(12) INTERNATIONAL APPLICATION PUBLISHED UNDER THE PATENT COOPERATION TREATY (PCT)

(19) World Intellectual Property Organization
International Bureau

(43) International Publication Date
13 August 2009 (13.08.2009)

PCT

(10) International Publication Number
WO 2009/099689 A1

(51) International Patent Classification:
G06T 1/00 (2006.01) *G06T 15/00* (2006.01)

(21) International Application Number:
PCT/US2009/030201

(22) International Filing Date:
6 January 2009 (06.01.2009)

(25) Filing Language: English

(26) Publication Language: English

(30) Priority Data:
12/024,848 1 February 2008 (01.02.2008) US

(71) Applicant *(for all designated States except US)*: **MICROSOFT CORPORATION** [US/US]; Sharon Rydberg (sharonr), LCA, International Patent Department, One Microsoft Way, 8/2321, Redmond, Washington 98052-6399 (US).

(72) Inventors: **QIU, Kan**; One Microsoft Way, Redmond, Washington 98052-6399 (US). **ABDO, Nadim Y.**; One Microsoft Way, Redmond, Washington 98052-6399 (US).

(81) Designated States *(unless otherwise indicated, for every kind of national protection available)*: AE, AG, AL, AM, AO, AT, AU, AZ, BA, BB, BG, BH, BR, BW, BY, BZ, CA, CH, CN, CO, CR, CU, CZ, DE, DK, DM, DO, DZ, EC, EE, EG, ES, FI, GB, GD, GE, GH, GM, GT, HN, HR, HU, ID, IL, IN, IS, JP, KE, KG, KM, KN, KP, KR, KZ, LA, LC, LK, LR, LS, LT, LU, LY, MA, MD, ME, MG, MK, MN, MW, MX, MY, MZ, NA, NG, NI, NO, NZ, OM, PG, PH, PL, PT, RO, RS, RU, SC, SD, SE, SG, SK, SL, SM, ST, SV, SY, TJ, TM, TN, TR, TT, TZ, UA, UG, US, UZ, VC, VN, ZA, ZM, ZW.

(84) Designated States *(unless otherwise indicated, for every kind of regional protection available)*: ARIPO (BW, GH, GM, KE, LS, MW, MZ, NA, SD, SL, SZ, TZ, UG, ZM, ZW), Eurasian (AM, AZ, BY, KG, KZ, MD, RU, TJ, TM), European (AT, BE, BG, CH, CY, CZ, DE, DK, EE, ES, FI, FR, GB, GR, HR, HU, IE, IS, IT, LT, LU, LV, MC, MK, MT, NL, NO, PL, PT, RO, SE, SI, SK, TR), OAPI (BF, BJ, CF, CG, CI, CM, GA, GN, GQ, GW, ML, MR, NE, SN, TD, TG).

Declarations under Rule 4.17:

— *as to applicant's entitlement to apply for and be granted a patent (Rule 4.17(ii))*

— *as to the applicant's entitlement to claim the priority of the earlier application (Rule 4.17(iii))*

Published:

— *with international search report (Art. 21(3))*

— *before the expiration of the time limit for amending the claims and to be republished in the event of receipt of amendments (Rule 48.2(h))*

(54) **Title**: GRAPHICS REMOTING ARCHITECTURE

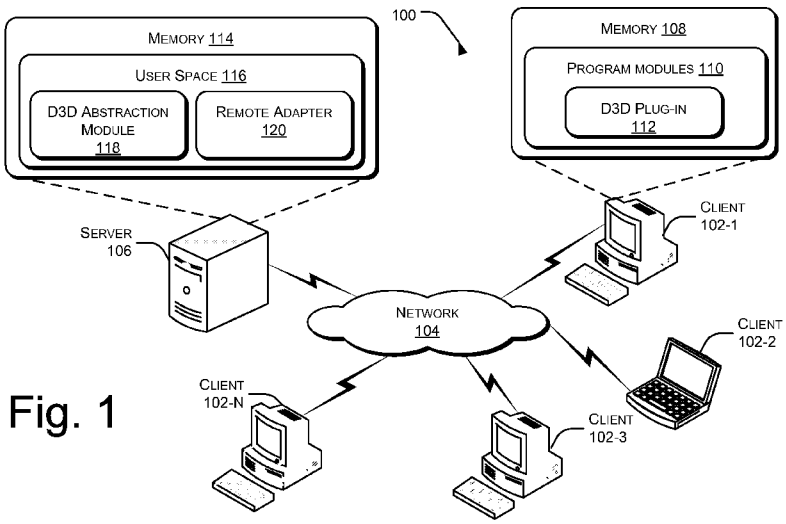


Fig. 1

(57) **Abstract**: Systems and methods to implement a graphics remoting architecture for rendering graphics images at remote clients are disclosed. In one implementation, when a D3D application hosted on a remote server is used by a remote client, the graphics associated with the D3D application are created and rendered at the remote client. For this, the D3D commands and D3D objects corresponding to the graphics are abstracted into data streams at the remote server. The data streams are then sent to the remote client. At the remote client, the D3D commands and D3D objects are extracted from the data streams and executed to create the graphics images. The graphics images are then rendered and displayed using output devices at the remote client.

