

PRE-NIÑOS EN UN ÁRBOL DE INTERFAZ DE USUARIO

REFERENCIA CRUZADA A LA SOLICITUD RELACIONADA

[0001] La presente solicitud reivindica la prioridad de la solicitud de patente no provisional U.S. número de serie 14/843.798 presentada el 2 de septiembre de 2015 y la solicitud de patente provisional U.S. número de serie 62/046,134, presentada el 4 de septiembre de 2014, la totalidad de cada una se incorpora aquí por referencia.

ANTECEDENTES

[0002] Una interfaz de usuario (UI) de un programa de aplicación tiene a menudo cientos o miles de elementos que cuando se representan se combinan de formas complejas para proporcionar una (con optimismo) experiencia de usuario visualmente atractiva que también es sencilla y consistente para usar. Durante el funcionamiento del programa, es común organizar la UI en una estructura de árbol, en la que cada elemento de un árbol tiene como máximo un elemento padre y cero o más elemento niño.

[0003] Para dibujar el árbol de la UI, es común realizar un recorrido de pre-orden, en el que un elemento dibuja, y luego cada uno de sus elementos niños dibuja en orden, recursivamente. Este tipo de recorrido establece un orden *z* tal que cuando se dibuja utilizando el algoritmo de un pintor, los niños de un elemento de la UI están visualmente encima del propio dibujo del propio elemento de la UI.

[0004] Por ejemplo, un diseñador de la UI puede que desee tener un botón azul rectangular que contiene algún texto en negro (e.g., "Click Me"). El diseñador

organiza los elementos de tal manera que el elemento padre dibuja un rectángulo azul y luego un elemento de texto niño dibuja la cadena de texto negro sobre (colocada dentro) ese rectángulo azul. La estructuración del árbol de la UI con el elemento de texto niño desacoplado de esta forma del elemento de botón padre proporciona una propiedad útil en la que el árbol es componible (composable). Por ejemplo, dado que el texto está desacoplado del botón, el diseñador puede tener en su lugar el botón que contenga una imagen (al tener una imagen del elemento niño) en lugar del texto, o puede tener tanto una imagen como un texto (otro niño) dibujado en el mismo botón. Si los elementos son objetos que contienen lógica/código (es decir, al menos el elemento de botón), la lógica/comportamiento configurado en el código del elemento de botón no cambia.

[0005] Sin embargo, considere que el elemento de botón recibe un enfoque (e.g., a través de un teclado, controlador de juego, ratón o similar) y el diseñador quiere cambiar la forma en que el elemento de botón se presenta al usuario para reflejar este estado de enfoque tal como dibujando un óvalo naranja detrás del botón para resaltar la apariencia del botón cuando se enfoca. En esta situación, el óvalo resaltado no comparte la propiedad de ser componible, ya que el botón (representado como el rectángulo azul) necesita conocer la apariencia de su resaltado (el óvalo naranja) para que el óvalo naranja se pueda dibujar antes de dibujar la apariencia del rectángulo azul del botón.

RESUMEN

[0006] Este Resumen se proporciona para introducir una selección de conceptos representativos en una forma simplificada que se describen más adelante en la Descripción Detallada. El presente Resumen no pretende identificar las características clave o características esenciales de la materia reivindicada ni

tampoco pretende ser utilizado de ninguna manera que limite el alcance de la materia reivindicada.

[0007] Brevemente, la tecnología descrita en el presente documento está dirigida a tener uno o más elementos de interfaz de usuario pre-niños en el árbol de interfaz de usuario que se puede dibujar antes de que se dibujen sus elementos de interfaz de usuario padre. Un elemento de la interfaz de usuario padre invoca cualquier elemento de interfaz de usuario pre-niño. Cada pre-niño puede incluir una lógica de pre-niño para dibujar una representación del elemento de interfaz de usuario pre-niño en una primera vez, que es antes de una segunda vez en la que se dibuja una representación de un elemento de interfaz de usuario padre. La lógica de pre-niño puede determinar si se dibuja la representación de pre-niño sobre la base de datos de estado actual, e.g., como un resaltado cuando los datos de estado actual indican que el elemento de interfaz de usuario padre tiene enfoque.

[0008] Otras ventajas pueden resultar evidentes a partir de la siguiente descripción detallada cuando se toma en conjunción con los dibujos.

BREVE DESCRIPCIÓN DE LOS DIBUJOS

[0009] La presente tecnología se ilustra a modo de ejemplo y no está limitada a las figuras adjuntas en las que números de referencia similares indican elementos similares y en las que:

[00010] La Fig. 1 es un diagrama de bloques que muestra un ejemplo de configuración de componentes que pueden usarse para proporcionar pre-niños en

un árbol de interfaz de usuario (UI) de un programa de aplicación, de acuerdo con una o más implementaciones de ejemplo.

[0011] Las Figs. 2A-2C son representaciones de una parte de ejemplo de un árbol de interfaz de usuario que tiene un padre, pre-niño y niño (figura 2A), y que muestra cómo los elementos del árbol dibujan sin enfoque (figura 2B) y con enfoque (figura 2C)), de acuerdo con una o más implementaciones de ejemplo.

[0012] Las Figs. 3 y 4 comprenden un diagrama de flujo que muestra etapas de ejemplo que se pueden tomar por un elemento de la UI cuando se invoca, que incluye invocar a pre-niños y niños, de acuerdo con una o más implementaciones de ejemplo.

[0013] Las Figs. 5 y 6 comprenden un diagrama de flujo que muestra ejemplos de etapas alternativas que se pueden tomar por un elemento de la UI cuando son invocados para invocar selectivamente a pre-niños y niños sobre la base de datos de estado, de acuerdo con una o más implementaciones de ejemplo.

[0014] La Fig. 7 es una representación de una parte de un árbol de interfaz de usuario de ejemplo en el que los pre-niños dibujan o no dibujan con base en los datos del estado actual, de acuerdo con una o más implementaciones de ejemplo.

[0015] Las Figs. 8A, 8B, 9A, 9B y 10 son representaciones de ejemplo de cómo la parte de árbol de ejemplo de la Fig. 7 se puede dibujar por elementos de la UI de pre-niños, el elemento de la UI padre y elementos de la UI de niños, que incluye con base en datos de estado actual, de acuerdo con una o más implementaciones de ejemplo.

[0016] La Fig. 11 es un diagrama de bloques que representa un entorno de cómputo de ejemplo en el que se pueden incorporar aspectos de la materia objeto de la presente invención.

DESCRIPCIÓN DETALLADA

[0017] La tecnología descrita en el presente documento está dirigida a tener un elemento padre con uno o más elementos niños, denominado elemento "pre-niño" o elementos pre-niños, que dibujan antes de que el elemento padre dibuje, a diferencia de los niños convencionales que se dibujan después de que el padre dibuja. Como se comprenderá, la tecnología pre-niño proporciona elementos de interfaz de usuario desacoplados en el árbol de interfaz de usuario que representan diferentes propiedades visuales, tales como un resaltado de enfoque o similar, que dibujan antes de que el padre dibuje y, de este modo, proporcionan la propiedad componible. Observe que, tal como se usa en el presente documento, un "resalte" como "dibujado" puede ser cualquier representación visible, audible y/o táctil de salida, tal como una forma bidimensional (e.g., cualquier polígono) que transporta un cierto estado de un elemento de interfaz de usuario a un usuario, e.g., enfocado, no enfocado, suspendido, seleccionado y así sucesivamente; (se describen aquí varios estados de ejemplo).

