (<Del> en teclados ingleses) el conjunto de objetos seleccionados de la pantalla. Evidentemente, esto elimina de la memoria los correspondientes objetos.

3.1.26. Poder realizar la traslación del objeto seleccionado a otras coordenadas.

Una vez realizada una selección de un conjunto de objetos sobre la ventana de dibujo, podemos mover los mismos a otra posición.

El proceso comienza con la selección de un objeto. Si poseemos un objeto seleccionado, podemos moverlo sobre la pantalla pulsando el botón izquierdo del ratón. Cuando movemos el ratón (manteniéndolo pulsado), deberemos ver una copia de la forma del objeto que se desplaza. De este modo, podemos situar el objeto que movemos de forma muy precisa sobre una posición de contacto sobre otro objeto. Al soltar el botón del ratón, el objeto quedará situado sobre las nuevas coordenadas. Otra posibilidad es tener una selección múltiple de objetos, en cuyo caso procederemos del mismo modo, solo que realizaremos la acción moviendo el conjunto de objetos al mantener pulsado el botón derecho del ratón.

3.1.27. Cambiar el color y/o grosor de una primitiva geométrica simple seleccionada.

Esta acción se realiza después de haber seleccionado un objeto. Además en modo L^AT_EX existen solamente dos tamaños de grosor de líneas, y por tanto esta acción también quedará restringida. Este proceso permite cambiar el color del lápiz tanto de relleno como de las líneas usadas para su representación, así como el grosor de las mismas.

Esta acción comienza con la selección de un objeto. Si poseemos un objeto seleccionado, podemos cambiar su color situándonos sobre la paleta de colores y pulsando el botón izquierdo (para cambiar el fondo) o el derecho (para cambiar el color de las líneas). Si nos situamos sobre la ventana de selección de grosores de línea, cambiaremos el grosor de las líneas de la primitiva con solo seleccionar un nuevo grosor sobre la ventana de grosores de línea. Estas acciones quedarán registradas en las zonas de memoria usadas para tal fin, esto es, el nuevo color de fondo y de tinta, se almacenarán para posteriores usos en la zona correspondiente, el nuevo grosor también será registrado, visualizándose los cambios sobre las ventanas correspondientes.

3.1.28. Deshacer la última operación realizada, volviendo al estado anterior.

Esta función permite volver al estado anterior cualquier acción emprendida. Por ejemplo, podemos borrar objetos y en caso de habernos equivocado, rectificar tal acción.

A medida que vamos trabajando, el producto va almacenando los distintos estados en que se halla la aplicación. Cuando realizamos una acción (sea la que fuere), se registrada en memoria el estado anterior a la acción realizada. En caso de desear deshacer la última acción, se accede a dicho registro de memoria y se identifica la operación anterior a la última acción realizada, reconstruyendo el estado de la aplicación al momento inmediato a la última acción emprendida. Esta función modifica el estado de la ventana de dibujo cambiando el aspecto y número de los objetos dibujados.

3.1.29. Definir un índice de ayuda de entrada a un archivo hipermedia.

Esta función asiste al usuario cuando tiene algún problema e intenta solucionarlo. Consiste en mostrar el índice de un archivo hipermedia que posee el producto. Este documento posee varias imágenes que hacen más comprensible y fácil de entender los procesos. Existen zonas dentro de las imágenes que son sensibles a pulsaciones del ratón. Si pulsamos el ratón sobre dichas zonas se accede a otra ventana que explica conceptos relacionados con el tema que estamos leyendo. Además deben poder localizarse todos los temas relacionados con una palabra dada.

3.1.30. Explicar el menú de la aplicación.

Normalmente, el usuario hace preguntas sobre que acciones puede realizar con los comandos el menú. Un ***comando*** es una posible selección del mismo. Este proceso asiste al usuario en como debe usar dichos comandos.

3.1.31. Clarificar algunos usos de funcionamiento que estén confusos.

Existen algunos puntos en la aplicación que precisan de un mayor énfasis en la explicación de los mismos, dado que todos los usuarios incurren frecuentemente en el mismo error. Estos vicios de pensamiento en que incurren muchos usuarios, es conveniente explicarlos de modo particular, para resaltar más su presencia y que sea más fácil su localización y aprendizaje; (a mayor número de usuarios con el mismo problema, mayor énfasis en intentar solucionarles esa cuestión).