WO 2009/099689 A1

GRAPHICS REMOTING ARCHITECTURE

BACKGROUND

[0001] Graphics software for creating high-end graphics applications, such as photo-editing, gaming, etc., may include animation and other tools that make use of complex calculations. Typically, such high-end graphics applications employ various technologies to provide graphics hardware support to manage the complex calculations. For example, graphics applications ran on Microsoft® Windows® operating system, may employ Direct3D rendering technologies to provide graphics hardware support. Direct3D is a three-dimensional (3D) graphics Application Programming Interface (API) that includes commands for rendering 3D images in graphics applications.

[0002] Although most personal computers (PCs) currently provide such graphics hardware support, there has been no efficient way of remoting such technologies. For example, in remote sessions, the graphics applications may not scale up with the window size or the number of user connections and can also create stability issues. Furthermore, graphics acceleration may not be available on server computers running Video Graphics Adapter (VGA) or on virtual machines. As a result, graphics applications running over remote sessions can cause poor end user experiences.

SUMMARY

[0003] This summary is provided to introduce simplified concepts related to Graphics remoting architecture, which are further described below in the Detailed Description. This summary is not intended to identify essential features of the claimed subject matter, nor is it intended for use in determining the scope of the claimed subject matter.

[0004] In an embodiment, a three dimension (3D) application hosted on a server computer is used by a remote client computer. The graphics associated with the 3D application are created and rendered at the remote client computer, by the use of 3D commands and 3D objects corresponding to graphics abstracted into data streams at the remote server and sent to the client computer. The 3D commands and 3D

objects are extracted from the data streams and executed to create the graphics images at the client computer.

BRIEF DESCRIPTION OF THE DRAWINGS

[0005] The detailed description is set forth with reference to the accompanying figures. In the figures, the left-most digit(s) of a reference number identifies the figure in which the reference number first appears. The use of the same reference numbers in different figures indicates similar or identical items.

[0006] Fig. 1 is a block diagram illustrating an exemplary network environment for implementing a 3D remoting architecture.

[0007] Fig. 2 is a block diagram illustrating an exemplary computing server for implementing remote rendering of 3D images.

[0008] Fig. 3 is a block diagram illustrating an exemplary client for implementing remote rendering of 3D images.

[0009] Fig. 4 is a block diagram illustrating an exemplary implementation of a 3D remoting architecture.

[0010] Fig. 5 is a flowchart illustrating an exemplary process for abstracting commands in 3D graphics applications.

[0011] Fig. 6 is a flowchart illustrating an exemplary process for rendering 3D images.

DETAILED DESCRIPTION

[0012] This disclosure relates to methods and systems for implementing a remoting architecture to facilitate rendering of three dimensional or 3D images at a remotely located client and achieve an enhanced end user experience in a remote session. Typically, in a remote environment, such as a Terminal Services® environment, any graphics-intense application commands, such as Direct3D or D3D commands, are executed at a server. After execution of the commands, the output, such as graphics images, is rendered with the help of the Graphics Processing Unit (GPU) of the server. The rendered graphics images are then sent to a client device as a bitmap image and are displayed through a display device at the client side. Running Direct3D or three dimension applications in such a manner is limited by architectural factors such as server availability, scalability, server stability, and

causes poor end user experience. In particular, architectural limitations may include a lack of frame rate control and inadequateness of progressive texture rendering when compression and other specialized methods are employed.

[0013] Techniques for implementing Direct3D remoting architecture to enable rendering of 3D graphics images at a remote client are disclosed in the following description. In one implementation, instead of rendering graphics images at a server computer and then sending the final bitmap images to a client device. The D3D commands can be sent directly to the client device that may then execute the D3D commands. A GPU at the client device end renders the output such as the graphics images and displays the images at the client end. Thus the graphics images can be rendered at the client device without employing the GPU at the server end. Furthermore, techniques for implementing frame rate control and progressive texture rendering methods to overcome the above mentioned architectural limitations are disclosed.

[0014] Methods and systems to implement a Direct3D remoting architecture for rendering 3D graphics images at a client device are disclosed in the following description.

Exemplary Network Environment

[0015] Fig. 1 illustrates an exemplary network or computing environment 100 for implementing a 3D or Direct3D (D3D) remoting architecture, which may implement techniques described herein, and which may be representative, in whole or in part, of elements described herein. The computing environment 100 is an example of a computing environment, and is not intended to suggest any limitation to the scope of use and/or functionality of the computer and network architectures used.