[0018] En general, tal como se describe en el presente documento, un elemento de interfaz de usuario padre puede mantener dos conjuntos de niños, es decir, un conjunto (que contiene uno o más pre-niños) que dibuja antes de que el elemento de interfaz de usuario padre dibuje y otro conjunto más (que contiene uno o más niños) que dibujan después de que el elemento de interfaz de usuario padre dibuja. Como resultado, un elemento de interfaz de usuario padre se puede hacer a sí mismo después de (sobre) un pre-niño. Por ejemplo, mientras está en

foco, el elemento de interfaz de usuario padre (rectángulo azul) puede traducirse sobre un elemento pre-niño (ovalito naranja) para resaltar el estado enfocado; sin embargo, debido a que el pre-niño es componible, el elemento oval naranja se puede reemplazar por otro elemento, e.g., un círculo rojo, cambiando el pre-niño en el árbol de interfaz de usuario. Aunque no esté enfocado, el "pre-niño del estado de enfoque" no se dibuja (o se dibuja como invisible).

[0019] Además, se puede usar un pre-niño diferente en otro estado. Por ejemplo, se puede dibujar un pre-niño de "estado suspendido" diferente antes de que el botón padre dibuje cuando el botón padre esté en el estado suspendido (e.g., un puntero del ratón está suspendido sobre él). Alternativamente, el mismo pre-niño se puede dibujar de manera diferente en diferentes estados, e.g., no del todo o invisible cuando no está enfocado o suspendido, como un triángulo verde cuando está suspendido, pero no enfocado, como un óvalo anaranjado cuando se enfoca y como un triángulo verde debajo de un óvalo anaranjado cuando está enfocado y suspendido. Tenga en cuenta que un pre-niño puede tener sus propios pre-niños y/o niños.

[0020] Se debe entender que cualquiera de los ejemplos de la presente invención, no son limitantes. Por ejemplo, aunque JavaScript® se usa en algunos de los ejemplos, se puede usar cualquier lenguaje de programación adecuado. Como otro ejemplo, se ejemplifican en este documento estados diferentes de un elemento de interfaz de usuario tales como enfocado y suspendido, sin embargo, pueden existir otros estados tales como seleccionados, presionados, escuchando para entrada de audio o gesto y así sucesivamente con respecto a un elemento de interfaz de usuario y, por lo tanto, se pueden beneficiar de la tecnología pre-niño. Como tal, la tecnología descrita aquí no está limitada a ninguna implementación particular, aspectos, conceptos, estructuras, funcionalidades o ejemplos descritos

en el presente documento. Más bien, cualquiera de las implementaciones, aspectos, conceptos, estructuras, funcionalidades o ejemplos descritos en el presente documento, no son limitantes y la presente tecnología se puede utilizar de diversas maneras que proporcionan beneficios y ventajas en la informática y en la tecnología de interfaz de usuario en general.

[0021] La Fig. 1 es un diagrama de bloques que muestra componentes de ejemplo que pueden usar tecnología pre-niño. En la Fig. 1, un programa de aplicación 102 incluye un árbol de interfaz de usuario 104. Como se conoce con un árbol de interfaz de usuario convencional, el árbol de interfaz de usuario se recorre en orden descendente según sea apropiado para representar elementos de interfaz de usuario en marcos de representación. Sin embargo, como se comprenderá, un contenedor padre puede tener dibujos pre-niños antes de que el contenedor primario dibuje y, por lo tanto, el recorrido es descendente, pero el dibujo (orden z) no es estrictamente descendente puesto que un padre puede invocar su pre-niño (inferior) para dibujar antes de que el padre dibuje.

[0022] Como se muestra en la Fig. 1, el árbol de interfaz de usuario 104 incluye un elemento padre superior 106 y un elemento padre 108 con sus propios niños, como se describe a continuación. Cada uno de los niños puede ser un padre de su propio niño o niños (no se muestra explícitamente en la figura 1).

[0023] El padre superior 106 puede ser un padre más alto, o puede ser uno (de posiblemente muchos) padres intervinientes (no mostrados explícitamente en la figura 1) entre el padre más alto y el elemento padre 108 que está representado en la Fig. 1. El elemento padre más alto en una o más implementaciones es una vista hospedadora que aloja al menos otra "vista" que comprende un elemento de interfaz de usuario en forma de un objeto. El padre superior 106, ya sea el padre

más alto o un padre interviniente por debajo del padre superior, se puede traducir a sí mismo como un elemento de interfaz de usuario, e.g., un contenedor del elemento de interfaz de usuario padre 108.

[0024] En una o más implementaciones, el elemento de interfaz de usuario padre comprende un objeto de vista que tiene código 110, datos generales 112 (e.g., datos de estado actual) y datos de estilo 114 que comprenden valores de propiedad, tales como posición de elemento, color de fondo, fuente, tamaño de fuente, color de fuente, datos de animación y así sucesivamente. Entre otras operaciones, el código 110 permite que el padre se dibuje a sí mismo, se comunique con su padre superior 106 y se comunique con sus niños.

[0025] El elemento de interfaz de usuario padre 108 ejemplificado en la Fig. 1 incluye pre-niños que comprenden elementos de interfaz de usuario pre-niño 116 (1) - 116 (m), y elementos de interfaz de usuario niño 118 (1) - 118 (n). Como se describe en este documento, el padre tiene el control de dibujar a sus pre-niños y niños, sin embargo, debido a que los pre-niños y los niños están desacoplados para ser componibles, el padre no necesariamente sabe nada de lo que sus pre-niños o niños harán cuando se invoque.

[0026] En general, y como se entiende, un sistema de entrada 120 o similar proporciona entrada al programa de aplicación 102. Ejemplos no limitantes de entrada pueden incluir entrada de teclado (e.g., entrada de teclado de computador, entrada de controlador de juego, entrada para dispositivo de control remoto), dispositivo de puntero (e.g., entrada del ratón, entrada táctil), entrada de gesto, entrada de observación de ojos, entrada de pista, entrada de voz y así sucesivamente. Como se sabe también, algunos de los datos de entrada (e.g. entrada de voz o entrada de gesto) se pueden convertir (e.g., por el sistema de

entrada 120 que incluye cualquier motor de reconocimiento acoplado a ellos) en comandos o similares antes de ser enviados al programa de aplicación 102. En general, la entrada afecta el árbol de nodos y posiblemente el estado de cada nodo, e.g., para permitir que un usuario navegue entre los elementos del árbol, e.g., para cambiar el enfoque/entrada directa a un elemento en particular, seleccionar un elemento, suspender sobre un elemento, desplazarse para ver nuevos elementos y así sucesivamente.

[0027] Tal como se utiliza aquí, el nodo que representa el elemento de interfaz de usuario en el árbol, el elemento de interfaz de usuario en sí mismo (e.g., un objeto) y la representación dibujada/representada de ese elemento de interfaz de usuario son generalmente intercambiables. Si no se especifica, el contexto del uso de dicho término se puede utilizar para distinguir entre ellos.

