РОССИЙСКАЯ ФЕДЕРАЦИЯ

(19) **RU** (11) **2 569 802**(13) **C2**

(51) МПК
G06F 3/0481 (2013.01)

ФЕДЕРАЛЬНАЯ СЛУЖБА
ПО ИНТЕЛЛЕКТУАЛЬНОЙ СОБСТВЕННОСТИ

(12) ОПИСАНИЕ ИЗОБРЕТЕНИЯ К ПАТЕНТУ

(21)(22) Заявка: **2010135599/08, 25.08.2005**

(24) Дата начала отсчета срока действия патента:
25.08.2005

Приоритет(ы):
(30) Конвенционный приоритет:
22.04.2005 US 11/111,983
Номер и дата приоритета первоначальной заявки, из которой данная заявка выделена:
2007138968 22.04.2005

(43) Дата публикации заявки: **27.02.2012** Бюл. № **6**

(45) Опубликовано: **27.11.2015** Бюл. № **33**

(56) Список документов, цитированных в отчете о поиске: **US 2001/0028368 A1, 11.10.2001. US 2002/0170058 A1, 14.11.2002. US 66006105 B1, 12.09.2003. US 6724403 B1, 20.04.2004. RU 2202859 C2, 20.04.2003.**

Адрес для переписки:
129090, Москва, ул. Б. Спасская, 25, строение 3, ООО "Юридическая фирма Городисский и Партнеры"

(72) Автор(ы):
ШЕХТЕР Грег (US),
САКС Джеван (US),
ФОРТЬЕР Крис (US)

(73) Патентообладатель(и):
МАЙКРОСОФТ ТЕКНОЛОДЖИ ЛАЙСЕНСИНГ, ЭлЭлСи (US)

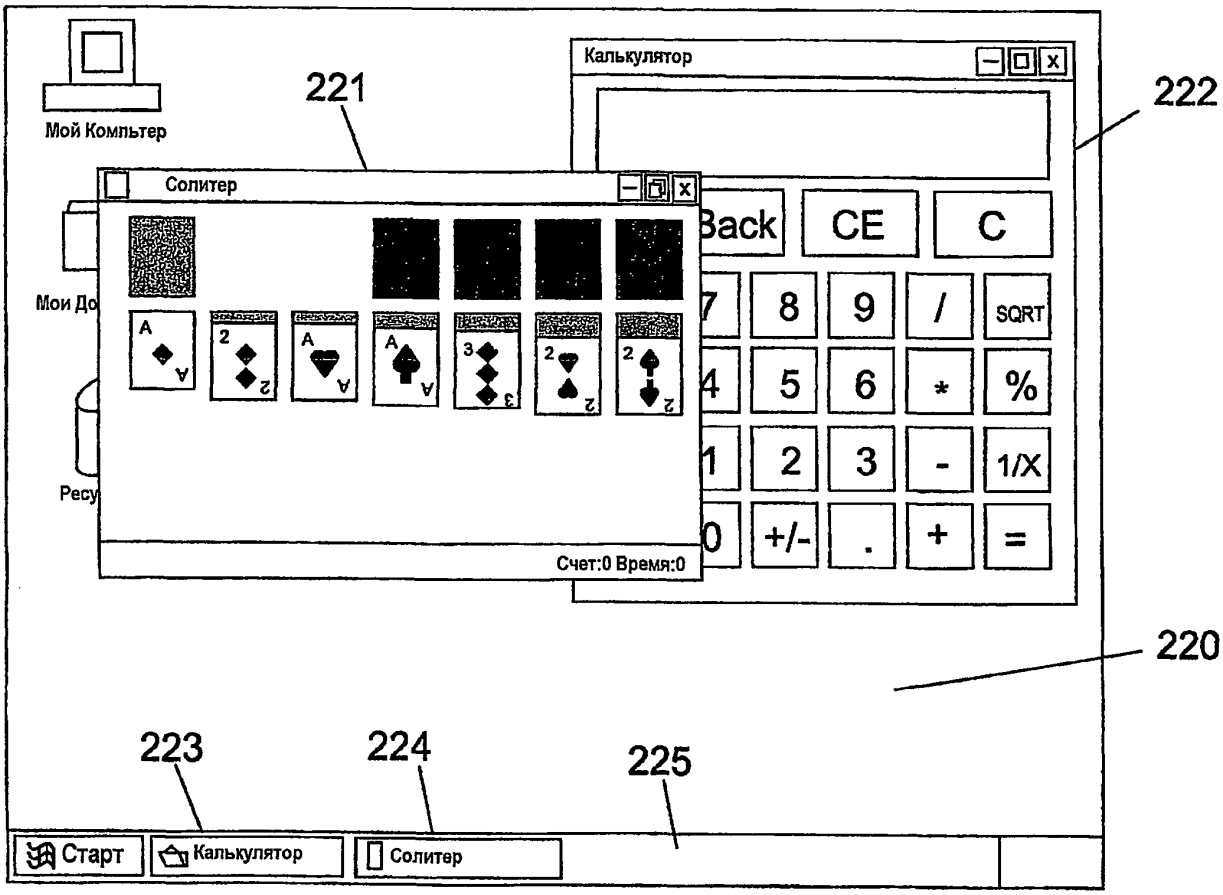
RU 2569802 C2

(54) ИНТЕРФЕЙС И СИСТЕМА ДЛЯ МАНИПУЛЯЦИИ ПИКТОГРАММАМИ АКТИВНЫХ ОКОН В АДМИНИСТРАТОРЕ ОКОН

(57) Реферат:

Изобретение относится к средствам манипулирования пиктограммами в администраторе окон. Технический результат заключается в уменьшении времени нахождения необходимого окна приложения. Отображают исходное окно, при этом исходное окно содержит контент исходного окна, который способен модифицироваться. Отображают целевое окно, причем по меньшей мере часть целевого окна содержит пиктограмму, причем пиктограмма включает в себя контент пиктограммы, соответствующий по меньшей мере части контента исходного окна. Модифицируют контент пиктограммы, когда контент исходного окна изменяется. Получают первый параметр для определения по меньшей мере части исходного окна, причем упомянутая модификация содержит определение по меньшей мере части исходного окна, которая должна быть нарисована в пиктограмме, на основе упомянутого первого параметра. Получают второй параметр для определения по меньшей части целевого окна, причем упомянутая модификация дополнительно содержит определение по меньшей мере части целевого окна для визуализации пиктограммы на основе упомянутого второго параметра. Модификация исходного окна отображается в пиктограмме. 2 н. и 4 з.п. ф-лы, 21 ил.

RU 2569802 C2



Фиг.2В

RU 2569802 C2

RU 2569802 C2

RUSSIAN FEDERATION

FEDERAL SERVICE
FOR INTELLECTUAL PROPERTY

(19) **RU** (11) **2 569 802**(13) **C2**

(51) Int. Cl.
G06F 3/0481 (2013.01)

(12) ABSTRACT OF INVENTION

(21)(22) Application: **2010135599/08, 25.08.2005**

(24) Effective date for property rights:
25.08.2005

Priority:
(30) Convention priority:
22.04.2005 US 11/111,983
Number and date of priority of the initial application, from which the given application is allocated:
2007138968 22.04.2005

(43) Application published: **27.02.2012** Bull. № **6**

(45) Date of publication: **27.11.2015** Bull. № **33**

Mail address:
129090, Moskva, ul. B. Spasskaja, 25, stroenie 3, OOO "Juridicheskaja firma Gorodisskij i Partnery"

(72) Inventor(s):
ShEKhTER Greg (US),
SAKS Dzhevan (US),
FORT'ER Kris (US)

(73) Proprietor(s):
MAJKROSOFT TEKNOLODZhI LAJSENSING, EhlEhlSi (US)

RU 2569802 C2

(54) INTERFACE AND SYSTEM FOR MANIPULATING THUMBNAILS OF ACTIVE WINDOWS IN WINDOW MANAGER

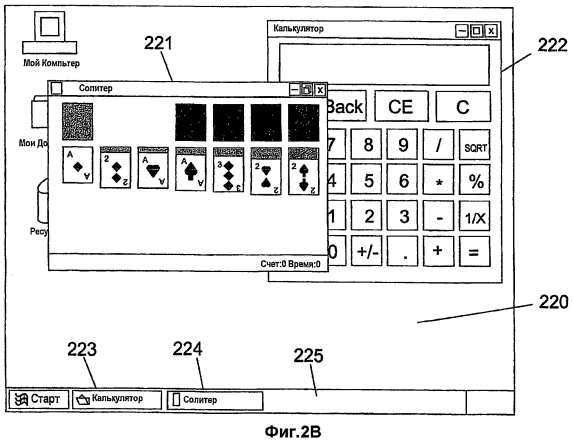
(57) Abstract:

FIELD: physics, computer engineering.

SUBSTANCE: invention relates to means of manipulating thumbnails in a window manager. The method comprises displaying a source window, the source window containing source window content which is capable of being modified; displaying a destination window, wherein at least a portion of the destination window contains a thumbnail, the thumbnail containing content thumbnail content corresponding to at least a portion of the source window content; modifying the thumbnail content when the source window content changes; obtaining a first parameter for defining at least a portion of the source window, wherein said modification comprises determining at least a portion of the source window to draw into the thumbnail based on said first parameter; obtaining a second parameter for defining at least a portion of the destination window, wherein said modification further comprises determining at least a portion of the destination window to render the thumbnail based on said second parameter. Modification of the source window is displayed in the thumbnail.

EFFECT: shorter time for finding the right application window.

6 cl, 21 dwg



Фиг.2В

RU 2569802 C2

ОБЛАСТЬ ТЕХНИКИ, К КОТОРОЙ ОТНОСИТСЯ ИЗОБРЕТЕНИЕ

[1] Аспекты настоящего изобретения относятся к пиктограммам окон и, в частности, к интерфейсу и системе манипуляции пиктограммами в администраторе окон.

УРОВЕНЬ ТЕХНИКИ

[2] Компьютерные системы отображают множество окон, в которых пользователи могут вводить данные, манипулировать данными или активизировать процедуры. При одновременной работе более одного приложения на дисплее компьютера может быть отображено несколько окон, при этом каждое окно соответствует приложению. Также для любого одного приложения может существовать множество окон. Например, если пользователь вводит текст в программу обработки текста, при этом работая в программе электронной таблицы, на дисплее могут быть открыты два окна. Одно из окон представляет собой окно приложения обработки текста, а второе окно представляет собой окно программы электронной таблицы. Если пользователь дополнительно рассматривает видео в приложении медиа-плейера, дополнительно присутствует окно, соответствующее приложению медиа-плейера. С увеличением количества активных приложений, увеличивается и количество окон, которые отображаются на дисплее компьютера.

[3] Пользователь, который одновременно использует множество приложений, часто сталкивается с наличием на дисплее множества окон, что является причиной переполненности рабочего стола, вызывающей замешательство и раздражение от беспорядка, увиденного на дисплее. Например, на дисплее может быть такое большое количество перекрывающихся окон, что пользователю, возможно, нужно будет тратить какое-то время на нахождение желаемого окна всякий раз, когда пользователь хочет завершить работу в другом приложении. С целью облегчения этой проблемы пользователь может минимизировать размер окна или выйти из всех приложений для уменьшения беспорядка (и количества окон на дисплее). Однако если окна минимизированы, то пользователь не имеет непосредственного доступа к соответствующему приложению. Если в определенном приложении, в котором окно было минимизировано, должно быть произведено действие, пользователь должен сначала определить местонахождение желаемого окна и открыть желаемое окно после определения его местоположения. Этот процесс отнимает много времени. Аналогично, если пользователь выходит из всех приложений, чтобы очистить перегруженность рабочего стола, то приложение больше не будет активным. В этом случае, пользователь должен повторно запустить приложение для завершения желаемой задачи, и потратить еще больше времени. Кроме того, если приложение выполняет текущую функцию, эта функция не будет выполняться в то время, когда приложение не является активным.

[4] Обычно, когда окно приложения минимизировано, кнопка панели задач может отображаться на панели задач для указания того, что приложение является активным. Хотя кнопка панели задач может указывать на то, что приложение является активным, часто на кнопке панели задач имеется только иконка или название приложения. Не имея дополнительной информации, пользователю надо будет открыть окно, чтобы рассмотреть его контент. Однако, открывая и закрывая окно просто, чтобы проверить контент окна (например, чтобы установить идентичность окна), пользователь тратит время и силы, которые приводят к снижению эффективности и производительности. Эта проблема усложняется с увеличением количества активных приложений и количества окон, поскольку пользователь должен открывать и закрывать множество окон, чтобы найти желаемое.

[5] Таким образом, в данной области техники существует необходимость в системе

и способе предоставления удобного программного доступа для манипуляции пиктограммами окон или приложений и визуализации пиктограмм с тем, чтобы обеспечить гибкость при отображении пиктограмм и контента пиктограммы. Также, в данной области техники существует необходимость в интерфейсе программирования и способе реализации интерфейса программирования для эффективного управления и реализации окон приложений и пиктограмм и показа окон и пиктограмм гибким способом.

РАСКРЫТИЕ ИЗОБРЕТЕНИЯ

[6] Аспекты настоящего изобретения предоставляют интерфейс, систему или способ для отображения окна на дисплее, таким образом решая одну или несколько проблем, описанных выше. Также предоставляется динамическая пиктограмма, соответствующая окну приложения, в котором модификации контента в окне приложения отображаются в соответствующей динамической пиктограмме. Модификация контента окна приложения дополнительно может быть отражена в соответствующей динамической пиктограмме в реальном масштабе времени.

КРАТКОЕ ОПИСАНИЕ ЧЕРТЕЖЕЙ

[7] На Фиг. 1А показан пример системы реализации изобретения, которая включает в себя вычислительное устройство общего назначения в виде компьютера.

[8] На Фиг.1В-1М показана компьютерная среда общего назначения, поддерживающая один или несколько аспектов настоящего изобретения.

[9] Фиг.2А представляет собой диаграмму в виде дерева графа сцены или дисплея, иллюстрирующую пример компонентов дисплея.