[0016] The computing environment 100 includes a plurality of client devices 102-1, 102-2, 102-3, …, 102-N (collectively referred to as clients 102). The clients 102 are communicatively associated through a network 104 with a server computer or server 106. The clients 102 may be any of a variety of conventional computing devices, for example, a desktop PC, a notebook or a portable computer, a Personal Digital Assistant (PDA), a workstation, a mobile computing device, or an Internet

appliance, etc. Each client 102 may include a memory that stores different program modules for rendering. For example, the client 102-1 includes a memory 108 that may store different program modules 110 such as a D3D Plug-in 112 and other modules to perform rendering.

[0017] The network 104 may be a wireless or a wired network, or a combination thereof. The network 104 may also be represented as a collection of individual networks interconnected with each other and functioning as a single large network (e.g., the Internet or an intranet). Examples of such individual networks include, but are not limited to, Local Area Networks (LANs), Wide Area Networks (WANs), and Metropolitan Area Networks (MANs). Further, the individual networks may also be wireless or wired networks, or a combination thereof.

[0018] The server 106 may be, for example, a general purpose computing device or a cluster of servers and mainframes. The server 106 includes a memory 114 having a user space 116 that corresponds to a client, such as client 102-1. The user space 116 further includes a D3D abstraction module 118 and a remote adapter 120. The remote adapter 120, at the server 106, establishes a connection with the client to send graphics commands abstracted by the D3D abstraction module 118 for rendering.

[0019] In one implementation, the D3D abstraction module 118 executes on a Terminal Server (TS)/D3D abstraction layer. The D3D abstraction layer is implemented between the physical hardware of the server 106 and the graphics applications such as D3D applications. The abstraction layer hides the implementation details of the hardware from such applications.

[0020] The D3D abstraction module 118 receives one or more D3D commands and D3D objects from a D3D application, when a client such as the client 102-1 uses the D3D application at the server 106 over a remote session. The D3D abstraction module 118 further abstracts the communication between the D3D application and graphics hardware that execute the D3D application.

[0021] The D3D abstraction module 118 abstracts the D3D commands and D3D objects into data streams that can be sent over the network 104. The D3D abstraction module 118 thus enables the server 106 to send the abstracted

commands and objects as data streams to one or more of the clients 102. The data streams are sent via the network 104 based on different remoting protocols, such as Remote Desktop Protocol (RDP), D3D Remoting Protocol, etc.

[0022] To send the data streams from the server 106 to the client 102-1, the remote adapter 120 establishes a communication channel with the D3D plug-in 112 of the client 102-1 over the network 104. The remote adapter 120 checks availability and capability of the display adapter at the client 102-1 from the D3D plug-in 112. On receiving confirmation from the D3D plug-in 112, the remote adapter 120 establishes the communication channel between the D3D abstraction module 118 and the D3D plug-in 112.

[0023] The D3D abstraction module 118 sends the abstracted D3D commands and D3D objects associated with the D3D application to the D3D plug-ins at the client 102-1. In one implementation, the D3D abstraction module 118 sends the abstracted D3D commands and the D3D objects as hidden data to the D3D plug-in 112.

[0024] At the client-end (i.e., client 102), the data streams are received and processed by one or more D3D plug-ins, such as the D3D plug-in 112. The D3D plug-in 112 receives the data streams, and parses the data streams to extract the D3D commands. Furthermore, the D3D plug-in creates actual D3D objects, executes the D3D commands, and renders the output such as graphics images using the GPU at the client end.

Exemplary Server

[0025] Fig. 2 is an exemplary server 106 for implementing remote rendering of 3D images. The server 106 can be a computing device that includes a processor 202, network interface(s) 204, input/output interface(s) 206, and the memory 114. The network interface(s) 204 may include one or more ports for connecting a number of computing devices, such as the clients 102 to each other, or to the server 106. The network interface(s) 204 can facilitate communications with a wide variety of networks and protocol types, including wired networks (e.g., LAN, cable, etc.) and wireless networks (e.g., WLAN, cellular, satellite, etc.).

[0026] The input/output interface(s) 206 may include, for example, a scanner a port, a mouse port, a keyboard port, etc., to receive/send data from/to the clients 102. In one implementation, the input/output interface(s) 206 of the server 106 send data, such as the D3D commands and the D3D objects over the network 104. The input/output interface(s) 206 of the server 106 receive data from one or more clients 102, such as data or contents that represent the rendered graphics images for synchronization purposes.

[0027] The memory 114 can be any computer-readable media in the form of both volatile memory, such as Random Access Memory (RAM), and/or non-volatile memory, such as Read Only Memory (ROM) and a flash RAM. The memory 114 typically includes data and/or program modules that are immediately accessible to and/or presently operated on by the processor 202.