3.1.32. Explicar como se debe usar la ayuda.

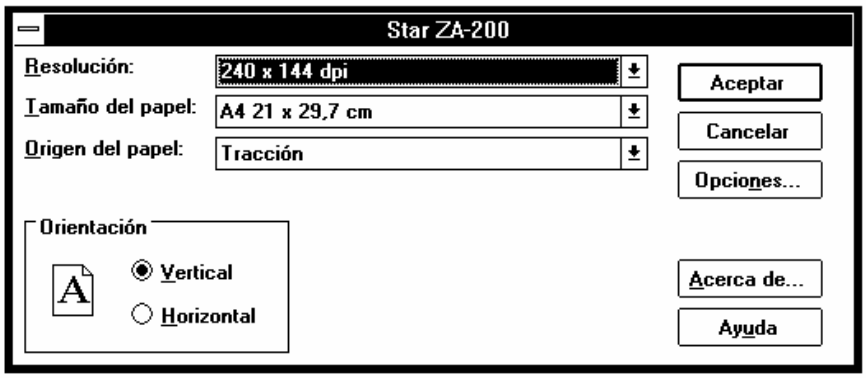
Esta función nos permite aprender como se usa un la ayuda de *Windows*. Este proceso no presenta ninguna entrada. El proceso consiste en presentar la propia ayuda de *Windows* para su uso.

3.2. Requerimientos de interfaces externos.

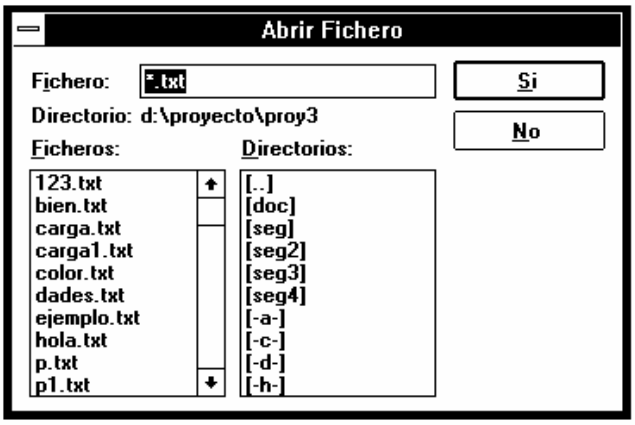
3.2.1 Interfaces de usuario.

En este apartado, vamos a hablar de los procesos existentes entre el ordenador y el usuario. Para ello haremos dos grandes grupos de clasificaciones, abordaremos en primer lugar el comportamiento del producto en la pantalla y posteriormente el comportamiento del teclado, ratón, tableta digitalizadora o cualquier otro dispositivo de entrada de datos semejante.

El usuario pretende ver cuando ejecuta una aplicación *Windows* un comportamiento semejante al de todas las aplicaciones. Por ejemplo, todas las aplicaciones cuando permiten configurar la impresora, realizan llamadas a *Windows* y muestran un cuadro de diálogo común diseñado por el fabricante. Si tenemos una impresora Star ZA-200, esperamos encontrar este cuadro de diálogo:



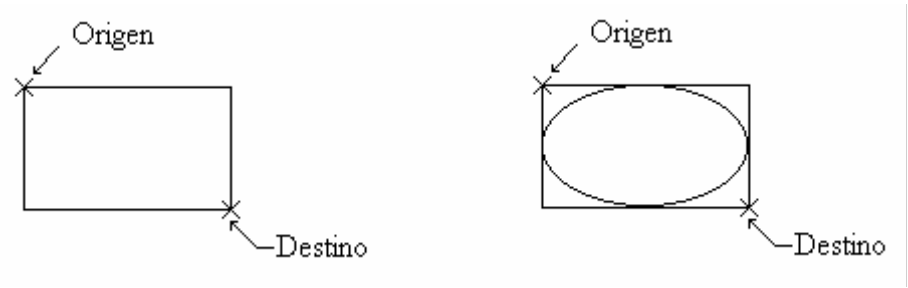
Por ello, la aplicación contendrá cuadros de dialogo para acceder a ficheros (obtener y almacenar), semejantes a el siguiente:



Otro aspecto a tener en cuenta, es que el programa deberá tener zonas de selección, iconos o botones que activen distintos estados de la máquina. Los menús de la aplicación deben seguir el orden que encontramos en todas las aplicaciones y que está plasmado en el documento CUA de IBM.