[10] На Фиг.2В показаны компоненты, отображенные на дисплее в соответствии с Фиг.2А.

[11] Фиг.3А представляет собой диаграмму в виде дерева графа сцены или дисплея, показывающую примеры компонентов отображения на дисплее настоящего изобретения.

[12] На Фиг.3В показано отображение на дисплее, соответствующее Фиг.3А.

[13] На Фиг.4А и 4В показан один из примеров применения настоящего изобретения при обеспечении пиктограмм окон приложений.

[14] На Фиг.5 показано другое приложение пиктограммы настоящего изобретения, в котором пиктограммы отображены в меню ALT-TAB.

[15] Фиг.6 представляет собой блок-схему последовательности операций, иллюстрирующую пример процесса предоставления пиктограмм.

ОСУЩЕСТВЛЕНИЕ ИЗОБРЕТЕНИЯ

[16] В помощь читателю документ разделен на секции. Эти секции включают в себя следующие: Краткий обзор; Вычислительная Среда Общего Назначения; Пиктограммы.

[17] Необходимо отметить, что между элементами в нижеприведенном описании существуют разнообразные связи. Необходимо отметить, что эти связи в общем случае, если не определено иное, могут быть прямыми или косвенными, и что настоящее описание не следует рассматривать в качестве ограничивающего в этом отношении.

КРАТКИЙ ОБЗОР

А. Статичные пиктограммы

[18] В одном из подходов пиктограмма может быть ассоциирована с приложением. При помощи пиктограммы может быть показана небольшая по размеру версия окна приложения для указания на исходный контент окна приложения, что позволяет пользователю примерно идентифицировать контент окна. Однако эти пиктограммы могут не предоставлять обновленный контент окна приложения. Таким образом, если в окне приложения были сделаны модификации, пиктограмма может быть устаревшей

и не может обеспечить надлежащее изображение фактического контента окна приложения. Таким образом, пользователю сложно должным образом идентифицировать окно желаемого приложения без необходимости открыть само окно приложения.

[19] Кроме того, пиктограммы не обладают гибкостью в терминах отображения контента, размещения, изменения размера и/или эффектов. Пиктограмма окна приложения обеспечивает статичное изображение исходного контента окна приложения. Однако, мало того что контент остается статичным и потенциально устаревшим, но также и особенности, и размещение пиктограммы может быть задано и может не изменяться в любой момент времени в зависимости от потребностей пользователя.

[20] Компьютерные операционные системы используют уровень программного обеспечения, ответственный за управление объектами пользовательского интерфейса, такие как пиктограммы, и предоставления услуг пользовательского интерфейса для приложений программного обеспечения. Уровень программного обеспечения может называться администратором окон рабочего стола (DWM). Логика визуализации, направление входных событий и интерфейсы прикладного программирования (API) администратора окон рабочего стола (DWM) все вместе воплощают пользовательскую политику интерфейса, которая, в свою очередь, определяет полное восприятие пользователем операционной системы. При визуализации рабочего стола DWM обычно обрабатывает обновления экрана, такие как перемещение окон, путем координации отрисовки непосредственно на экране дисплея. Следовательно, если перекрывающее окно закрывают или отодвигают, заново отображенное расположенное снизу окно или рабочий стол должно быть перерисовано. Это замедляет процесс, при котором напрасно расходуются время и ресурсы. Также, если на дисплее окна нарисованы непосредственно в конкретном положении и в конкретной конфигурации, как это обычно бывает, соответствующее изображение окна устанавливается в конкретном положении и конфигурации. Модификации положения или конфигурации окна потребуют большого расхода ресурсов для перерисовки изображения.

B. Динамические пиктограммы

[21] В другом аспекте настоящего изобретения предоставлены интерфейс, система и способ отображения пиктограммы на дисплее, в которых администратор окон рабочего стола (DWM) обеспечивает интерфейс прикладного программирования (API), регистрирующий исходное окно и целевое окно. В этом примере целевое окно представляет собой окно, в котором может быть визуализирована пиктограмма. Пиктограмма, визуализированная в пределах целевого окна, может представлять, по меньшей мере, часть соответствующего исходного окна. Следовательно, пиктограмма может представлять ассоциацию между исходным окном и целевым окном путем предоставления, по меньшей мере, части исходного окна, нарисованного в, по меньшей мере, части целевого окна. DWM может получить дескриптор окна (HWND), соответствующий исходному окну, и HWND, соответствующий целевому окну. Приложение может дополнительно получить дескриптор, представляющий регистрацию пиктограммы, соответствующей исходному окну.

[22] В другом аспекте настоящего изобретения предоставлены интерфейс, система и способ отображения пиктограммы на дисплее, в которых администратор окон рабочего стола (DWM) обновляет или модифицирует пиктограмму, ассоциированную с соответствующим исходным окном. В этом примере DWM может принимать данные, определяющие размер целевого окна, область исходного окна, которая должна быть нарисована в целевом окне в виде пиктограммы, непрозрачность пиктограммы,

видимость пиктограммы, специальные эффекты пиктограммы и т.д.

[23] Дополнительный аспект настоящего изобретения дополнительно предоставляет динамическую пиктограмму, содержащую экспортирование интерфейса прикладного программирования для регистрации дескриптора пиктограммы, поддержки списка регистраций пиктограмм, рисование пиктограммы или целевого окна на основе исходного окна, причем исходное окно может быть окном приложения, обновления свойства пиктограммы и отмены регистрации динамической пиктограммы. Кроме того, модификация в исходном окне может автоматически обрабатываться для выполнения соответствующей модификации в целевом окне.

[24] Необходимо отметить, что в нижеприведенном описании между элементами существуют разнообразные связи. Необходимо отметить, что эти связи в общем случае, если не определено иное, могут быть прямыми или косвенными и что настоящее описание не следует рассматривать в качестве ограничивающего в этом отношении.

[25] Аспекты настоящего изобретения предоставляют интерфейс прикладного программирования (API), систему и способ управления отображением элементов отображения, таких как окна, или другое представление окон, такое как пиктограммы на дисплее. В одном из примеров настоящего изобретения альтернативная версия окна приложения представляет собой более мелкую версию окна приложения и отображается на дисплее. Более мелкая версия окна приложения, которая отличается от исходного окна приложения, может быть дополнительно отображена в любом местоположении на дисплее. В качестве альтернативы, исходное окно приложения может быть минимизировано таким образом, что исходное окно приложения не будет отображаться до тех, пор пока на дисплее отображается более мелкая версия окна приложения. Кроме того, более мелкая версия окна приложения сама может быть минимизирована, например, если на дисплее необходимо дополнительное место. Хотя альтернативная версия окна приложения может представлять собой более мелкую версию приложения, как описано, необходимо отметить, что альтернативная версия окна приложения может быть больше исходного окна приложения или иметь любой другой размер в зависимости от потребностей пользователя.

[26] Например, альтернативная версия окна приложения может представлять собой пиктограмму окна приложения. Пиктограмма обычно представляет собой небольшое изображение или представление большего графического изображения, или части большего изображения, соответствующего окну приложения. Пиктограмма также может представлять ассоциацию между исходным окном, предоставляя источник или контент пиктограммы, и целевым окном, которое может представлять собой окно, в котором пиктограмма предоставляет контент исходного окна. Пиктограммы обеспечивают преимущество, заключающееся в способности отображать сразу нескольких окон в удобном формате, минимизируя использование места на дисплее. Также, пиктограммы могут использоваться для более легкого выбора и сортировки изображений и окон.

[27] В другом аспекте настоящего изобретения пиктограмма или альтернативная версия окна приложения может быть отображена на дисплее при необходимости. Например, зависание курсора над кнопкой панели задач может вызывать отображение пиктограммы окна приложения. В одном из примеров пиктограмма представляет собой динамическую пиктограмму и может изменяться при изменении исходного окна приложения. Например, изменения, сделанные в окне приложения, отображаются в динамической пиктограмме, соответствующей окну приложения. Такие изменения, например, могут отражаться в динамической пиктограмме в реальном масштабе

времени.

[28] В другом аспекте настоящего изобретения пиктограммы или динамические пиктограммы могут быть размещены в пределах другого окна, или целевого окна. Например, множество динамических пиктограмм из различных приложений могут быть включены в целевое окно таким образом, что целевое окно отображает пиктограммы из каждого из различных приложений. Кроме того, любая из динамических пиктограмм, отображаемая в целевом окне, может отражать текущий статус соответствующего окна приложения. В качестве альтернативы, пиктограмма может быть отображена в другой пиктограмме, или пиктограммы могут быть расположены каскадом таким образом, что первая пиктограмма может быть отображена во второй пиктограмме, которая, в свою очередь, может быть отображена в третьей пиктограмме и т.д.

[29] В другом аспекте настоящего изобретения предоставлены система и способ для подготовки окон без вывода на экран. Затем может быть выполнена композиция окон на экране, таким образом экономя компьютерные ресурсы. Кроме того, подготовка окон без вывода на экран и композиция окон на экране предоставляет возможность для функционирования пиктограммы, как описано в настоящем изобретении.

ВЫЧИСЛИТЕЛЬНАЯ СРЕДА ОБЩЕГО НАЗНАЧЕНИЯ

[30] На Фиг.1А показан пример подходящей вычислительной среды 100, в которой может быть осуществлено изобретение. Вычислительная среда 100 представляет собой только один из примеров подходящей вычислительной среды и не предназначена для любого ограничения объема использования или функциональных возможностей изобретения. Вычислительную среду 100 не следует интерпретировать как имеющую какую-либо зависимость или требования относительно любых компонентов или их комбинации, проиллюстрированных в иллюстративной операционной среде 100.

[31] Настоящее изобретение может функционировать в многочисленных других вычислительных средах или конфигурациях общего назначения или специального назначения. Примеры известных вычислительных систем, сред и/или конфигураций, которые могут быть подходящими для использования с изобретением, включают в себя без ограничений персональные компьютеры, серверные компьютеры, карманные или портативные устройства, мультипроцессорные системы, системы на основе микропроцессоров, программируемую бытовую электронику, PC сети, миникомпьютеры, мэйнфреймы, распределенную вычислительную среду, которая включает любую из вышеуказанных систем или устройств и т.п.

[32] как показано на Фиг.1А, иллюстративная система для реализации изобретения включает в себя вычислительное устройство общего назначения в виде компьютера 110. Компоненты компьютера 110 могут включать без ограничения блок 120 обработки, системную память 130 и системную шину 121, которая соединяет различные системные компоненты, включая системную память с блоком 120 обработки. Системная шина 121 может быть любой из нескольких типов шинных структур, включая шину памяти или контроллер памяти, периферийную шину и локальную шину с использованием любой из множества архитектур шин. В качестве неограничивающего примера такая архитектура включает в себя шину промышленной стандартной архитектуры (ISA), шину микроканальной архитектуры (MCA), шину расширенной ISA (EISA), локальную шину ассоциации по стандартам в области видеоэлектроники (VESA) и шину взаимодействия периферийных компонентов (PCI), также известную как шина расширения.

[33] Компьютер 110 обычно включает в себя множество читаемых компьютером

сред. Читаемая компьютером среда включает в себя энергозависимую и энергонезависимую среду, съемную и несъемную среду. В качестве неограничивающего примера читаемые компьютером среды могут содержать компьютерные носители информации и среду для передачи данных и включают в себя без ограничения RAM, ROM, EEPROM, флэш-память или другую технологию памяти, CD-ROM, цифровые универсальные диски (DVD) или другое оптическое дисковое устройство хранения, магнитные кассеты, магнитную ленту, магнитное дисковое устройство хранения или другие магнитные устройства хранения, или любую другую среду, которая может использоваться для хранения желаемой информации и которая может быть доступна компьютеру 110. Среды для передачи данных обычно реализуют читаемые компьютером инструкции, структуры данных, программные модули или другие данные в модулированном сигнале данных, таком как несущая, или другой механизм передачи, и включают в себя любые среды передачи информации. Термин "модулированный сигнал данных" означает сигнал, который имеет один или несколько наборов характеристик или измененный таким образом, чтобы кодировать информацию в сигнал. В качестве неограничивающего примера среда для передачи данных включают в себя проводные среды, такие как проводная сеть или непосредственное проводное соединение, и беспроводные среды, такие как акустические, радиочастотные, инфракрасные и другие беспроводные среды. Комбинации любых из вышеперечисленных сред также должны быть включены в объем читаемых компьютером сред.

[34] Системная память 130 включает в себя компьютерные носители информации в виде энергозависимой и/или энергонезависимой памяти, такой как постоянное запоминающее устройство 131 (ROM) и запоминающее устройство 132 с произвольным доступом (RAM). Базовая система 133 ввода/вывода (BIOS), содержащая основные процедуры, которые помогают передавать информацию между элементами в пределах компьютера 110, например, во время запуска, обычно хранится в ROM 131. RAM 132 обычно содержит данные и/или программные модули, которые являются непосредственно доступными и/или в настоящее время исполняются блоком 120 обработки. В качестве неограничивающего примера на Фиг.1 показана операционная система 134, прикладные программы 135, другие программные модули 136 и данные 137 программ.

[35] Компьютер 110 также может включать в себя другие съемные/несъемные, энергозависимые/энергонезависимые компьютерные носители информации. Только в качестве примера, на Фиг.1 показан жесткий диск 141, который считывает или записывает данные в несъемную, энергонезависимую магнитную среду, магнитный дисковод 151, который считывает или записывает данные на съемный, энергонезависимый магнитный диск 152, и оптический дисковод 155, который считывает или записывает данные на съемный, энергонезависимый оптический диск 156, такой как CD-ROM, или другие оптические среды. Другие съемные/несъемные, энергозависимые/энергонезависимые компьютерные носители информации, которые могут использоваться в иллюстративной операционной среде, включают без ограничения магнитные кассетные ленты, карты флэш-памяти, цифровые универсальные диски, цифровую видеоленту, твердотельную RAM, твердотельную ROM и т.п. Жесткий диск 141 обычно соединен с системной шиной 121 через интерфейс несъемной памяти, такой как интерфейс 140, и магнитный дисковод 151 и оптический дисковод 155 обычно соединены с системной шиной 121 интерфейсом съемной памяти, таким как интерфейс 150.

