(19) OFICINA ESPAÑOLA DE PATENTES Y MARCAS

ESPAÑA

(11) Número de publicación: **2 634 676**

(51) Int. CI.:

G06F 9/44 (2006.01)
G06F 3/0481 (2013.01)

(12) TRADUCCIÓN DE PATENTE EUROPEA T3

(86) Fecha de presentación y número de la solicitud internacional: **25.08.2005 PCT/US2005/030314**

(87) Fecha y número de publicación internacional: **02.11.2006 WO06115531**

(96) Fecha de presentación y número de la solicitud europea: **25.08.2005 E 05789924 (7)**

(97) Fecha y número de publicación de la concesión europea: **26.04.2017 EP 1872194**

(54) Título: **Interfaz y sistema para manipular miniaturas de ventanas en directo en un gestor de ventanas**

(30) Prioridad:

22.04.2005 US 111983

(45) Fecha de publicación y mención en BOPI de la traducción de la patente:
28.09.2017

(73) Titular/es:

MICROSOFT TECHNOLOGY LICENSING, LLC (100.0%)
One Microsoft Way
Redmond, WA 98052, US

(72) Inventor/es:

SCHECHTER, GREG;
SAKS, JEVAN y
FORTIER, CHRIS

(74) Agente/Representante:

CARVAJAL Y URQUIJO, Isabel

DESCRIPCIÓN

Interfaz y sistema para manipular miniaturas de ventanas en directo en un gestor de ventanas

Campo de la invención

Los aspectos de la presente invención se refieren a miniaturas de ventanas y, en particular a una interfaz y un sistema de manipulación de miniaturas en un gestor de ventanas.

Antecedentes de la invención

Los sistemas informáticos visualizan una diversidad de ventanas en las que los usuarios pueden introducir datos, manipular datos o activar procedimientos. Si se ejecuta más de una aplicación simultáneamente, pueden visualizarse varias ventanas en la pantalla del ordenador, correspondiendo cada ventana a una aplicación. Puede haber también múltiples ventanas para una aplicación. Por ejemplo, si un usuario está introduciendo texto en un programa de procesamiento de textos mientras también está trabajando en un programa de hoja de cálculo, puede haber dos ventanas abiertas en la pantalla. Una de las ventanas es la ventana de la aplicación de procesamiento de textos y la segunda ventana es la ventana del programa de hoja de cálculo. Si el usuario está viendo adicionalmente un vídeo en una aplicación de reproductor multimedia, habrá una ventana adicional que corresponde a la aplicación del reproductor multimedia. A medida que el número de aplicaciones activas aumenta, lo hace el número de ventanas que se visualizan en la pantalla del ordenador.

Un usuario que usa múltiples aplicaciones simultáneamente a menudo se enfrenta con una multitud de ventanas en la pantalla que provocan un escritorio desordenado que conduce a confusión y frustración del desorden observado en la pantalla. Por ejemplo, puede haber tantas ventanas solapándose en la pantalla que un usuario pueda tener que desperdiciar tiempo en encontrar una ventana deseada cada vez que el usuario desea completar una tarea en una aplicación diferente. Para aliviar este problema el usuario puede minimizar ventanas o abandonar la aplicación correspondiente. Si necesita realizarse una acción en una aplicación particular en la que se ha minimizado la ventana, el usuario debe localizar en primer lugar la ventana deseada y abrir la ventana deseada después de localizar la ventana. Este proceso lleva mucho tiempo. Análogamente, si el usuario abandona la aplicación completamente para liberar el desorden del escritorio, entonces la aplicación ya no está activa. En este caso, el usuario debe volver a lanzar la aplicación para completar la tarea deseada y desperdiciar incluso más tiempo. También, si la aplicación realiza una función en curso, esa función se pierde durante el tiempo que la aplicación no estaba activa.

Típicamente, cuando se minimiza una ventana de aplicación, puede visualizarse un botón de barra de tareas en una barra de tareas para indicar que la aplicación está activa. Aunque el botón de la barra de tareas puede indicar que la aplicación está activa, a menudo hay únicamente un icono o un nombre del botón de la aplicación en la barra de tareas. Sin información adicional, el usuario tendría que abrir la ventana para ver los contenidos de la ventana. Sin embargo, abrir y cerrar la ventana simplemente para comprobar los contenidos de la ventana (por ejemplo, para determinar la identidad de la ventana), el usuario desperdicia tiempo y esfuerzo que produce una pérdida de eficacia y productividad. Este problema se agrava cuando el número de aplicaciones activas y el número de ventanas aumenta a medida que el usuario debe abrir y cerrar muchas aplicaciones para hallar la ventana deseada.

En el documento US 2002/0170058 A1, se desvela un método del procesamiento visual de ficheros de imágenes en un editor de imágenes, en el que se produce una pluralidad de ficheros de miniaturas de la imagen en dicho editor a partir del contenido de los ficheros de imagen y se visualizan y actualizan en un menú desplegable. Sin embargo, no es evidente a partir de dicha publicación cómo aplicar esta materia objeto a la funcionalidad de un gestor de ventanas. También, no proporciona ninguna solución para el caso en el que el usuario únicamente esté interesado en ser notificado acerca de cambios de únicamente una región de una ventana.

En el documento US 6606105 B1, se desvelan técnicas para visualizar miniaturas de diferentes capas de un documento que se está editando. Cada miniatura se visualiza en una jerarquía que corresponde a la estructura de dicho documento. Sin embargo, no se desvela medio para mantener a un usuario actualizado sobre cambios en una ventana arbitraria.

El documento US 2001/0028368 A1 se refiere a un “navegador instantáneo” para visualizar miniaturas que muestran contenido de ficheros que se han abierto recientemente o están abiertos en la actualizad. Las mismas desventajas que las indicadas anteriormente para las publicaciones anteriores pueden observarse con esta publicación.

Además, el documento WO 03/025899 A1 se refiere a los denominados parches inteligentes, que corresponden a comunicaciones activas para usuarios remotos y que se actualizan de acuerdo con el estado de la comunicación respectiva. Esta publicación no supera las desventajas observadas anteriormente para cualquiera de las publicaciones anteriormente analizadas.

El documento US 5333256 contiene materia objeto relacionada con visualizar iconos que corresponden a programas de aplicación. Sin embargo, dichos iconos no reflejan el contenido visible de la respectiva aplicación sino que únicamente son representaciones abstractas icónicas de las aplicaciones. No se proporcionan medios realmente para informar a un usuario sobre cambios de región en una ventana de aplicación.

En el documento US 2003/0229894 A1, se proporciona una técnica que soporta la búsqueda de una escena

deseada, tal como una imagen, mientras se visualizan imágenes en movimiento, tal como en un flujo de vídeo. De nuevo, no es evidente cómo esta publicación podría ayudar a notificar a un usuario sobre cambios que tienen lugar en una región de una ventana de aplicación.

Por lo tanto, existe una necesidad en la técnica de un sistema y método que mantengan a un usuario actualizado sobre cambios en una región de interés en una ventana de aplicación, independientemente de si dicha ventana de aplicación es actualmente visible o no, tal como cuando se minimiza u oculta detrás de otras ventanas de aplicación. En otras palabras, un usuario puede a menudo estar interesado en cambios que tienen lugar únicamente en una región particular de una ventana, pero puede no estar interesado en cambios fuera de esa región en la misma ventana.

Sumario de la invención

Los aspectos de la presente invención proporcionan una interfaz, sistema o método para visualizar una ventana en una pantalla tratando de esta manera uno o más problemas analizados anteriormente. También se proporciona una miniatura dinámica que corresponde a una ventana de aplicación en la que se reflejan modificaciones de contenido en una ventana de aplicación en la miniatura dinámica correspondiente. Una modificación de contenido de la ventana de aplicación puede reflejarse adicionalmente en la miniatura dinámica correspondiente en tiempo real.

Breve descripción de los dibujos

La Figura 1A ilustra un ejemplo de un sistema para implementar la invención que incluye un dispositivo informático de fin general en forma de un ordenador.

Las Figuras 1B a 1M muestran un entorno informático de fin general que soporta uno o más aspectos de la presente invención.

La Figura 2A es un gráfico de escena o diagrama de árbol de visualización que ilustra un ejemplo de componentes de una visualización.

La Figura 2B demuestra componentes de una visualización de muestra que corresponden a la Figura 2A.

La Figura 3A es un gráfico de escena o diagrama de árbol de visualización que ilustra un ejemplo de componentes de una visualización de la presente invención.

La Figura 3B demuestra la visualización que corresponde a la Figura 3A.

Las Figuras 4A y 4B demuestran un ejemplo de una aplicación de la presente invención al proporcionar miniaturas de ventanas de aplicación.

La Figura 5 ilustra otra aplicación de una miniatura de la presente invención en la que se visualizan miniaturas en un menú ALT-TAB.

La Figura 6 es un diagrama de flujo que ilustra un ejemplo de un proceso para proporcionar miniaturas

Descripción detallada

Este documento se divide en secciones para ayudar al lector. Estas secciones incluyen: Vista general; Entorno informático de fin general; Miniaturas.

Se observa que se exponen diversas conexiones entre los elementos en la siguiente descripción. Se observa que estas conexiones en general y, a menos que se especifique de otra manera, pueden ser directas o indirectas y que esta memoria descriptiva no se pretende que esté limitada en este aspecto.

Vista general

A. Miniaturas estáticas

En un enfoque, una miniatura puede estar asociada con la aplicación. Con una miniatura, puede visualizarse una versión pequeña de la ventana de aplicación para indicar el contenido original de la ventana de aplicación, permitiendo al usuario identificar contenido aproximado de la ventana. Sin embargo, estas miniaturas pueden no proporcionar contenido actualizado de la ventana de aplicación. Por lo tanto, si se realizaran modificaciones a la ventana de aplicación, la miniatura puede quedar desactualizada y puede no proporcionar la imagen apropiada del

contenido real de la ventana de aplicación. Por lo tanto, es difícil que el usuario identifique apropiadamente la ventana de aplicación deseada sin tener que abrir la misma ventana de aplicación.

También, las miniaturas carecen de flexibilidad en términos de visualización de contenido, colocación, redimensionamiento y/o efectos. Una miniatura de una ventana de aplicación proporciona una imagen estática del contenido original de la ventana de aplicación. Sin embargo, no únicamente el contenido permanece estático y potencialmente desactualizado, sino también las características y colocación de la miniatura pueden estas predeterminadas y pueden no variar basándose en las necesidades del usuario en cualquier momento dado.

Los sistemas operativos informáticos emplean una capa de software responsable de gestionar los objetos de interfaz de usuario tales como miniaturas, y proporcionar servicios de interfaz de usuario a aplicaciones de software. La capa de software puede denominarse como el Gestor de Ventanas de Escritorio (DWM). La lógica de representación visual, encaminamiento de eventos de entrada y las interfaces de programación de aplicación (API) del Gestor de Ventanas de Escritorio (DWM) incorporan de manera colectiva la política de interfaz de usuario, que a su vez define la experiencia global del usuario del sistema operativo. Al representar visualmente el escritorio, el DWM típicamente procesa actualizaciones de pantalla, tales como el movimiento de las ventanas, coordinando la pintura directamente en la pantalla. Por lo tanto, si una ventana solapante se cierra o se aleja, debe volverse a pintar la ventana o escritorio subyacente recién visualizado. Esto es un proceso lento en el que se desperdician innecesariamente tiempo y recursos. También, cuando se pintan ventanas directamente en una posición particular y en una configuración particular en la pantalla como se hace típicamente, la imagen correspondiente de la ventana se establece en esa posición y configuración particulares. Las modificaciones a la posición o configuración de la ventana requerirían un gran gasto de recursos para volver a pintar la imagen.

B. Miniaturas dinámicas

En otro aspecto de la presente invención, se proporciona una interfaz, sistema y método para visualizar una miniatura en una pantalla en la que un Gestor de Ventanas de Escritorio (DWM) proporciona una Interfaz de Programación de Aplicación (API) que registra una ventana de origen y una ventana de destino. En este ejemplo, la ventana de destino es una ventana en la que puede representarse visualmente una miniatura. La miniatura representada en la ventana de destino puede representar al menos una porción de una ventana de origen correspondiente. Por lo tanto, la miniatura puede representar una asociación entre la ventana de origen y la ventana de destino proporcionando al menos una porción de la ventana de origen dibujada en al menos una porción de la ventana de destino. El DWM puede recibir un manejador de ventana (HWND) que corresponde a la ventana de origen y un HWND que corresponde a la ventana de destino. La aplicación puede recibir adicionalmente un manejador que representa el registro de la miniatura que corresponde a la ventana de origen.

En otro aspecto de la presente invención, se proporciona una interfaz, sistema y método para visualizar una miniatura en una pantalla en la que un Gestor de Ventanas de Escritorio (DWM) actualiza o modifica una miniatura asociada con una ventana de origen correspondiente. En este ejemplo, el DWM puede recibir datos que definen un tamaño de una ventana de destino, una región de una ventana de origen para dibujar en una ventana de destino como una miniatura, opacidad de una miniatura, visibilidad de una miniatura, efectos especiales de una miniatura, etc.

Un aspecto adicional de la presente invención proporciona adicionalmente una miniatura dinámica que comprende exportar una interfaz de programación de aplicación para registrar un manejador de miniatura, que mantiene una lista de registros de miniatura, dibujar una miniatura o ventana de destino basándose en una ventana de origen, en la que la ventana de origen puede ser una ventana de aplicación, actualizar las propiedades de una miniatura, y anular el registro de la miniatura dinámica. Además, la modificación a una ventana de origen puede manejarse automáticamente para efectuar una correspondiente modificación en una ventana de destino.

Se observa que se exponen diversas conexiones entre elementos en la siguiente descripción. Se observa que estas conexiones en general, y a menos que se especifique de otra manera, pueden ser directas o indirectas y que esta memoria descriptiva no se pretende que esté limitada en este aspecto.

Los aspectos de la presente invención proporcionan un sistema y método de interfaz de programación de aplicación (API), para control de la visualización de los elementos de visualización tales como ventanas u otra representación de ventanas tales como miniaturas en una pantalla. En un ejemplo de la presente invención, una versión alternativa de una ventana de aplicación es una versión más pequeña de la ventana de aplicación y se visualiza en la pantalla. La versión más pequeña de la ventana de aplicación puede visualizarse adicionalmente en una localización en la pantalla que es diferente de la ventana de aplicación original. Como alternativa, la ventana de aplicación original puede minimizarse de manera que la ventana de aplicación original no se visualiza mientras la versión más pequeña de la ventana de aplicación se visualiza en la pantalla. También, la versión más pequeña de la ventana de aplicación puede minimizarse ella misma, por ejemplo, si se desea espacio adicional en la pantalla. Aunque la versión alternativa de la ventana de aplicación puede ser una versión más pequeña de la aplicación como se describe, se observa que la versión alternativa de la ventana de aplicación puede ser mayor que la ventana de aplicación original

o de cualquier otro tamaño dependiendo de las necesidades del usuario.

Por ejemplo, la versión alternativa de la ventana de aplicación puede ser una miniatura de la ventana de aplicación. Una miniatura es típicamente una imagen pequeña o representación de un gráfico más grande, o una parte de una imagen más grande, que corresponde a la ventana de aplicación. La miniatura puede representar también una asociación entre una ventana de origen, que proporciona el origen o contenido de la miniatura, y una ventana de destino que puede ser una ventana en la que la miniatura proporciona el contenido desde la ventana de origen. Las miniaturas proporcionan la ventaja de la capacidad de visualización de varias ventanas a la vez en un formato conveniente mientras se minimiza el uso de espacio en la pantalla. También, las miniaturas pueden usarse al seleccionar y clasificar imágenes y ventanas más fácilmente.