[0028] Como se entiende también, el programa de aplicación 102 atraviesa al menos parte del árbol a través de un sistema de representación de aplicaciones o similar, e.g., para construir marcos de representación para la salida. La salida puede ser a un dispositivo o dispositivos de salida 122 adecuados, e.g., una pantalla de dispositivo interno, una pantalla acoplada externa tal como un monitor de PC, un monitor remoto, un altavoz, un mecanismo de vibración y/o similar.

[0029] Para organizar la interfaz de usuario en una estructura de árbol, se puede especificar un padre con niños convencionales "post-dibujados" (cero o más) como:

```
Clase Elemento UI {  
    Padre Elemento UI;  
    Lista <Elemento UI> niños;  
}
```

La lista puede estar vacía o no especificada si no hay niños para un elemento padre determinado.

[0030] Tal como se describe en el presente documento, se puede especificar un padre con niños pre-niños (cero o más) y niños convencionales (cero o más) como:

```
Clase Elemento UI {
    Padre Elemento UI;
    Lista <Elemento UI> pre-niños;
    Lista <Elemento UI> niños;
}
```

Al igual que con la lista de niños, la lista pre-niños puede estar vacía (o no especificada) para un elemento padre dado si no hay pre-niños.

[0031] Para dibujar el árbol desde el nodo raíz, se puede usar una función recursiva tal como se ejemplifica a continuación:

```
Función dibujoArbol (Elemento UI raíz) {
    Paracada (pre-niño en raiz.pre-niños) {
        DibujoArbol (pre-niño);
    }
    Raiz.dibujo ();
    Paracada (niño en raíz.niños) {
        DibujoArbol (niño);
    }
}
```

[0032] Como se puede ver, en lugar de tener cada dibujo del elemento, entonces cada uno de sus elementos niños dibuja en orden, recursivamente, la

recursividad es la misma, sin embargo, para cualquier padre dado que dibuja, cada uno de los pre-niños dibuja antes que el padre y cada uno de los niños dibuja después que el padre.

[0033] En la Fig. 2A se muestra un ejemplo en árbol parcial, con los elementos dibujados representados en las Figs. 2B y 2C. La Fig. 2A muestra un elemento padre 220 que representa un rectángulo con esquinas redondeadas y un borde, que tiene un elemento pre-niño 222 y un elemento niño 224. El padre típicamente está asociado con mucha más información de estilo (apariencia u otra), como color de fondo, datos de frontera (e.g., borde blanco, cuatro píxeles), fuente, tamaño de fuente, datos de posición de texto y así sucesivamente, sin embargo, esto no se muestra en la FIG. 2A para fines de simplicidad. De manera similar, el pre-niño 222 y el niño 224 pueden tener datos de estilo adicionales, e.g., el pre-niño está configurado para dibujar un óvalo de cierto tamaño, sombreado con un patrón de relleno de línea diagonal, pero puede tener muchas otras apariencias u otros valores de propiedad en su conjunto de estilos.

[0034] En el ejemplo de las Figs. 2A - 2C, el pre-niño 222 sólo es aplicable cuando el elemento padre tiene foco. Como se describe en este documento, la aplicabilidad o no se puede llevar a cabo de diversas maneras. Una forma es que el padre 220 informe a su pre-niño 222 del estado actual del padre, con lo cual el pre-niño 222 opera de acuerdo con el estado actual, e.g., dibuja un óvalo si está enfocado, o no dibuja nada (o dibuja de forma invisible) si no enfocado. Otra forma de determinar la aplicabilidad es tener un padre organizado con categorías de pre-niños que corresponden al estado o estados actuales del padre, e.g., el padre puede tener un subconjunto de uno o más pre-niños de "enfoque" que solo se llaman si el padre tiene foco, un subconjunto de uno o más pre-niños "suspendido" que sólo se llaman si el padre está actualmente suspendido sobre, un subconjunto

de uno o más pre-niños "seleccionados" que sólo se llaman si el padre está seleccionado, y así sucesivamente.

[0035] El elemento niño 224 mostrado en la Fig. 2A es un elemento de texto que tiene el texto "Click Me" aunque como se entiende, este niño es componible y por lo tanto puede ser cambiado en el árbol en cualquier momento a un elemento diferente. Además, el niño puede estar dispuesto para recuperar su texto desde otra fuente, tal como la red, de manera que, por ejemplo, el texto puede cambiar en cualquier momento en que el elemento 224 se vuelva a dibujar si cambian los datos de la fuente detrás del texto.

[0036] En el ejemplo simplificado de las Figs. 2A - 2C, la Fig. 2B representa el botón cuando está dibujado en el estado no enfocado, es decir, como el botón traducido 226 solo, sin que se dibuje el óvalo del pre-niño. La Fig. 2C representa el botón cuando se dibuja en el estado enfocado, concretamente como el botón 228 resaltado enfocado traducido, con el resalte procedente del óvalo del pre-niño que se está dibujando antes del elemento de botón y el elemento niño de texto. Al igual que con los niños, este pre-niño es componible y por lo tanto se puede cambiar en el árbol en cualquier momento a un elemento diferente, e.g., para dibujar un resalte hexagonal debajo del padre en lugar de un resaltado oval. Además, el pre-niño se puede disponer para recuperar datos de otra fuente tal como la red para obtener al menos parte de lo dibujado, (aunque probablemente esto no se hará cuando se usa únicamente para fines de realce).

[0037] Como se puede ver en el ejemplo anterior en el que el pre-niño sólo dibujó como resalte cuando el padre se enfocó, cualquier elemento que incluya un elemento pre-niño puede depender del estado. Tenga en cuenta que a medida que se representa esta parte del árbol, un elemento de interfaz de usuario padre

de nivel superior (no mostrado) borra el resaltado cuando dibuja, de modo que el resaltado no continúe apareciendo si cambia el enfoque.

[0038] El elemento de interfaz de usuario pre-niño puede ser un objeto (un objeto de vista) codificado con lógica que obtiene datos de estado del padre y, a continuación, puede basar su decisión de dibujo en los datos de estado actuales. La forma en que un elemento de interfaz de usuario (que incluye cualquier pre-niño) aparece o se comporta (e.g., anima) para el usuario depende de un conjunto de valores de propiedad denominado estilo del elemento.

[0039] En una o más implementaciones, un estilo comprende así un conjunto de valores de propiedad tales como el color, el tamaño de la fuente, la fuente, la posición del niño y los datos de tamaño, los datos de animación y así sucesivamente. Un estilo puede estar asociado con cada vista en el momento de carga y puede especificar comportamientos con base en el estado, para no volver a calcular el árbol de la interfaz de usuario para eventos frecuentes, como cambios de enfoque, por ejemplo.