[36] Драйверы (дисководы) и ассоциированные с ними компьютерные носители

информации, описанные выше и показанные на Фиг.1А, обеспечивают хранение читаемых компьютером инструкций, структур данных, программных модулей и других данных для компьютера 110. На Фиг.1А, например, жесткий диск 141 показан как хранящий операционную систему 144, прикладные программы 145, другие программные модули 146 и данные 147 программ. Следует отметить, что эти компоненты могут быть теми же самыми или отличаться от операционной системы 134, прикладных программ 135, других программных модулей 136 и данных 137 программ. Операционной системе 144, прикладным программам 145, другим программным модулям 146 и данным 147 программ в настоящем описании присвоены другие ссылочные позиции, как минимум, для иллюстрации того, что они представляют собой различные копии. Пользователь может вводить команды и информацию в компьютер 110 через устройства ввода, такие как клавиатура 162 и указывающее устройство 161, обычно называемое мышью, трекболом или сенсорной панелью. Другие устройства ввода (не показаны) могут включать в себя микрофон, джойстик, игровую консоль, спутниковую антенну, сканер или т.п. Эти и другие устройства ввода часто соединены с блоком 120 обработки посредством пользовательского интерфейса 160 ввода, который соединен с системной шиной, но может быть соединен другим интерфейсом и структурами шин, такими как параллельный порт, игровой порт или универсальная последовательная шина (USB). Монитор 191 или другой тип устройства отображения также соединен с системной шиной 121 через интерфейс, такой как видео интерфейс 190. Помимо монитора компьютеры также могут включать в себя другие периферийные устройства вывода, такие как спикеры 197 и принтер 196, которые могут быть связаны посредством периферийного интерфейса 190 вывода.

[37] Компьютер 110 может работать в сетевой среде, используя логические соединения, с одним или несколькими удаленными компьютерами, такими как удаленный компьютер 180. Удаленный компьютер 180 может представлять собой персональный компьютер, сервер, маршрутизатор, сетевой PC, одноранговое устройство или другой общий сетевой узел, и обычно включает в себя многие или все элементы, описанные выше в отношении компьютера 110, хотя на Фиг.1А показано только устройство 181 памяти. Логические соединения, показанные на Фиг.1А, включают в себя локальную сеть (LAN) 171 и глобальную сеть (WAN) 173, но могут также включать в себя другие сети. Такие сетевые среды представляют собой обычные офисные, корпоративные компьютерные сети, интранет и Интернет.

[38] При использовании в сетевой среде LAN компьютер 110 соединен с LAN 171 через сетевой интерфейс или адаптер 170. При использовании в сетевой среды WAN компьютер 110 обычно включает в себя модем 172 или другое средство для установки связи с WAN 173, такой как Интернет. Модем 172, который может быть внутренним или внешним, может быть соединен с системной шиной 121 через пользовательский интерфейс 160 ввода или другой подходящий механизм. В сетевой среде программные модули, показанные в отношении компьютера 110, или их части, могут храниться в удаленном устройстве памяти. В качестве неограничивающего примера на Фиг.1А показаны удаленные прикладные программы 185 как постоянно хранящиеся в устройстве 181 памяти. Очевидно, что показанные сетевые соединения являются иллюстративными, и могут использоваться другие средства установки линий связи между компьютерами.

[39] Очевидно, что показанные сетевые соединения являются иллюстративными, и могут использоваться другие средства установки линий связи между компьютерами. Предполагается наличие любого из хорошо известных протоколов, таких как TCP/IP,

Ethernet, FTP, HTTP и т.п., и система может работать в конфигурации клиент-сервер, давая возможность пользователю извлекать веб-страницы из веб-сервера. Для отображения и манипуляции данными на веб-страницах может использоваться любой из различных обычных web-браузеров.

[40] Интерфейс программирования (или просто "интерфейс") может быть рассмотрен в качестве любого механизма, процесса, протокола для предоставления возможности одному или нескольким сегментам кода осуществлять связь или получать доступ к функциональным возможностям, предоставляемым одним или несколькими другими сегментами кода. В качестве альтернативы, интерфейс программирования может быть рассмотрен как один или несколько механизмов, способов, вызовов функции, модулей, объектов и т.д. системного компонента, способного к соединению для передачи данных с одним или несколькими механизмами, способами, вызовами функций, модулями и т.д. другого компонента(компонентов). Термин "сегмент кода" в предыдущем предложении охватывает одну или несколько инструкций или строк кода, и включает в себя, например, модули кода, объекты, подпрограммы, функции и т.д., независимо от применяемой терминологии или от того, скомпилированы ли сегменты кода отдельно, или предоставлены в качестве исходного кода, промежуточного кода или объектного кода, используются ли сегменты кода в во время выполнения системы или процесса, или расположены ли они на одной или различных машинах или распределены во множестве машин, или реализованы ли функциональные возможности, представленные сегментами кода, полностью в программном обеспечении, полностью в аппаратных средствах, или в виде комбинации аппаратных средств и программного обеспечения.

[41] Абстрактно, интерфейс программирования может быть рассмотрен в общем, как показано на Фиг.1B или Фиг.1C. На Фиг.1B показан интерфейс Интерфейс_1 в виде канала, через который осуществляется связь между первым и вторым сегментом кода. На Фиг.1C показан интерфейс как содержащий объекты интерфейсов I1 и I2 (которые могут быть или могут не быть частью первого и второго сегментов кода), что позволяет первому и второму сегментам кода системы осуществлять связь через среду M. С точки зрения Фиг.1C можно рассматривать объекты интерфейсов I1 и I2 как отдельные интерфейсы одной и той же системы, и можно считать, что объекты I1 и I2 плюс среда M составляют интерфейс. Хотя на Фиг.1B и 1C показан двунаправленный поток и интерфейсы с каждой стороны потока, конкретные реализации могут иметь поток информации только в одном направлении (или не иметь никакого потока информации, как описано ниже) или могут иметь объект интерфейса только на одной стороне. В качестве неограничивающего примера термины, такие как интерфейс прикладного программирования (API), точка ввода, метод, функция, подпрограмма, вызов удаленной процедуры и интерфейс объектной модели программных компонентов (COM) охвачены в пределах определения интерфейса программирования.

[42] Аспекты такого интерфейса программирования могут включать в себя метод, посредством которого первый сегмент кода передает информацию (где "информация" используется в своем самом широком смысле и включает данные, команды, запросы и т.д.) во второй сегмент кода; метод, посредством которого второй сегмент кода принимает информацию; и структура, последовательность, синтакс, организация, схема, таймирование и контент данной информации. В этом отношении, основная транспортная среда может не иметь значения для работы интерфейса, независимо от того, является ли среда проводной или беспроводной, или их комбинацией, при условии что информация передается таким образом, как определено интерфейсом. В определенных ситуациях информация может не передаваться в одном или обоих направлениях в обычном смысле,

поскольку передача информации может осуществляться либо при помощи другого механизма (например, информации, помещенной в буфер, файл и т.д., отдельно от потока информации между сегментами кода), либо не осуществляться, когда один сегмент кода просто получает доступ к функциональным возможностям, выполняемым вторым сегментом кода. В данной ситуации любые или все из этих аспектов могут иметь значение, например, в зависимости от того, являются ли сегменты кода частью системы в свободно соединенной или жестко соединенной конфигурации, и таким образом этот список следует рассматривать как иллюстративный и неограничивающий.

[43] Этот взгляд на интерфейс программирования известен специалистам в данной области техники и является очевидным из вышеприведенного подробного описания изобретения. Однако имеются другие способы реализации интерфейса программирования, и, если они не исключены в явном виде, они также охватываются в соответствии с формулой изобретения, приведенной в конце этого описания. Такие другие пути могут казаться более сложными или комплексными, чем упрощенное представление, показанное на Фиг.1B и 1C, но, тем не менее, они выполняют подобную функцию с целью достижения того же самого конечного результата. Ниже кратко описаны некоторые иллюстративные альтернативные реализации интерфейса программирования.

РАЗЛОЖЕНИЕ

[44] Осуществление связи одного сегмента кода с другим может осуществляться косвенно путем разбивки линии связи на множество дискретных линий связи. Это схематично изображено на Фиг.1D и 1E. Как показано, некоторые интерфейсы могут быть описаны в терминах делимых наборов функциональных возможностей. Таким образом, функциональные возможности интерфейса по Фиг.1B и 1C могут быть разложены для достижения того же результата, который, с математической точки зрения, может быть представлен как 24, или 2 по 2 по 2 по 3 раза. Соответственно, как показано на Фиг.1D, функция, предоставляемая интерфейсом Интерфейс_1, может быть подразделена таким образом, чтобы преобразовать линии связи интерфейса во множество интерфейсов: Интерфейс_1A, Интерфейс_1B, Интерфейс_1C и т.д., достигая того же самого результата. Как показано на Фиг.1E, функция, предоставляемая интерфейсом I1, может быть подразделена на множество интерфейсов I1a, I1b, I1c и т.д. для достижения того же самого результата. Аналогично, интерфейс I2 второго сегмента кода, который принимает информацию от первого сегмента кода, может быть разложен на множество интерфейсов I2a, I2b, I2c, и т.д. При разложении количество интерфейсов, включенных в 1-ый сегмент кода, не должны соответствовать количеству интерфейсов, включенных во 2-ой сегмент кода. В любом из случаев по Фиг.1D и 1E функциональность интерфейсов Интерфейс_1 и I1 остается той же самой что и на Фиг.1B и 1C, соответственно. Разложение интерфейсов может также соответствовать ассоциативным, коммутативным и другим математическим свойствам так, что разложение может быть трудно определяемым. Например, упорядочивание операций может быть неважным, и, следовательно, функция, выполняемая интерфейсом, может быть заранее до достижения интерфейса выполнена другой частью кода или интерфейса или выполнена отдельным компонентом системы. Более того, специалисту в области программирования может быть очевидным существование множества способов создания запросов различных функций, которые достигают того же самого результата.

ПЕРЕОПРЕДЕЛЕНИЕ

[45] В некоторых случаях, можно проигнорировать, добавить или переопределить некоторые аспекты (например, параметры) интерфейса программирования, тем не

менее достигая намеченного результата. Это показано на Фиг.1F и 1G. Например, предположим, что интерфейс Интерфейс_1 по Фиг.1B включает в себя вызов функции Square (входные данные, точность, выходные данные) (функция возведения в квадрат), вызов, который включает в себя три параметра, входные данные, точность, выходные данные, и которая вызывается из 1-ого сегмента кода во 2-ой сегмент кода. Если средний параметр, точность, в данном сценарии не представляет интереса, как показано на Фиг.1F, его с таким же успехом можно проигнорировать или даже заменить бессмысленным (в этой ситуации) параметром. Также можно добавить дополнительный, незначимый параметр. Так или иначе, функция возведения в квадрат может быть достигнута, при условии, что выходные данные выдаются после того, как входные данные были возведены в квадрат вторым сегментом кода. Точность может быть очень значимым параметром при дальнейшей обработке или в другой части вычислительной системы; однако, после узнавания того, как установлено, что точность не является необходимой для конкретной операции возведения, она может быть заменена или проигнорирована. Например, вместо того, чтобы передавать реальное значение точности, можно передавать бессмысленную величину, такую как дата рождения, не влияя неблагоприятным образом на результат. Аналогично, как показано на Фиг.1G, интерфейс I1 заменен интерфейсом I1', переопределенный с целью игнорирования или добавления параметров к интерфейсу. Аналогично интерфейс I2 может быть переопределен как интерфейс I2', переопределенный для игнорирования ненужных параметров, или параметров, которые могут быть обработаны в другом месте. Суть этого заключается в том, что в некоторых случаях интерфейс программирования может включать в себя аспекты, такие как параметры, которые не являются необходимыми для некоторой цели, и поэтому они могут быть проигнорированы или переопределены или обработаны в другом месте для других целей.

ЛИНЕЙНОЕ КОДИРОВАНИЕ

[46] Также возможно слияние некоторых или всех функциональных возможностей двух отдельных модулей кода таким образом, что "интерфейс" между ними изменяет форму. Например, функциональные возможности по Фиг.1B и 1C могут быть преобразованы в функциональные возможности по Фиг.1H и 1I, соответственно. На Фиг.1H предыдущие 1-ый и 2-ой сегменты кода по Фиг.1B слиты в модуль, содержащий оба сегмента. В этом случае, сегменты кода могут все еще осуществлять связь друг с другом, но интерфейс может быть адаптирован к виду, который является более подходящим для единичного модуля. Таким образом, например, может отсутствовать необходимость в формальных операторах Call и Return, но подобная обработка, или ответ(ответы) в соответствии с интерфейсом Интерфейс_1 нем не менее может иметь место. Аналогично, как показано на Фиг.1I, часть (или весь) интерфейса I2 по Фиг.1C может быть линейно записан в интерфейсе I1, формируя интерфейс I1". Как показано, интерфейс I2, разделен на I2a и I2b, и часть интерфейса, I2a кодирована линейно с интерфейсом I1, формируя интерфейс I1". В качестве конкретного примера, предположим, что интерфейс, I1 по Фиг.1C выполняет вызов функции возведения в квадрат (входные данные, выходные данные), который получает интерфейс I2, который после обработки значения, пришедшего в виде входных данных (для вычисления результата возведения в квадрат) во второй сегмент кода, передает назад результат возведения в квадрат в виде выходных данных. В этом случае, обработка, выполняемая вторым сегментом кода (возведения в квадрат входных данных), может быть выполнена первым сегментом кода без вызова интерфейса.