En otro aspecto de la presente invención, la miniatura o versión alternativa de la ventana de aplicación puede visualizarse en la pantalla cuando se solicita. Por ejemplo, desplazando un cursor sobre un botón de la barra de tareas puede invocar la pantalla de una miniatura de la ventana de aplicación. En un ejemplo, la miniatura es una miniatura dinámica y puede modificarse a medida que se modifica la ventana de aplicación original. Por ejemplo, los cambios realizados a la ventana de aplicación se reflejan en la miniatura dinámica que corresponde a la ventana de aplicación. Tales cambios pueden reflejarse en la miniatura dinámica en tiempo real, por ejemplo.

En otro aspecto de la presente invención, las miniaturas o miniaturas dinámicas pueden disponerse en otra ventana, o ventana de destino. Por ejemplo, múltiples miniaturas dinámicas de diferentes aplicaciones pueden incluirse en una ventana de destino de manera que la ventana de destino visualiza miniaturas de cada una de las diferentes aplicaciones. También, cualquiera de las miniaturas dinámicas visualizadas en la ventana de destino puede reflejar el estado actual de la correspondiente ventana de aplicación. Como alternativa, una miniatura puede visualizarse en otra miniatura o las miniaturas pueden estar en cascada de manera que una primera miniatura puede visualizarse en una segunda miniatura que, a su vez, puede visualizarse en una tercera miniatura, etc.

En otro aspecto más de la presente invención, se proporciona un sistema y método para preparar ventanas fuera de la pantalla. Componer las ventanas puede entonces conseguirse en pantalla ahorrando por lo tanto recursos informáticos. También, preparar ventanas fuera de la pantalla y componer las ventanas en pantalla posibilita la funcionalidad de miniatura como se describe en la presente invención.

Entorno informático de fin general

La Figura 1A ilustra un ejemplo de un entorno 100 de sistema informático adecuado en el que puede implementarse la invención. El entorno 100 de sistema informático es únicamente un ejemplo de un entorno informático adecuado y no se pretende para sugerir ninguna limitación en cuanto al alcance de uso o funcionalidad de la invención. Ni debería interpretarse que el entorno 100 informático tiene alguna dependencia o requisito relacionado con algún componente o combinación de componentes ilustrados en el entorno 100 de operación ejemplar.

La invención es operacional con numerosos otros entornos de sistema informático o configuraciones de fin general o de fin especial. Ejemplos de sistemas informáticos bien conocidos, entornos y/o configuraciones que pueden ser adecuados para uso con la invención incluyen, pero sin limitación, ordenadores personales, ordenadores de servidor, dispositivos de mano y portátiles, sistemas multiprocesador, sistemas basados en microprocesador, electrónica de consumo programable, PC de red, miniordenadores, ordenadores centrales, entornos informáticos distribuidos que incluyen cualquiera de los sistemas o dispositivos anteriores, y similares.

Con referencia a la Figura 1A, un sistema ilustrativo para implementar la invención incluye un dispositivo informático de fin general en forma de un ordenador 110. Los componentes del ordenador 110 pueden incluir, pero sin limitación, una unidad 120 de procesamiento, una memoria 130 de sistema y un bus 121 de sistema que acopla diversos componentes de sistema que incluyen la memoria de sistema a la unidad 120 de procesamiento. El bus 121 de sistema puede ser cualquiera de varios tipos de estructuras de bus incluyendo un bus de memoria o controlador de memoria, un bus periférico y un bus local que usa cualquiera de una diversidad de arquitecturas de bus. A modo de ejemplo, y no como limitación, tales arquitecturas incluyen el bus de la Arquitectura Estándar de la Industria (ISA), el bus de la Arquitectura Micro Canal (MCA), bus ISA mejorado (EISA), bus local de la Asociación de Normalización en la Electrónica de Vídeo (VESA) y bus de Interconexión de Componentes Periféricos (PCI) también conocido como bus Mezzanine.

El ordenador 110 típicamente incluye una diversidad de medios legibles por ordenador. Medio legible por ordenador incluye tanto medio volátil como no volátil, extraíble, no extraíble. A modo de ejemplo, y no como limitación, medio legible por ordenador puede comprender medio de almacenamiento informático y medio de comunicación e incluye, pero sin limitación, RAM, ROM, EEPROM, memoria flash u otra tecnología de memoria, CD-ROM, discos versátiles digitales (DVD) u otro almacenamiento de disco óptico, casetes magnéticos, cinta magnética, almacenamiento de disco magnético u otros dispositivos de almacenamiento magnético, o cualquier otro medio que pueda usarse para almacenar la información deseada y que pueda accederse mediante el ordenador 110. Medio de comunicación

típicamente incorpora instrucciones legibles por ordenador, estructuras de datos, módulos de programa u otros datos en una señal de datos modulada tal como una onda portadora u otro mecanismo de transporte e incluye cualquier medio de suministro de información. La expresión “señal de datos modulada” significa una señal que tiene una o más de sus características establecidas o cambiadas de tal manera para codificar información en la señal. A modo de ejemplo, y no como limitación, medio de comunicación incluye medio alámbrico tal como una red alámbrica o conexión alámbrica directa, y medio inalámbrico tal como medios acústicos, de RF, infrarrojos y otros. Combinaciones de lo anterior deberían incluirse también dentro del alcance de medio legible por ordenador.

La memoria 130 de sistema incluye medio de almacenamiento informático en forma de memoria volátil y/o no volátil tal como memoria 131 de solo lectura (ROM) y memoria 132 de acceso aleatorio (RAM). Un sistema 133 básico de entrada/salida (BIOS), que contiene las rutinas básicas que ayudan a transferir información entre elementos en el ordenador 110, tal como durante el arranque, se almacena típicamente en la ROM 131. La RAM 132 típicamente contiene datos y/o módulos de programa que son inmediatamente accesibles a y/o que se operan actualmente mediante la unidad 120 de procesamiento. A modo de ejemplo, y no como limitación, la Figura 1 ilustra el sistema 134 operativo, programas 135 de aplicación, otros módulos 136 de programa y datos 137 de programa.

El ordenador 110 puede incluir también otro medio de almacenamiento informático extraíble/no extraíble, volátil/no volátil. A modo de ejemplo únicamente, la Figura 1 ilustra una unidad 140 de disco duro que lee desde o escribe en medio magnético no extraíble, no volátil, una unidad 151 de disco magnético que lee desde o escribe en un disco 152 magnético extraíble no volátil, y una unidad 155 de disco óptico que lee desde o escribe en un disco 156 óptico extraíble no volátil tal como un CD ROM u otro medio óptico. Otro medio de almacenamiento informático extraíble/no extraíble, volátil/no volátil que puede usarse en el entorno de operación ejemplar incluye, pero sin limitación, casetes de cinta magnética, tarjetas de memoria flash, discos versátiles digitales, cinta de vídeo digital, RAM de estado sólido, ROM de estado sólido y similares. La unidad 141 de disco duro está conectada típicamente al bus 121 de sistema a través de una interfaz de memoria no extraíble tal como la interfaz 140, y la unidad 151 de disco magnético y la unidad 155 de disco óptico están típicamente conectadas al bus 121 de sistema mediante una interfaz de memoria extraíble, tal como la interfaz 150.

Las unidades y su medio de almacenamiento informático asociado analizados anteriormente e ilustrados en la Figura 1A, proporcionan almacenamiento de instrucciones legibles por ordenador, estructuras de datos, módulos de programa y otros datos para el ordenador 110. En la Figura 1A, por ejemplo, la unidad 141 de disco duro se ilustra como que almacena el sistema 144 operativo, programas 145 de aplicación, otros módulos 146 de programa y datos 147 de programa. Obsérvese que estos componentes pueden ser los mismos que o diferentes del sistema 134 operativo, programas 135 de aplicación, otros módulos 136 de programa y datos 137 de programa. El sistema 144 operativo, programas 145 de aplicación, otros módulos 146 de programa y datos 147 de programa se les proporciona diferentes números en este punto para ilustrar que, como mínimo, son copias diferentes. Un usuario puede introducir comandos e información en el ordenador 20 a través de dispositivos de entrada tales como un teclado 162 y dispositivo 161 apuntador, denominados comúnmente como un ratón, bola de mando o panel táctil. Otros dispositivos de entrada (no mostrados) pueden incluir un micrófono, palanca de mando, control de juegos, antena parabólica, escáner o similares. Estos y otros dispositivos de entrada a menudo están conectados a la unidad 120 de procesamiento a través de una interfaz 160 de entrada de usuario que está acoplada al bus de sistema, pero pueden conectarse mediante otra interfaz y estructuras de bus, tales como un puerto paralelo, puerto de juegos o un bus serie universal (USB). Un monitor 191 u otro tipo de dispositivo de visualización está también conectado al bus 121 de sistema mediante una interfaz, tal como una interfaz 190 de vídeo. Además del monitor, los ordenadores pueden incluir también otros dispositivos de salida periféricos tales como los altavoces 197 y la impresora 196, que pueden conectarse a través de una interfaz 190 periférica de salida.

El ordenador 110 puede operar en un entorno en red usando conexiones lógicas a uno o más ordenadores remotos, tal como un ordenador 180 remoto. El ordenador 180 remoto puede ser un ordenador personal, un servidor, un encaminador, un PC de red, un dispositivo entre pares u otro nodo de red común, y típicamente incluye muchos o todos los elementos anteriormente descritos con relación al ordenador 110, aunque únicamente se ha ilustrado un dispositivo 181 de almacenamiento de memoria en la Figura 1A. Las conexiones lógicas representadas en la Figura 1A incluyen una red 171 de área local (LAN) y una red 173 de área extensa (WAN), pero pueden incluir también otras redes. Tales entornos de red son comunes en oficinas, redes informáticas en las empresas, intranets e Internet.

Cuando se usa en un entorno de interconexión de red LAN, el ordenador 110 está conectado a la LAN 171 a través de una interfaz o adaptador de red. Cuando se usa en un entorno de interconexión de red WAN, el ordenador 110 típicamente incluye un módem 172 u otro medio para establecer comunicaciones a través de la WAN 173, tal como Internet. El módem 172, que puede ser interno o externo, puede conectarse al bus 121 de sistema mediante la interfaz 160 de entrada de usuario u otro mecanismo apropiado. En un entorno en red, los módulos de programa representados con relación al ordenador 110, o porciones del mismo, pueden almacenarse en el dispositivo de almacenamiento de memoria remoto. A modo de ejemplo, y no como limitación, la Figura 1A ilustra los programas 185 de aplicación remotos como que residen en el dispositivo 181 de memoria. Se apreciará que las conexiones de red mostradas son ejemplares y que pueden usarse otros medios de establecimiento de un enlace de

comunicaciones entre los ordenadores.

Se apreciará que las conexiones de red mostradas son ejemplares y que pueden usarse otros medios de establecimiento de un enlace de comunicaciones entre los ordenadores. Se supone la existencia de diversos protocolos bien conocidos tales como TCP/IP, Ethernet, FTP, HTTP y similares, y el sistema puede operarse en una configuración cliente-servidor para permitir que un usuario recupere páginas web desde un servidor basado en web. Cualquiera de diversos exploradores web convencionales puede usarse para visualizar y manipular datos en páginas web.

Una interfaz de programación (o de manera más sencilla “interfaz”) puede observarse como cualquier mecanismo, proceso, protocolo, para posibilitar que uno o más segmento o segmentos de código comuniquen con o accedan a la funcionalidad proporcionada mediante uno o más otros segmentos de código. Como alternativa, una interfaz de programación puede observarse como uno o más otro mecanismo o mecanismo, método o métodos, llamada o llamadas de función, módulo o módulos, objeto u objetos, etc., de un componente de un sistema que pueda acoplar de manera comunicativa a uno o más otro mecanismo o mecanismos, método o métodos, llamada o llamadas de función, módulo o módulos, etc., de otro componente o componentes. La expresión “segmento de código” en la sentencia anterior se pretende que incluya una o más instrucciones o líneas de código, e incluye, por ejemplo, módulos de código, objetos, subrutinas, funciones y así sucesivamente, independientemente de la terminología aplicada o si los segmentos de código se compilan de manera separada, o si los segmentos de código se proporcionan como código fuente, intermedio u objeto, si los segmentos de código se utilizan en un sistema o proceso de tiempo de ejecución o si están localizados en la misma o diferentes máquinas o distribuidos a través de múltiples máquinas, o si la funcionalidad representada por los segmentos de código se implementa completamente en software, completamente en hardware, o una combinación de hardware y software.

De manera ideal, una interfaz de programación puede observarse de manera genérica, como se muestra en la Figura 1B o en la Figura 1C. La Figura 1B ilustra una interfaz la Interfaz1 como un conducto a través del cual comunican el primer y segundo segmentos de código. La Figura 1C ilustra una interfaz como que comprende los objetos de interfaz I1 y I2 (que pueden o pueden no ser parte del primer y segundo segmentos de código), que posibilita que el primer y segundo segmentos de código de un sistema comuniquen mediante el medio M. En vista de la Figura 1C, se puede considerar los objetos de interfaz I1 y I2 como interfaces separadas del mismo sistema y se puede considerar que los objetos I1 y I2 más el medio M comprenden la interfaz. Aunque las Figuras 1B y 1C muestran flujo e interfaces bidireccionales en cada lado del flujo, ciertas implementaciones pueden tener únicamente información de flujo en una dirección (o ninguna información de flujo como se describe a continuación) o pueden tener únicamente un objeto de interfaz en un lado. A modo de ejemplo, y no como limitación, términos tales como interfaz de programación de aplicación (API), punto de entrada, método, función, subrutina, llamada de procedimiento remoto, e interfaz de modelo de objeto de componente (COM), se abarcan dentro de la definición de interfaz de programación.

Los aspectos de una interfaz de programación de este tipo pueden incluir el método mediante el cual el primer segmento de código transmite información (donde “información” se usa en su sentido más amplio e incluye datos, comandos, solicitudes, etc.) al segundo segmento de código; el método mediante el cual el segundo segmento de código recibe la información; y la estructura, secuencia, sintaxis, organización, esquema temporización y contenido de la información. En este sentido, el mismo medio de transporte subyacente puede ser no importante para la operación de la interfaz, ya sea el medio alámbrico o inalámbrico, o una combinación de ambos, siempre que la información se transporte de la manera definida por la interfaz. En ciertas situaciones, la información puede no pasarse en una o ambas direcciones en el sentido convencional, ya que la transferencia de información puede ser mediante cualquier otro mecanismo (por ejemplo información colocada en una memoria intermedia, fichero, etc., separado del flujo de información entre los segmentos de código) o no existente, como cuando un segmento de código simplemente accede a funcionalidad realizada por un segundo segmento de código. Cualquiera o todos estos aspectos pueden ser importantes en una situación dada, por ejemplo, dependiendo de si los segmentos de código son parte de un sistema en una configuración acoplada de forma holgada o acoplada de forma estrecha, y por esto esta lista debería considerarse ilustrativa y no limitante.

Esta noción de una interfaz de programación es conocida para los expertos en la materia y es evidente a partir de la anterior descripción detallada de la invención. Sin embargo, existen otras maneras para implementar una interfaz de programación, y, a menos que se excluya expresamente, estas se pretende también que se abarquen por las reivindicaciones expuestas en el final de esta memoria descriptiva. Tales otras maneras pueden parecer ser más sofisticadas o complejas que la vista simplista de las Figuras 1B y 1C, pero, sin embargo, realizan una función similar para conseguir el mismo resultado global. Describiremos ahora brevemente algunas implementaciones ilustrativas alternativas de una interfaz de programación.