[0040] Un estilo puede estar asociado con un elemento de interfaz de usuario de cualquier manera, tal como codificado en el elemento, o aplicado cuando sea necesario. Por ejemplo, en una o más implementaciones, un sistema de estilo flexible permite procesar el mismo elemento de interfaz de usuario de manera diferente en diferentes circunstancias (e.g., en diferentes plataformas o incluso en la misma plataforma para una presentación de tamaño de tableta frente a una pantalla de diez pies). Dicho sistema de estilo proporciona flexibilidad, ya que un conjunto de valores visuales y de otra propiedad pueden estar asociado con un objeto de vista en una situación (tal como cuando se ejecuta en una plataforma de teléfono inteligente) y un conjunto diferente de valores visuales y de otra propiedad

asociados con el mismo objeto de vista en otra situación (como cuando se ejecuta en un sistema de juegos y entretenimiento).

[0041] Independientemente de cómo se diseña un elemento de interfaz de usuario, sus propiedades de estilo pueden depender del estado. Con este fin, además de propiedades como el color y el tamaño, los estilos son capaces de especificar transiciones, tales como, por ejemplo, una fuente puede tener un tamaño cuando se enfoca y otro tamaño cuando no está enfocado. Tales transiciones estilizadas también permiten animaciones, e.g., un elemento de interfaz de usuario puede animar a mover y/o fundirse en un menú cuando el usuario entra por primera vez en ese menú (e.g., usando las propiedades de estilo entrantes que cambian la posición y/u opacidad del elemento sobre múltiples marcos), y luego animar para mover y/o desaparecer del menú (usando propiedades similares de estilo de salida) cuando el usuario realiza una acción que hace que el elemento salga de ese menú.

[0042] En una o más implementaciones, un elemento de interfaz de usuario de objeto de vista mantiene estados, que incluyen un estado de "entrada" cuando se mueve hacia (y una vez se mueve hacia) una región visible y un estado de "salida" cuando se mueve fuera de la región visible. Un estilo puede especificar diferentes datos para usar en cada estado, que incluye datos de animación de estilo; por ejemplo, en el estado de entrada, una vista se puede desvanecer en una región visible por transición (sobre una pluralidad de tramas de representación) a su valor de opacidad para que se vuelva menos transparente hasta que esté completamente sólida y posteriormente desaparezca de una región visible en el estado de salida por inversión de esta transición de opacidad. Las propiedades de estilo de la vista pueden incluir valores para el estado de entrada, incluidos los datos de animación (e.g., los valores de opacidad inicial y final y el número de

marcos de representación u otro valor de tiempo correspondiente), como las propiedades de estilo para el estado de salida.

[0043] Una vista también puede mantener datos relacionados con el estilo (e.g., con base en un conjunto de indicadores booleanos) que comprenden información adicional con respecto al estado interactivo actual de la vista. Por ejemplo, los valores de indicador pueden indicar cuándo una vista es: 1) enfocada dentro de su alcance de entrada, 2) suspendido sobre (la vista tiene entrada de puntero sobre ella), 3) escuchando (la vista está escuchando para entradas de comandos tal como entrada de voz y/o posiblemente entrada de gesto), 4) seleccionada (la vista está activa/invocada/activada) y/o 5) pulsada (la vista tiene un puntero capturado o el botón de invocación presionado). Cada uno de estos estados de vista puede tener asociado un subconjunto de valor de propiedad de estilo, e.g., de manera que, como en el ejemplo anterior, la fuente de una vista puede tener un tamaño cuando se enfoca y otro tamaño cuando no se enfoca, etc. De este modo, aunque un estilo en una vista puede ser estático porque los valores de propiedad que se utilizan para una vista asociada se pueden fijar para una sesión de tiempo de ejecución, el conjunto de propiedades de estilo es dinámico con respecto a qué subconjunto de los valores de propiedad se puede aplicar para la vista en cualquier momento, con base en un estado actual dentro de los estados variables de una vista.

[0044] Se entiende por lo tanto que un elemento de interfaz de usuario pre-niño puede utilizar datos de estado que establecen cómo dibujará el pre-niño. De hecho, una decisión directa es si dibujar todo, por ejemplo, en la Fig. 2B, el estado no estaba enfocado, de modo que el pre-niño 222 (Fig. 2A) no dibujo en absoluto (o dibujo de forma invisible). En la Fig. 2C, el pre-niño 222 (fig. 2A) hizo el dibujo. Aunque no se muestra, se entiende que un pre-niño puede tener un conjunto más

complejo de valores de propiedad, e.g., dibujar un óvalo blanco cuando no está enfocado, o dibujar un óvalo anaranjado cuando se enfoca. De hecho, un pre-niño puede ser un objeto de vista y, por tanto, hacer cualquier cosa que un objeto de vista pueda hacer, que incluye posiblemente producir animaciones relativamente elaboradas, audio, retroalimentación táctil, etc. (Aunque se debe señalar que un padre puede anular a sus niños hasta cierto punto, como para tener todos sus niños en transición dentro y fuera de un contenedor de la misma manera, independientemente de lo que especifique el valor de la propiedad niño).

[0045] Las Figs. 3 y 4 comprenden un diagrama de flujo que muestra etapas de ejemplo que se pueden tomar con respecto a dibujar cualquier elemento de interfaz de usuario pre-niño, el elemento de interfaz de usuario padre y, a continuación, cualquier elemento de interfaz de usuario niño. Se entiende que las Figs. 3 y 4 se pueden aplicar a cualquier elemento de interfaz de usuario padre dado, y que cada elemento de interfaz de usuario pre-niño y elemento niño (que también puede ser un padre) también pueden contener una lógica similar a la de las Figs. 3 y 4; es decir, la lógica puede operar recursivamente.

[0046] La etapa 302 representa la recepción de cualquier información de estado desde un contenedor padre de nivel superior. Observe que el contenedor de nivel más alto, e.g., un hospedador de vista, puede obtener cualquier información de estado del programa de aplicación y, de este modo, proporcionar dicha información al siguiente nivel inferior, si procede. Además, el hospedador de vistas puede ejecutar una lógica similar a la de las Figs. 3 y 4.

[0047] La etapa 304 evalúa si el padre que ha sido invocado para dibujar (durante el recorrido del árbol) tiene cualquier pre-niño. Si es así, las etapas 306, 308, 310 y 312 dan a cada pre-niño la oportunidad de dibujar. Tenga en cuenta

que con base en la información de estado pasada en la etapa 308, un pre-niño no puede dibujar, e.g., un pre-niño puede dibujar si los datos de estado indican que el padre tiene enfoque, pero de otra manera no dibuja.

[0048] Cuando se ha dado a cualquier pre-niño la oportunidad de dibujar, la etapa 314 evalúa si el padre mismo dibuja. Por ejemplo, considere que el padre es en sí mismo un pre-niño (o un niño) que dibuja o no en función de los datos del estado actual. Si el padre se dibuja, se ejecuta la etapa 316. De lo contrario, se evita esta etapa, y el proceso continúa a la Fig. 4.

[0049] La etapa 402 de la Fig. 4 evalúa si hay cualquier niño que pueda dibujar. Las etapas 404, 406, 408 y 410 representan dar a cada niño la oportunidad de dibujar. Como se puede ver, las Figs. 3 y 4, por lo tanto, tienen cualquier dibujo de pre-niño como sea apropiado, a continuación, el elemento de interfaz de usuario padre dibuja, si procede, luego, cualquier niño dibuja como sea apropiado.