En cuanto a las entradas realizadas a través de teclado, debemos destacar algunos aspectos, como tener teclas de acceso rápido a los elementos del menú, así como a los elementos dentro de los cuadros de diálogo. Por supuesto deben respetarse los comportamientos estándares dentro de los cuadros de diálogo, como por ejemplo, usar la tecla <Tab > para avanzar campos de texto.

Un punto a destacar es el de dibujo de formas geométricas. Es deseable que el usuario indique el mínimo número de puntos para definir un objeto. Por ejemplo, para pintar un punto, con una pulsación de ratón es suficiente. En cambio, para hacer un rectángulo o una elipse, se precisan solamente dos puntos. Esto podemos observarlo en la siguiente figura:



En cuanto al producto, es deseable que tenga una ayuda al estilo *Windows*, o sea, mediante un hipertexto que permita la fácil comprensión de todos los concepto que atañen al producto. Para realizar dicho hipertexto, existen unas normas de escritura que vamos a comentar seguidamente. Existen tres aspectos que definen un hipertexto. El primero es la audiencia para la que se desea preparar dicho documento; en función del nivel al que se desea llegar con dicha audiencia, se deberá definir el contenido del mismo. El segundo aspecto trata de encontrar la estructura interna del contenido que estamos intentando plasmar. Hay que intentar organizar el contenido de la forma más lógica posible, pensando lo que vamos a decir en cada pantalla del hipertexto. Y el tercer aspecto se refiere a que un documento electrónico debe poseer la propiedad de ser claro en contenido. Frases cortas y bien dichas acompañadas de imágenes dan mejores resultados de comprensión que largos textos explicativos.

En caso de tener una tableta digitalizadora, el programa se comportará del mismo modo que el descrito para el ratón, dado que el funcionamiento es muy similar. Por último debemos decir algo que viene diciendo *Windows* desde hace mucho tiempo, el usuario es y debe ser el primer elemento del sistema, ya que es un ser humano, y por consiguiente el interfaz con el ordenador debe ser amigable, fácil de aprender y que ayude al usuario. Una de las mayores ventajas del ratón en los programas es que

el usuario no debe recordar nombres extraños para realizar tareas que pueden resultar tan simples como pulsar un botón de un ratón.

3.2.2 Interfaces hardware.

Inicialmente, el presente producto puede usar todos los dispositivos que admite *Windows*. Hacer una lista de todo el hardware compatible *Windows* es demasiado extenso y contraproducente. Hay que destacar el concepto de independencia tan bien logrado por MICROSOFT. Problemas tan frecuentes en MSDOS como tener aplicaciones que admiten uno o varios modelos de pantalla EGA, VGA, CGA, etc.. y que en algunos casos los ficheros para representar gráficos dependen del modelo de pantalla, quedan atrás.

3.2.3 Interfaces software.

El producto que estamos describiendo está desarrollado bajo el sistema operativo MSDOS en su versión 6.2, y sobre dicho sistema, se ha usado *Windows* 3.1 como entorno sobre el que se va a ejecutar la aplicación. En última instancia, la aplicación usa las funciones de los interfaces de *Windows* y por tanto, realmente no depende de MSDOS. De hecho, existen versiones superiores de *Windows* que son completos sistemas operativos, como *Windows NT* y un producto que está aun en desarrollo denominado Chicago (o *Windows* 95 según sus últimas denominaciones). En principio, el presente producto debe usar las funciones de los tres interfaces que presenta *Windows* 3.1. Dado que el API de *Windows* 3.0 posee cerca de 600 funciones y en su versión 3.1 llegan a las 1000, haré una pequeña descripción de cada uno de los interfaces a nivel global. El API de *Windows* posee tres grupos de funciones: funciones de gestión del interfaz de *Windows*, funciones referentes al interfaz del dispositivo gráfico (GDI) y funciones de servicios del sistema.

Funciones de gestión del interfaz *Windows*.

Estas funciones gestionan el procesamiento de los mensajes, la creación de ventanas, su movimiento y alteración, creando el sistema de salidas. A continuación se muestra una tabla describiendo este tipo de funciones.