Factorización

Una comunicación desde un segmento de código a otro puede conseguirse indirectamente descomponiendo la comunicación en múltiples comunicaciones discretas. Esto se representa esquemáticamente en las Figuras 1D y 1E.

Como se muestra, algunas interfaces pueden describirse en términos de conjuntos divisibles de funcionalidad. Por lo tanto, la funcionalidad de la interfaz de las Figuras 1B y 1C puede factorizarse para conseguir el mismo resultado, así como matemáticamente se puede proporcionar 24, o 2 veces 2 veces 3 veces 2. Por consiguiente, como se ilustra en la Figura 1D, la función proporcionada por la interfaz Interfaz1 puede subdividirse para convertir las comunicaciones de la interfaz en múltiples interfaces, Interfaz 1A, Interfaz 1B, Interfaz 1C, etc., mientras se consigue el mismo resultado. Como se ilustra en la Figura 1E, la función proporcionada por la interfaz I1 puede subdividirse en múltiples interfaces I1a, I1b, I1c, etc., mientras se consiga el mismo resultado. De manera similar, la interfaz I2 del segundo segmento de código que recibe información desde el primer segmento de código puede factorizarse en múltiples interfaces I2a, I2b, I2c, etc. Cuando se realiza factorización, el número de interfaces incluidas con el 1er segmento de código no necesita coincidirse con el número de interfaces incluidas con el 2º segmento de código. En cualquiera de los casos de las Figuras 1D y 1E, el espíritu funcional de las interfaces Interfaz1 y I1 permanece siendo el mismo que con las Figuras 1B y 1C, respectivamente. La factorización de las interfaces puede seguir también propiedades asociativas, conmutativas y otras matemáticas de manera que la factorización puede ser difícil de reconocer. Por ejemplo, la ordenación de las operaciones puede no ser importante, y en consecuencia, una función llevada a cabo por una interfaz puede llevarse a cabo bien con antelación de alcanzar la interfaz, por otra pieza de código o interfaz, o realizarse por un componente separado del sistema. Además, un experto en la materia de la programación puede apreciar que existe una diversidad de maneras de realizar diferentes llamadas de función que consiguen el mismo resultado.

Redefinición

En algunos casos, puede ser posible ignorar, añadir o redefinir ciertos aspectos (por ejemplo, parámetros) de una interfaz de programación mientras se consigue aún el resultado pretendido. Esto se ilustra en las Figuras 1F y 1G. Por ejemplo, suponiéndose que la interfaz Interfaz 1 de la Figura 1B incluye una llamada de función Cuadrado (entrada, precisión, salida), una llamada que incluye tres parámetros, entrada, precisión y salida, y que se emite desde el 1er segmento de código al 2º segmento de código. Si la precisión del parámetro medio no es de interés para un escenario dado, como se muestra en la figura 1F, podría asimismo ignorarse o incluso sustituirse por un parámetro sin significado (en esta situación). Se puede añadir también un parámetro adicional sin interés. En cualquier caso, la funcionalidad del cuadrado puede conseguirse, siempre que la salida que se devuelva después de la entrada sea cuadrada por el segundo segmento de código. La precisión puede ser un parámetro muy significativo para alguna porción aguas abajo u otra del sistema informático; sin embargo, una vez que se reconoce que la precisión no es necesaria para el fin limitado de calcular el cuadrado, puede sustituirse o ignorarse. Por ejemplo, en lugar de pasar un valor de precisión válido, podría pasarse un valor sin significado, tal como una fecha de cumpleaños sin afectar adversamente el resultado. De manera similar, como se muestra en la Figura 1G, la interfaz I1 se sustituye por la interfaz I1', redefinida para ignorar o añadir parámetros a la interfaz. La Interfaz I2 puede redefinirse de manera similar como la interfaz I2', redefinida para ignorar parámetros innecesarios, o parámetros que pueden procesarse en otro lugar. El punto aquí es que en algunos casos una interfaz de programación puede incluir aspectos, tales como parámetros, que no son necesarios para algún fin, y de esta manera pueden ignorarse o redefinirse o procesarse en otro lugar para otros fines.

Codificación en línea

Puede ser factible también unir alguna o toda la funcionalidad de dos módulos de código separados de manera que la "interfaz" entre ellos cambie la forma. Por ejemplo, la funcionalidad de las Figuras 1B y 1C puede convertirse a la funcionalidad de las Figuras 1H y 1I, respectivamente. En la Figura 1H, los segmentos de código 1º y 2º anteriores de la Figura 1B se unen en un módulo que contiene ambos de ellos. En este caso, los segmentos de código pueden aún estar en comunicación entre sí pero la interfaz puede adaptarse a una forma que es más adecuada al módulo único. Por lo tanto, por ejemplo, las sentencias de llamada y retorno formales pueden ya no ser necesarias, aunque puede efectuarse aún el procesamiento similar o respuesta o respuestas conforme a la interfaz Interfaz1. De manera similar, se muestra en la Figura 1I, que parte (o toda) la interfaz I2 de la Figura 1C puede escribirse en línea en la interfaz I1 para formar la interfaz I1". Como se ilustra, la interfaz I2 se divide en I2a y I2b, y la porción de interfaz I2a se ha codificado en línea con la interfaz I1 para formar la interfaz I1". Para un ejemplo concreto, considérese que la interfaz I1 de la Figura 1C realiza una llamada de función cuadrado (entrada, salida), que se recibe mediante la interfaz I2, que después de procesar el valor pasado con la entrada (para calcular el cuadrado de una entrada) por el segundo segmento de código, pasa de nuevo el resultado cuadrado con la salida. En un caso de este tipo, el procesamiento realizado por el segundo segmento de código (entrada de cuadrado) puede realizarse mediante el primer segmento de código sin una llamada a la interfaz.

Divorcio

Puede conseguirse una comunicación de un segmento de código a otro indirectamente descomponiendo la comunicación en múltiples comunicaciones discretas. Esto se representa esquemáticamente en las Figuras 1J y 1K. Como se muestra en la Figura 1J, se proporciona una o más pieza o piezas de código (interfaz o interfaces de divorcio, puesto que divorcian las funciones de funcionalidad y/o interfaz de la interfaz original) para convertir las comunicaciones en la primera interfaz, Interfaz 1, para adaptarlas a una interfaz diferente, en este caso las

interfaces Interfaz2A, Interfaz2B e Interfaz2C. Esto puede hacerse, por ejemplo, cuando hay una base de aplicaciones instalada diseñada para comunicar con, es decir, un sistema operativo de acuerdo con un protocolo de la Interfaz1, pero a continuación el sistema operativo se cambia para usar una interfaz diferente, en este caso las interfaces Interfaz2A, Interfaz2B y Interfaz2C. El punto es que la interfaz original usada mediante el 2º segmento de código se cambie de manera que ya no sea compatible con la interfaz usada por el 1er segmento de código, y de esta manera se use un intermediario para hacer a las interfaces, antigua y nueva, compatibles. De manera similar, como se muestra en la Figura 1K, puede introducirse un tercer segmento de código con la interfaz de divorcio DI1 para recibir las comunicaciones desde la interfaz I1 y con la interfaz de divorcio DI2 para transmitir la funcionalidad de la interfaz a, por ejemplo, las interfaces I2a y I2b, rediseñadas para funcionar con DI2, pero para proporcionar el mismo resultado funcional. De manera similar, DI1 y DI2 pueden funcionar juntas para traducir la funcionalidad de las interfaces I1 y I2 de la Figura 1C a un nuevo sistema operativo, mientras se proporciona el mismo resultado funcional o similar.

Reescritura

Otra posible variante más es reescribir dinámicamente el código para sustituir la funcionalidad de la interfaz con otra cosa pero que consiga el mismo resultado global. Por ejemplo, puede haber un sistema en el que un segmento de código presentado en un lenguaje intermedio (por ejemplo Microsoft IL, Java ByteCode, etc.) se proporcione a un compilador o intérprete en tiempo de ejecución (JIT) en un entorno de ejecución (tal como el proporcionado por la plataforma .Net, el entorno de tiempo de ejecución de Java u otros entornos de tipo de tiempo de ejecución similares). El compilador JIT puede escribirse para convertir dinámicamente las comunicaciones desde el 1er segmento de código al 2º segmento de código, para adaptarlas a una interfaz diferente como puede requerirse por el 2º segmento de código (ya sea el original o un 2º segmento de código diferente). Esto se representa en las Figuras 1L y 1M. Como puede observarse en la Figura 1L, este enfoque es similar al escenario de Divorcio anteriormente descrito. Puede hacerse, por ejemplo, cuando una base de aplicaciones instalada se diseña para comunicar con un sistema operativo de acuerdo con un protocolo de Interfaz1, pero a continuación el sistema operativo se cambia para usar una interfaz diferente. El compilador JIT podría usarse para adaptarse a las comunicaciones al vuelo desde las aplicaciones de base instalas a la nueva interfaz del sistema operativo. Como se representa en la Figura 1M, este enfoque de reescribir dinámicamente la interfaz o interfaces puede aplicarse para factorizar dinámicamente, o alterar de otra manera la interfaz o interfaces también.

Se observa también que los escenarios anteriormente descritos para conseguir el mismo resultado o similar que una interfaz mediante realizaciones alternativas pueden combinarse también de diversas maneras, en serie y/o en paralelo o con otro código intermedio. Por lo tanto, las realizaciones alternativas anteriormente presentadas no son mutuamente exclusivas y pueden mezclarse, coincidirse y combinarse para producir el mismo escenario o equivalentes a los escenarios genéricos presentados en las Figuras 1B y 1C. Se ha de observar también que, con la mayoría de las construcciones de programación, existen otras maneras similares de conseguir la misma funcionalidad o similar de una interfaz que no puede describirse en el presente documento, pero sin embargo se representan mediante el espíritu y alcance de la invención, es decir, se observa que es al menos parcialmente la funcionalidad representada mediante, y los resultados ventajosos posibilitados por, una interfaz que subyace al valor de una interfaz.

Miniaturas

Las Figuras 4A y 4B demuestran un ejemplo de una aplicación de la presente invención al proporcionar miniaturas de ventanas de aplicación. En el ejemplo de la Figura 4A, se ilustra una barra de tareas 400 que contiene botones de barra de tareas. Cada uno de los botones de barra de tareas corresponde a una aplicación activa. El botón 402 de barra de tareas corresponde a un programa de aplicación del Solitario que se ejecuta actualmente en el ordenador. Análogamente, el botón 403 de barra de tareas corresponde a una ventana de programa de aplicación de procesador de textos, el botón 405 de barra de tareas corresponde a una ventana de programa de aplicación de reproductor multimedia y el botón 406 de barra de tareas corresponde a una ventana de programa de aplicación de la Calculadora. Se proporciona una miniatura 401 del Solitario para la ventana del programa de aplicación del Solitario y se proporciona una miniatura 404 del Reproductor Multimedia para el programa de aplicación del Reproductor Multimedia. Las miniaturas correspondientes pueden invocarse en una diversidad de maneras. En un ejemplo no limitante, las miniaturas pueden invocarse desplazando un cursor sobre el botón de barra de tareas correspondiente. Por ejemplo, la miniatura 401 del Solitario puede invocarse desplazando el curso sobre el botón 402 de barra de tareas del Solitario.

También se ilustra en la Figura 4A, que las miniaturas dinámicas pueden proporcionar contenido actualizado o “en directo” que corresponde al programa de aplicación. Por ejemplo, la aplicación de Reproductor Multimedia puede reproducir un vídeo. A medida que el vídeo se reproduce en la ventana de aplicación de Reproductor Multimedia (no mostrado), el vídeo también se reproduce en la miniatura 404 del Reproductor Multimedia. Si la misma ventana de aplicación del reproductor multimedia se minimiza, la miniatura 404 del reproductor multimedia puede presentar la última escena en la ventana de aplicación del reproductor multimedia antes de la minimización. Como alternativa, la miniatura 404 del reproductor multimedia puede continuar visualizando el vídeo a medida que continúa

reproduciendo en la aplicación de reproductor multimedia incluso aunque la misma ventana de aplicación del reproductor multimedia esté minimizada.

La Figura 4B ilustra otro ejemplo de miniaturas de la presente invención. Una barra de tareas 410 se ilustra conteniendo un botón 411 de barra de tareas del Solitario que corresponde a una ventana de aplicación del Solitario (no mostrado), un botón 412 de barra de tareas de procesamiento de textos que corresponde a una ventana de aplicación de procesamiento de textos (no mostrado), un botón 413 de barra de tareas del reproductor multimedia que corresponde a una ventana de aplicación del reproductor multimedia (no mostrado) y un botón 414 de barra de tareas de la calculadora que corresponde a una ventana de aplicación de la calculadora (no mostrado). En este ejemplo, se proporciona una miniatura 415 de la Calculadora. La miniatura 415 de la Calculadora visualiza los contenidos actuales y actualizados de la ventana de aplicación de la Calculadora. En una realización de la presente invención, la miniatura 415 de la Calculadora puede actualizar también la ventana de aplicación. Por ejemplo, puede realizarse un cálculo a través de la miniatura 415 de la Calculadora para conseguir un resultado matemático. Este resultado puede visualizarse en la miniatura 415 de la Calculadora así como la ventana de aplicación de la Calculadora (no mostrado). En este ejemplo, se selecciona la tecla “=” en la miniatura 415 de la Calculadora y se visualiza un resultado en la miniatura de la Calculadora (es decir, “4”). Este resultado puede reflejarse también en la ventana de aplicación de la Calculadora también (no mostrado) incluso aunque el cálculo se realizara a través de la miniatura 415 de la Calculadora.

La Figura 5 ilustra otra aplicación de una miniatura de la presente invención en la que se visualizan miniaturas en un menú ALT-TAB. Como ilustra la Figura 5, una pantalla 500 visualiza ventanas de aplicación. En este caso, una porción de una ventana 502 de aplicación de la Calculadora y una porción de una ventana 501 de aplicación del Solitario son visibles en la pantalla 500. Cada una de las aplicaciones tiene un botón de barra de tareas correspondiente en la barra de tareas. La aplicación de la Calculadora tiene un botón 503 de barra de tareas de la Calculadora correspondiente y la aplicación del Solitario tiene un botón 504 de barra de tareas del Solitario correspondiente. Una aplicación de reproductor multimedia también está activa (no mostrada) y un botón 506 de barra de tareas del Reproductor Multimedia correspondiente también está presente en la barra de tareas.

En el ejemplo ilustrado en la Figura 5, se proporciona un menú 505 ALT-TAB que visualiza miniaturas dinámicas de aplicaciones activas. En este ejemplo, cada una de las miniaturas dinámicas visualiza contenido “en directo” (es decir, actualizado) de una ventana de aplicación correspondiente. El contenido de la miniatura dinámica puede actualizarse en tiempo real. A medida que se modifican o actualizan las ventanas de aplicación, la miniatura dinámica correspondiente puede actualizarse también en consecuencia. La actualización de la miniatura dinámica puede conseguirse también en tiempo real. En este ejemplo, el menú 505 ALT-TAB contiene una miniatura dinámica de cada una de las aplicaciones activas, ya sea la ventana de aplicación de la correspondiente aplicación visible en la pantalla o no. Se ilustra una miniatura 507 de la Calculadora, miniatura 508 del Reproductor Multimedia y miniatura 509 del Solitario en el menú 505 ALT-TAB seleccionándose la miniatura 509 del Solitario. Cada una de las miniaturas dinámicas puede proporcionar contenido “en directo”. Por ejemplo, puede reproducirse un vídeo en la miniatura 508 del Reproductor Multimedia en tiempo real.