РАЗДЕЛЕНИЕ

[47] Осуществление связи одного сегмента кода с другим может быть достигнуто косвенно, разделением линии связи на множество дискретных линий связи. Это схематично показано на Фиг.1J и 1K. Как показано на Фиг.1J, предоставлены одна или несколько частей кода (интерфейс(интерфейсы) Разделения, так как они разделяют функциональные возможности и/или функции интерфейса от исходного интерфейса), для преобразования линий связи первого интерфейса, Интерфейс_1, для согласования с другим интерфейсом, в этом случае интерфейсами Интерфейс2A, Интерфейс2B и Интерфейс2C. Это может быть сделано, например, там, где есть установленная база приложений, разработанных для осуществления связи, скажем, с операционной системой в соответствии с протоколом Интерфейс_1, но затем операционная система изменилась таким образом, что стала использовать другой интерфейс, в этом случае интерфейсы Интерфейс2A, Интерфейс2B и Интерфейс2C. Суть этого заключается в том, что исходный интерфейс, используемый 2-ым Сегментом кода, изменился таким образом, что он больше не является совместимым с интерфейсом, используемым 1-ым Сегментом кода, и таким образом используется посредник, чтобы сделать старые и новые интерфейсы совместимыми. Аналогично, как показано на Фиг.1K, может вводиться третий сегмент кода с интерфейсом 1D Разделения для приема линий связи от интерфейса I1 и с интерфейсом DI2 Разделения для взаимодействия с, например, интерфейсами I2a и I2b, перепроектированных для работы с DI2, при этом обеспечивая тот же самый функциональный результат. Аналогично, DI1 и DI2 могут работать вместе для обеспечения функциональных возможностей интерфейсов I1 и 12 по Фиг.1C в новой операционной системе, обеспечивая тот же самый или подобный функциональный результат.

ЗАМЕНА

[48] Еще один возможный вариант представляет собой динамическое переписывание (замену) кода для замены функциональных возможностей интерфейса чем-то еще, но который дает тот же результат. Например, может иметь место система, в которой сегмент кода, представленный на промежуточном языке (например, Microsoft IL, Java ByteCode и т.д.), направляется в динамический компилятор (JIT) или интерпретатор в среде исполнения (например, инфраструктурой .Net, динамической средой Ява или другой подобной средой динамического типа). Компилятор JIT может быть написан таким образом, чтобы выполнять динамическое преобразование линий связи из 1-ого Сегмента кода во 2-ой Сегмент кода, то есть, согласование их с другим интерфейсом, как может потребоваться 2-ым Сегментом кода (или исходным или другим 2-ым Сегментом кода). Это показано на Фиг.1L и 1M. Как можно видеть на Фиг.1L, этот подход подобен сценарию Разделения, описанному выше. Это можно сделать, например, там, где установленная база данных приложений разработана для осуществления связи с операционной системой в соответствии с протоколом Интерфейс_1, но затем операционная система изменилась таким образом, что она использует другой интерфейс. Компилятор JIT может быть использован для согласования “на лету” линий связи из инсталлированных базой приложений с новым интерфейсом операционной системы. Как показано на Фиг.1M, этот подход динамического переписывания интерфейса (интерфейсов) может быть применен для динамического разложения, или, какого-либо другого изменения интерфейса(интерфейсов).

[49] Также следует отметить, что вышеописанные сценарии для достижения того же самого или подобного результата, что и обеспечиваемый интерфейсом через альтернативные варианты осуществления, также могут быть объединены различными способами, последовательно и/или параллельно или посредством другого чередующегося

кода. Таким образом, альтернативные варианты осуществления, представленные выше, не являются взаимоисключающими и могут быть смешаны, согласованы и объединены для получения сценариев, таких же или эквивалентных общим сценариям, представленным на Фиг.1B и 1C. Также следует отметить, что, как и большинство программных конструкций, существуют другие аналогичные способы достижения тех же самых или подобных функциональных возможностей интерфейса, которые могут быть не раскрыты в настоящем описании, но тем не менее, соответствуют сущности и объему изобретения, то есть, следует отметить, что это представляет собой, по меньшей мере частично, функциональные возможности, представленные, и выигрышные результаты, определяемые интерфейсом, который определяет достоинства данного интерфейса.

ПИКТОГРАММЫ

[50] На Фиг.4A и 4B показан один из примеров приложения настоящего изобретения для предоставления пиктограмм окон приложений. В примере по Фиг.4A показана панель 400 задач, которая содержит кнопки панели задач. Каждая из кнопок на панели задач соответствует активному приложению. Кнопка 402 панели задач соответствует прикладной программе Солитер в настоящее время, выполняемой компьютером. Аналогично, кнопка 403 панели задач соответствует окну прикладной программы обработки текста, кнопка 405 панели задач соответствует окну прикладной программы Медиа Плейер, и кнопка 406 панели задач соответствует окну прикладной программы Калькулятор. Пиктограмма 401 Солитер предоставлена для окна прикладной программы Солитер, а пиктограмма 404 Медиа Плейер предоставлена для прикладной программы Медиа Плейер. Соответствующие пиктограммы могут быть вызваны множеством способов. В одном неограничивающем примере пиктограммы могут быть вызваны при нависании курсора над соответствующей кнопкой панели задач. Например, пиктограмма 401 Солитер может быть вызвана нависанием курсора над кнопкой 402 Солитер панели задач.

[51] Также показанные на Фиг.4A, динамические пиктограммы могут обеспечивать обновляемый или "текущий" контент, соответствующий прикладной программе. Например, приложение Медиа Плейер может проигрывать видео. При проигрывании видео в окне приложения Медиа Плейер (не показано), видео также проигрывается в пиктограмме 404 Медиа Плейер. Если окно приложения Медиа Плейер минимизировано, то пиктограмма 404 Медиа Плейер может показывать последнюю перед уменьшением сцену в окне приложения Медиа Плейер. В качестве альтернативы, пиктограмма 404 Медиа Плейер может продолжать показывать видео до тех пор, пока продолжается воспроизведение в приложении Медиа Плейер, даже в том случае, когда окно приложения Медиа Плейер минимизировано.

[52] На Фиг.4B показан другой пример пиктограмм настоящего изобретения. Показано, что панель 410 задач содержит кнопку 411 Солитер панели задач, соответствующую окну приложения Солитер (не показано), кнопку 412 обработки текста панели задач, соответствующую окну приложения обработки текста (не показано), кнопку 413 Медиа Плейер панели задач, соответствующую окну приложения Медиа Плейер (не показано), и кнопку 414 Калькулятор панели задач, соответствующую окну приложения Калькулятор (не показано). В этом примере представлена пиктограмма 415 Калькулятор. Пиктограмма 415 Калькулятор отображает текущий и обновленный контент окна приложения Калькулятор. В одном из вариантов осуществления настоящего изобретения пиктограмма 415 Калькулятор также может обновлять окно приложения. Например, вычисление может быть выполнено через пиктограмму 415

Калькулятор для получения математического результата. Этот результат может быть отображен на пиктограмме 415 Калькулятор, а также в окне приложения Калькулятор (не показано). В этом примере клавишу "=" выбирают в пиктограмме 415 Калькулятор, и результат отображается на пиктограмме Калькулятор (т.е. "4"). Этот результат также может быть отражен в окне приложения Калькулятор (не показано), даже если вычисление было выполнено посредством пиктограммы 415 Калькулятор.

[53] На Фиг.5 показано другое применение пиктограммы настоящего изобретения, в котором пиктограммы отображены в меню ALT-TAB. Как показано на Фиг.5, дисплей 500 отображает окна приложений. В этом случае, на дисплее 500 видны часть окна 502 приложения Калькулятор и часть окна 501 приложения Солитер. У каждого из приложений есть соответствующая кнопка панели задач на панели задач. У приложения Калькулятор есть соответствующая кнопка 503 Калькулятор панели задач, а у приложения Солитер есть соответствующая кнопка 504 Солитер панели задач. Приложение Медиа Плейер также является активным (не показано), и соответствующая кнопка 506 Медиа Плейер панели задач также присутствует на панели задач.

[54] В примере, показанном на Фиг.5, представлено меню 505 ALT-TAB, которое отображает динамические пиктограммы активных приложений. В этом примере каждая из динамических пиктограмм показывает "текущий" (то есть, обновленный) контент соответствующего окна приложения. Контент динамической пиктограммы может обновляться в реальном масштабе времени. По мере модификации или обновления окон приложений соответствующая динамическая пиктограмма также может соответственно обновляться. Обновление динамической пиктограммы также может происходить в реальном времени. В этом примере меню 505 ALT-TAB содержит динамическую пиктограмму каждого из активных приложений, независимо от того видимо или нет окно приложения соответствующего приложения. Пиктограмма 507 Калькулятор, пиктограмма 508 Медиа Плейер и пиктограмма 509 Солитер показаны в меню 505 ALT-TAB с выбранной пиктограммой 509 Солитер. Каждая из динамических пиктограмм может предоставлять "текущий" контент. Например, в пиктограмме 508 Медиа Плейер может воспроизводиться видео в реальном масштабе времени.

[55] Фиг.2A и 2B представляют собой диаграммы, показывающие компоненты, отображенные на дисплее. Рабочий стол на Фиг.2A представлен в дереве графы сцены или дисплея как элемент 200 и содержит обои 201 рабочего стола и два приложения, Калькулятор 202 и Солитер 203. Кроме того, рабочий стол дополнительно содержит панель 208 задач. В этом способе представления рамки вокруг окон и клиентская область каждого окна представляют собой все части графа сцены. Перемещение окна становится изменяющимися значениями преобразований на дереве дисплея. Каждое из приложений в этом примере содержит оконную рамку 204, 205, соответственно, и, соответственно, контент 206, 207 окна. Панель 208 задач содержит контент 209 окна. На Фиг.2A элементы на дисплее визуализированы слева направо. Следовательно, обои 201 рабочего стола визуализированы сзади окна 203 приложения Калькулятор, которое визуализировано сзади окна 202 приложения Солитер.

[56] На Фиг.2B показано изображение на дисплее, как представлено на Фиг.2A. На заднем плане изображение на дисплее содержит обои 220 рабочего стола. Окно 222 приложения Калькулятор показано перед обоями рабочего стола, а окно 221 приложения Солитер показано перед окном 222 приложения Калькулятор и обоями 220 рабочего стола. Как показано на Фиг.2B, каждое из окон приложения может быть представлено кнопками панели задач на панели 225 задач. В этом примере приложение Калькулятор представлено кнопкой 223 панели задач на панели 225 задач, и прикладная программа

Солитер представлена кнопкой 224 панели задач на панели 225 задач. Каждое окно 222 приложения Калькулятор и окно 221 приложения Солитер содержит оконную рамку, и контент окна, отображенное на обоях 220 рабочего стола.

[57] Любой подузел дерева, изображенного на Фиг.2А, может использоваться в качестве контента другого узла дерева. Следуя этому, контент одной части дерева можно увидеть в другом местоположении в другом месте на дереве. Существует множество способов визуализации узла на дереве в другом местоположении в другом месте на дереве. В качестве одного из неограничивающих примеров, может быть использована Visual Brush. "Visual Brush" представляет собой способ, используемый для ссылки на существующую часть графа сцены, но для визуализации его с "текущим" набором графического контекста, такого как положение, масштаб, поворот, непрозрачность, эффект и т.д. Это заставляет часть дерева, визуализированного в одном месте быть визуализированным в любом другом месте. Например, если нужна динамическая пиктограмма окна приложения Калькулятор, то можно вызвать соответствующий интерфейс программирования (например, интерфейс прикладного программирования или API) для предоставления дескриптора HWND хоста пиктограммы или дескриптора окна. Этот пример показан на Фиг.3А и 3В.

[58] Фиг.3А представляет собой диаграмму графа сцены или дерева, иллюстрирующую пример компонентов изображения на дисплее настоящего изобретения. Рабочий стол представлен на Фиг.3А как элемент или узел 300 и содержит обои 301 рабочего стола и два приложения, Калькулятор 302 и Солитер 303. Каждое из приложений в этом примере содержит оконную рамку 305, 306, соответственно, и контент 307, 308 окна, соответственно. Рабочий стол в этом примере дополнительно содержит панель 307 задач.

[59] В этом примере предоставлена пиктограмма окна приложения Калькулятора путем вызова соответствующего API. Как показано на Фиг.3А, дескриптор окна HWND 304 хоста пиктограммы создан приложением и может регистрировать пиктограмму приложения, такую как приложение Калькулятор. Например, на Фиг.3А панель 307 задач содержит HWND 304 хоста пиктограммы и контент 310 окна. HWND 304 хоста пиктограммы содержит пиктограмму 311 Калькулятор и может присоединять пиктограмму окна приложения к хосту пиктограммы. Пунктирная стрелка на Фиг.3А показывает, что пиктограмма 311 Калькулятор HWND 304 хоста пиктограммы указывает на или перерисовывает Калькулятор из окна приложения Калькулятор. В результате предоставляется "текущая" или динамическая пиктограмма, которая представляет обновленный контент соответствующего окна приложения (в этом случае, окна приложения Калькулятор), а не просто статические представления контента окна приложения.

[60] Как и в примере на Фиг.2А, элементы на дисплее, представленные на Фиг.3А, визуализированы слева направо. Следовательно, обои рабочего стола 301 визуализированы сзади окна приложения Калькулятор 302, которое визуализировано сзади окна приложения Солитер 303 и панели 208 задач. На Фиг.3В показано изображение на дисплее, соответствующее Фиг.3А. Как показано на Фиг.3В, изображение на дисплее содержит обои 320 рабочего стола на заднем плане. Окно 322 приложения Калькулятор отображается перед обоями 320 рабочего стола, а окно 321 приложения Солитер отображается перед окном 322 приложения Калькулятор и обоями 320 рабочего стола. Также отображается панель 326 задач. Каждое из приложений ассоциировано с соответствующей кнопкой панели задач на панели 326 задач. В этом примере приложение Калькулятор ассоциировано с кнопкой 324 панели задач, а

приложение Солитер ассоциировано с кнопкой 325 панели задач. Кроме того, на Фиг.3В показано, что представлена пиктограмма, которая ассоциирована с приложением Калькулятор. Прикладная программа Калькулятор и окно 322 ассоциировано с кнопкой 324 панели задач на панели 326 задач, а прикладная программа Солитер и окно 321 ассоциировано с кнопкой 325 панели задач.