[0028] In one implementation, the memory 114 includes a user space 116 corresponding to each of the clients 102, and program data 208. Each user space 116 may include a D3D abstraction module 118, and a remote adapter 120 in addition to other modules 210. The program data 208 may store parameters including D3D commands 212, D3D objects 214, data streams 216, and other program data 218.

[0029] The D3D abstraction module 118 can be configured to monitor data traffic that includes the D3D commands 212 and the D3D objects 214. As described in further detail below, the D3D abstraction module 118 can be configured to receive the D3D commands 212 and the D3D objects 214 from the D3D application through a D3D API. The D3D API includes functions that can be used to render 3D images in the D3D application. In another implementation, the D3D abstraction module 118 can be configured to receive the D3D commands 212 and the D3D objects 214 directly from the D3D application.

[0030] In one implementation, the received D3D commands 212 may be high frequency commands, which may require one or more network roundtrips from the clients 102. Such network roundtrips may cause delay in execution of the high frequency commands. To avoid such network roundtrips, the D3D abstraction module 118 can be configured to assign a handle or an identifier to each of the D3D

objects in the abstraction layer. The D3D abstraction module 118 then records the handle in the data streams 216 along with other parameters for creating the D3D object on the client-end. The handles can be stored in the other program data 218 in the memory 114 of the server 106.

[0031] The D3D abstraction module 118 abstracts the D3D commands 212 and the D3D objects 214 into the data streams 216 and sends them over the network 104. For this, the remote adapter 120 enquires about the availability and capability of the display adapter of the client 102-1. Upon receiving confirmation of availability and capability from the D3D plug-in 112 at the client 102-1, the remote adapter 120 establishes a communication channel between the D3D abstraction module 118 and the D3D plug-in 112. The D3D abstraction module 118 sends the abstracted D3D commands 212 and the D3D objects 214 to the D3D plug-in 112 of the client 102-1 via the communication channel.

[0032] In addition to abstraction, the D3D abstraction module 118 can be configured to perform centralized resource encoding. The centralized resource encoding encompasses compression and decimation methods known in the art for encoding the available D3D resources. This enables efficient use of the bandwidth of the server 106 for multiple applications and reconnections. For example, a D3D application may utilize D3D resources, such as a texture map, mesh data, or a normal bitmap image that may be larger in size than the available bandwidth. In such cases, the D3D abstraction module 118 employs known compression and/or decimation methods to reduce the size of the resources. The resource encoding process can be centralized so that a system wide persistent caching can be performed to use the bandwidth of the server 106 in an efficient manner.

Exemplary Client

[0033] Fig. 3 illustrates an exemplary client 102 for implementing remote rendering of 3D images. The exemplary client 102 can be any of the clients 102, for example client 102-1, and includes a processor 302, network interface(s) 304, input/output interface(s) 306, the memory 108, and input/output device(s) 310. The network interface(s) 304 facilitate communication between the clients 102 across the network 104.

[0034] In the exemplary implementation, the network interface(s) 304 may include one or more ports for connecting the client 102-1 either to the server 106 or to the clients 102 other than the client 102-1. The network interface(s) 304 can facilitate communications with a wide variety of networks and protocol types, including wired networks (e.g.,LAN, cable, etc.) and wireless networks (e.g., WLAN, cellular, satellite, etc.).

[0035] The input/output interface(s) 306 may include, for example, a scanner a port, a mouse port, a keyboard port, etc., to receive/send data from/to the clients 102. In one implementation, the input/output interface(s) 306 of the client 102-1 receive data, such as, for example, the D3D commands 212 and the D3D objects 214 over the network 104.

[0036] The input/output interface(s) 306 of the client 102-1 send data to the server 106, such as data or contents that represent the rendered graphics images for synchronization purposes. The input/output interface(s) 306 can display data, such as rendered graphics images by the input/output devices 310.

[0037] The memory 108 can be any computer-readable media in the form of both volatile memory, such as Random Access Memory (RAM) and/or non-volatile memory, such as Read Only Memory (ROM) and/or a flash RAM. The memory 108 includes data and/or program modules that may be immediately accessible to and/or operated on by the processor 302. In the exemplary implementation, the memory 108 includes the program modules 110 and a program data 308. The program modules 110 may include the D3D Plug-in 112, and other modules 312. The program data 308 may store parameters including data streams 314, graphics images 316, and other program data 318.

[0038] The D3D Plug-in 112 receives the data streams 314 that include D3D commands 212 and D3D objects 214, and extracts the D3D commands 212. Furthermore, the D3D plug-in 112 executes the D3D commands 212 and renders output graphics images by using the GPU at the client end. In one implementation, the D3D plug-in 112 of the client 102-1 may be configured to receive the data streams 314 from the server over the network 104. The D3D Plug-in 112 can be

configured to parse the data streams 314 using any parsing methods known in the art, and extract D3D commands 212 from the data streams 314.