Las Figuras 2A y 2B son diagramas que ilustran componentes de una visualización de muestra. El escritorio se representa en la Figura 2A en un gráfico de escena o árbol de visualización como el elemento 200 y contiene el fondo 201 de escritorio y dos aplicaciones, la Calculadora 202 y el Solitario 203. Además, el escritorio contiene adicionalmente una barra de tareas 208. En este método de representación, los marcos alrededor de las ventanas y el área de cliente de cada ventana son todos parte del gráfico de la escena. El movimiento de la ventana se hace cambiando valores de transformación en el árbol de visualización. Cada una de las aplicaciones en este ejemplo contiene un marco 204, 205 de ventana, respectivamente, y contenido 206, 207 de ventana, respectivamente. La barra de tareas 208 contiene el contenido 209 de ventana. En la Figura 2A, los elementos en la pantalla se representan visualmente de izquierda a derecha. Por lo tanto, el fondo 201 de escritorio se representa visualmente detrás de la ventana 203 de aplicación de la Calculadora que se representa visualmente detrás de la ventana 202 de aplicación del Solitario.

La Figura 2B ilustra la pantalla como se representa en la Figura 2A. La pantalla contiene el fondo 220 de escritorio en el fondo. La ventana 222 de aplicación de la Calculadora se visualiza delante del fondo de escritorio y la ventana 221 de aplicación del Solitario se visualiza delante de la ventana 222 de aplicación de Calculadora y del fondo 220 de escritorio. Como se muestra en la Figura 2B, cada una de las ventanas de aplicación puede representarse mediante botones de barra de tareas en la barra de tareas 225. En este ejemplo, la aplicación de la Calculadora se representa mediante el botón 223 de barra de tareas en la barra de tareas 225 y el programa de aplicación del Solitario se representa mediante el botón 224 de barra de tareas en la barra de tareas 225. Cada una de la ventana 222 de aplicación de la Calculadora y la ventana 221 de aplicación del Solitario contienen un marco de ventana y el contenido de ventana como se visualiza en el fondo 220 de escritorio.

Cualquier subnodo del árbol como se representa en la Figura 2A puede usarse como el contenido de otro nodo del árbol. Haciendo esto, los contenidos de una parte del árbol pueden observarse en otra localización en otro lugar en el árbol. Puede haber muchos métodos de conseguir una reproducción de un nodo en el árbol en otra localización en

otro lugar en el árbol. Como un ejemplo no limitante, puede usarse Visual Brush. “VisualBrush” es una técnica usada para referenciar una parte existente del gráfico de la escena, pero para representarla visualmente con el conjunto de contexto de gráficos “actual”, tal como posición, escala, rotación, opacidad, efecto, etc. Esto provoca que se represente visualmente una porción del árbol en un lugar a representar en otro lugar. Por ejemplo, si se desea una miniatura dinámica de la ventana de aplicación de la Calculadora, puede solicitarse una interfaz de programación apropiada (por ejemplo, una Interfaz de Programación de Aplicación o API) para proporcionar un HWND de Anfitrión de miniatura o manejador de ventana. Este ejemplo se demuestra en las Figuras 3A y 3B.

La Figura 3A es un gráfico de escena o diagrama de árbol de visualización que ilustra un ejemplo de componentes de una visualización de la presente invención. El escritorio se representa en la Figura 3A como el elemento o nodo 300 y contiene el fondo 301 de escritorio y dos aplicaciones, la Calculadora 302 y el Solitario 303. Cada una de las aplicaciones en este ejemplo contiene un marco 305, 306 de ventana, respectivamente, y contenido 307, 308 de ventana, respectivamente. El escritorio en este ejemplo contiene adicionalmente una barra de tareas 307.

En este ejemplo, se proporciona una miniatura de la ventana de aplicación de la Calculadora solicitando una API apropiada. Como se ilustra en la Figura 3A, se crea un manejador de ventana HWND 304 de Anfitrión de Miniatura mediante una aplicación y puede registrar una miniatura de una aplicación tal como la aplicación de la Calculadora. Por ejemplo, en la Figura 3A, la barra de tareas 307 contiene un HWND 304 de Anfitrión de Miniatura y contenido 310 de ventana. El HWND 304 de Anfitrión de Miniatura contiene una miniatura 311 de la Calculadora y puede adjuntar una miniatura de la ventana de aplicación al anfitrión de miniatura. La flecha discontinua en la Figura 3A ilustra que la miniatura 311 de calculadora del HWND 304 de Anfitrión de la Miniatura señala o vuelve a dibujar la Calculadora a partir de la ventana de aplicación de la Calculadora. Como resultado, se proporciona una miniatura “en directo” o dinámica que presenta contenido actualizado de la correspondiente ventana de aplicación (en este caso, la ventana de aplicación de la Calculadora) en lugar de simplemente vistas estáticas del contenido de la ventana de aplicación.

Como en el ejemplo de la Figura 2A, los elementos en la pantalla representados en la Figura 3A se representan visualmente de izquierda a derecha. Por lo tanto, el fondo 301 de escritorio se representa visualmente detrás de la ventana de aplicación de la Calculadora 302 que se representa visualmente detrás de la ventana de aplicación del Solitario 303 y de la barra de tareas 208. La Figura 3B demuestra la visualización que corresponde a la Figura 3A. Como ilustra la Figura 3B, la pantalla contiene el fondo 320 de escritorio en el fondo. La ventana 322 de aplicación de la Calculadora se visualiza delante del fondo 320 de escritorio y la ventana 321 de aplicación del Solitario se visualiza delante de la ventana 322 de aplicación de la Calculadora y del fondo 320 de escritorio. También se visualiza la barra de tareas 326. Cada una de las aplicaciones está asociada a un botón de barra de tareas correspondiente en la barra de tareas 326. En este ejemplo, la aplicación de la Calculadora está asociada con el botón 324 de la barra de tareas y la aplicación del Solitario está asociada con el botón 325 de la barra de tareas. Además, la Figura 3B ilustra que se proporciona una miniatura que está asociada con la aplicación de la Calculadora. El programa y ventana 322 de la aplicación de la Calculadora están asociados con el botón 324 de la barra de tareas en la barra de tareas 326 y el programa y ventana 321 de la aplicación del Solitario están asociados con el botón 325 de la barra de tareas.

También se ilustra en la Figura 3B una miniatura 323 dinámica de la ventana 322 de aplicación de la Calculadora. La miniatura 323 se proporciona mediante el nodo 311 de Miniatura de la Calculadora del HWND 304 de Anfitrión de Miniatura que señala o vuelve a dibujar el marco 305 de ventana y el contenido 308 de ventana de la aplicación 302 de la Calculadora como se representa en la Figura 3A. El resultado es una miniatura 323 dinámica de la ventana 322 de aplicación de la Calculadora que es “en directo”, es decir, la miniatura se modifica a medida que se realizan modificaciones correspondientes a la misma ventana de aplicación. Como se ilustra en la Figura 3B, el valor de “243” calculado en la ventana 322 de aplicación de la Calculadora también se refleja en la miniatura 323 correspondiente.

En un aspecto de la presente invención, las miniaturas dinámicas se consiguen mediante registro, anulación de registro y actualización de las propiedades de las miniaturas. Tal registro, anulación de registro y actualización de las propiedades o modificación de miniaturas puede conseguirse a través de interfaces de programación de aplicación (API) en el Gestor de Ventanas de Escritorio (DWM). Se describen tres API ejemplares para conseguir las miniaturas de la presente invención.

Ejemplo 1 de API - registro de la miniatura

Una miniatura puede generarse desde una ventana, por ejemplo, una ventana de aplicación. La ventana de aplicación, en este caso, es la ventana de origen que proporciona a la ventana de aplicación las bases para el contenido de la miniatura. En un ejemplo, la miniatura es una versión miniaturizada de la ventana de aplicación y contiene todos los contenidos de la ventana de aplicación. Como alternativa, la miniatura puede contener únicamente una porción seleccionada de la ventana de origen.

El contenido de la miniatura desde la ventana de origen se representa visualmente en una segunda área. Esta

segunda área se designa a la ventana de destino que adapta el contenido de la miniatura basándose en la ventana de origen. La ventana de destino puede formar la base de la misma miniatura. Por ejemplo, un marco de ventana pequeño en una localización distinta de la ventana de aplicación, un área designada en la pantalla o un menú emergente, pueden usarse como la ventana de destino para visualizar la miniatura basándose en la ventana de aplicación.

Cuando se registra una miniatura, puede realizarse una operación que establece la relación entre la ventana de origen y la ventana de destino. Puede especificarse un manejador de ventana (HWND) que designa la ventana de destino. La ventana de destino especificada puede servir como el objetivo de la representación visual de la miniatura y es de propiedad del proceso que solicita o registra la miniatura. Si el proceso que registra la miniatura también tiene propiedad de la ventana de destino, entonces pueden evitarse modificaciones indeseadas en la aplicación por otras aplicaciones. De manera similar, puede especificarse también un HWND para la ventana de origen.

Puede especificarse también el tamaño de un mapa de bits almacenado en caché de la miniatura. Por ejemplo, puede indicarse un tamaño mínimo de manera que se informa al sistema del último mapa de bits conocido de la ventana de origen. De esta manera, si la ventana de origen se minimiza posteriormente y no es visible en la pantalla, no obstante se mantiene un mapa de bits de la ventana y la miniatura aún está disponible. El tamaño puede variar basándose en las necesidades de la aplicación solicitante. Si hay múltiples registros de la misma ventana de origen, un proceso puede determinar qué tamaño de miniatura se aplicará basándose en las necesidades actuales. Por ejemplo, en el caso de múltiples registros de tamaño de miniatura variable, puede usarse el tamaño más grande.

La API puede contener también un manejador que representa el registro de la miniatura. Este manejador puede ser único para el proceso que está emitiendo la llamada de manera que puede anularse el registro de la miniatura posteriormente únicamente desde el proceso con el que se registró.

Un ejemplo de una API para registrar una miniatura es como sigue:

```
Typedef HANDLE HTHUMBNAIL;
HRESULT
DwmRegisterThumbnail (
   HWND dstWindow,
   HWND srcWindow,
   SIZE *lpMinimizedSize,
   OUT HTHUMBNAIL *lphThumbnail
);
```

En el que dstWindow representa la ventana de destino, srcWindow representa la ventana de origen, lpMinimizedSize representa el tamaño en el que el sistema almacena en caché el último mapa de bits conocido de la ventana de origen (srcWindow) una vez que se minimiza srcWindow y lphThumbnail es un valor retornado que representa el manejador único que representa el registro realizado en la llamada.

Ejemplo 2 de API - modificar o establecer propiedades de miniatura

Después del registro de la miniatura, el contenido de la ventana de origen, o porción del mismo, se coloca como la miniatura en la ventana de destino. Por ejemplo, cuando se solicita, una API, Actualizar propiedades de miniatura, puede modificar o actualizar propiedades de la representación visual de la miniatura en una ventana de destino. Por ejemplo, pueden usarse las propiedades de UpdateThumbnail para establecer propiedades de las miniaturas tales como visibilidad, forma, rectángulos, etc. Una API, Actualizar propiedades de miniatura, puede indicar que se actualice el manejador de una miniatura. Las actualizaciones a miniaturas incluyen cualquier modificación de la miniatura, incluyendo pero sin limitación, situación de la miniatura, tamaño de la miniatura, porción de la ventana de origen a usarse para la miniatura, porción de la ventana de destino a usarse para la miniatura, opacidad o transparencia de la miniatura, visibilidad de la miniatura, efectos especiales de la miniatura (por ejemplo, giro, distorsión, fragmentación, etc.), etc.

En otro ejemplo, las miniaturas pueden actualizarse o modificarse a partir de los procesos en los que se registraron. En un ejemplo, se solicita un manejador de miniatura de manera repetitiva a una ventana de destino de manera que la miniatura puede moverse alrededor de la ventana de destino. En este caso, la miniatura puede reposicionarse en la ventana de destino. Por ejemplo, en una ventana ALT-TAB, las miniaturas pueden reorganizarse en orden variable volviendo a solicitar la miniatura a la ventana de destino.

Pueden usarse parámetros al modificar o establecer propiedades de miniatura. Existen muchos ejemplos de parámetros para modificar o establecer propiedades de miniatura, tales como pero sin limitación, banderas. A través del uso de los parámetros, la estructura de la API puede modificarse más fácilmente haciendo a la API más flexible para mejoras futuras. Los parámetros pueden designar propiedades de la miniatura a usarse y pueden ellos mismos

definirse por constantes especificadas. Por ejemplo, los parámetros pueden definir y corresponder a un parámetro de ventana de origen que define un área o porción de una ventana de origen a usarse como una miniatura en una ventana de destino cuando se desea usar menos que la ventana de origen completa para la miniatura. En este ejemplo, únicamente se usa la porción especificada de la ventana de origen como se indica mediante el parámetro de la ventana de origen para la miniatura en la ventana de destino. En un ejemplo, la porción de la ventana de origen a usarse como la miniatura en la ventana de destino se especifica como coordenadas de la pantalla. Por ejemplo, si la porción deseada es un rectángulo en la ventana de origen, la API puede especificar un parámetro para indicar las coordenadas de un rectángulo que realiza la intersección con la ventana de origen para obtener el rectángulo final a usarse como la miniatura. Como un ejemplo, pueden establecerse las coordenadas (0, 0) como la esquina superior izquierda del área de la ventana. Puede establecerse un parámetro que define y que corresponde al parámetro de ventana de origen deseado para indicar que únicamente se ha de usar una porción especificada de la ventana de origen como la miniatura, definiéndose la porción especificada de la ventana de origen en términos de las coordinadas proporcionadas. Aunque hay muchas maneras en las que especificar las coordinadas y la porción deseada de la ventana de origen, un ejemplo puede incluir usar la porción de la ventana de origen definida por la intersección del rectángulo proporcionado con la ventana de origen real intersecada.

También, la porción de la ventana de destino a usarse para la miniatura puede especificarse mediante un parámetro que define y corresponde a un parámetro de ventana de destino. En este caso, si toda la ventana de destino no desea usarse para la miniatura y únicamente ha de usarse una porción de la ventana de destino para la miniatura, entonces el área de la ventana de destino en la que puede representarse visualmente la miniatura puede especificarse mediante el parámetro de ventana de destino. Si este es el caso, un enfoque sería representar visualmente únicamente en el área designada deseada como se define mediante el parámetro de ventana de destino (es decir, recortando la imagen cuando tiene lugar la representación visual fuera del área designada). También, un método de determinación de la localización de la porción de la ventana de destino en la que puede dibujarse la miniatura es determinar la intersección de la porción de la ventana de destino a usarse y la misma ventana de destino. Esto evita representar visualmente fuera del área designada. Además, puede usarse un parámetro que define y que corresponde a un parámetro de ventana de destino para designar un área dentro del área designada a usarse como una miniatura desde la ventana de origen. En este ejemplo, si no se define un parámetro de ventana de destino, toda la ventana designada puede usarse para el contenido de la miniatura.

Cualquier número de otras propiedades de miniatura puede especificarse en la API. Por ejemplo, puede especificarse la opacidad o transparencia de la miniatura. En un aspecto de este ejemplo de la presente invención, puede asignarse un parámetro para designar la opacidad (o transparencia) de la miniatura, por ejemplo, puede asignarse 255 como totalmente opaco mientras que puede asignarse 0 como totalmente transparente. Por lo tanto, si se establece un parámetro que define y que corresponde a un parámetro de opacidad, puede aplicarse un valor numérico asignado designado para la opacidad de la miniatura. Como resultado, la miniatura puede mostrar valores variables de opacidad según se desee por un usuario.