[61] Также на Фиг.3В показана динамическая пиктограмма 323 окна 322 приложения Калькулятор. Пиктограмма 323 предоставлена узлом 311 Пиктограммы Калькулятор для HWND 304 хоста пиктограммы, который указывает на или перерисовывает оконную рамку 305 и контент 308 окна приложения 302 Калькулятор, как изображено на Фиг.3А. Результат представляет собой динамическую пиктограмму 323 окна 322 приложения Калькулятор, которая является "текущей", т.е. пиктограмма меняется при соответствующих модификациях, происходящих в самом окне приложения. Как показано на Фиг.3В, значение "243", вычисленное в окне 322 приложения Калькулятор, также отражается в соответствующей пиктограмме 323.

[62] В одном из аспектов настоящего изобретения динамические пиктограммы формируются посредством таких свойств пиктограммы, как регистрация, отмена регистрации и обновление. Такие свойства как регистрация, отмена регистрации и обновление или модификация пиктограмм могут выполняться посредством интерфейсов прикладного программирования (API) в администраторе окон рабочего стола (DWM). Описаны три иллюстративных API для обеспечения пиктограмм настоящего изобретения.

Пример 1 API - Регистрация пиктограммы

[63] Пиктограмма может быть сгенерирована из окна, например, окна приложения. Окно приложения, в этом случае, представляет собой исходное окно, поскольку окно приложения обеспечивает основу для контента пиктограммы. В одном из примеров пиктограмма представляет собой миниатюризированную версию окна приложения и содержит весь контент окна приложения. В качестве альтернативы, пиктограмма может содержать только выбранную часть исходного окна.

[64] Контент пиктограммы исходного окна визуализирован во второй области. Эта вторая область определяется целевым окном, которое подстраивает контент пиктограммы на основе исходного окна. Целевое окно может само формировать основу пиктограммы. Например, для показа пиктограммы на основе окна приложения в качестве целевого окна может использоваться небольшая оконная рамка в местоположении, отличающемся от местоположения окна приложения, обозначенная область на дисплее или всплывающее меню.

[65] Когда пиктограмма зарегистрирована, может быть выполнена операция, которая устанавливает отношения между исходным окном и целевым окном. Может быть определен дескриптор (HWND) окна, который определяет целевое окно. Определенное целевое окно может служить в качестве мишени визуализации пиктограммы и принадлежит процессу, вызывающему или регистрирующему пиктограммы. Если процесс, регистрирующий пиктограмму, также владеет целевым окном, то можно избежать нежелательных изменений в приложении, вызываемых другими приложениями. Аналогично, может также быть определен HWND для исходного окна.

[66] Также может быть определен размер кэшированного изображения в растровом формате пиктограммы. Например, может быть указан минимальный размер таким образом, что системе сообщают о последнем известном изображении исходного окна в растровом формате. Таким образом, если исходное окно затем минимизируется и не видимо на дисплее, при этом, тем не менее, сохраняется изображение окна в растровом

формате, и пиктограмма все еще доступна. Размер может изменяться в зависимости от потребностей вызывающего приложения. Если имеет место множество регистраций одного и того же исходного окна, процесс может определить, какой размер пиктограммы применить, исходя из существующих потребностей. Например, в случае множества регистраций изменяющихся размеров пиктограммы может использоваться наибольший размер.

[67] API также может содержать дескриптор, представляющий регистрацию пиктограммы. Этот дескриптор может быть уникальным для процесса, который осуществляет вызов таким образом, что впоследствии регистрация пиктограммы может быть отменена только из процесса, при помощи которого она зарегистрирована.

[68] Ниже приведен пример API для регистрации пиктограммы:

```
Typedef HANDLE HTHUMBNAIL;

HRESULT

DwmRegisterThumbnail (

HWND                    dstWindow,

HWND                    srcWindow,

SIZE                    *lpMinimizedSize,

OUT HTHUMBNAIL          *lphThumbnail

);
```

[69] В котором dstWindow представляет целевое окно, srcWindow представляет исходное окно, lpMinimizedSize представляет размер, в котором система кэширует последнее известное изображение исходного окна в растровом формате (srcWindow) после минимизации srcWindow, и lphThumbnail представляет собой возвращаемое значение, представляющее уникальный дескриптор, представляющий регистрацию, осуществленную при вызове.

Пример 2 API - модификация или установка свойств пиктограммы

[70] После регистрации пиктограммы контент исходного окна, или его часть, размещают в виде пиктограммы в целевом окне. Например, при вызове, свойства API Обновление пиктограммы могут изменить или обновить свойства визуализации пиктограммы в целевом окне. Например, могут использоваться свойства UpdateThumbnail для установки свойств пиктограмм, таких как видимость, форма, прямоугольники, и т.д. Свойства API Обновление пиктограммы могут указывать на дескриптор пиктограммы, предназначенный для обновления. Обновление пиктограммам включает в себя любую модификацию пиктограммы, включая без ограничений расположение пиктограммы, размер пиктограммы, часть исходного окна, которая должна использоваться для пиктограммы, часть целевого окна, которая должна использоваться для пиктограммы, непрозрачность или прозрачность пиктограммы, видимость пиктограммы, специальные эффекты пиктограммы (например, вращение, деформация, разрушение и т.д.) и т.д.

[71] В другом примере пиктограммы могут обновляться или изменяться из процессов, в которых они были зарегистрированы. В одном из примеров дескриптор пиктограммы неоднократно вызывается в целевом окне таким образом, что пиктограмма может

перемещаться внутри целевого окна. В этом случае, местоположение пиктограммы может изменяться в пределах целевого окна. Например, в окне ALT-TAB можно изменить порядок размещения пиктограмм путем вызова пиктограммы в целевом окне.

[72] При модификации или установке свойств пиктограммы могут использоваться параметры. Существует много примеров параметров для модификации или установки свойств пиктограммы, например, без ограничения, флаги. С помощью этих параметров структура API может легко изменяться, делая API более гибким для будущих обновлений. Параметры могут определять свойства пиктограммы, которые должны быть использованы, и сами могут определяться указанными константами. Например, параметры могут определять и соответствовать параметру исходного окна, который определяет область или часть исходного окна, предназначенного для использования в качестве пиктограммы в целевом окне, когда желательно использовать для пиктограммы размер, меньший чем все исходное окно. В этом примере для пиктограммы в целевом окне используется только определенная часть исходного окна, как указано параметром исходного окна. В одном из примеров часть исходного окна, предназначенная для использования, в качестве пиктограммы в целевом окне, определена в виде координат на дисплее. Например, если желаемая часть представляет собой прямоугольник в пределах исходного окна, API может определить параметр, чтобы указать координаты прямоугольника, пересекающегося с исходным окном для получения конечного прямоугольника, который будет использоваться в качестве пиктограммы. В качестве примера, координаты (0,0) могут быть установлены как верхний левый угол области окна. Параметр, определяющий и соответствующий желаемому параметру исходного окна, может быть установлен для того, чтобы указать, что только определенная часть исходного окна должна быть использована в виде пиктограммы, при этом определенная часть исходного окна определяется в терминах предоставляемых координат. Несмотря на то, что существует множество способов определения координат и желаемой части исходного окна, один из примеров может включать в себя использование части исходного окна, определенной пересечением предоставленного прямоугольника с пересекаемым фактическим исходным окном.

[73] Кроме того, часть целевого окна, которая должна быть использована для пиктограммы, может быть определена при помощи параметра, который определяет и соответствует параметру целевого окна. В этом случае, если желательно использовать не все целевое окно, которое должно быть использовано для пиктограммы, а только часть целевого окна, которое должно быть использовано для пиктограммы, то область целевого окна, в котором может быть визуализирована пиктограмма, может быть определена параметром целевого окна. В этом случае, один из подходов мог бы представлять собой визуализацию только в пределах желаемой определенной области, как определено параметром целевого окна (т.е., усечение изображения при визуализации происходит за пределами определяемой области). Кроме того, один из способов определения местоположения части целевого окна, в котором может быть нарисована пиктограмма, представляет собой определение пересечения части целевого окна, предназначенного для использования, и самого целевого окна. Это предотвращает визуализацию за пределами заданной области. Кроме того, параметр, определяющий и соответствующий параметру целевого окна, может использоваться для определения области в пределах целевого окна, предназначенного для использования в качестве пиктограммы, из исходного окна. В этом примере, если параметр целевого окна не определен, то для контента пиктограммы может быть использовано все заданное окно.

[74] В API может быть определено любое число других свойств пиктограммы.

Например, может быть определена непрозрачность или прозрачность пиктограммы. В одном из аспектов этого примера настоящего изобретения может быть назначен параметр для определения непрозрачности (или прозрачности) пиктограммы, например, 255 может быть назначен для указания полной непрозрачности, в то время как 0 может быть назначен для указания полной прозрачности. Таким образом, если установлен параметр, определяющий и соответствующий параметру непрозрачности, может быть применено определенное назначенное цифровое значение для непрозрачности пиктограммы. В результате пиктограмма может иметь различную степень непрозрачности по желанию пользователя.

[75] В качестве дополнительного примера, пиктограмма может быть сделана видимой или невидимой по необходимости. Может быть предоставлен параметр, определяющий и соответствующий параметру видимости. Параметр видимости может представлять собой Булево значение таким образом, что когда параметр видимости представляет собой значение TRUE, пиктограмма будет видимой, в то время как пиктограмма будет невидимой, если параметр видимости имеет значение FALSE. Настоящее изобретение не ограничено никакой конкретной модификацией пиктограммы, поскольку любая модификация пиктограммы находится в пределах объема настоящего изобретения.

[76] Ниже приведен пример API обновления или изменения пиктограммы DwmUpdateThumbnailProperties:

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

[77] где dwFlags определяют параметры или флаги, соответствующие параметрам, которые определяют последующие свойства, которые должны быть использованы. В этом примере параметр rcSource представляет собой параметр исходного окна, который определяет область исходного окна (srcWindow), предназначенную для использования в виде пиктограммы, rcDestination представляет собой параметр целевого окна, который определяет целевое окно (dstWindow), в котором визуализируется пиктограмма, Opacity представляет собой параметр непрозрачности, который управляет непрозрачностью пиктограммы (например, устанавливаемый от 0 до 255), и fVisible представляет собой параметр видимости, который определяет пиктограмму как невидимую или видимую на основе Булева значения.

Пример 3 API - Отмена регистрации пиктограммы

[78] Ассоциация между исходным и целевым окнами удаляется, когда она больше не нужна. Например, если окно больше не является активным, ассоциация между окном и пиктограммой может осуществляться путем отмены регистрации ассоциации. Это приводит к освобождению ресурсов. Более того, пиктограммы могут дополнительно освобождаться при завершении процесса регистрации. Ниже приведен пример API для отмены регистрации пиктограммы DwmUnregisterThumbnail:

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

[79] Фиг.6 представляет собой блок-схему последовательности операций, иллюстрирующую пример процесса предоставления пиктограмм, включая "текущие" или динамические пиктограммы настоящего изобретения. На этапе 601, предоставляется API, при помощи которого приложение может запросить первое окно верхнего уровня или исходное окно показать пиктограмму, которая также может быть расположена в секции второго окна верхнего уровня или целевого окна. Следовательно, первое окно верхнего уровня или исходное окно может быть визуализировано в виде объекта самого высокого уровня в пределах целевого окна. Один из неограничивающих примеров этой реализации представляет собой окно ATL-TAB, в котором пиктограмма, соответствующая исходному окну, предоставляется в пределах окна ALT-TAB (то есть, целевого окна).

[80] На этапе 602, генерируется вызов Регистратора для DWM, который может предоставить дескриптор пиктограммы. Также может иметь место множество случаев вызова в одном и том же исходном окне или одном и том же целевом окне. В примере ALT-TAB множество случаев вызова в одном и том же исходном окне и одном и том же целевом окне приводит к размещению пиктограмм в другом порядке в пределах окна ALT-TAB, а также предоставление увеличенного и/или стилизованного представления пиктограммы приложения при выборе пиктограммы. В другом примере Z-порядок пиктограммы может быть изменен последующими вызовами регистрации. Если нужна модификация z-порядка пиктограммы относительно других элементов дисплея, новая пиктограмма может быть зарегистрирована при помощи другого вызова регистрации пиктограммы. После установки свойств для новой пиктограммы, которая соответствует старой пиктограмме, регистрация старой пиктограммы может быть отменена. Таким образом, в этом примере, z-порядок пиктограммы может быть изменен посредством множества регистраций пиктограммы.

[81] В качестве альтернативы, может быть вызвано множество исходных окон для одного и того же целевого окна. В этом примере, вызовы множества исходных окон для одного целевого окна приводят к предоставлению множества различных пиктограмм в пределах целевого окна.

[82] В другом примере динамические пиктограммы могут быть отображены в пределах других динамических пиктограмм. Например, первая динамическая пиктограмма с более высоким z-порядком может быть размещена в более динамической пиктограмме приложения с более низким z-порядком. В этом примере пиктограммы могут быть вложены в другие пиктограммы. Более того, к пиктограммам или к любым вложенным пиктограммам могут быть применены эффекты. В качестве альтернативы, пиктограммы любого z-порядка могут быть размещены в виде каскада в другой пиктограмме любого z-порядка. Например, по меньшей мере, одна пиктограмма (независимо от z-порядка) может быть помещена в большую динамическую пиктограмму приложения любого z-порядка. Аналогично, большая динамическая пиктограмма может быть помещена в другую динамическую пиктограмму любого z-порядка. Таким образом, пиктограммы могут быть размещены в виде каскада независимо от z-порядка.

[83] На этапе 603, DWM регистрирует пару процесс/ID для пиктограммы. DWM может дополнительно поддерживать глобальный список регистраций пиктограмм, в котором каждому приложению соответствует дескриптор пиктограммы (HWND). Глобальный

список может поддерживаться множеством способов в зависимости от потребностей пользователя. Например, один из удобных способов поддержки глобального списка состоит в индексации пиктограмм целевым окном (dstWindow) таким образом, что любое желаемое целевое окно может быть эффективно доступным. Кроме того, пиктограммы в любом конкретном целевом окне могут быть дополнительно организованы в логическом порядке, таком как в порядке наложения или основанном на z-порядке.