[0039] The D3D plug-in 112 can be configured to create D3D objects 214 with the use of handles received along with the data streams 314. As discussed above, a handle has an entry that includes the address of a resource associated with a graphics application such as a D3D application, so that the objects such as the D3D objects 214 can be identified, examined, created, and/or modified. Therefore, the D3D objects can be assigned handles to create the actual D3D objects 214 at the client end by the D3D plug-in 112. The D3D plug-in 112 can then execute the D3D commands 212, and render the graphics images 316 using the GPU at the client. The rendering of the graphics images 316 can be performed by rendering methods known in the art.

[0040] In addition to rendering, the D3D Plug-in 112 may be configured to track changes in geometry of the window displayed on the desktop of the server 106 to change accordingly the representation of the server desktop at the client 102-1. The changes in window geometry can include window resizing, window movement or positioning, and/or visible region changes in the desktop of the server 106.

[0041] To support changes in window geometry, a proxy window can be used at the client 102-1 to reproduce the geometrical changes occurring at the window of the server 106. The proxy window ensures interoperability with any window activated in the same session, such as windows at the server 106 and windows at the clients 102. The proxy window can be established at the clients 102, for example, when the desktop composition feature is disabled. Desktop composition is a feature that changes the way pixels of a graphics image are displayed on the display screen of the client and can be controlled by a Desktop Window Manager of the server 106.

[0042] The D3D Plug-in 112 may also be configured to present Direct3D graphics in the full screen exclusive mode on the client 102 side, for example, at the client 102-1. The full screen exclusive mode of display automatically captures, sets/restores screen resolution and pixel format without intervention from a user. A client window may be a simple window used for displaying images. However, a Direct3D application runs under a full screen exclusive mode of display, that may

not be supported by the simple window format. The D3D Plug-in 112 presents the graphics under full screen exclusive mode with full screen representation at the window of the clients, and enhancing the user experience.

[0043] The D3D Plug-in 112 may be configured to send information related to rendered graphics images 316 to the server 106 in case the server requires the information for synchronization. In one implementation, the D3D plug-in 112 renders the output graphics images at the window of the client 102-1 based on a client-side-only rendering protocol. In such a case, the content to be rendered is stored at the clients 102 by default, which can also be sent to the server 106 in case the server needs the content. Therefore, the D3D plug-in 112 can be further configured to allow tracking and synchronization between the perceived, invisible desktop of the server 106 and the visible client 102 side representation of the server 106 desktop.

[0044] Furthermore, to achieve synchronization and content coherency, the D3D plug-in 112 can be configured to track various Direct3D commands 212 issued from the server 106 and read/write accesses made to the desktops of the clients 102. The D3D plug-in 112 can further maintain updated information about each pixel sent/received by both the server 106 and the clients 102, such as the client 102-1. The information about the pixels can be updated for synchronization, for example when the desktop composition is disabled.

Exemplary Architecture

[0045] Fig. 4 illustrates an exemplary implementation of Direct3D (D3D) remoting architecture 400. Architecture 400 is a logical view of the D3D remoting architecture showing related components used to render graphics images in remote sessions. The architecture 400 can include a group of components that operate at a server side 402 (e.g., server 106). The components of the server side 402 receive D3D commands from the D3D application, abstract the D3D commands, and D3D objects as a data streams to be sent over the network to a client side 404 (e.g., client 102). The client side 404 includes a group of components related to a client 102, such as the client 102-1. The components at the client side 404 receive the data stream, extract the D3D commands, create actual D3D objects, execute the D3D

commands, and render the output graphics images at the client end. The architecture 400 may further include components accessed by the server 106 or the clients 102 in the same session referred to as a session space 406, and components accessed by the server 106 or the clients 102 in other sessions referred to as a global space 408.

[0046] In one implementation, the D3D application 410 at the server side 402 receives one or more D3D commands 212 as input from users. The D3D application 410 can be a high-end graphics application, such as a gaming application, that involves complex calculations. For such applications, the D3D commands 212 and D3D objects 214 are abstracted as the data streams 216 at a Terminal Server® TS/ D3D abstraction layer 412. In one implementation, a D3D abstraction module 118 can execute on the Terminal Server® TS/ D3D abstraction layer 412 to abstract the commands and objects. Furthermore, the D3D abstraction module 118 assigns handles to one or more of the D3D objects 214, such that the actual D3D objects 214 can be created using the handles at the client end.