Tipo de funciones API	Propósito
Mensaje	Leen y procesan los mensajes de *Windows* de la cola de la aplicación.
Creación de ventana	Crean, destruyen, alteran y devuelven información sobre ventanas.
Mover y Mostrar	Muestran, ocultan, mueven y devuelven información en pantalla viendo el número y las posiciones de las ventanas.
Entrada	Controlan los sistemas de dispositivo, deshabilitan entradas de los dispositivos del sistema y especifican la acción especial de *Windows* cuando una aplicación recibe entradas de estos.
Hardware	Alteran y preguntan por el estado de los dispositivos de entrada (p.e. ratón y teclado)
Pintado	Preparan una ventana para pintar y ofrecen funciones gráficas de propósito general.
Cajas de dialogo	Crean, cambian, comprueban y retiran cajas de diálogo y los controles dentro de ellas.
Desplazamiento	Controlan el desplazamiento de una ventana y de las barras de desplazamiento.
Menú	Crean, alteran y borran menús.
Información	Devuelven información sobre el número y la localización de ventanas en la pantalla.
Sistema	Devuelven información sobre la métrica, el color y la hora del sistema.

Portapapeles	Cambian datos entre aplicaciones de *Windows* y el portapapeles.
Error	Muestran en pantalla errores del sistema.
Cursor de texto	Gestionan la marca del cursor de textos.
Cursor del ratón	Fija, mueven, ocultan, muestran y restringen los movimientos del cursor.
Funciones Hook	Manejan los ***hooks*** del sistema, que son recursos compartidos que instalan un tipo específico de función filtro. Una ***función filtro*** es una función "callback" proporcionada por la aplicación, especificada por la función *SetWindows* , que procesa eventos antes de que alcance el lazo de mensajes de la aplicación.
Propiedades	Crean y acceden a la lista de propiedades de la ventana. Una ***lista de propiedades*** es un área de almacenamiento que contiene handles para los datos que la aplicación desea asociar a una ventana.
Rectángulo	Fija y pide información sobre rectángulos en el área de trabajo de una ventana.

Funciones de interfaz del dispositivo gráfico (GDI)

Estas funciones llevan a cabo operaciones gráficas independientes del dispositivo en las aplicaciones *Windows*.

Tipo de funciones API	Propósito
Contexto de dispositivo	Crean, borran y restauran contextos de dispositivo (que enlazan la aplicación *Windows*, un gestor de dispositivo y un dispositivo de salida).
Herramientas de dibujo	Crean y borran las herramientas de dibujo que usan el GDI cuando genera salidas de un dispositivo.
Paleta de colores	Proporciona un método independiente del dispositivo para acceder a los colores de un dispositivo que se muestra en pantalla.
Atributos de dibujo	Afectan a la apariencia de la imagen de salida de una aplicación *Windows* (que tiene cuatro formatos: brocha, línea, bitmaps y texto).
Mapeado	Fijan e indagan información del modo de mapeado GDI.
Coordenadas	Hacen la conversión entre las coordenadas de la pantalla y del usuario y determinan la localización de un punto individual.
Región	Crean, modifican y obtienen información sobre regiones. Una ***región*** es un área elíptica o poligonal, dentro de una ventana, que puede ser proporcionada como salida gráfica.
Recorte	Crean, cambian y comprueban las regiones de recorte.
Salida de línea	Crean salidas de línea simples y complejas usando una herramienta de dibujo.
Elipse y polígono	Dibujan elipses y polígonos.
Bitmap	Muestran bitmaps en pantalla.
Texto	Escriben texto en la superfice de un dispositivo de exposición, obtienen información del texto, cambian el alineamiento del texto y modifican la justificación del texto.
Fuentes	Seleccionan, crean, borran y obtienen información sobre fuentes.
Metafile	Crean, copian, cierran, borran, recuperan y obtienen información sobre Metafiles.
Control de impresora.	Obtienen las capacidades de la impresora y alteran su estado de inicialización.
Funciones de escape de la impresora	Permiten a una aplicación acceder a las facilidades de un dispositivo específico no accesibles directamente a través del GDI.

Entorno	Fija e indaga información relativa al entorno del dispositivo de salida.

Funciones del interfaz de servicios del sistema

Estas funciones reservan memoria local y global, gestionan las tareas, cargan los recursos de los programas, manipulan cadenas, cambian los archivos de inicialización de *Windows*, ofrecen ayudas para la depuración (debugging), realizan entradas y salidas en archivos y dan acceso a puertos de comunicaciones, así como crean los sonidos.