Como un ejemplo adicional, la miniatura puede hacerse visible o invisible según sea necesario. Puede proporcionarse un parámetro que define y que corresponde a un parámetro de visibilidad. El parámetro de visibilidad puede ser un valor booleano de manera que cuando el parámetro de visibilidad es VERDADERO, la miniatura se hace visible mientras que la miniatura se hace invisible si el parámetro de visibilidad es FALSO. La presente invención no está limitada a ninguna modificación particular de la miniatura ya que ninguna modificación de la miniatura está dentro del alcance de la presente invención.

Un ejemplo de una API para actualizar o modificar una miniatura es DwmUpdateThumbnailProperties como sigue:

```
HRESULT
DwmUpdateThumbnailProperties (
     HTHUMBNAIL hThumbnail,
   DWM_THUMBNAIL_PROPERTIES *ptnProperties );
#define DWM_TNP_RECTDESTINATION 0X00000001
#define DWM_TNP_RECTSOURCE 0X00000002
#define DWM_TNP_OPACIDAD 0X00000004
#define DWM_TNP_VISIBILIDAD 0X00000008
typedef struct_DWM_THUMBNAIL_PROPERTIES
{
     DWORD dwFlags;
     RECT rcDestination;
     RECT rcSource;
     BYTE opacidad;
     BOOL fVisible;
} DWM_THUMBNAIL_PROPERTIES;
```

donde dwFlags define parámetros o banderas que corresponden a parámetros que especifican propiedades

posteriores a usarse. En este ejemplo, el parámetro rcSource es un parámetro de ventana de origen que especifica la región de la ventana de origen (srcWindow) a usar como la miniatura, rcDestination es un parámetro de ventana de destino que especifica la ventana de destino (dstWindow) en la que se va a representar visualmente la miniatura, opacity es un parámetro de opacidad que controla la opacidad de la miniatura (por ejemplo, establecida de 0 a 255) y fVisible es un parámetro de visibilidad que especifica la miniatura como invisible o visible basándose en un valor booleano.

Ejemplo 3 de API - anulación de registro de la miniatura

Una asociación entre la ventana de origen y de destino se elimina cuando ya no es necesaria. Por ejemplo, si una ventana ya no está activa, la asociación entre la ventana y la miniatura puede cumplirse anulando el registro de la asociación. Esto da como resultado liberación de recursos. Además, las miniaturas pueden liberarse adicionalmente cuando se acaba el proceso de registro. Un ejemplo de una API para anular el registro de una miniatura es DwmUnregisterThumbnail como sigue:

```
HRESULT
DwmUnregisterThumbnail(
   HTHUMBNAIL hThumbnail
);
```

La Figura 6 es un diagrama de flujo que ilustra un ejemplo de un proceso para proporcionar miniaturas, que incluye miniaturas “en directo” o dinámicas, de la presente invención. En la etapa 601, se proporciona una API mediante la cual una aplicación puede solicitar una primera ventana de nivel superior o ventana de origen para visualizar una miniatura que puede localizarse también en una sección de una segunda ventana de nivel superior o ventana de destino. Por lo tanto, la primera ventana de nivel superior o ventana de origen puede representarse visualmente como el objeto más superior en la ventana de destino. Un ejemplo no limitante de esta implementación es una ventana ATL-TAB en la que se proporciona una miniatura que corresponde a una ventana de origen en la ventana ALT-TAB (es decir, la ventana de destino).

En la etapa 602, se realiza una solicitud de registro al DWM que puede proporcionar el manejador de la miniatura. Puede haber también múltiples solicitudes en la misma ventana de origen o la misma ventana de destino. En el ejemplo de ALT-TAB, múltiples solicitudes en la misma ventana de origen y en la misma ventana de destino dan como resultado la reorganización de las miniaturas en la ventana ALT-TAB así como proporcionan una representación más grande y/o estilizada de una miniatura de una aplicación cuando se selecciona la miniatura. En otro ejemplo, el orden en Z de la miniatura puede modificarse mediante llamadas de registro posteriores. Si se desea una modificación del orden en z de la miniatura con relación a otros elementos de visualización, puede registrarse una nueva miniatura con otra llamada de registro de miniatura. Después de que se establecen las propiedades para la nueva miniatura que corresponden a la miniatura antigua, puede anularse el registro de la miniatura antigua. Por lo tanto, en este ejemplo, el orden en z de la miniatura puede cambiarse mediante registro múltiple de la miniatura.

Como alternativa, puede haber llamadas a múltiples ventanas de origen para la misma ventana de destino. En este ejemplo, las llamadas a múltiples ventanas de origen para una ventana de destino dan como resultado proporcionar diferentes miniaturas en la ventana de destino.

En otro ejemplo más, las miniaturas dinámicas pueden visualizarse dentro de otras miniaturas dinámicas. Por ejemplo, una primera miniatura dinámica con un orden en z superior puede colocarse en una miniatura dinámica más grande de una aplicación con un orden en z inferior. En este ejemplo, las miniaturas pueden embeberse dentro de otras miniaturas. Además, pueden aplicarse efectos a cualquiera de las miniaturas o a cualquier miniatura embebida. Como alternativa, las miniaturas de cualquier orden en z pueden estar en cascada en otra miniatura de cualquier orden en z. Por ejemplo, al menos una miniatura (independientemente del orden en z) puede colocarse en una miniatura dinámica más grande de una aplicación de cualquier orden en z. Análogamente, la miniatura dinámica más grande puede colocarse en otra miniatura dinámica de cualquier orden en z. Por lo tanto, las miniaturas pueden estar en cascada independientemente del orden en z.

En la etapa 603, el DWM registró el par proceso/ID para la miniatura. El DWM puede mantener adicionalmente una lista global de registros de miniatura en la que se empareja cada aplicación con un correspondiente manejador de miniatura (HWND). La lista global puede mantenerse en una diversidad de maneras dependiendo de las necesidades del usuario. Por ejemplo, una manera conveniente de mantener la lista global es indexar las miniaturas mediante la ventana de destino (dstWindow) de manera que cualquier ventana de destino deseada pueda accederse de manera eficaz. También, las miniaturas en cualquier ventana de destino particular pueden organizarse adicionalmente en un orden lógico, tal como en orden de solapamiento o basándose en orden en z.

En otro ejemplo, el DWM puede “fotografiar” los contenidos de una ventana (es decir, grabar los contenidos actuales de la ventana en el momento) antes de que se minimice la ventana. El DWM fotografía los contenidos de la ventana,

por ejemplo, en un mapa de bits de un tamaño particular. El tamaño puede determinarse mediante la API de registro y es un tamaño apropiado para mapeo de miniatura. Después de que se minimice la ventana, los contenidos de la ventana ya están conservados de manera que una miniatura que corresponde a la ventana de aplicación puede aún estar disponible. Como alternativa, las ventanas que están minimizadas pueden “aparcarse” temporalmente fuera de la pantalla en lugar de minimizarse realmente. De esta manera, las aplicaciones pueden continuar enviando información y datos para actualizar las miniaturas. Por lo tanto, si tiene lugar el registro de miniatura después de que se “minimice” una ventana de aplicación, las miniaturas pueden aún registrarse y actualizarse como si la ventana no se hubiera minimizado. Como alternativa, las miniaturas pueden sustituirse por iconos o barras de título si el registro de la miniatura no es factible.

En la etapa 604, se determinan e implementan actualizaciones para las propiedades de la miniatura. Al modificar o actualizar las propiedades de la miniatura, el DWM puede notificar a la lista global en consecuencia. Por ejemplo, la aplicación de DWM puede añadir miniaturas en su propia escenografía (por ejemplo, la Figura 2A o la Figura 3A) en la que el nodo de ventana de nivel superior apropiado que representa la ventana objetivo se vuelve a dibujar desde la ventana de origen (es decir, sreWindow) basándose en instrucciones de dibujo adicionales. Cualquier modificación o efectos especificados adicionales a la miniatura pueden incorporarse también, por ejemplo, cambio de escala, posicionamiento, opacidad, invisibilidad o efectos especiales.

En la etapa 605, la miniatura se modifica basándose en instrucciones recibidas. Por ejemplo, una API UpdateThumbnail puede modificar la visibilidad del nodo de la miniatura si el parámetro visible es VERDADERO o la API UpdateThumbnail puede aumentar (o reducir) la opacidad de la miniatura. Después de que se realicen todas las modificaciones deseadas a la miniatura, el proceso de DWM determina si se desea una anulación de registro de la miniatura. Este proceso de determinación de si se desea una anulación de registro de la miniatura puede realizarse de manera separada del proceso UpdateThumbnail. En este caso, puede tener lugar la anulación de registro debido a una llamada de API UnregisterThumbnail API (no mostrada). En la etapa 606, se determina si se recibe una API de anulación de registro por el proceso de DWM. Si se recibe la API de anulación de registro, se elimina el emparejamiento del proceso y el ID de la miniatura correspondiente (por ejemplo, de la lista global de registros de miniatura mantenida en el proceso de DWM). En la etapa 607, el flujo de instrucciones de ventana de nivel superior se vuelve a emitir como resultado de que se anule el registro de la miniatura. Como resultado, la miniatura se elimina.

En un ejemplo, un cursor se desplaza sobre un botón de barra de tareas dando como resultado la visualización de la miniatura de la aplicación correspondiente. Cuando el cursor se desplaza sobre el botón de barra de tareas, el proceso crea un nuevo nivel superior de manejador de ventana (HWND) que alberga una miniatura. El proceso solicita una API de registro que usa el manejador de ventana (HWND) de la aplicación como el origen del contenido para la miniatura (es decir, srcWindow). El HWND de nivel superior nuevo que alberga la miniatura es la ventana de destino (dstWindow). El contenido de la ventana de origen, o una porción deseada de la misma, se coloca a continuación en la ventana de destino para visualizar la miniatura. La miniatura puede contener todo el contenido de ventana de origen o puede contener únicamente una porción predeterminada o seleccionada como se ha expuesto anteriormente. La API Actualizar Miniatura, puede modificar o mover la miniatura. Por ejemplo, la miniatura puede reposicionarse con relación al correspondiente icono o pueden aplicarse efectos especiales a la miniatura (por ejemplo, transparencia, invisibilidad, distorsión, etc.). Cuando el cursor se aleja del botón de barra de tareas, puede anularse el registro del HWND de nivel superior que alberga la miniatura de manera que se oculta la miniatura. Como alternativa, la miniatura puede actualizarse para que sea invisible de manera que la miniatura está registrada aún y puede invocarse posteriormente. En un método alternativo, cada ventana después de la creación puede asignarse a una miniatura creada para ella con la barra de tareas. En este ejemplo, el código de la barra de tareas puede usarse para manipular la visibilidad de las miniaturas y el rectángulo de destino para la colocación de la miniatura, por ejemplo.

Se entiende que los aspectos de la presente invención pueden tomar muchas formas y realizaciones. Las realizaciones mostradas en el presente documento se pretenden para ilustrar en lugar de para limitar la invención, apreciándose que pueden realizarse diversas variaciones sin alejarse del espíritu del alcance de la invención. Aunque se han mostrado y descrito realizaciones ilustrativas de la invención, se pretende una amplia gama de modificaciones, cambios y sustituciones en la divulgación anterior y en algunos casos algunas características de la presente invención pueden emplearse sin uso correspondiente de las otras características. Por consiguiente, es apropiado que las reivindicaciones adjuntas se consideren ampliamente y de una manera coherente con el alcance de la invención.

REIVINDICACIONES

1. Un método implementado por ordenador de visualización de una ventana (221, 222, 321, 322, 501, 502) en un dispositivo de visualización, comprendiendo el método:

visualizar una ventana (221, 222, 321, 322, 501, 502) de origen, comprendiendo la ventana de origen primer contenido que puede modificarse;

minimizar dicha ventana (221, 222, 321, 322, 501, 502) de origen; visualizar una ventana (505) de destino en la misma pantalla,

en el que al menos una porción de la ventana de destino contiene una miniatura (323, 401, 404, 415, 507, 508, 509), incluyendo la miniatura segundo contenido que corresponde a al menos una porción del contenido de la ventana de origen;

visualizar automáticamente, en dicha miniatura (323, 401 ,404, 415, 507, 508, 509), un último contenido visualizado mediante la ventana de origen antes de la minimización; y

modificar el segundo contenido de la miniatura cada vez que cambie el primer contenido de la ventana de origen;

recibir un primer parámetro para definir al menos una porción de la ventana de origen, en el que dicha modificación comprende determinar al menos una porción de la ventana de origen para dibujar en la miniatura basándose en dicho primer parámetro; y

recibir un segundo parámetro para definir al menos una porción de las ventanas de destino, en el que dicha modificación comprende adicionalmente determinar al menos una porción de la ventana de destino para representar visualmente la miniatura basándose en dicho segundo parámetro;

en el que la modificación de la ventana de origen se visualiza en la miniatura.

2. El método implementado por ordenador de la reivindicación 1, en el que la miniatura (323, 401, 404, 415, 507, 508, 509) se modifica en tiempo real a medida que se modifica el primer contenido de la ventana (221, 222, 321, 322, 501, 502) de origen.

3. El método implementado por ordenador de la reivindicación 1, en el que modificar el segundo contenido de la miniatura (323, 401, 404, 415, 507, 508, 509) comprende adicionalmente modificar la visibilidad de la miniatura.

4. El método de la reivindicación 1, en el que la etapa de modificar el segundo contenido de la miniatura (323, 401, 404, 415, 507, 508, 509) tiene lugar automáticamente cada vez que cambia el primer contenido de la ventana (221, 222, 321, 322, 501, 502) de origen.

5. Un sistema para visualizar una ventana (221, 222, 321, 322, 501, 502) en un dispositivo de visualización, comprendiendo el sistema:

una pantalla para visualizar una ventana (221, 222, 321, 322, 501, 502) de origen que comprende contenido que puede modificarse y una ventana (505) de destino visualizada en dicha pantalla;

un gestor para registrar una miniatura (323, 401, 404, 415, 507, 508, 509), estando la miniatura en dicha ventana de destino y que corresponde a al menos una porción de la ventana de origen, comprendiendo dicho registro:

recibir un manejador de ventana para dicha ventana de origen;

establecer una asociación entre la ventana de origen y la miniatura de modo que al menos una porción del contenido de la ventana de origen se dibuja en la miniatura;

recibir un primer parámetro para definir al menos una porción de la ventana de origen, en el que dicha modificación comprende determinar al menos una porción de la ventana de origen para dibujar en la miniatura basándose en dicho primer parámetro; y

recibir un segundo parámetro para definir al menos una porción de la ventana de destino, en el que dicha modificación comprende adicionalmente determinar al menos una porción de la ventana de destino para representar visualmente la miniatura basándose en dicho segundo parámetro;

y

recibir un manejador único que representa la asociación entre la ventana de origen y la miniatura;

un procesador; y

memoria que almacena instrucciones ejecutables por ordenador que, cuando se ejecutan mediante el procesador, provocan que el sistema realice un método de modificación del segundo contenido de la miniatura cada vez que cambia el contenido de la ventana de origen, comprendiendo dicho método las etapas de:

determinar al menos una porción de la ventana de origen para dibujar en la miniatura basándose en un primer parámetro para definir al menos una porción de la ventana de origen; y

determinar la miniatura para representar visualmente basándose en un segundo parámetro para definir al menos una porción de la ventana de destino en la que representar visualmente la miniatura.