[0047] For abstracting the D3D commands 212 and the D3D objects, the D3D abstraction module 118 communicates with a D3D API 414; however, in certain cases applications may implement a service side GPU (i.e., local GPU Driver 416) if it's available. If this is the case (or by default per OS policy) the GPU driver 416 may be present locally in the server 106 and can communicate with a Win32k process 418. In other words, remote adapter 120, representing remote GPU at a client (e.g., clients 102 such as the client 102-1) , can be running side by side with local GPU Driver 416 present; however local GPU Driver 416 may not be present for this remoting architecture to function. Local GPU Driver 416 illustrates how the remoting architecture could be integrated with existing local graphics architecture.

[0048] The Win32k process 418 is a Windows® operating system kernel process providing the lowest-level abstraction for computing resources, such as the memory, the processor(s), and input/output devices. The Win32k process 418 communicates with a Graphical Device Interface (GDI) GDI App 420 that performs the graphics related tasks. The GDI App 420 can be a Microsoft®

Windows® operating system interface used for representing graphical objects and transmitting the graphical objects to output devices, such as monitors and printers. The GDI App 420 can perform graphics related tasks such as drawing lines and curves, and rendering fonts.

[0049] In one implementation, for performing such graphics related tasks, the GDI App 420 uses a GDI32 process 422. The GDI32 process 422 includes functions for the Windows® operating system GDI of the server to create objects and process information for displaying images on the display window of a client 102. The GDI32 process 422 performs graphics related functions, such as creating D3D objects 214 for the D3D applications 410.

[0050] After creating the D3D objects 214, the D3D commands 212 and the D3D objects 214 are communicated to the D3D application 410 and D3D API 414. During the communication, the D3D abstraction module 118 at the D3D abstraction layer 412 abstracts the D3D commands 212 and the D3D objects 214 into the data streams 216. The D3D abstraction module 118 also abstracts handles of the D3D objects 214 into the data streams 216.

[0051] The D3D abstraction module 118 sends the data streams 216 to the clients 102 via the network 104. For this, the remote adapter 120 enquires about the availability and the capability of the display adapters of the clients 102 such as the client 102-1. On receiving a confirmation from the D3D plug-in 112 of the client 102-1, the remote adapter 120 establishes a communication channel to send the data streams 216 over the network 104.

[0052] At the client side 404, the D3D plug-in 112 receives data streams 314 via the communication channel established by the remote adapter 120. The data streams 314 received by the D3D plug-in 112 includes the D3D commands 212, the D3D objects 214, and handles to recreate the actual D3D objects 214. The D3D plug-in 112 parses the received data streams 314 using any parsing methods known in the art and extracts the D3D commands 212 and the D3D objects 214.

[0053] The D3D plug-in 112 then executes the D3D commands 212 to display the graphics associated with the D3D application 410. For this, the D3D plug-in 112 communicates with a D3D API 424 of the clients 102 such as the client 102-1. The

D3D API 424 communicates the D3D commands 212 and D3D objects 214 between the D3D plug-in 112 and the graphics hardware that executes the graphics applications. The D3D API 424 also includes commands for rendering graphics images that can be used to display the graphics.

[0054] The D3D API 424 further interacts with a Win32k process 426 to perform graphics related tasks such as creation of actual D3D objects 214 using the handles. The Win32k process 426 (a Windows® operating system kernel process) may be similar to the Win32k process 418 of the server 106. For the creation of actual D3D objects 214, the Win32k process 426 interacts with a GDI32 process 428, a Graphical Device Interface process that creates the actual D3D objects 214.

[0055] After creating the actual D3D objects 214, the GDI32 process 428 at the client side 404 communicates the D3D objects 214 to the D3D plug-in 112 via the D3D API 424. Further, the D3D plug-in 112 executes the D3D commands 212 using the D3D objects 214 and renders the graphics images 316. The D3D plug-in 112 renders the graphics images 316 similar to a GPU driver. The rendered graphics images 316 are then displayed through the input/output devices 310 of the client 102-1. In an implementation, each of the clients 102 can be a remote terminal server client 430. The D3D plug-ins at the clients 102 can keep the D3D objects 214 alive in a session during the rendering process, even when the clients 102 get disconnected and try to reconnect later. After successful reconnection, the server 106 and the clients 102 will resume the execution of the D3D application 410.

[0056] In an implementation, displaying the graphics of the D3D application 410 involves using one or more D3D resources such as texture map, normal map, etc. that can be shared by one or more of the clients 102. For the purpose of sharing the resources, the D3D plug-in 112 can be configured to assign a handle or an identifier to each of the D3D resources; however, use of handles may require one or more network roundtrips from the clients 102. To avoid continuous network roundtrips, a centralized server component, can be used as a proxy for the clients 102 to provide the assigned handles. For example, a RDPDD (Remote Desktop Protocol Device Driver) 432 or a RDPWD (Remote Desktop Protocol Win32k Driver) 434

can be used in the session space 406 or global space 408 respectively as a proxy for the clients 102.