[0050] Se debe observar que las Figs. 3 y 4 proporcionan flexibilidad significativa a un diseñador, sin embargo, hay alternativas a la lógica de ejemplo establecida en el mismo. Por ejemplo, puede existir una regla tal que, a menos que un padre se aplique a los datos del estado actual, entonces no se aplica a pre-niños o niños. En tal situación, se entiende que se puede usar una etapa similar a la etapa 314 (e.g., directamente después de la etapa 302) para saltarse el resto de la lógica y finalizar si el padre no se aplica con base en los datos de estado actual. De manera similar, se puede aplicar una regla tal que, a menos que un padre en sí mismo aplique a los datos del estado actual, entonces no se aplican los pre-niños (pero se pueden aplicar niños regulares), por lo que se puede usar una etapa similar a la etapa 314 después de la etapa 302 para saltarse a la Fig. 4. Como otra

alternativa, se puede establecer una regla tal que, a menos que un padre se aplique a los datos del estado actual, entonces los pre-niños pueden aplicar (pero los niños regulares no pueden), por lo que algo similar a la rama "no" de la etapa 314 puede terminar la lógica en lugar de continuar con la Fig. 4.

[0051] Como otra consideración, se debe observar que un padre puede escoger dibujar solamente un subconjunto de sus niños. Por ejemplo, considere que un contenedor tiene cientos de elementos y, por lo tanto, sólo se puede presentar un subconjunto a la vez (e.g., veinte), con desplazamiento utilizado para que el usuario vea otros. En tal caso, los niños (y posiblemente los pre-niños) se pueden dibujar selectivamente, e.g., con base en la técnica de virtualización, sólo los elementos que son visibles necesitan ser instanciados y dibujados para conservar los recursos.

[0052] Las Figs. 5 y 6 son diagramas de flujo alternativos que son similares a las Figs. 3 y 4, sin embargo, en las Figs. 5 y 6, el padre sabe cuáles de sus pre-niños y niños se aplican a un estado dado. Como se describió anteriormente, una alternativa factible es que un padre tenga un subconjunto de uno o más pre-niños de "enfoque" que sólo se invocan si el padre tiene un enfoque, un subconjunto de uno o más pre-niños "seleccionados" que son sólo llamados si el padre está seleccionado, y así sucesivamente. Por ejemplo:

```

Clase Elemento UI {
    Padre Elemento UI;
    Lista <Elemento UI> PreNiños;
    Lista <Elemento UI> EnfoquepreNiños;
    Lista <Elemento UI> Seleccionado pre-Niños;
    Lista <Elemento UI> Niños;
    Lista <Elemento UI> EnfoqueNiños;

```

```
Lista <Elemento UI> SeleccionadoNiños;  
}
```

[0053] En esta alternativa, un padre puede tener cero o más pre-niños "generales" que dibujan independientemente del estado, cero o más pre-niños foco que el padre invoca solamente para dibujar si en el estado enfocado, y/o cero o más pre-niños seleccionados que el padre sólo invoca para dibujar si está en el estado seleccionado. Los niños regulares pueden tener una división similar con base en el estado. Observe que puede haber otros estados, algunos de los cuales se ejemplifican en el presente documento, e.g., suspendido, escuchando, presionado, etc.

[0054] Así, si la etapa 502 de la Fig. 5 determina que hay al menos un pre-niño, las etapas 504, 506 y 508 invocan selectivamente sólo a los pre-niños que coinciden con el estado o estados actuales, o están listados como pre-niños "generales" (no dependientes del estado). Tenga en cuenta que, si se invoca en la etapa 508, un pre-niño también puede usar datos de estado para tomar decisiones relacionadas con el dibujo. Las etapas 510 y 512 repiten el proceso para cualquier otro pre-niño.

[0055] La etapa 514 representa el padre que se dibuja en sí mismo. Tenga en cuenta que, en esta alternativa, el padre (que también es un niño a un contenedor superior) sólo se invoca para dibujar si coincide con el estado actual, o aparece como un elemento "general" que dibuja independientemente del estado y, por lo tanto, típicamente dibuja. Sin embargo, como se ha expuesto anteriormente, una vez invocado cualquier elemento también puede usar información de estado para tomar decisión(es), que incluye si se debe dibujar en absoluto.

[0056] La Fig. 6 es similar a la Fig. 5 con respecto a los niños post-dibujo. La etapa 602 de la Fig. 6 determina que hay al menos un niño y, de ser así, las etapas 604, 606 y 608 dibujan selectivamente solamente a aquellos niños que coinciden con el estado o estados actuales o se especifican como niños "generales" (no dependientes del estado) a ser invocados para dibujar. Al igual que ocurre con los pre-niños, si un niño es invocado en la etapa 608, el niño puede usar los datos del estado para tomar decisiones relacionadas con el dibujo. Las etapas 610 y 612 repiten el proceso para cualquier otro niño.

[0057] Observe que en la alternativa de las Figs. 5 y 6, debido a la invocación selectiva, se puede invocar un elemento de pre-niño o niño, solo si fue invocado por su padre. Como se puede apreciar fácilmente, esto es menos flexible que en las Figs. 3 y 4, en los que se puede invocar a los niños y los pre-niños, incluso si el propio padre no aplica al estado actual.

[0058] Volviendo a un ejemplo de árbol y representaciones traducidas con base en el estado que puede resultar, a modo de ejemplo, considerar que un diseñador quiere que se dibuje lo siguiente para un botón con base en ciertos estados, según sea apropiado; (en este ejemplo, un botón sólo puede estar en el estado seleccionado si también está en el estado enfocado y no puede estar tanto en el estado suspendido como en el estado seleccionado al mismo tiempo):

Ninguno suspendido, seleccionado ni enfocado = botón de dibujo

Suspendido = botón de dibujo sobre el triángulo

Enfocado (no suspendido) = botón de dibujo sobre el ovalo

Suspendido y enfocado = botón de dibujo sobre el ovalo sobre el triángulo

Seleccionado y enfocado = botón de dibujo sobre ovalo;

tanto el botón como el oval se animan a una posición de desplazamiento

(de $X + 10$, $Y + 10$).

[0059] La Fig. 7 es una parte de un árbol de interfaz de usuario de ejemplo 770 que cumple con las especificaciones de diseñador anteriores. El árbol de la interfaz de usuario de ejemplo 770 (parte) incluye un elemento de botón padre 771 que tiene dos pre-niños 772 y 773 y dos niños 774 y 775. Como se comprenderá, los pre-niños 772 y 773 proporcionan las propiedades con bases en el estado deseado e.g., dibujar (o no) con base en un estado enfocado, estado suspendido y/o estado seleccionado (o ninguno de estos estados).

[0060] En este ejemplo, el pre-niño "si se enfoca" 773 tiene en sí un pre-niño de nivel inferior 777 y un niño 778. Observe que los nodos representados de la Fig. 7 no tienen la intención de mostrar un conjunto completo de datos de estilo y no pretenden mostrar el código/lógica real que puede estar presente, sino que contienen un lenguaje simple como "si se enfocan" o "si están suspendidos, no enfocados" para transmitir cualquier dependencia de niño o pre-niño en el estado o estados actuales.