Tipo de funciones API	Propósito
Manejo de módulos	Fijan e indagan información respecto a módulos de *Windows*.
Manejo de memoria	Gestionan la memoria local y global del sistema.
Segmento	Reservan, liberan, bloquean y desbloquean segmentos de memoria.
Interrupciones del sistema operativo	Permiten a programas en lenguaje ensamblador, realizar llamadas a ciertas interrupciones DOS y NETBIOS sin codificarlas directamente.
Tareas	Cambian los estados de ejecución de una tarea, recuperan información relacionada con la tarea y obtienen información sobre el entorno en que se está ejecutando dicha tarea.
Manejo de recursos	Localizan y cargan recursos de aplicación de un archivo ejecutable *Windows*.
Manipulación de cadenas	Ofrecen funciones de manipulación de cadenas usadas con frecuencia, tales como copiar cadenas, comparar y convertir en mayúsculas.
Manejo de átomos	Crean y manejan átomos (un ***átomo*** es un índice entero distintivo para una cadena).
Archivo de inicialización	Fija e indaga información sobre los archivos de inicialización *Windows* (por ejemplo de WIN.INI).
Comunicaciones	Realizan comunicaciones en serie y paralelo a través de los puertos del sistema.
Sonido	Crea sonidos y música para ser ejecutados por el generador de sonidos del sistema.
Macros de utilidad	Obtiene los contenidos de bytes, palabras y enteros largos (LONG).
Archivos de entrada	Crean, abren, cierran, leen y escriben en archivos.
Depurado	Ayudan a capturar errores en los módulos y aplicaciones *Windows*.
Herramientas de optimización	Gestionan como interactúan las herramientas de desarrollo de software *Windows Profiler* y *Swap* con una aplicación que esta siendo desarrollada.
Ejecución de aplicaciones	Permiten a una aplicación *Windows* ejecutar otro programa *Windows*.

Windows posee tres librerías de enlace dinámico denominadas USER.EXE, KERNEL.EXE y GDI.EXE que contienen a las funciones antes descritas. Dichas librerías poseen la característica de ser comunes a todas las aplicaciones que existen instaladas en el sistema, produciéndose el enlace con la aplicación en tiempo de ejecución, de ahí que se denominen *dinámicas.*

3.2.4 Interfaces de comunicaciones.

Este producto no posee ningún requisito para ejecutarse en una red. El único requisito en la instalación del producto en modo multiusuario, es que cada usuario tenga su propio fichero de configuración. O sea, se deberá poner el producto en un subdirectorio compartido del servidor y un subdirectorio local para cada usuario con

un fichero de configuración propio para mantener el estado en que deja cada usuario el producto (por ejemplo el tipo de modo de ejecución en que esta trabajando).

En cuanto a la comunicación del presente producto con otras aplicaciones, se deberá incluir la transferencia del dibujo como un mapa de bits al portapapeles. En versiones de *Windows* como el *Windows TG* y superiores existe la posibilidad de copiar el contenido del portapapeles al portafolio y compartir este mapa de bits sin tener que crear un fichero BMP con *PaintBrush*.

3.3. Requerimientos de eficiencia.

El producto en principio está pensado para un único usuario bajo *Windows*, aunque deberán poderse ejecutar varias instancias del mismo sobre un terminal. Por tanto puede ser considerado como multiusuario. Si lo desea puede instalarse una sola vez en un servidor de ficheros, teniendo cada usuario su subdirectorio de trabajo con el fichero de configuración de la aplicación. De este modo, todos los terminales *Windows* tienen acceso a una única copia del producto en disco. El número de usuarios simultáneos conectados no depende del producto, sino del sistema, y en principio puede ser cualquiera.

En el parámetro **Files** del fichero *Config.sys* de *MSDOS* se debe indicar el mínimo número de instancias que desea poder ejecutar simultáneamente. Este número seguramente no deberá modificarlo ya que *Windows* lo corrige en su instalación. El número máximo de primitivas gráficas que pueden se dibujadas está restringido a 4.294.967.295.

Dado que el presente producto esta desarrollado en *Borland C++* y que en dicho entorno existe una herramienta denominada *Turbo Profiler*, la cual sirve para analizar el rendimiento de un programa, deseamos que se analice con una carga inicial aproximada de 500 primitivas la aplicación. Esta utilidad puede mostrar exactamente donde se emplea el tiempo del programa, tanto el número de veces que se ejecuta una instrucción como la cantidad de tiempo empleado en la ejecución de esta. Una característica de *Windows* es que es un sistema dirigido por eventos; esto significa que todas las aplicaciones *Windows* poseen un bucle en el programa principal testeando la cola de mensajes de la aplicación. Evidentemente, cuanto más tiempo se esté ejecutando el producto en dicha tarea (dando vueltas testeando la cola), mejor para el kernel de *Windows*. Este proceso puede usarse para decidir que equipo debe usar un usuario en función de la carga media que espera usar, y analizar el comportamiento con la misma.