[0057] Furthermore, one or more methods such as progressive texture rendering methods, frame rate control methods and auto-reconnection methods can be implemented in the D3D remoting architecture. In one implementation, the D3D application 410 may also deal with D3D resources such as meshes and textures that consume more network bandwidth in order to achieve better end user experience. In such cases, the progressive texture rendering method can be configured to allocate the bandwidth efficiently and achieve better end user experience.

[0058] Further, the D3D abstraction module 118 can use frame rate control methods that allow the server 106 to regulate the bandwidth consumption by limiting the frame transfer rate based on the type of the D3D application 410. Such frame rate control methods can improve scalability and consistency among multiple connections and applications.

[0059] In addition, auto-reconnection methods can be implemented to provide seamless remote auto-reconnection experience from the D3D application 410. For example, when a client 102 gets disconnected during the execution of the D3D application 410, a new client that is capable of supporting the running D3D application 410 may be connected. In one implementation, during the reconnection of a new client, the D3D application 410 receives a message such as DEVICE_LOST or DEVICE_REMOVED. The D3D abstraction module 118 then restores the D3D application 410 to the pre-disconnection state and resumes the execution of the D3D application 410. In another implementation, during the reconnection of a new client, the D3D abstraction module 118 performs complete abstraction of the D3D commands 212 and the D3D objects that are needed to recreate the object content at the server side 402.

[0060] Additional methods can be employed to ensure that the server 106 does not accumulate unbounded number of D3D commands 212 during a disconnected session. In the present implementation, to avoid unbounded number of D3D commands 212 at the server 106, one or more kernel threads that call the D3D application 410 can be blocked.

Exemplary Processes

[0061] Exemplary methods of implementing Direct3D remoting architecture are described below. Some or all of these methods may, but need not, be implemented at least partially in an architecture such as that shown in Figs. 1-4. Also, it should be understood that certain acts in the methods need not be performed in the order described, may be modified, and/or may be omitted entirely, depending on the circumstances.

[0062] Fig. 5 illustrates an exemplary process 500 for abstracting commands in D3D graphics applications at the server end. The order in which the exemplary process 500 is described is not intended to be construed as a limitation, and any number of the described method blocks can be combined in any order to implement the method, or an alternate method. Additionally, individual blocks may be deleted from the method without departing from the spirit and scope of the subject matter described herein. Furthermore, the method can be implemented in any suitable hardware, software, firmware, or combination thereof.

[0063] At block 502, one or more graphics application commands can be identified and received from a graphics application. The D3D abstraction module 118 may monitor and receive the graphics application commands from the D3D application 410 at the D3D abstraction layer 412.

[0064] At block 504, the application commands are communicated to the graphics hardware drivers through an API. The D3D abstraction module 118 may communicate the received commands 212 to the graphics hardware driver through the D3D API 414.

[0065] At block 506, the application commands and objects generated by the graphics hardware driver are abstracted in a data stream. The D3D abstraction module 118 may abstract the generated D3D commands 212 and the D3D objects 214 as hidden data in the data streams 216 to be sent over the network 104. Further, the D3D abstraction module 118 assigns a handle or an identifier to one or more D3D objects. The D3D abstraction module 118 records the handle in the data streams 216 along with other parameters that can be used to create the D3D objects 214 at the client 102-1.

[0066] At block 508, the data streams are sent over the network to the client. The D3D abstraction module 118 may send the data streams 216 to one or more of the clients 102 for example the client 102-1. The data streams 216 are then received by the D3D plug-in 112 of the client 102-1 as the data streams 314.

[0067] In one implementation, the D3D commands 212 and the D3D objects 214 in the data streams 216 can be compressed using any compression and decimation methods known in the art. Such techniques can be used, if, for example, the D3D resources such as textures, mesh data, and so on are large in size, and are unable to fit in the bandwidth of the network 104. The compressed D3D resources can be then sent over the network 104 through the remote adapter 120.

[0068] The remote adapter 120 enables the D3D abstraction module 118 to communicate with the D3D plug-in 112 of the client 102-1. For this, the remote adapter 120 checks availability and the capability status of the display adapter of the client 102-1. Once the confirmation is received from the D3D plug-in 112, the remote adapter 120 establishes a communication channel between the D3D abstraction layer 412 and the D3D plug-in 112. The D3D abstraction module 118 sends the data streams 216 via the established communication channel over the network 108 to the clients 102.

[0069] Fig. 6 illustrates an exemplary process for rendering 3D images. The order in which the exemplary process 600 is described is not intended to be construed as a limitation, and any number of the described method blocks can be combined in any order to implement the method, or an alternate method. Additionally, individual blocks may be deleted from the method without departing from the spirit and scope of the subject matter described herein. Furthermore, the method can be implemented in any suitable hardware, software, firmware, or combination thereof.