[0061] Se considera que el botón representado por el elemento de botón padre 771 no está enfocado actualmente (y por lo tanto tampoco seleccionado), y no está actualmente suspendido sobre. Cuando se procesa el árbol, no se aplican pre-niños en este estado y, por lo tanto, el nodo del botón dibuja, pero el elemento del botón no tiene ningún pre-niño aplicable que dibujó antes que el padre. De este modo, el elemento de botón 771 se dibuja con su conjunto de propiedades, y luego sus niños 774 y 775 dibujan, dando como resultado la representación de la Fig. 8A. Por ejemplo, traducido dentro de algo del contenedor padre de más alto nivel 880. Como se ve, el elemento de botón padre 771 dibuja un rectángulo 881 con un borde y esquinas redondeadas, el elemento niño 774 dibuja el texto "XYZ" 884 y se dibuja una imagen 885, Por ejemplo, recuperados a través de una URL especificada por el elemento niño 775.

[0062] Considere que, en algún otro momento, el botón se está suspendido sobre (pero no está enfocado). En este ejemplo, el elemento pre-niño 772 se aplica a este estado y, por lo tanto, el elemento pre-niño 772, con base en su conjunto de propiedades, dibuja el triángulo 882 (con borde) especificado como se representa en la Fig. 8B, después de lo cual se dibuja el botón 881 con su conjunto de propiedades (mediante el elemento de botón padre 771), después de lo cual los niños del botón 774 y 775 se dibujan con sus conjuntos de propiedades respectivas, dando como resultado el texto 884 y la imagen 885 respectivamente.

[0063] En algún momento posterior, el botón tiene enfoque, pero en este momento no está suspendido sobre. Por lo tanto, se aplica el pre-niño 773, que en general dibuja un óvalo 883 (con borde, como se muestra en las Figs. 9A y 9B). Tenga en cuenta que antes que se dibuje el óvalo 883, el pre-niño de nivel inferior 777 del pre-niño 773 no dibuja, porque en este momento el botón no está suspendido sobre, en este ejemplo. Además, después de que el óvalo 883 es dibujado, el niño de nivel inferior 778 del pre-niño 773 no dibuja, porque el botón tampoco está seleccionado. El elemento de botón padre 771 dibuja entonces su rectángulo, luego sus niños 774 y 775 dibujan su texto e imagen, respectivamente. Por ejemplo, el resultado puede aparecer así en la representación de la Fig. 9A,

[0064] En otra situación, el botón tiene enfoque y también se encuentra actualmente suspendido sobre. Por lo tanto, el pre-niño 772 no se aplica (debido al enfoque), pero se aplica el pre-niño 773, que en general dibuja un óvalo 883. Sin embargo, antes que se dibuje el óvalo, el pre-niño de nivel inferior 777 del pre-niño 773 dibuja un triángulo invertido 889 (180°) (Fig. 9B), porque el botón está suspendido sobre, en este estado de ejemplo. Además, después que se dibuja el óvalo, el niño de nivel inferior 778 del pre-niño 773 no dibuja, porque el botón no está seleccionado. El elemento de botón padre 771 dibuja a continuación, sobre el

ovalo 883 que se dibujó sobre el triángulo invertido 889, entonces sus niños 774 y 775 dibujan su texto 884 e imagen 885, respectivamente. Por ejemplo, el resultado puede aparecer así en la representación de la Fig. 9B.

[0065] En otro ejemplo, considere que el botón está en el estado seleccionado (que en este ejemplo significa también enfocado, pero no suspendido). Por lo tanto, se aplica el pre-niño 773, y su niño de nivel inferior 778 (porque está seleccionado), pero el pre-niño de nivel inferior 777 no se aplica (porque no está suspendido sobre). En este ejemplo, el padre y sus niños se dibujan después de un ovalo 883(a), en la Fig. 10 se dibuja por el pre-niño de nivel inferior 778 del pre-niño 773. El pre-niño 778 especifica una animación para mover el óvalo a un desplazamiento de $X + 10$, $Y + 10$ en un segundo y medio, mostrado como el ovalo 883(b). De forma similar, el elemento padre del botón tiene los mismos parámetros de animación cuando se selecciona, y así el botón y sus niños "se mueven" con el óvalo. Tenga en cuenta que, en una implementación típica, el pre-niño configura los parámetros de animación, pero no completa la animación antes de volver, de modo que la animación del botón padre y la animación pre-niño están sincronizadas (sustancial o exactamente) "movimientos" interpolados en medio segundo del número de marcos de representación en este ejemplo. El resultado, por ejemplo, puede aparecer, como en la representación de la Fig. 10, donde las líneas discontinuas representan la posición inicial de los diversos elementos (sólo 881(a) y 883(a) son visibles en el ejemplo de la Fig. 10), y las líneas continuas representan una posición posterior animada, e.g., al final de la animación que muestra las posiciones posteriores de los elementos traducidos 881(b), 883(b), 884(b) y 885(b).

[0066] Como se puede ver, la tecnología descrita en el presente documento utiliza pre-niños en un árbol de interfaz de usuario de una manera que permite

desacoplar elementos de interfaz de usuario de elementos de interfaz de usuario que representan a niños, permitiendo así expresar elegantemente interfaces de usuario componibles. Los pre-niños se pueden usar para representar un resaltado (e.g., un resaltado de enfoque) que dibuja antes de que un elemento padre dibuje, sin embargo, la propiedad del árbol sigue siendo componible.

[0067] Uno o más aspectos están dirigidos hacia la invocación de un elemento de interfaz de usuario pre-niño de un árbol de interfaz de usuario, en el que el pre-niño incluye una lógica pre-niño para dibujar una representación de un elemento de interfaz de usuario pre-niño en un primer momento. El uno o más aspectos incluyen dibujar un elemento de interfaz de usuario padre en una segunda vez que es posterior a la primera vez. Un elemento de interfaz de usuario niño se puede invocar para dibujar una representación del elemento de interfaz de usuario niño en una tercera vez que es posterior a la segunda vez

[0068] La lógica pre-niño puede determinar si se debe dibujar la representación del elemento de interfaz de usuario pre-niño, que incluye la obtención de datos de estado en el elemento de interfaz de usuario pre-niño y determinando si se dibuja la representación del elemento de interfaz de usuario con base en los datos de estado. La representación del elemento de interfaz de usuario pre-niño se puede dibujar como un resaltado en la primera vez en una posición que aparece debajo del elemento padre una vez que se dibuja la representación del elemento padre en la segunda vez.

[0069] Invocar el elemento de interfaz de usuario pre-niño se puede basar en que el elemento de interfaz de usuario padre determine que el elemento de interfaz de usuario pre-niño se debe invocar con base en datos de estado.

[0070] Dibujar la representación del elemento de interfaz de usuario niño, del elemento de interfaz de usuario pre-niño, puede incluir animar la representación del elemento de interfaz de usuario niño sobre una pluralidad de tramas de representación.

[0071] El elemento de interfaz de usuario pre-niño puede tener un elemento de interfaz de usuario pre-niño de nivel inferior en el árbol de interfaz de usuario. Si es así, se puede invocar el elemento de interfaz de usuario pre-niño inferior. El elemento de interfaz de usuario pre-niño puede tener un elemento de interfaz de usuario niño de nivel inferior en el árbol de interfaz de usuario y, de ser así, se puede invocar el elemento de interfaz de usuario niño de nivel inferior.