Un ejemplo de comportamiento del producto bajo la observación del *Turbo Profiler* puede ser:

```
Turbo Profiler for Windows Version 2.1  Thu Oct 13 09:37:52 1994
Program: D:\PROYECT\DEBUGED\PAINT.EXE
Execution Profile
Total time: 133.59 sec
% of total: 19 %
       Run: 1 of 1
    Filter: All
    Show: Time
     Sort: Frequency
WINMAIN             115.26 sec    86%    |**************
TApplication::Id    8.2790 sec    6%     | ***
TApplication::Pr    5.5805 sec    4%     | **
TPaintApp::InitM    1.8874 sec    1%     |
TApplication::Pr    1.3947 sec    1%     |
TApplication::Pr    1.1008 sec    1%     |
----------------    -------       ----
        |                    |            |
Nombre              nº líneas             Tiempo
de la llamada.      analizadas.   consumido en dicha línea.
```

3.4. Obligaciones de diseño.

3.4.1. Estándares cumplidos.

El producto debe intentar cumplir la norma CUA, para facilitar el aprendizaje a los usuarios.

En cuanto a la nominación de nombres, se deberá intentar seguir la *notación húngara* de prefijos normalizados. Esta notación no es una obligación, aunque si una recomendación. En esta notación, un nombre posee dos partes: un prefijo y la parte explícita de su cometido, pues bien, como norma, en la parte donde se expresa el cometido de dicho nombre, estará formado por palabra unidas sin nexos, que deberán comenzar con la primera letra de cada palabra en mayúsculas.

En cuanto a la norma principal que debe cumplir el producto, es usar el API de *Windows* plenamente y siguiendo sus reglas para poder decir que realmente se comporta como un programa *Windows*.

3.4.2. Limitaciones hardware.

Las limitaciones hardware del producto son las limitaciones que posee *Windows*. Dado que MICROSOFT nos asegura que el sistema *Windows* estará disponible (con el paso del tiempo) en todos los ordenadores y funcionará en todos los dispositivos existentes, las limitaciones hardware se reducen bastante. En principio, la independencia del producto respecto a los dispositivos está solucionada con *Windows* del mejor modo posible existente, ya que los fabricantes proporcionan a *Windows* sus controladores de dispositivos (*drivers*), reduciendo el tiempo de programación a miles de programadores que ya tienen suficientes problemas como para preocuparse de como usar esa nueva pantalla 3D que ha sacado el fabricante X y que hace maravillas. Además, mejor que el fabricante nadie hará dicho driver. Desde luego, *Windows* hace fácil el uso y la programación de las aplicaciones; voy a poner un pequeño ejemplo de ello: el producto en cuestión permite imprimir los dibujos realizados en una impresora matricial, en una de inyección a color, en una láser, en un trazador y hasta emitir vía fax el dibujo, sin tener desarrollada ninguna rutina especial para cada dispositivo y con la máxima resolución que admite dicho dispositivo. Existen algunos dispositivos que todavía no poseen drivers bajo *Windows*, pero con el tiempo, seguro que salen a la luz. Los movimientos de las grandes multinacionales nos indican que el futuro va a ser de *Windows* y OS/2, por ejemplo, podemos ver como el AS/400 de IBM permite tener terminales *Windows* accediendo de modo transparente al servidor.

Una de las limitaciones hardware que debe existir en el producto es la resolución que usará en los dispositivos conectados. Se debe intentar usar la máxima que admita *Windows*.

Otra limitación es la cantidad de memoria que admite el producto, ya que existe un límite en el número de primitivas gráficas (el cual es de 4.294.967.295). Creemos que nadie llegará al mismo considerando que estas deben ser dibujadas a mano sobre la pantalla.

El único requisito real del sistema es que necesita de un hardware potente para soportar grandes soluciones, pero esto en vez de verse como un problema, se puede ver como una virtud.