[84] В другом примере DWM может "захватывать" контенты окна (т.е. спасать текущие контенты окна) перед минимизацией этого окна. DWM захватывает контенты окна, например, в виде изображения в растворном формате конкретного размера. Размер может быть определен API регистрацией и является подходящим размером для отображения пиктограммы. После минимизации окна контент окна сохраняется таким образом, что пиктограмма, соответствующая окну приложения, может все еще быть доступной. В качестве альтернативы, окна, которые минимизированы, могут быть временно "оставленными" без вывода на экран, а не действительно минимизироваться. В этом случае, приложения могут продолжать посылать информацию и данные для обновления пиктограммы. Таким образом, если происходит регистрация пиктограммы после "минимизации" окна приложения, пиктограммы все еще могут регистрироваться и обновляться, как будто окно не было минимизировано. В качестве альтернативы, пиктограммы могут быть заменены иконками или заголовками окна, если регистрация пиктограммы не выполнима.

[85] На этапе 604, определяются и происходят обновления свойств пиктограмм. При модификации или обновлении свойств пиктограммы DWM может модифицировать глобальный список соответствующим образом. Например, приложение DWM может добавить пиктограммы к своему собственному графу сцены (например, см. Фиг.2А или Фиг.3А), в котором перерисовывается подходящий узел окна верхнего уровня, который представляет целевое окно, из исходного окна (т.е., sreWindow) на основе дополнительных инструкций для выполнения рисования. Также в пиктограмму могут быть включены любые дополнительные определенные изменения или эффекты, например, масштабирование, позиционирование, непрозрачность, невидимость или специальные эффекты.

[86] На этапе 605, пиктограмма модифицируется на основе полученных инструкций. Например, API UpdateThumbnail может изменять видимость узла пиктограммы, если параметр fVisible равен TRUE, или API UpdateThumbnail может увеличить (или уменьшить) непрозрачность пиктограммы. После выполнения всех желательных модификаций в пиктограмме процесс DWM определяет, нужна ли отмена регистрации пиктограммы. Этот процесс определения, если отмена регистрации пиктограммы нужна, может быть выполнен отдельно от процесса UpdateThumbnail. В этом случае, отмена регистрации может происходить в результате вызова API UnregisterThumbnail (не показано). На этапе 606, определяют, получен ли процессом DWM API отмены регистрации. Если API отмены регистрации получен, удаляется спаривание процесса и соответствующего ID пиктограммы (например, из глобального списка регистрации пиктограмм, поддерживаемого в процессе DWM). На этапе 607, в результате отмены регистрации пиктограммы повторно выдается поток инструкций окна верхнего уровня. В результате пиктограмма удаляется.

[87] В одном из примеров курсор нависает над кнопкой панели задач, приводя к отображению пиктограммы соответствующего приложения. Когда курсор нависает над кнопкой панели задач, процесс создает новый дескриптор (HWND) окна верхнего

уровня, которое хостирует пиктограмму. Процесс вызывает API регистрацию, которая использует дескриптор (HWND) окна приложения в качестве источника контента для пиктограммы (т.е., srcWindow). Новый HWND верхний уровень, который хостирует пиктограмму, представляет собой целевое окно (dstWindow). Затем контент из исходного окна, или желаемая его часть, размещается в целевом окне для отображения пиктограммы. Пиктограмма может содержать весь контент исходного окна или может содержать только заданную или выбранную часть как описано выше. UpdateThumbnail API может изменять или перемещать пиктограмму. Например, пиктограмма может быть перемещена относительно соответствующей иконки, или в пиктограмме могут быть применены специальные эффекты (например, прозрачность, невидимость, искажение и т.д.). При отведении курсора в сторону от кнопки панели задач регистрация HWND верхнего уровня, хостирующего пиктограмму, может быть отменена, таким образом, что пиктограмма будет скрыта. В качестве альтернативы, пиктограмма может быть обновлена, чтобы стать невидимой таким образом, что пиктограмма все еще будет зарегистрирована и может быть вызвана впоследствии. В альтернативном способе каждому окну после создания может быть назначена пиктограмма, созданная для этого при помощи панели задач. В этом примере код панели задач может быть использован, например, для манипуляции видимостью пиктограмм и целевым прямоугольником для размещения пиктограммы.

[88] Является очевидным, что аспекты настоящего изобретения могут относиться ко множеству форм и вариантов осуществления. Варианты осуществления, раскрытые в настоящем описании, предназначены для иллюстрации, а не ограничения изобретения, очевидно, что могут быть сделаны изменения без отступления от сущности и объема изобретения. Хотя показаны и описаны иллюстративные варианты осуществления изобретения, для вышеприведенного описания подразумевается широкий диапазон модификаций, изменений и замен, и в некоторых случаях некоторые признаки настоящего изобретения могут использоваться без соответствующего использования других признаков. Следовательно, очевидно, что прилагаемая формула изобретения должна рассматриваться максимально широко в соответствии с объемом настоящего изобретения.

Формула изобретения

1. Реализуемый компьютером способ отображения окна на устройстве отображения, причем способ содержит этапы:

отображения исходного окна, при этом исходное окно содержит контент исходного окна, который способен модифицироваться;

отображения целевого окна, причем по меньшей мере часть целевого окна содержит пиктограмму, причем пиктограмма включает в себя контент пиктограммы, соответствующий по меньшей мере части контента исходного окна; и

модификации контента пиктограммы, когда контент исходного окна изменяется,

получения первого параметра для определения по меньшей мере части исходного окна, причем упомянутая модификация содержит определение по меньшей мере части исходного окна, которая должна быть нарисована в пиктограмме, на основе упомянутого первого параметра; и

получения второго параметра для определения по меньшей части целевого окна, причем упомянутая модификация дополнительно содержит определение по меньшей мере части целевого окна для визуализации пиктограммы на основе упомянутого второго параметра;

причем модификация исходного окна отображается в пиктограмме.

2. Способ по п. 1, дополнительно содержащий этапы

минимизации исходного окна и

автоматического отображения в пиктограмме последнего контента, отображенного посредством исходного окна до минимизации.

3. Способ по п. 1, в котором пиктограмма модифицируется в реальном времени, когда первый контент исходного окна модифицируется.

4. Способ по п. 1, в котором модификация контента пиктограммы дополнительно содержит модификацию видимости пиктограммы.

5. Способ по п. 1, в котором этап модификации контента пиктограммы происходит автоматически, когда контент исходного окна изменяется.

6. Система для отображения окна на устройстве отображения, причем система содержит:

дисплей для отображения исходного окна, содержащего контент, который способен модифицироваться, и целевого окна;

администратор для регистрации пиктограммы, причем пиктограмма находится в целевом окне и соответствует по меньшей мере части исходного окна, при этом упомянутая регистрация содержит:

получение дескриптора окна для упомянутого исходного окна;

установку ассоциации между исходным окном и пиктограммой так, чтобы по меньшей мере часть контента исходного окна была нарисована в пиктограмме;

получение первого параметра для определения по меньшей мере части исходного окна, причем упомянутая модификация содержит определение по меньшей мере части исходного окна, которая должна быть нарисована в пиктограмме, на основе упомянутого первого параметра; и

получение второго параметра для определения по меньшей части целевого окна, причем упомянутая модификация дополнительно содержит определение по меньшей мере части целевого окна для визуализации пиктограммы на основе упомянутого второго параметра; и

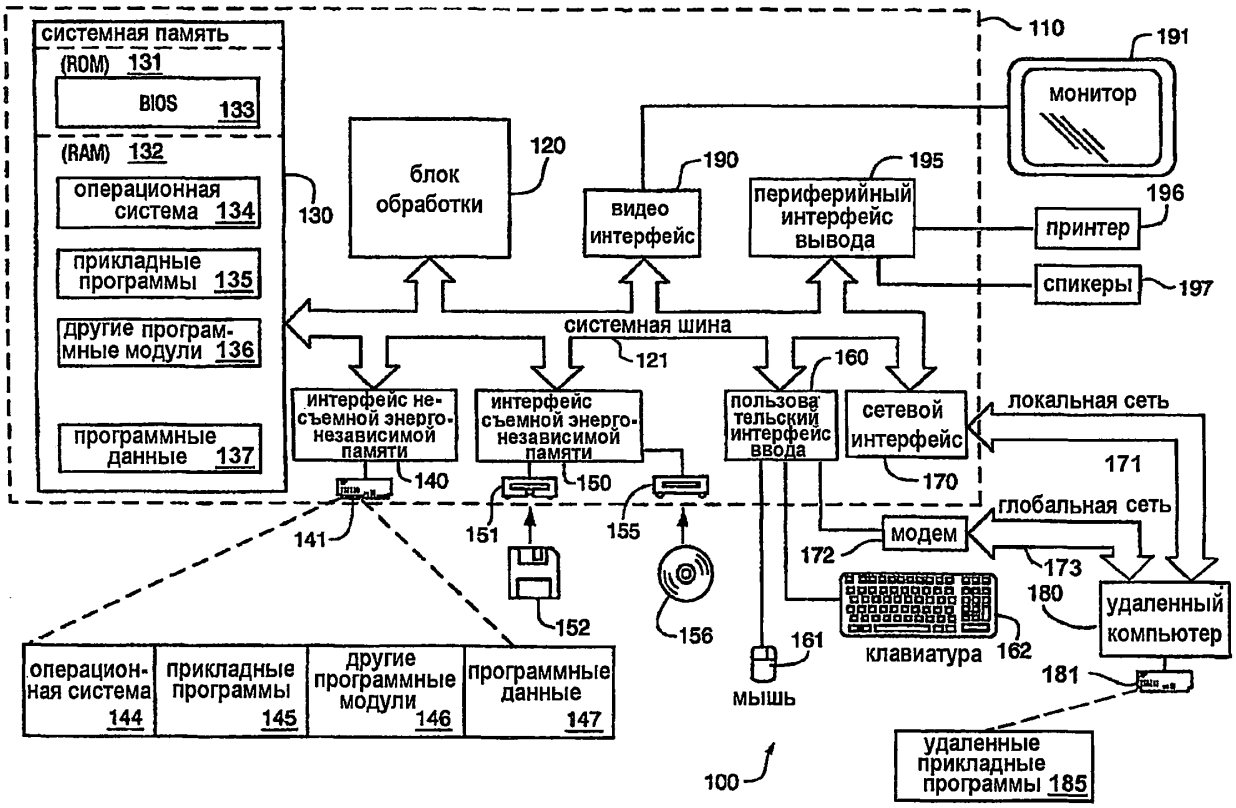
получение уникального дескриптора, представляющего ассоциацию между исходным окном и пиктограммой;

процессор и

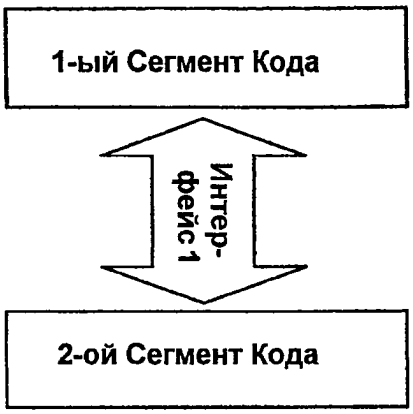
память, хранящую компьютерно-выполняемые инструкции, которые при выполнении процессором вынуждают систему осуществлять способ модификации контента пиктограммы, когда контент исходного окна изменяется, причем упомянутый способ содержит этапы, на которых:

определяют по меньшей мере часть исходного окна, которая должна быть нарисована в пиктограмме, на основе первого параметра для определения по меньшей мере части исходного окна и

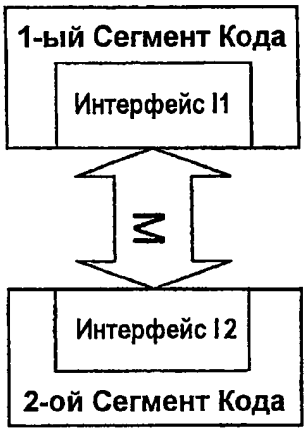
определяют пиктограмму для визуализации на основе второго параметра для определения по меньшей мере части целевого окна, в которой должна быть визуализирована пиктограмма.