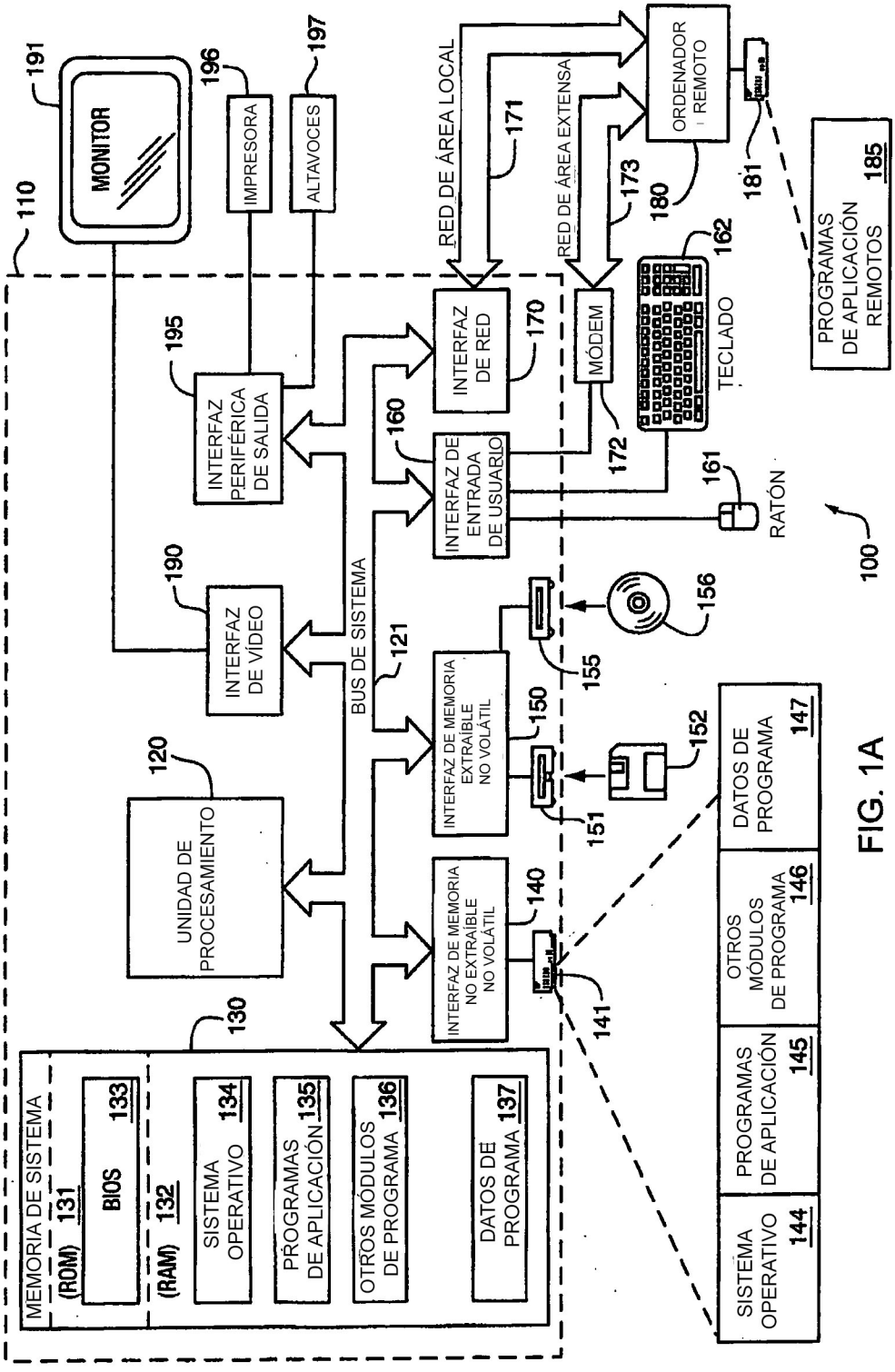

110
MONITOR
191
196
IMPRESORA
197
ALTAVOCES
RED DE ÁREA LOCAL
171
RED DE ÁREA EXTENSA
173
ORDENADOR REMOTO
180
181
162
TECLADO
PROGRAMAS DE APLICACIÓN REMOTOS
185
195
INTERFAZ PERIFÉRICA DE SALIDA
INTERFAZ DE RED
170
MÓDEM
172
160
INTERFAZ DE ENTRADA DE USUARIO
161
RATÓN
100
190
INTERFAZ DE VÍDEO
BUS DE SISTEMA
121
156
155
150
INTERFAZ DE MEMORIA EXTRAÍBLE NO VOLÁTIL
152
151
120
UNIDAD DE PROCESAMIENTO
INTERFAZ DE MEMORIA NO EXTRAÍBLE NO VOLÁTIL
140
141
130
MEMORIA DE SISTEMA
(ROM) 131
BIOS 133
(RAM) 132
SISTEMA OPERATIVO 134
PROGRAMAS DE APLICACIÓN 135
OTROS MÓDULOS DE PROGRAMA 136
DATOS DE PROGRAMA 137
DATOS DE PROGRAMA 147
OTROS MÓDULOS DE PROGRAMA 146
PROGRAMAS DE APLICACIÓN 145
SISTEMA OPERATIVO 144

FIG. 1A

FIG. 1C

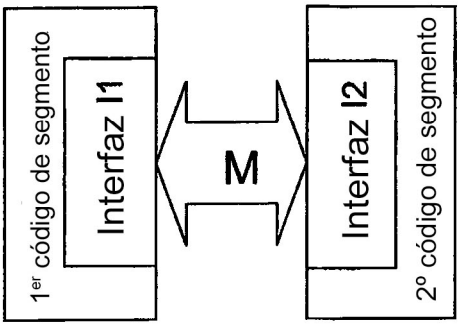


FIG. 1E

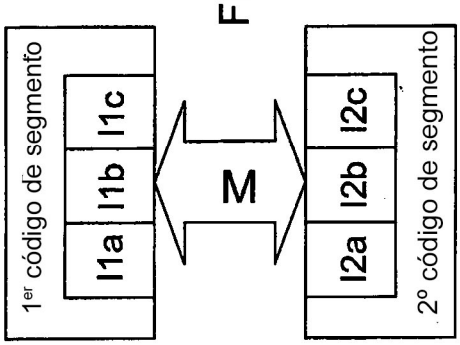


FIG. 1B

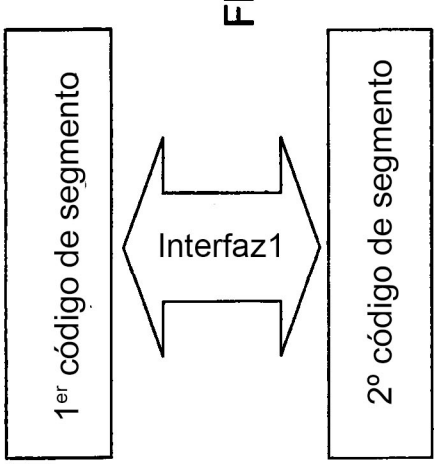
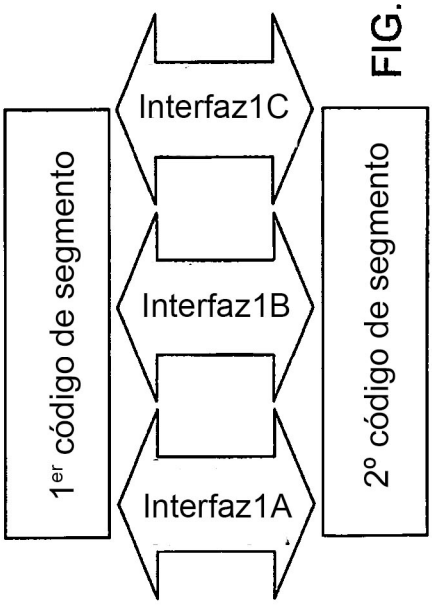


FIG. 1D



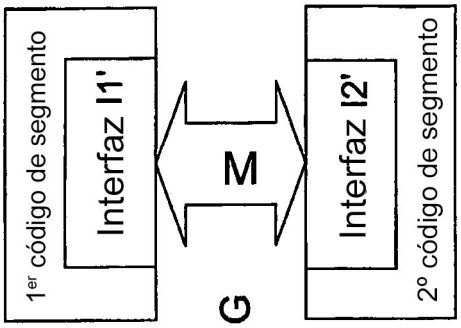


FIG. 1G

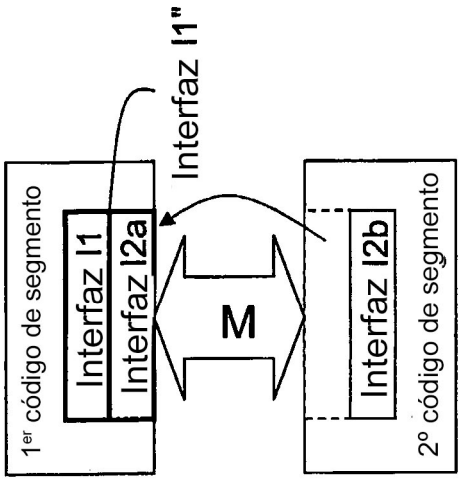


FIG. 1I

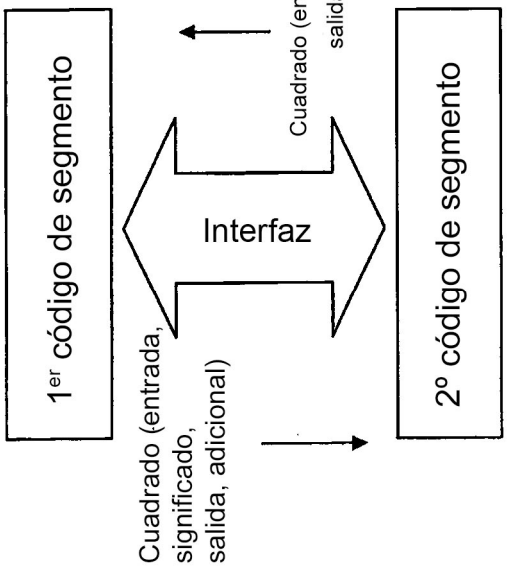


FIG. 1F

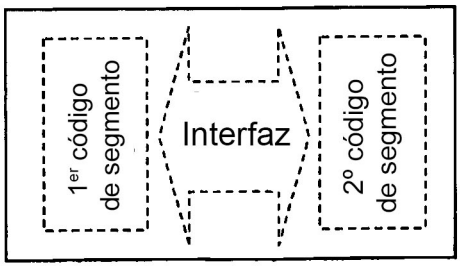


FIG. 1H

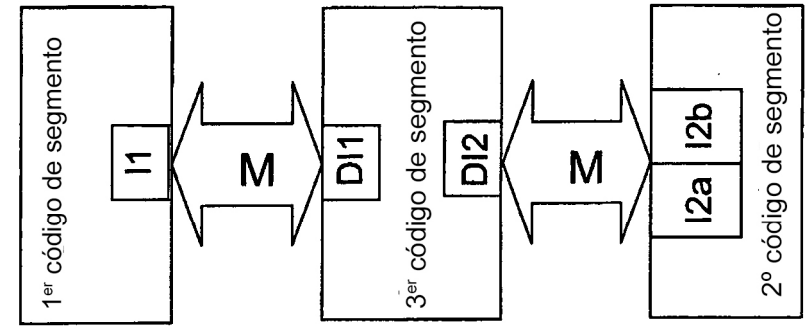

1er código de segmento
I1
M
DI1
3er código de segmento
DI2
M
I2a
I2b
2º código de segmento

FIG. 1K

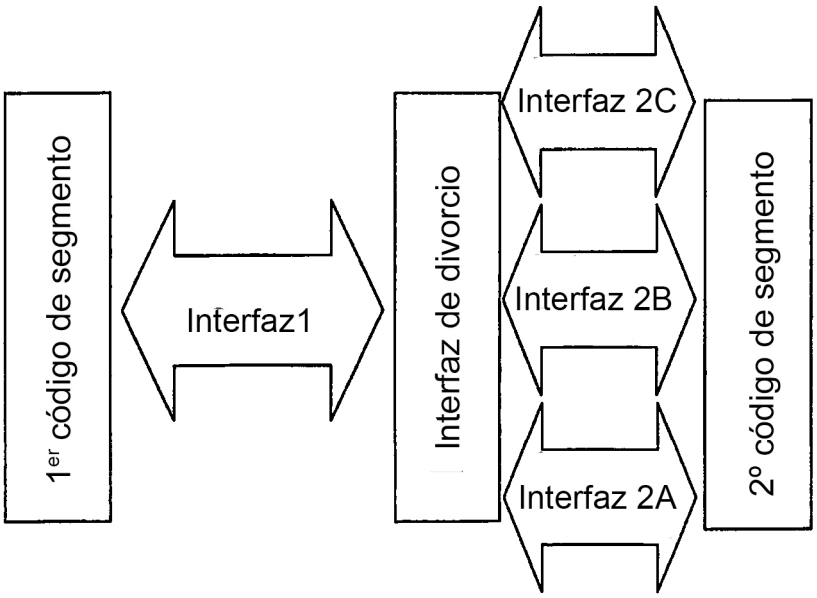

1er código de segmento
Interfaz1
Interfaz de divorcio
Interfaz 2C
Interfaz 2B
Interfaz 2A
2º código de segmento

FIG. 1J

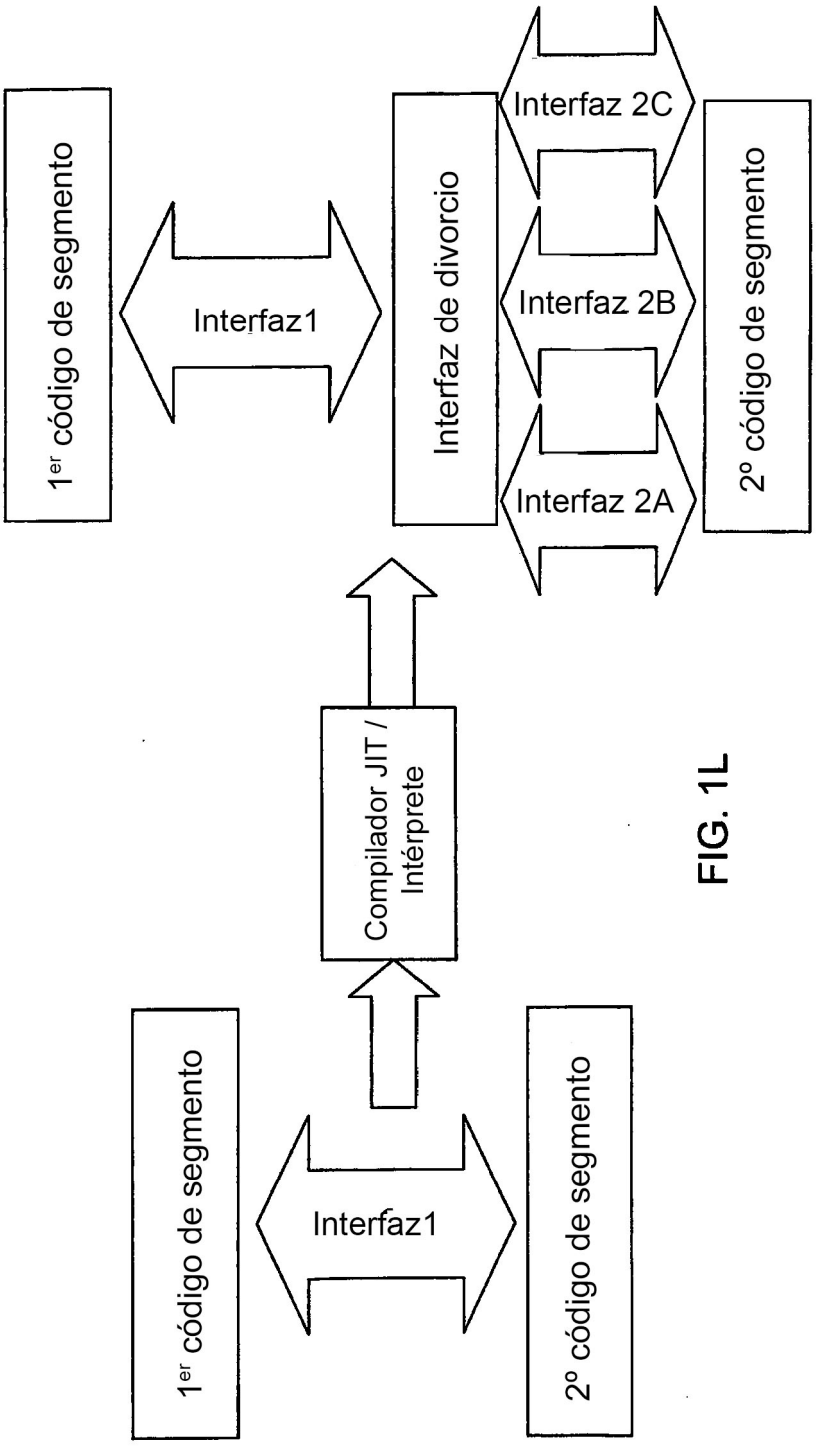

1er código de segmento
Interfaz1
Interfaz de divorcio
Interfaz 2C
Interfaz 2B
Interfaz 2A
2º código de segmento
Compilador JIT / Intérprete
1er código de segmento
Interfaz1
2º código de segmento