3.5. Atributos.

3.5.1 Seguridad.

Dado que el producto debe funcionar bajo *Windows*, es importante tener algunos conceptos claros respecto al mismo. Cuando *Windows* detecta una violación de alguna norma por parte de alguna aplicación, normalmente detecta dicha anomalía y lanza un aviso al usuario indicando el tipo de error cometido y procediendo a la descarga y el restablecimiento del sistema. Normalmente, este tipo de problemas por parte de las aplicaciones ocurren cuando desean usar recursos de sistema *Windows* sin contar con la aprobación del mismo. Este concepto se puede observar frecuentemente cuando ejecutamos aplicaciones *DOS* desde *Windows*. Cuando una aplicación *DOS* necesita gran cantidad de memoria, usa un gestor de memoria virtual propio que permite el acceso a la memoria extendida como si fuera memoria convencional, por ejemplo BORLAND posee la tecnología VROOMM (Virtual Run-Time Object-Oriented Memory Manager).

El producto que estamos describiendo, debe comprobar los valores de retorno de las llamadas que realicen al sistema. Muchas de las funciones que se pueden llamar del interfaz de programación de aplicaciones (API) de *Windows*, devuelven un valor que indica el éxito o fracaso de la función. Otras funciones que no pueden devolver un único valor, frecuentemente asignan un valor específico de error a una variable denominada *errno.* La aplicación debe tener un gestor de errores que actúe ante situaciones de error avisando al usuario y eliminando la aplicación del sistema en caso necesario.

En cuanto a operaciones restringidas por la lógica, podemos destacar que la aplicación estando en modo *Windows* no debe poder exportar a modo LaTeX en cambio si puede leer metaficheros generados en modo LaTeX. Por supuesto, es una violación leer documentos *Windows* estando en modo LaTeX.

Por último decir que las necesidades del producto en cuanto a copias de seguridad son insignificantes y pueden ser solucionadas con el proceso de copias de seguridad que proporciona MSDOS en cualquiera de sus versiones.

3.5.2 Facilidades de Mantenimiento.

En el punto 3.6.1. denominado bases de datos, se explica como deben poder hacer el mantenimientos los usuarios LaTeX del producto, por tanto, remito los comentarios a dicho apartado.

En cuanto a instalaciones y reinstalaciones del producto en otras máquinas también se ha hablado del tema al hablar del concepto de aplicación independiente en el punto 2.1.

Un punto a considerar en esta apartado es como conseguir que el producto entre en otras regiones de distinto idioma sin necesitar que sea traducido por el creador del programa. Esto esta bien conseguido en *Windows* gracias al concepto de recurso. Los recursos son datos (constantes) que están incluidos en el archivo .EXE del programa, pero que no residen en el segmento de datos normal del mismo. Al tener todas las constantes definidas en una zona única y accesible fácilmente con herramientas que nos permiten la edición de las mismas (BORLAND posee una herramienta fabulosa denominada *Resource Workshop*), se puede hacer una traducción en cualquier parte del mundo. Como consecuencia de lo dicho, para permitir esta característica, todas las constantes que tengan una implicación directa en la traducción deberán estar definidas como recursos.

3.6. Otros requerimientos.

3.6.1. Bases de datos.

El producto no utiliza ningún gestor de bases de datos, ya que en principio no esta pensado para ello. Debe poseer un fichero de configuración de la aplicación, en el que se guardará información referente al modo de funcionamiento seleccionado y a opciones de exportación del formato LaTeX.

Dado que el producto deseamos que sea WYSIWYG, el usuario debe poder configurar las fuentes LaTeX como desee. Para ello, en el fichero de configuración se almacenará la información necesaria para realizar la correspondencia entre las fuentes LaTeX y las fuentes *Windows*.

Para realizar tal correspondencia de fuentes LaTeX con fuentes *Windows*, definiremos unas palabras reservadas para los nombres de las fuentes en LaTeX. Estas palabras dependen del idioma del usuario (al igual que para identificar los tamaños que el usuario desea aplicar). A estos nombres de fuentes, les asociamos todos los tamaños que admite LaTeX en dicha fuente.

La expresión que identifica una fuente *Windows* en el fichero de configuración, deberá tener todas las características que este admite:

- Alto de un carácter.