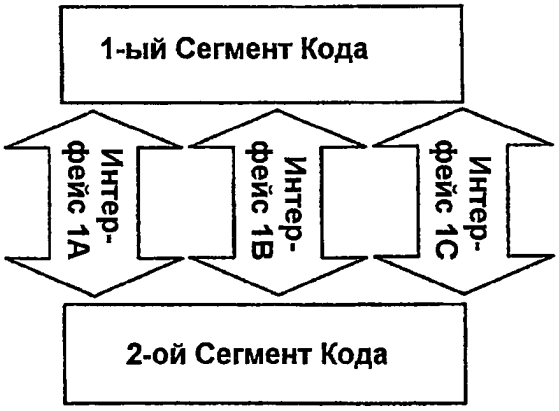
Фиг.1A



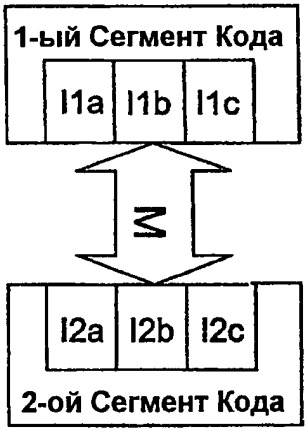
Фиг.1B



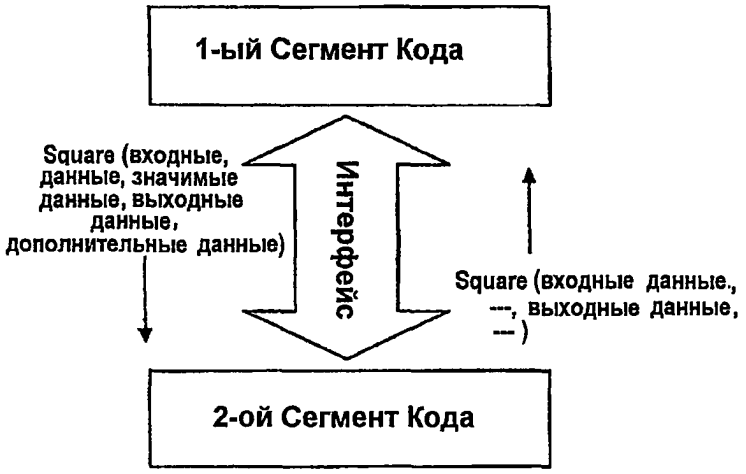
Фиг.1C



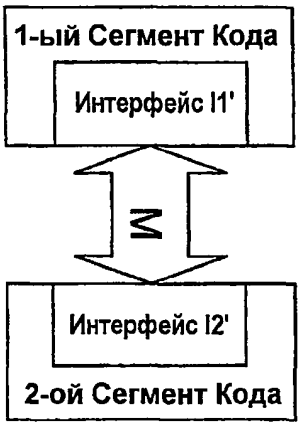
Фиг.1D



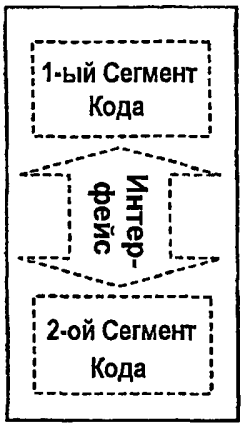
Фиг.1E



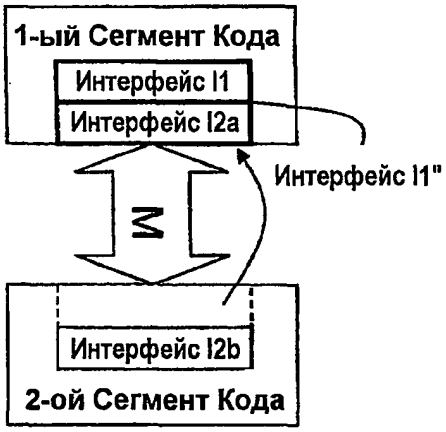
Фиг.1F



Фиг.1G



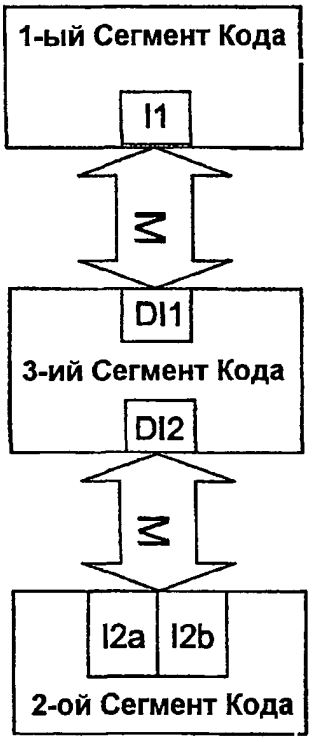
Фиг.1H



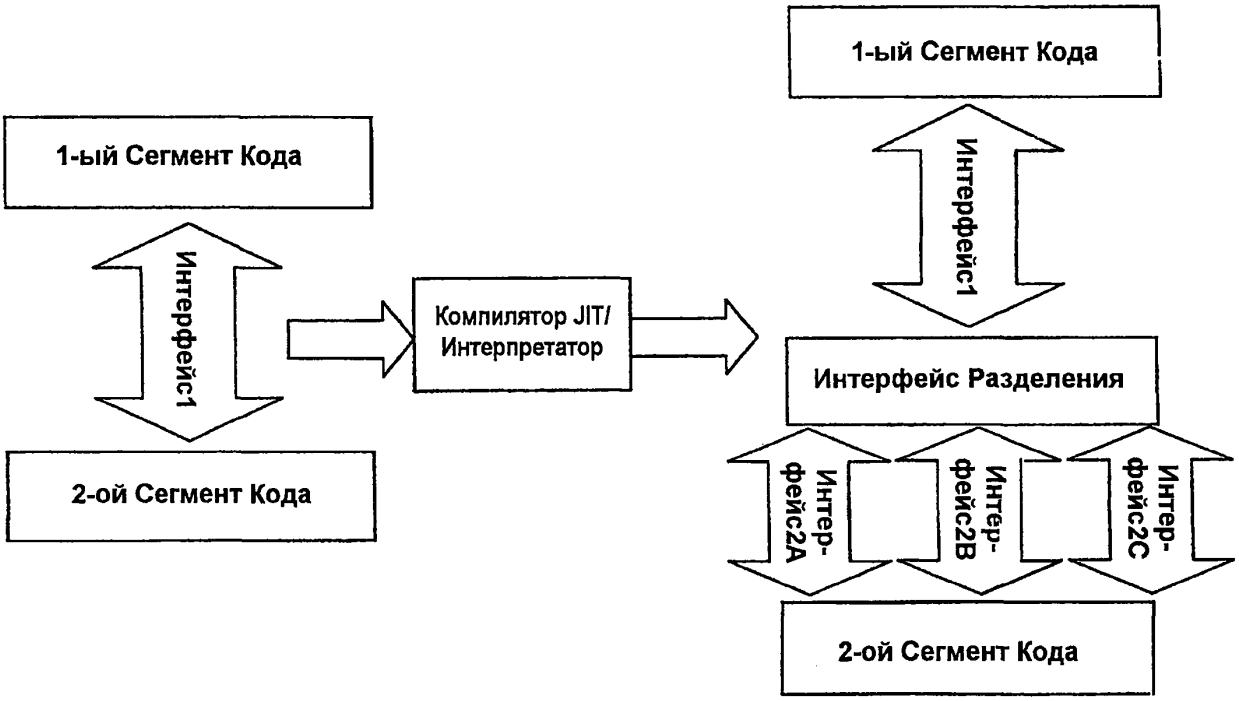
Фиг.1I


1-ый Сегмент Кода
Интерфейс1
Интерфейс Разделения
Интер-фейс2A
Интер-фейс2B
Интер-фейс2C
2-ой Сегмент Кода

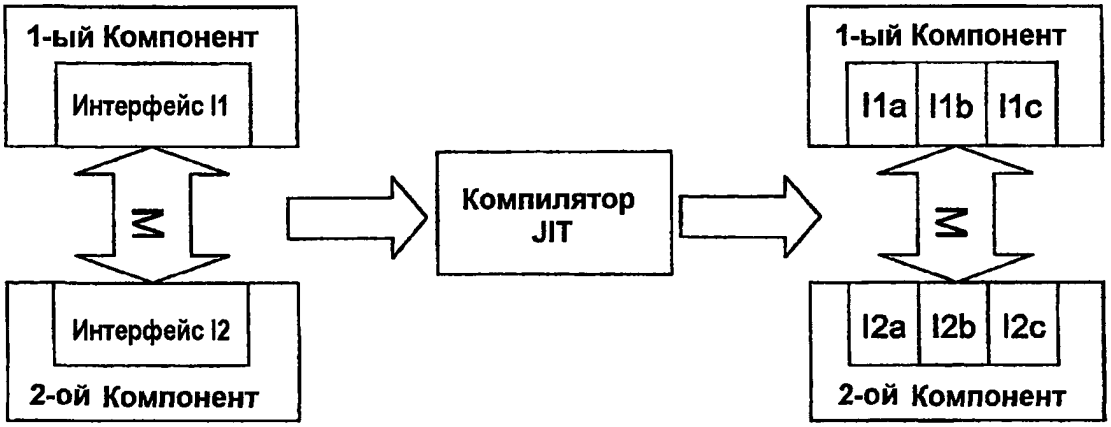
Фиг.1J


1-ый Сегмент Кода
I1
M
DI1
3-ий Сегмент Кода
DI2
M
I2a
I2b
2-ой Сегмент Кода

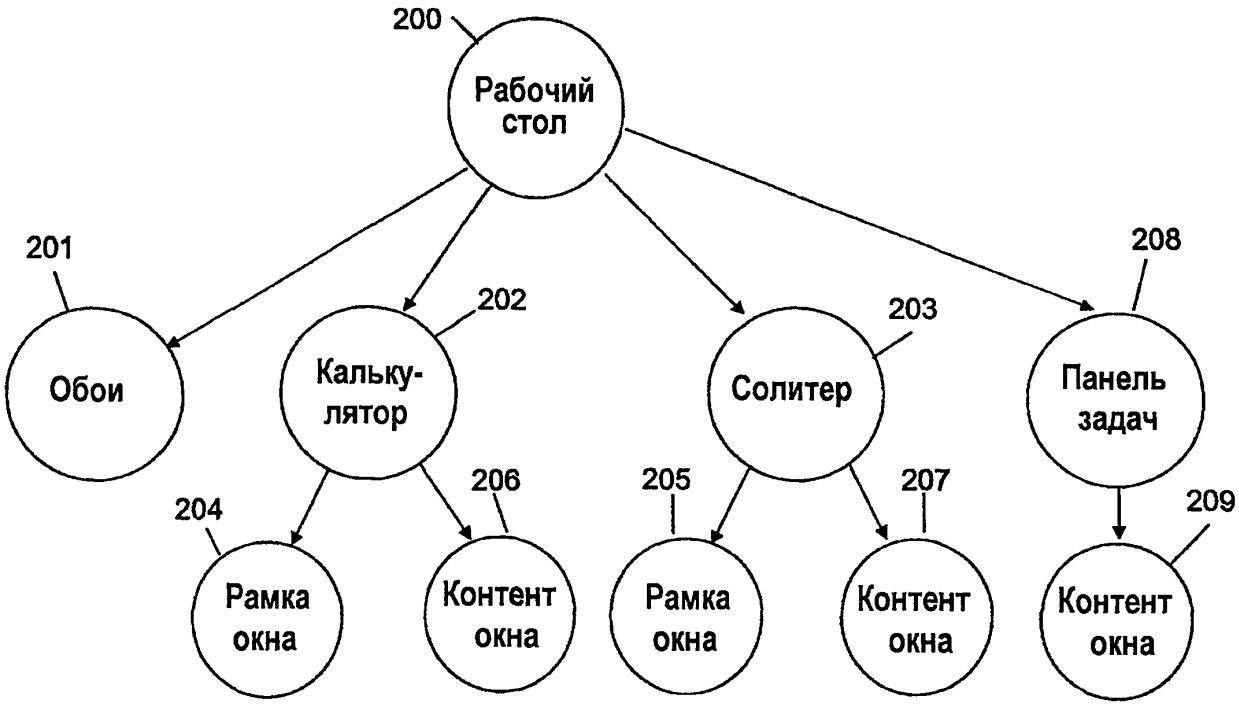
Фиг.1K


1-ый Сегмент Кода
Интерфейс1
2-ой Сегмент Кода
Компилятор JIT/
Интерпретатор
1-ый Сегмент Кода
Интерфейс1
Интерфейс Разделения
Интер-
фейс2A
Интер-
фейс2B
Интер-
фейс2C
2-ой Сегмент Кода

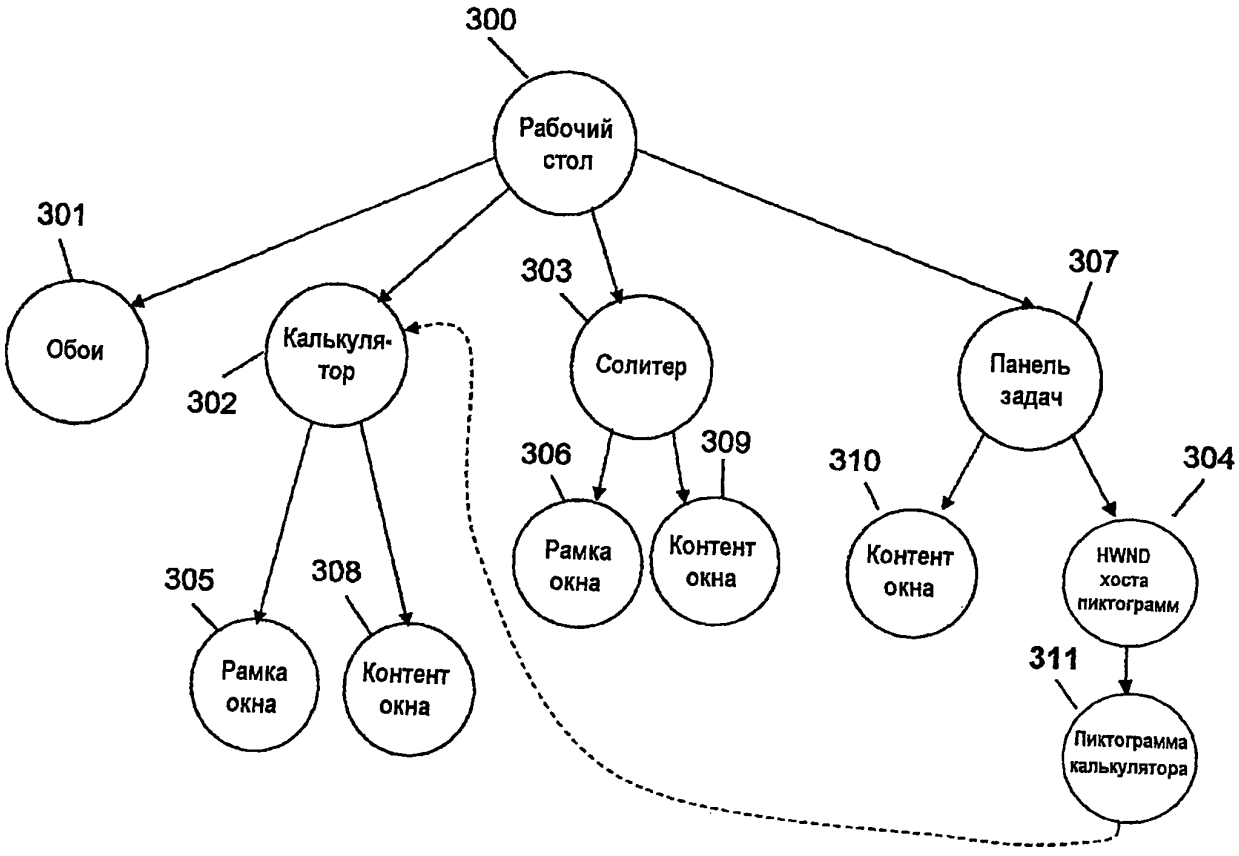
Фиг.1L


1-ый Компонент
Интерфейс I1
M
Интерфейс I2
2-ой Компонент
Компилятор
JIT
1-ый Компонент
I1a
I1b
I1c
M
I2a
I2b
I2c
2-ой Компонент