FIG. 1L

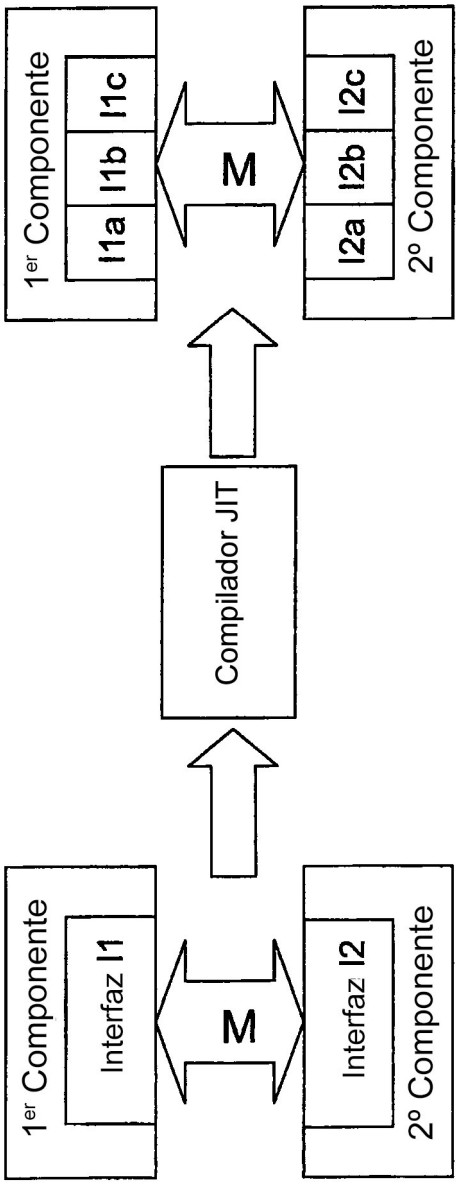

1er Componente
I1a
I1b
I1c
M
I2a
I2b
I2c
2º Componente
Compilador JIT
1er Componente
Interfaz I1
M
Interfaz I2
2º Componente

FIG. 1M

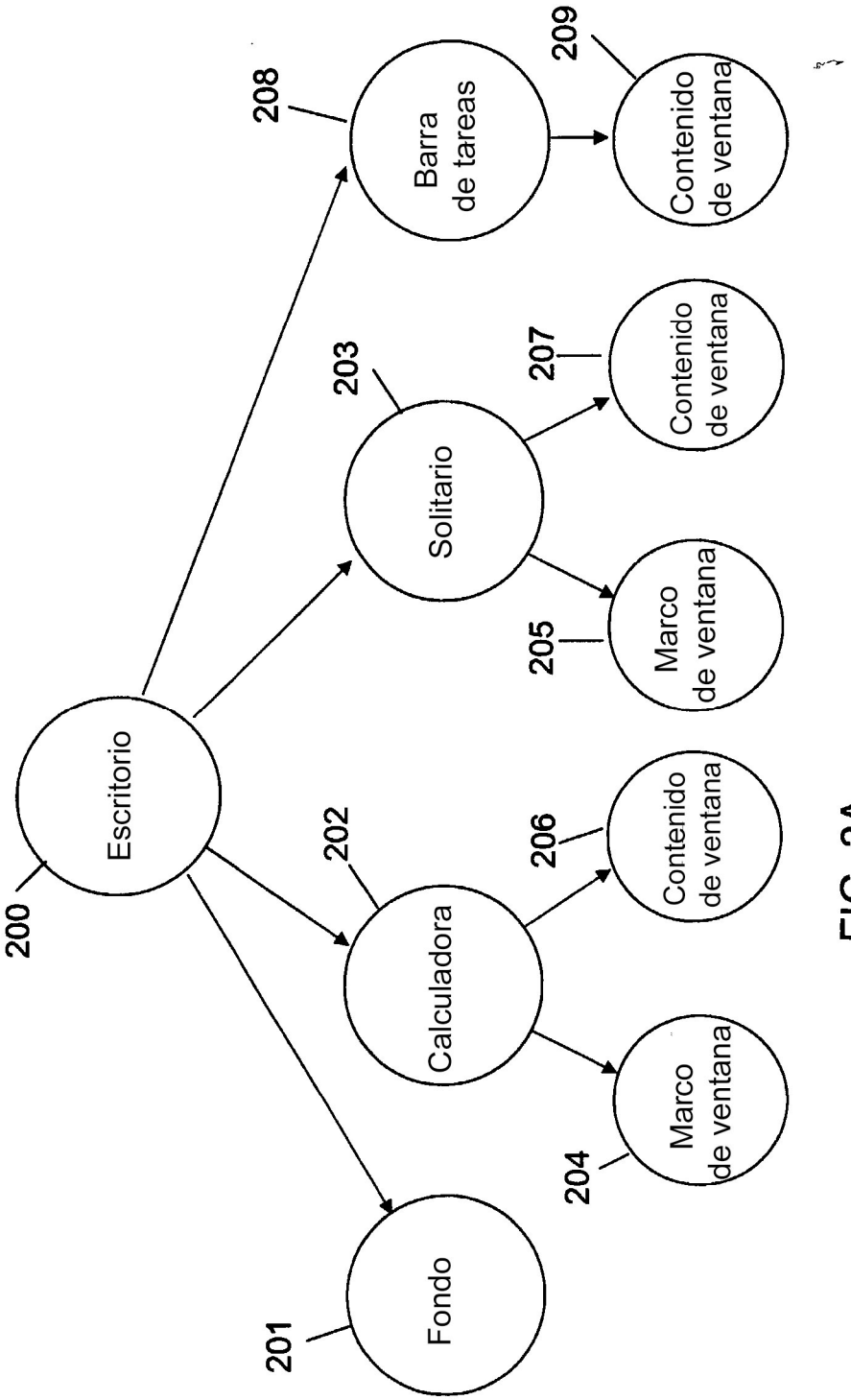

200
Escritorio
201
Fondo
202
Calculadora
203
Solitario
204
Marco de ventana
205
Marco de ventana
206
Contenido de ventana
207
Contenido de ventana
208
Barra de tareas
209
Contenido de ventana

FIG. 2A

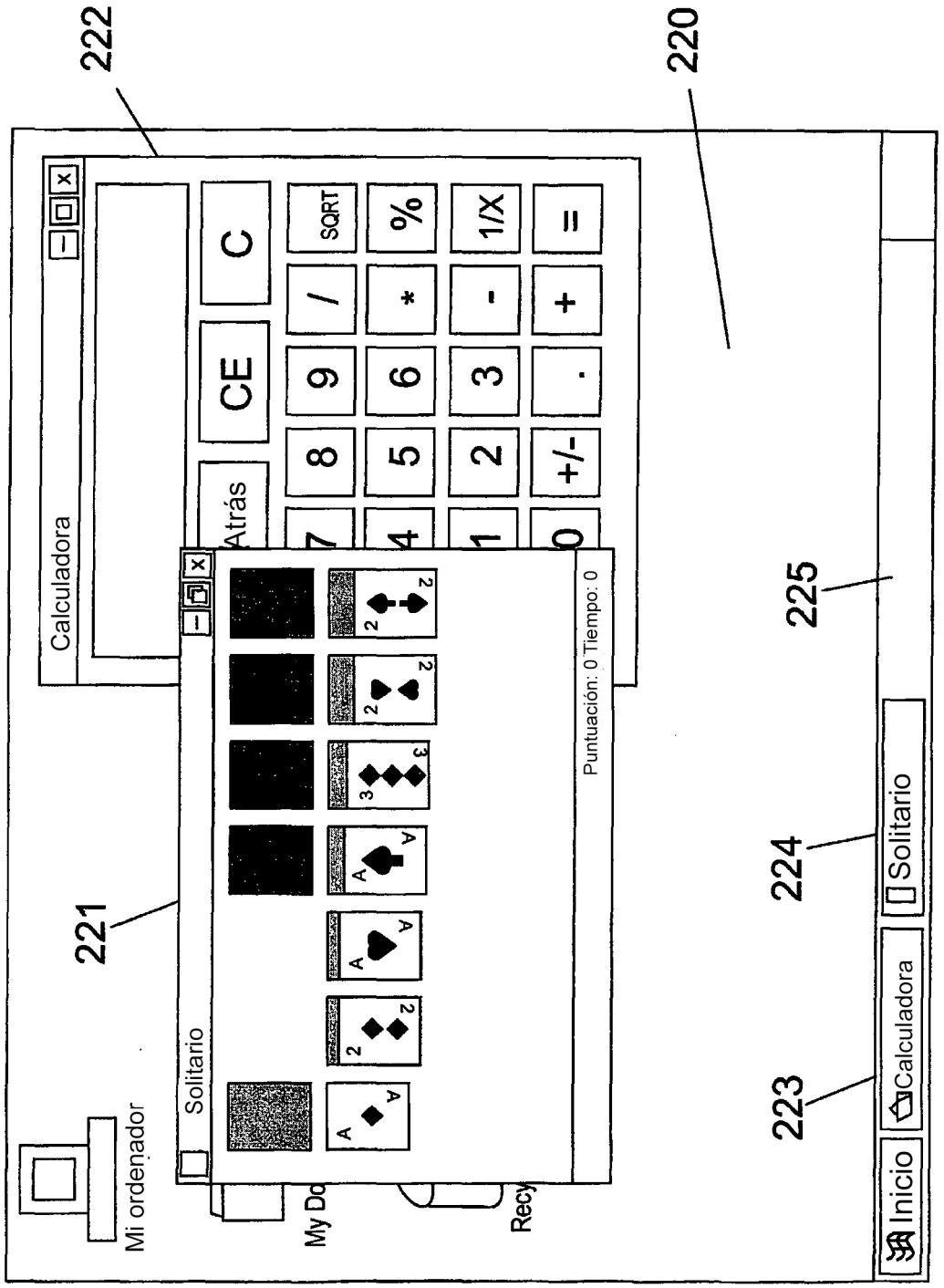

222
220
Calculadora
Atrás
CE
C
SQRT
%
1/X
=
9
/
*
-
+
8
6
3
.
5
2
+/-
7
4
1
0
221
Solitario
Puntuación: 0 Tiempo: 0
225
224
223
Mi ordenador
Inicio
Calculadora
Solitario

Fig. 2B

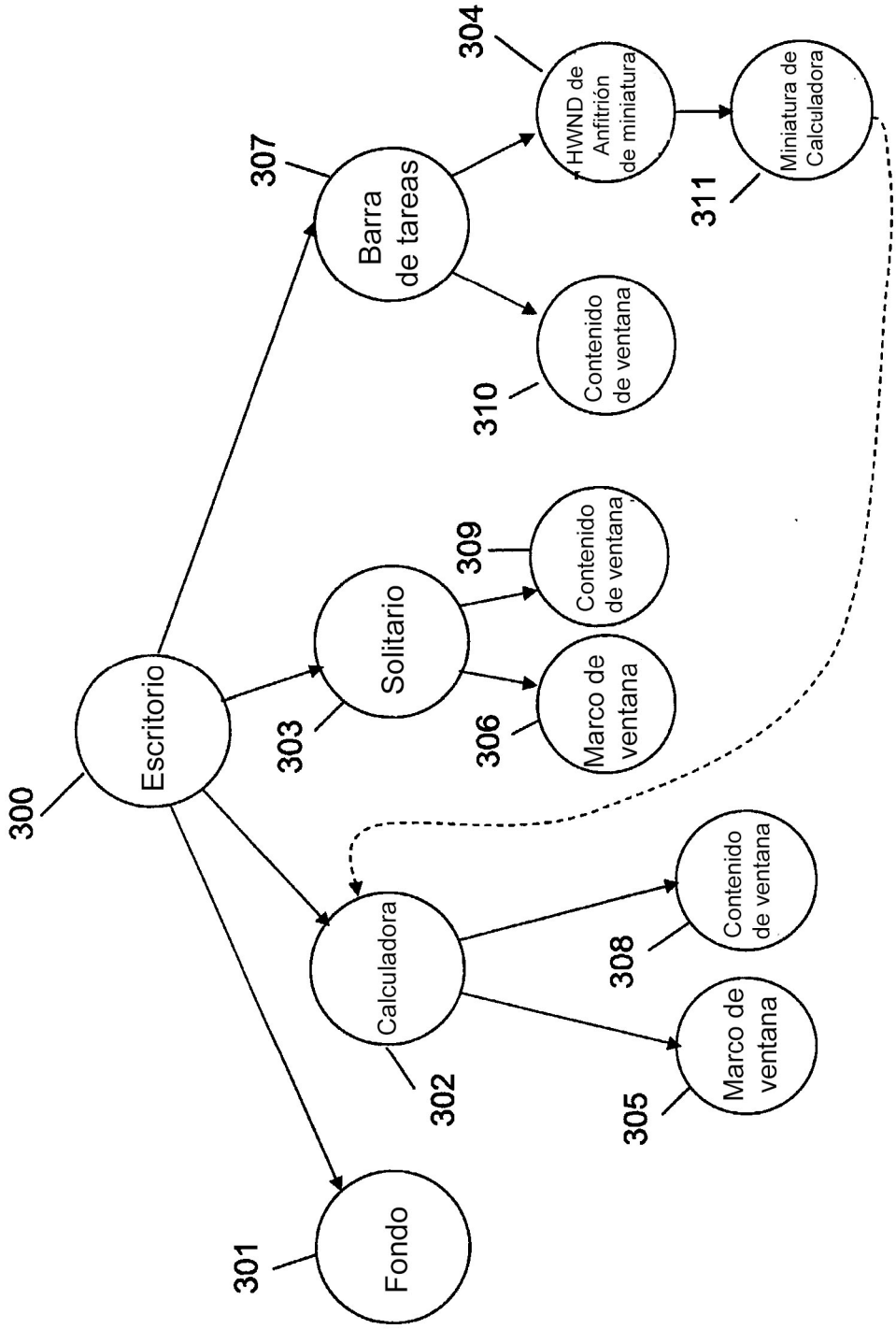

300
Escritorio
301
Fondo
302
Calculadora
303
Solitario
305
Marco de ventana
306
Marco de ventana
307
Barra de tareas
308
Contenido de ventana
309
Contenido de ventana
310
Contenido de ventana
311
Miniatura de Calculadora
304
HWND de Anfitrión de miniatura