- Ancho de un carácter.
- Ángulo de rotación de la cadena.
- Ángulo de rotación de cada carácter.
- Intensidad de negrita.
- Itálica.
- Subrayado.
- Tachado de letras.
- Juego de caracteres del tipo de letra.
- Precisión deseada de salida.
- Como cortar los caracteres que quedan fuera de la región de recorte.
- Calidad.
- Familia de anchuras.
- Nombre identificador del tipo de fuente (imprenta).

Este fichero debe leerse únicamente en el periodo de inicialización de la aplicación. Además del formato de metafichero de *Windows*, el producto debe soportar otro tipo de metaficheros que son propios para el producto. Este tipo de metaficheros debe poseer la propiedad de poder ser editado por cualquier editor de textos. Además, la información contenida en el mismo, debe poder leerse perfectamente por cualquier usuario. De este modo se consiguen grandes propiedades con pequeño esfuerzo.

3.6.2. Operaciones.

Dado que el programa solo presenta una pantalla de edición se desea que puedan existir varias instancias del mismo en memoria. Como sabemos, cuando *Windows* carga por segunda vez un mismo programa, lo que en realidad hace es compartir el código de los programas cargados, así como los datos estáticos (recursos). Consecuencia de ello, se deduce que cargar varias veces en memoria el presente producto aumentará la productividad del usuario sin ser un costo de memoria.

Deben existir dos modos de funcionamiento en el producto: el modo *Windows* y el modo $LaTeX$. Esto es debido a que desde el modo *Windows* podemos hacer exportaciones a metaficheros con formato *Windows*, el propio de la aplicación y a mapa de bits. En cambio la exportación a formato $LaTeX$ precisa de restricciones importantes que deben ser cumplidas para conseguir un producto WYSIWYG.

3.6.3. Requerimientos de adaptación a situaciones.

Un objetivo que debe cumplir el programa es ser independiente del subdirectorio en que se instala. Durante el proceso de arranque, el programa debe de analizar en que subdirectorio se encuentra para saber donde se halla el fichero de configuración.

Windows proporciona una propiedad importante. Supongamos que compramos una impresora láser y la conectamos al sistema. Automáticamente y sin tener que cambiar absolutamente nada en la configuración del programa, el producto usará dicha impresora usando la máxima resolución de la misma. Por ello, un requisito importante es que el programa funcione internamente en la más alta resolución posible. De este modo, cuando *Windows* imprime los objetos sobre un dispositivo (sea cual fuere, pantalla en modo VGA, impresora matricial o láser, etc..) no perderá resolución, funcionado a la mas alta que el dispositivo admita. A continuación se expresa una tabla en la que podemos comparar las resoluciones de algunos dispositivos.

Dispositivo físico	**Resolución en milímetros**
VGA	0.3250
LaserJet	0.0846
Plotter de Plumas	0.0254
Imprenta	0.0106
MM_HIMETRIC	0.0100

En *Windows*, se nos proporciona un entorno que traduce los requisitos gráficos de la aplicación al sistema en el que se halla. Las aplicaciones *Windows* pueden moverse desde gráficos monocromos (Hércules) a EGA o SVGA con mínimas adaptaciones. Además el interfaz de dispositivos gráficos de *Windows* (GDI) va más allá del soporte de entorno de video para incluir el soporte de dispositivos de salida impresa (las mismas primitivas que se usan sobre el dispositivo de video, se usan sobre un trazador o una impresora laser).

Así, bajo *Windows*, las aplicaciones operan en un único entorno virtual independiente del hardware del sistema real mientras *Windows* mapea la pantalla virtual de la aplicación sobre la pantalla real. De este modo, una aplicación que se ejecuta con 256 colores en SVGA se hace corresponder con otra de 16 colores para EGA o VGA y en niveles de gris para un sistema monocromo.

Por supuesto, hay límites para el soporte de *Windows* y existen dispositivos que por su novedad, no poseen drivers disponibles. A pesar de esto, según aparecen nuevos dispositivos de video e impresión, los fabricantes de hardware proporcionan rápidamente sus propios drivers, con lo cual, el programador de aplicaciones no tiene que preocuparse de programar dicho interfaz y además, es seguro que el suministrado por el fabricante para *Windows* funciona correctamente.

Otro punto a tener en cuenta a la hora de instalar el programa por usuarios $\LaTeX$, son las fuentes de que dispone el usuario en su equipo. El producto debe tener un sistema en el cual sea fácil incluirle nuevas fuentes y tamaños de las mismas. Esta característica se obtendrá al poder usar un editor de texto para modificar el fichero de configuración de la aplicación.