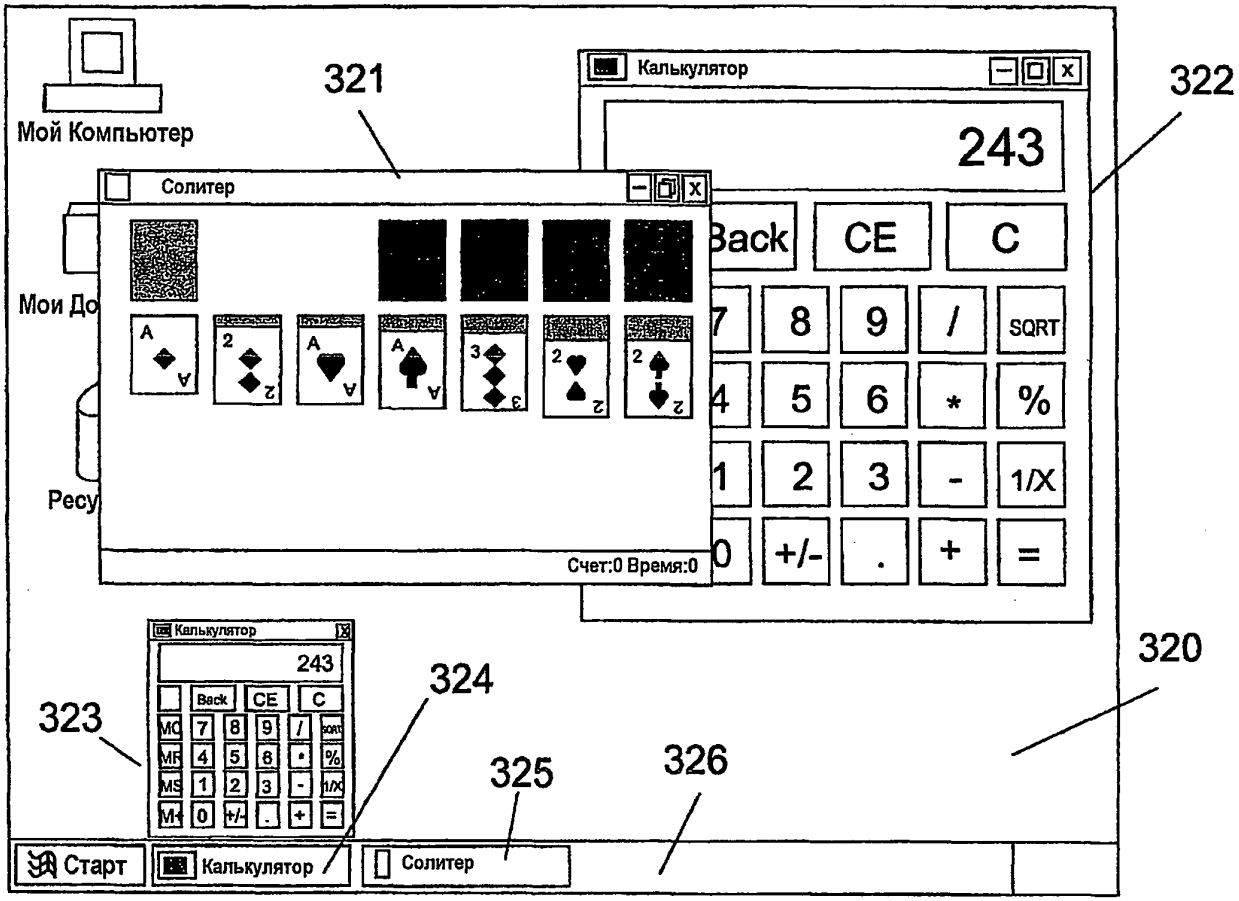
Фиг.1M


200
Рабочий стол
201
Обои
202
Кальку-лятор
203
Солитер
208
Панель задач
204
Рамка окна
206
Контент окна
205
Рамка окна
207
Контент окна
209
Контент окна

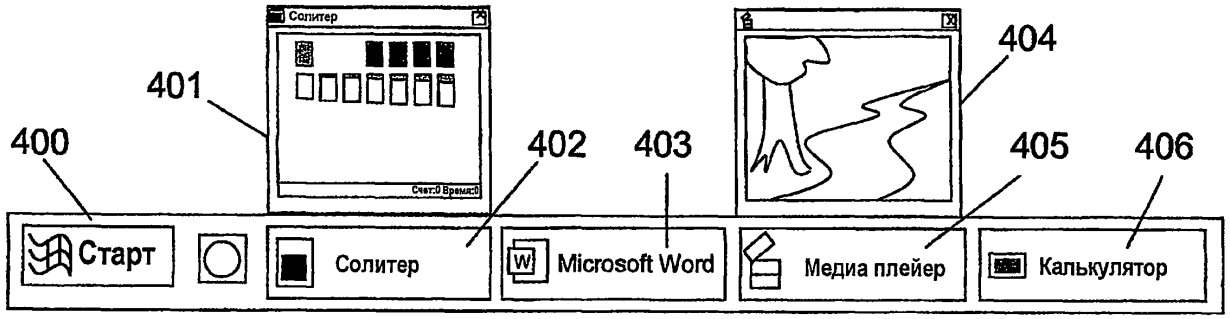
Фиг.2А


300
Рабочий стол
301
Обои
302
Калькуля-тор
303
Солитер
307
Панель задач
305
Рамка окна
308
Контент окна
306
Рамка окна
309
Контент окна
310
Контент окна
304
HWND хоста пиктограмм
311
Пиктограмма калькулятора

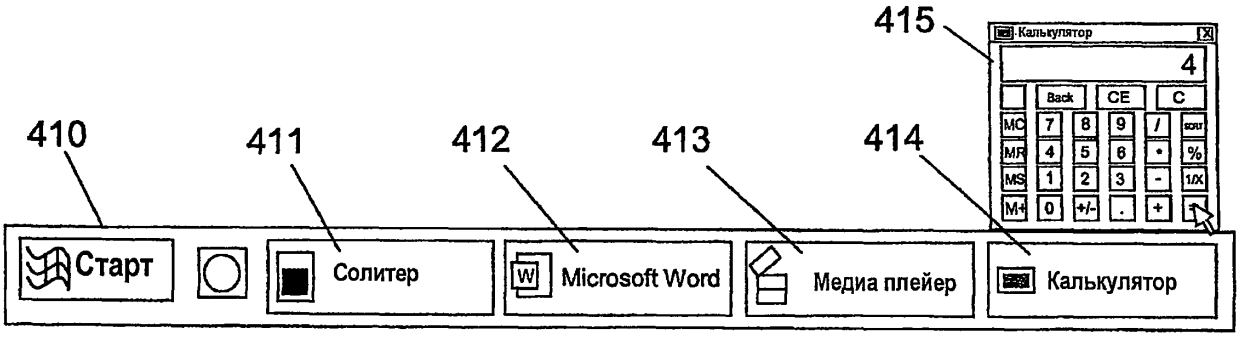
Фиг.3А



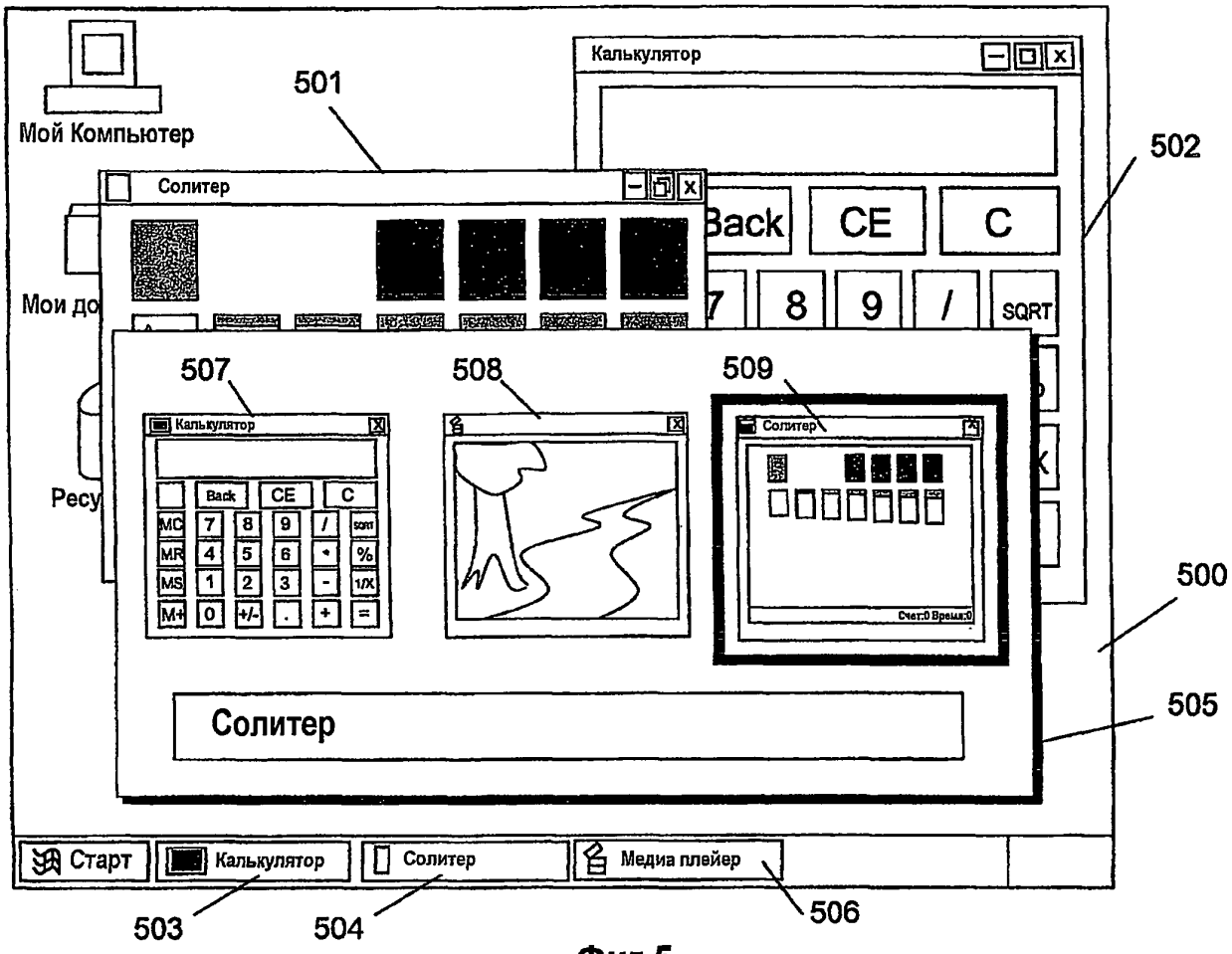
Фиг.3В



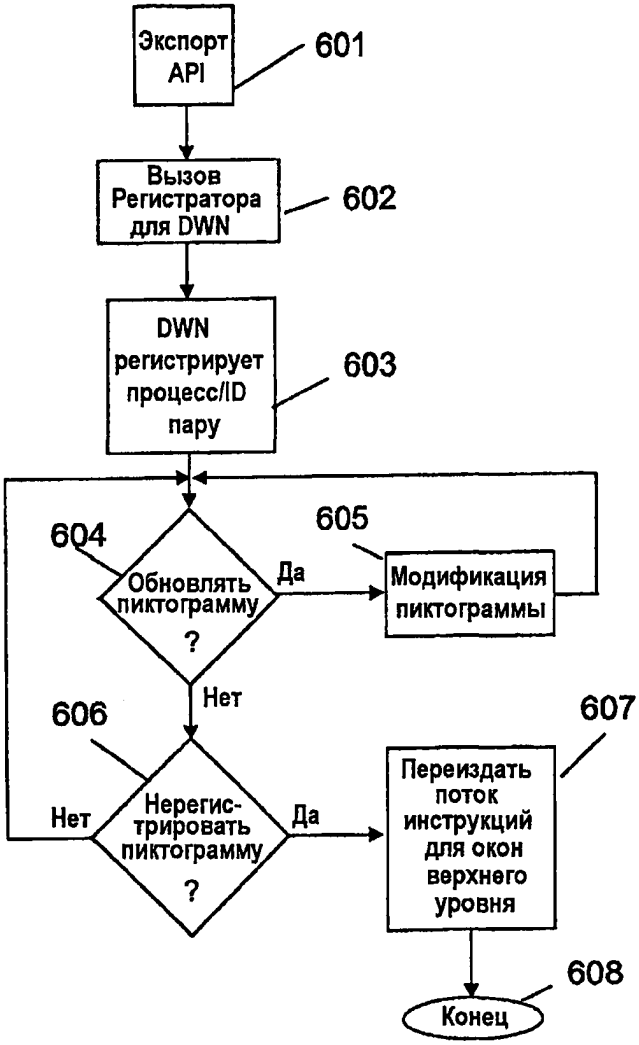
Фиг.4А



Фиг.4В


Мой Компьютер
501
Калькулятор
502
Солитер
Back
CE
C
8
9
SQRT
507
508
509
Калькулятор
Солитер
Счет:0 Время:0
500
505
Солитер
Старт
Калькулятор
Солитер
Медиа плейер
503
504
506

Фиг.5


Экспорт API
601
Вызов Регистратора для DWN
602
DWN регистрирует процесс/ID пару
603
604
Обновлять пиктограмму ?
Да
605
Модификация пиктограммы
Нет
606
Нерегис-трировать пиктограмму ?
Нет
Да
607
Переиздать поток инструкций для окон верхнего уровня
608
Конец

Фиг.6