[0072] Uno o más aspectos están dirigidos hacia un programa de aplicación que atraviesa al menos parte del árbol de interfaz de usuario para **hacer representaciones** de elementos de interfaz de usuario correspondientes a nodos del árbol de interfaz de usuario. Esto incluye recorrer al menos parte del árbol con base en la entrada de un sistema de entrada acoplado al programa de aplicación que tiene representaciones de nodo de al menos parte del árbol representado en un dispositivo de salida. Un nodo padre en el árbol de la interfaz de usuario corresponde a un elemento de interfaz de usuario padre que cuando se representa aparece como una representación del nodo padre en el dispositivo de salida. Un nodo pre-niño está asociado con el nodo padre y corresponde a un elemento de interfaz de usuario pre-niño que cuando se representa aparece como una representación del nodo pre-niño en el dispositivo de salida. Cuando el nodo pre-niño dibuja, el nodo pre-niño dibuja antes de que el nodo padre dibuje.

[0073] El nodo pre-niño puede corresponder a un objeto invocado que contiene una lógica pre-niño que determina si se debe dibujar con base en datos de estado

asociados con el nodo padre. Los datos de estado pueden corresponder a un estado enfocado, un estado suspendido, un estado de escucha, un estado seleccionado y/o un estado presionado.

[0074] El nodo pre-niño puede estar asociado con un conjunto de valores de propiedad de estilo que se utilizan para determinar si se dibuja la representación del nodo pre-niño cuando se invoca y, en caso afirmativo, cómo se dibuja la representación del nodo pre-niño. La representación del nodo pre-niño puede ser una forma que aparece como resaltado en relación con la representación del nodo padre.

[0075] El nodo pre-niño puede tener un nodo pre-niño de nivel inferior en el árbol de interfaz de usuario que, si se invoca, dibuja antes del nodo pre-niño. El nodo pre-niño puede tener un nodo niño de nivel inferior en el árbol de la interfaz de usuario que, si se invoca, dibuja después del nodo pre-niño y antes del nodo padre.

[0076] Uno o más aspectos están dirigidos hacia la invocación de un elemento de interfaz de usuario pre-niño, que incluye proporcionar datos de estado al elemento de interfaz de usuario pre-niño. En el presente documento se describe determinar si se debe dibujar una representación del elemento de interfaz de usuario pre-niño con base en los datos de estado y, si es así, dibujar una representación del elemento de interfaz de usuario pre-niño. También se describe aquí dibujar una representación del elemento de interfaz de usuario padre después de dibujar la representación del elemento de interfaz de usuario pre-niño, invocar un elemento de interfaz de usuario niño y dibujar una representación del elemento de interfaz de usuario niño después de dibujar la representación del elemento de interfaz de usuario padre.

[0077] El elemento de interfaz de usuario pre-niño puede tener un elemento de interfaz de usuario pre-niño de nivel inferior; se puede hacer una determinación de si se debe dibujar una representación del elemento de interfaz de usuario pre-niño de nivel inferior con base en los datos de estado y, en caso afirmativo, dibujar una representación del elemento de interfaz de usuario pre-niño de nivel inferior antes de dibujar la representación del elemento de interfaz de usuario pre-niño.

[0078] Puede haber al menos otro elemento de interfaz de usuario pre-niño del elemento padre. Si es así, aquí se describe invocar cada otro elemento de interfaz de usuario pre-niño, que incluye proporcionar datos de estado al mismo.

[0079] Dibujar la representación del elemento de interfaz de usuario pre-niño puede comprender dibujar un resaltado que es relativo a la representación del elemento de interfaz de usuario padre. Dibujar la representación del elemento de interfaz de usuario pre-niño puede comprender animar la representación del elemento de interfaz de usuario pre-niño junto con animar la representación del elemento de interfaz de usuario padre.

EJEMPLO DISPOSITIVO DE COMPUTACIÓN

[0080] Las técnicas descritas en el presente documento se pueden aplicar a cualquier dispositivo o conjunto de dispositivos (máquinas) capaces de ejecutar programas y procesos. Se puede entender, por lo tanto, que se contemplan computadores personales, computadores portátiles, dispositivos portátiles, portátiles y otros dispositivos informáticos y objetos informáticos de todo tipo, que incluye teléfonos celulares, computadores de tableta/pizarra, consolas de juegos y entretenimiento y similares para uso en conexión con diversas implementaciones que incluye aquellas ejemplificadas en este documento. Por consiguiente, el

mecanismo de cómputo de propósito general descrito más adelante en la Fig. 11 es sólo un ejemplo de un dispositivo informático.

[0081] Las implementaciones se pueden efectuar parcialmente a través de un sistema operativo, para uso por un desarrollador de servicios para un dispositivo u objeto, y/o incluidas en el software de aplicación que opera para realizar uno o más aspectos funcionales de las diversas implementaciones descritas en el presente documento. El software se puede describir en el contexto general de las instrucciones ejecutables por computador, tales como módulos de programa, que son ejecutados por uno o más computadores, tales como estaciones de trabajo cliente, servidores u otros dispositivos. Los versados en el arte apreciarán que los sistemas informáticos tienen una variedad de configuraciones y protocolos que se pueden utilizar para comunicar datos y, por lo tanto, ninguna configuración o protocolo particular se considera limitante.

[0082] La Fig. 11 ilustra, por tanto, un ejemplo de un entorno de sistema informático adecuado 1100 en el que uno o varios aspectos de las implementaciones descritas en el presente documento se pueden efectuar, aunque como se ha explicado anteriormente, el entorno del sistema de cómputo 1100 es sólo un ejemplo de un entorno informático adecuado y no pretende sugerir ninguna limitación en cuanto al alcance de uso o funcionalidad. Además, el entorno del sistema informático 1100 no se pretende que sea interpretado como que tiene una dependencia relativa a cualquier componente o combinación de componentes ilustrada en el entorno de sistema de cómputo 1100 de ejemplo.

[0083] Haciendo referencia a la Fig. 11, un dispositivo de ejemplo para efectuar una o más implementaciones incluye un dispositivo informático de propósito general en forma de un computador 1110. Los componentes del

computador 1110 pueden incluir, pero no se limitan a, una unidad de procesamiento 1120, una memoria del sistema 1130 y un bus de sistema 1122 que acopla varios componentes de sistema que incluyen la memoria del sistema para la unidad de procesamiento 1120.

[0084] El computador 1110 incluye típicamente una variedad de medios legibles de máquina (e.g., computador) y puede ser cualquier medio disponible al que pueda acceder una máquina tal como el computador 1110. La memoria del sistema 1130 puede incluir medios de almacenamiento de computador en la forma de memoria volátil y/o no volátil, tal como memoria de sólo lectura (ROM) y/o memoria de acceso aleatorio (RAM), y medios de disco duro, medios de almacenamiento ópticos, medios de flash y así sucesivamente; como se usa en el presente documento, medios de almacenamiento legibles por máquina/legibles por computador almacenan datos que no incluyen señales transitorias (aunque se puede en otros tipos de medios legibles por máquina/legibles por computador que no son medios de almacenamiento). A modo de ejemplo, y no limitante la memoria del sistema 1130 también puede incluir un sistema operativo, programas de aplicación, otros módulos de programa y datos de programa.