FIG. 3A

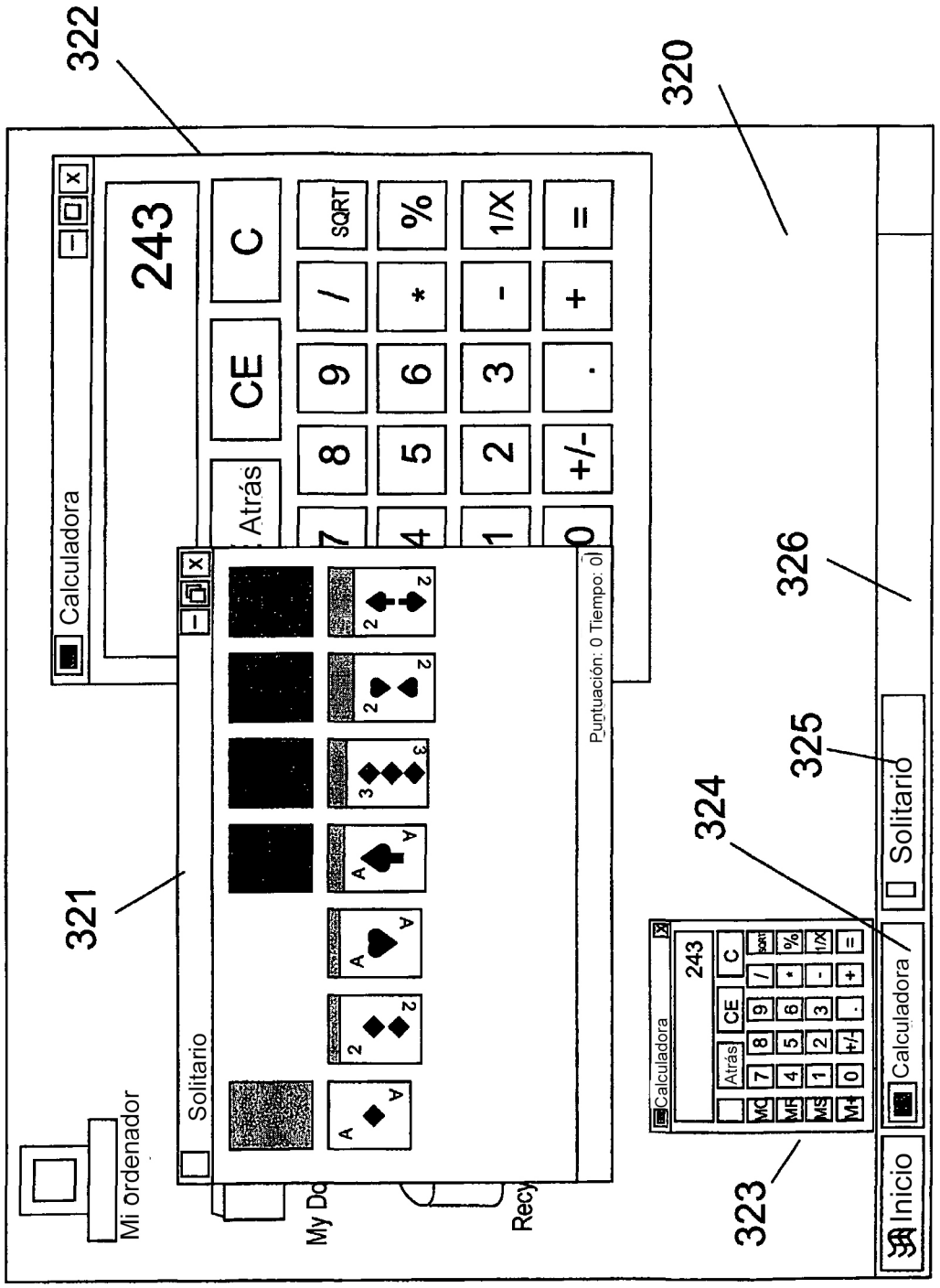

322
320
321
326
325
324
323
Calculadora
243
C
CE
Atrás
SQRT
%
1/X
=
+
.
+/-
0
Solitario
Puntuación: 0 Tiempo: 0
Mi ordenador
Inicio
Calculadora
Solitario

Fig. 3B

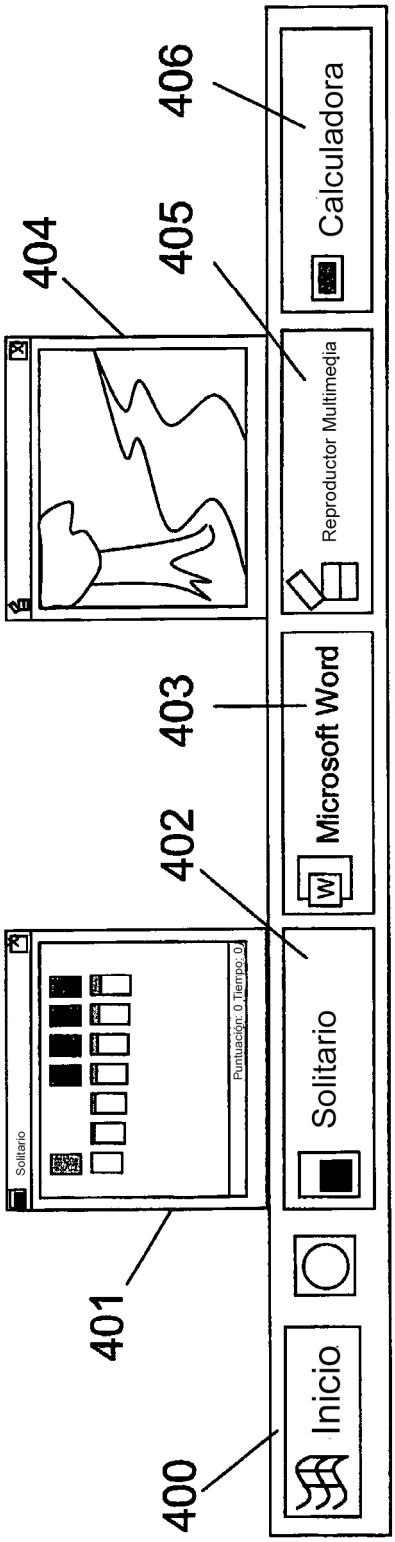

Inicio
Solitario
Microsoft Word
Reproductor Multimedia
Calculadora
Solitario
Puntuación: 0 Tiempo: 0
400
401
402
403
404
405
406

Fig. 4A

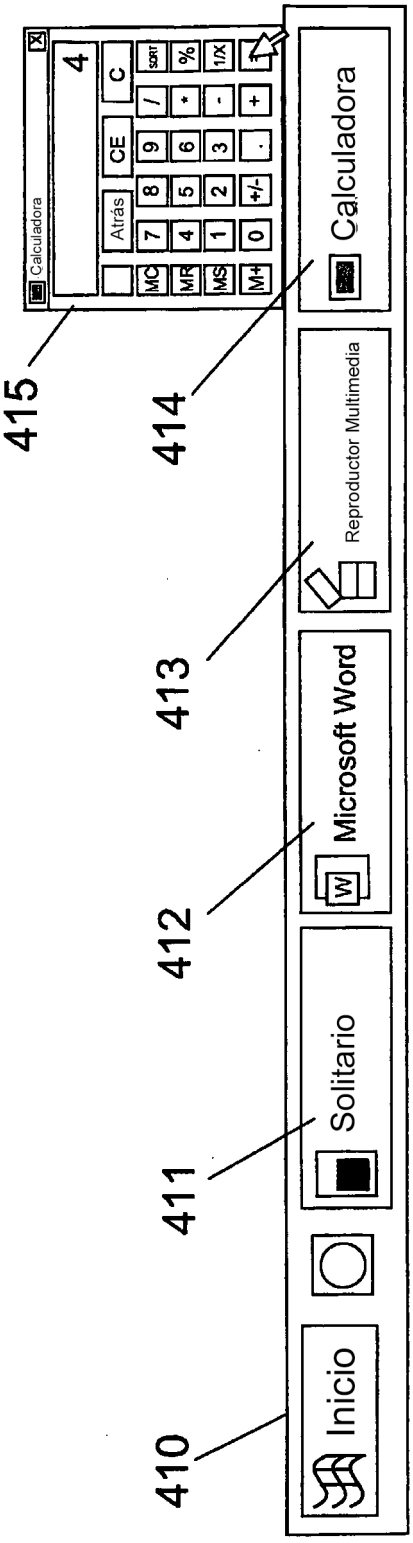

Inicio
Solitario
Microsoft Word
Reproductor Multimedia
Calculadora
Calculadora
410
411
412
413
414
415

Fig. 4B

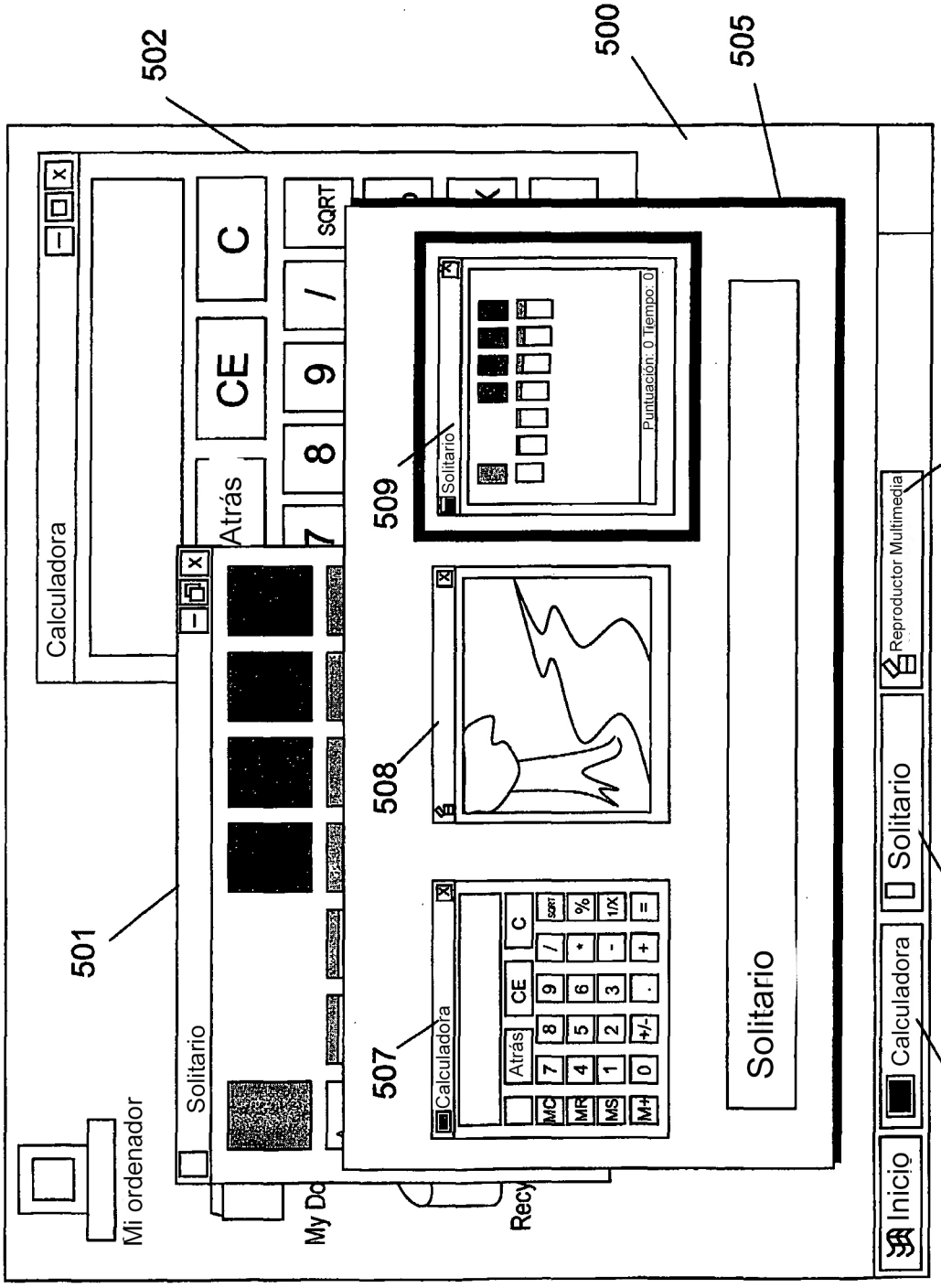

502
500
505
501
509
508
507
504
503
506
Calculadora
Atrás
CE
C
SQRT
Solitario
Puntuación: 0 Tiempo: 0
Reproductor Multimedia
Inicio
Calculadora
Solitario
Mi ordenador

Fig. 5

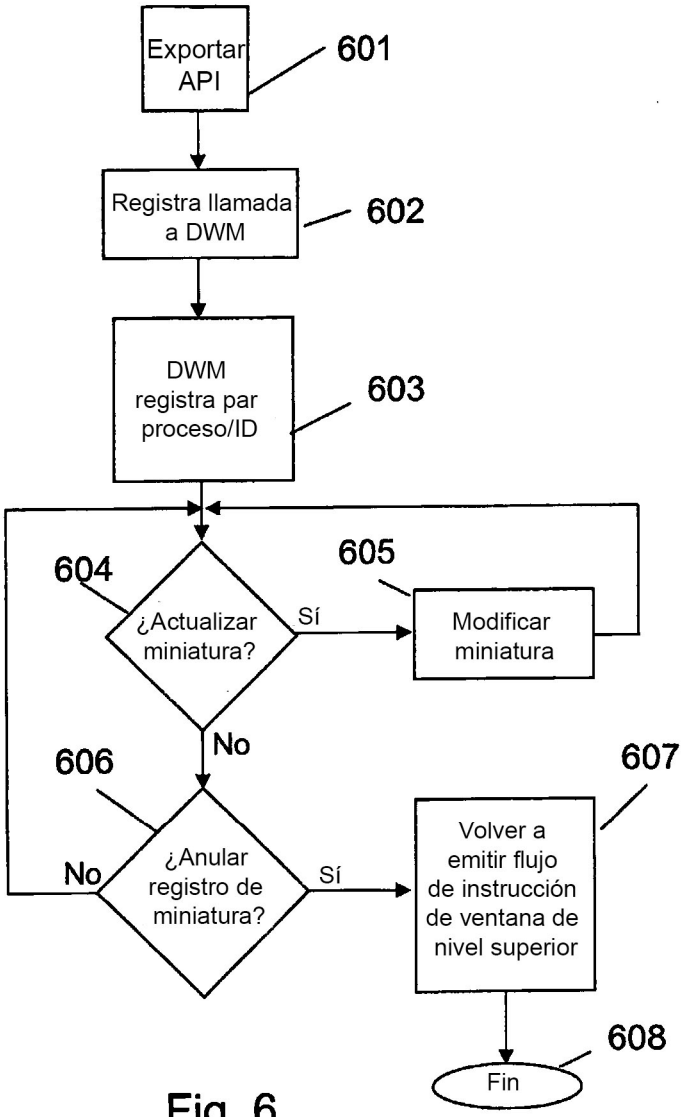

Exportar API
601
Registra llamada a DWM
602
DWM registra par proceso/ID
603
604
¿Actualizar miniatura?
Sí
605
Modificar miniatura
No
606
¿Anular registro de miniatura?
No
Sí
607
Volver a emitir flujo de instrucción de ventana de nivel superior
608
Fin

Fig. 6