[0085] Un usuario puede introducir comandos e información en el computador 1110 mediante uno o más dispositivos de entrada 1140. Un monitor u otro tipo de dispositivo de visualización está también conectado al bus de sistema 1122 a través de una interfaz, tal como interfaz de salida 1150. Además de un monitor, los computadores también pueden incluir otros dispositivos de salida periféricos, como altavoces y una impresora, que se pueden conectar a través de la interfaz de salida 1150.

[0086] El computador 1110 puede funcionar en un entorno en red o distribuido utilizando conexiones lógicas a uno o más computadores remotos, tal como el computador remoto 1170. El computador remoto 1170 puede ser un computador personal, un servidor, un enrutador, una red PC, un dispositivo parecido u otro nodo de red común, o cualquier otro dispositivo de consumo o transmisión de medios remoto, y puede incluir cualquiera o todos los elementos descritos anteriormente con relación al computador 1110. Las conexiones lógicas representadas en la Fig. 11 incluyen una red 1172, tal como una red de área local (LAN) o una red de área extensa (WAN), pero también pueden incluir otras redes/buses. Tales entornos de red son comunes en hogares, oficinas, redes informáticas de toda la empresa, intranets e internet.

[0087] Como se mencionó anteriormente, aunque se han descrito implementaciones de ejemplo en conexión con diversos dispositivos de cómputo y arquitecturas de red, los conceptos subyacentes se pueden aplicar a cualquier sistema de red y cualquier dispositivo o sistema informático en el que sea deseable implementar dicha tecnología.

[0088] También hay múltiples formas de implementar la misma o similar funcionalidad, e.g., una API apropiada, un conjunto de herramientas, un código de controlador, un sistema operativo, un control, un objeto de software independiente o descargable, etc., que permite que las aplicaciones y los servicios tomen ventaja de las técnicas aquí proporcionadas. De este modo, las implementaciones de la presente invención se contemplan desde el punto de vista de una API (u otro objeto de software), así como de un objeto de software o hardware que efectúa una o más implementaciones como se describe en el presente documento. De este modo, varias implementaciones descritas aquí pueden tener aspectos que

están completamente en hardware, en parte en hardware y parcialmente en software, así como completamente en software.

[0089] La palabra "ejemplo" se usa aquí para significar servir como un ejemplo, instancia o ilustración. Para evitar dudas, la materia objeto de la presente invención no está limitada por tales ejemplos. Además, cualquier aspecto o diseño descrito en el presente documento como "ejemplo" no se debe interpretar necesariamente como preferente o ventajoso con respecto a otros aspectos o diseños, ni pretende excluir estructuras de ejemplo equivalentes y técnicas conocidas por los versados en el arte. Además, en la medida en que los términos "incluye", "tiene", "contiene" y otras palabras similares se usan, para evitar dudas, dichos términos tienen la intención de ser inclusivos de una manera similar al término "que comprende" como una palabra de transición abierta sin excluir cualquier elemento adicional u otros elementos cuando se emplea en una reivindicación.

[0090] Como se ha mencionado, las diversas técnicas descritas aquí se pueden implementar en conexión con hardware o software o, cuando sea apropiado, con una combinación de ambos. Tal como se utiliza en el presente documento, los términos "componente", "módulo", "sistema" y similares se refieren asimismo a una entidad relacionada con el computador, hardware, una combinación de hardware y software, software o software en ejecución. Por ejemplo, un componente puede ser, pero no está limitado a ser, un proceso que se ejecuta en un procesador, un procesador, un objeto, un ejecutable, un subproceso de ejecución, un programa y/o un computador. A modo de ilustración, tanto una aplicación que se ejecuta en un computador y el computador pueden ser un componente. Uno o más componentes pueden residir dentro de un proceso y/o

subproceso de ejecución y un componente se puede localizar en un computador y/o distribuir entre dos o más computadores.

[0091] Los sistemas mencionados anteriormente se han descrito con respecto a la interacción entre varios componentes. Se puede apreciar que tales sistemas y componentes pueden incluir aquellos componentes o subcomponentes especificados, algunos de los componentes o subcomponentes especificados, y/o componentes adicionales, y según diversas permutaciones y combinaciones de los anteriores. Los subcomponentes también se pueden implementar como componentes acoplados comunicativamente a otros componentes en lugar de incluirse en componentes primarios (jerárquicos). Además, se puede observar que uno o más componentes se pueden combinar en un único componente que proporciona funcionalidad agregada o dividida en varios subcomponentes separados, y que se puede proporcionar una o más capas intermedias, tal como una capa de gestión, se pueden proporcionar acoplados comunicativamente a tales subcomponentes para proporcionar una funcionalidad integrada. Cualquier componente descrito en el presente documento también puede interactuar con uno o más de otros componentes no descritos específicamente en el presente documento, pero generalmente conocidos por los versados en el arte.

[0092] A la vista de los sistemas de ejemplo descritos en el presente documento, también se pueden apreciar metodologías que se pueden implementar de acuerdo con la materia descrita, con referencia a los diagramas de flujo/diagramas de flujo de las diversas figuras. Aunque con fines de simplicidad de explicación, las metodologías se muestran y describen como una serie de bloques, se debe entender y apreciar que las diversas implementaciones no están limitadas por el orden de los bloques, ya que algunos bloques pueden ocurrir en órdenes diferentes y/o concurrentemente con otros bloques de lo que se representa y

describe en el presente documento. Cuando se ilustra un flujo no secuencial o ramificado a través de diagramas de flujo/diagramas de flujo, se puede apreciar que se pueden implementar otras diversas ramas, trayectorias de flujo y órdenes de los bloques que consiguen el mismo resultado o un resultado similar. Además, algunos bloques ilustrados son opcionales en la implementación de las metodologías aquí descritas.

CONCLUSIÓN

[0093] Aunque la invención es susceptible de diversas modificaciones y construcciones alternativas, ciertas implementaciones ilustradas de esta se muestran en los dibujos y han sido descritas en detalle anteriormente. Se debe entender, sin embargo, que no hay intención de limitar la invención a las formas específicas descritas, sino que, por el contrario, se pretende cubrir todas las modificaciones, construcciones alternativas y equivalentes que caen dentro del espíritu y alcance de la invención.

[0094] Además de las diversas implementaciones descritas en el presente documento, se debe entender que se pueden utilizar otras implementaciones similares o se pueden hacer modificaciones y adiciones a la implementación o ejecuciones descritas para llevar a cabo la misma función o función equivalente de la implementación correspondiente(s) sin desviarse de ello. Además, múltiples chips de procesamiento o dispositivos múltiples pueden compartir el rendimiento de una o más funciones descritas en el presente documento, y de manera similar, el almacenamiento se puede efectuar a través de una pluralidad de dispositivos. En consecuencia, la invención no se limita a ninguna implementación única, sino que se debe interpretar en amplitud, espíritu y alcance de acuerdo con las reivindicaciones adjuntas.