[0070] At block 602, data streams are received from the server. The D3D plug-in 112 of the client 102-1, may receive the data streams 314 from the D3D abstraction module 118 at the server 106.

[0071] At block 604, the D3D commands and the D3D objects are extracted from the data stream. The D3D Plug-in 112 may extract the D3D commands 212 and the

D3D objects 214 from the data streams 314. Furthermore, the D3D Plug-in 112 can execute the D3D commands 212 with the help of the GPU of the client 102-1.

[0072] At block 606, the output in the form of graphics images is rendered through input/output devices at the client. The output of the D3D application 410, such as 3D graphics images 316, can be rendered through the input/output devices 310 of the client 102-1 using any of the rendering algorithms known in the art.

[0073] Any of the acts described above with respect to any method may be implemented by a processor or other computing device based on instructions stored on one or more computer-readable media. Computer-readable media can be any available media that can be accessed locally or remotely by the resource modeling application. By way of example, and not limitation, computer-readable media may comprise computer storage media and communication media. Computer storage media includes volatile and nonvolatile, removable and non-removable media implemented in any method or technology for storage of information such as computer-readable instructions, data structures, program modules, or other data. Computer storage media includes, but is not limited to, RAM, ROM, EEPROM, flash memory or other memory technology, CD-ROM, digital versatile disks (DVD) or other optical storage, magnetic cassettes, magnetic tape, magnetic disk storage or other magnetic storage devices, or any other medium which can be used to store the desired information and which can be accessed by the resource modeling application. Communication media typically embodies computer-readable instructions, data structures, program modules, or other data on any information delivery media. Combinations of the any of the above should also be included within the scope of computer-readable media.

CONCLUSION

[0074] Although the invention has been described in language specific to structural features and/or methodological acts for implementing self-described rendering of data, it is to be understood that the invention is not necessarily limited to the specific features or acts described. Rather, the specific features and acts are disclosed as exemplary forms of implementing the invention.

CLAIMS

What is claimed is:

1. A server computer comprising:

a memory;

a processor operatively coupled to the memory; and

a user space in the memory, that corresponds to one or more clients, wherein the user space includes a abstraction module to monitor data traffic that includes (three dimension) 3D commands and 3D objects.

2. The server computer of claim 1, wherein the memory includes program data that stores the 3D commands, the 3D objects, and data streams.

3. The server computer of claim 1, wherein the user space includes a remote adapter that enquires as to availability and capability of a display adapter at a client device.

4. The server computer of claim 3, wherein the remote adapter establishes a communication channel between the abstraction module and a plug-in of the client device, wherein the plug-in receives commands and 3d objects.

5. The server computer of claim 1, wherein the abstraction module is configured to receive the 3D commands and 3D objects from a 3D application through an application program interface (API) that includes functions to render 3D images in the 3D application.

6. The server computer of claim 1, wherein the abstraction module assigns identifiers to the 3D objects.

7. The server computer of claim 1, wherein the abstraction module abstracts the 3D commands and 3D objects into a data stream and sends the data stream to the one or more clients.

8. The server computer of claim 1, wherein the abstraction module performs centralized encoding by compression and decimination.

9. The server computer of claim 1, wherein the 3D commands are high frequency commands.

10. A client computer comprising:

a memory;

a processor operatively coupled to the memory;

a 3D plug-in in the memory that receives that receives data streams the include 3D commands and 3D objects, and extracts the 3D commands; and

a graphics processing unit (GPU) that renders the extracted 3D commands.

11. The client computer of claim 10, wherein the 3D plug-in parses the data streams and extracts 3D commands.

12. The client computer of claim 10, wherein the 3D plug-in creates 3D objects by identifying handles associated with the 3D objects in the data streams.

13. The client computer of claim 12, wherein the handles includes entries that include an address of a resource associated with a graphics application.

14. The client computer of claim 10, wherein the 3D plug-in presents 3D graphics in full screen user interface.

15. The client computer of claim 10, wherein the 3D plug-in sends synchronization information to a server.

16. The client computer of claim 10, wherein the 3D plug-in renders output graphics based on a client-side-only rendering protocol.

17. The client computer of claim 10, wherein the GPU generates a proxy window to produce geometrical changes occurring at a window in a server computer that sends the data streams.

18. A method implemented on computer executable media comprising:

receiving and identifying from graphics application commands;

communicating the graphics application commands to graphics drivers; and

abstracting the application commands and objects generated by the graphics drivers in a data stream.

19. The method of claim 18, wherein communicating is through an application program interface.

20. The method of claim 18, wherein the abstracting includes abstracting generated commands and objects as hidden data.

21. The method of claim 18 further comprising sending the data streams over a network to one or more clients.

22. The method of claim 21 further comprising compressing the data streams.

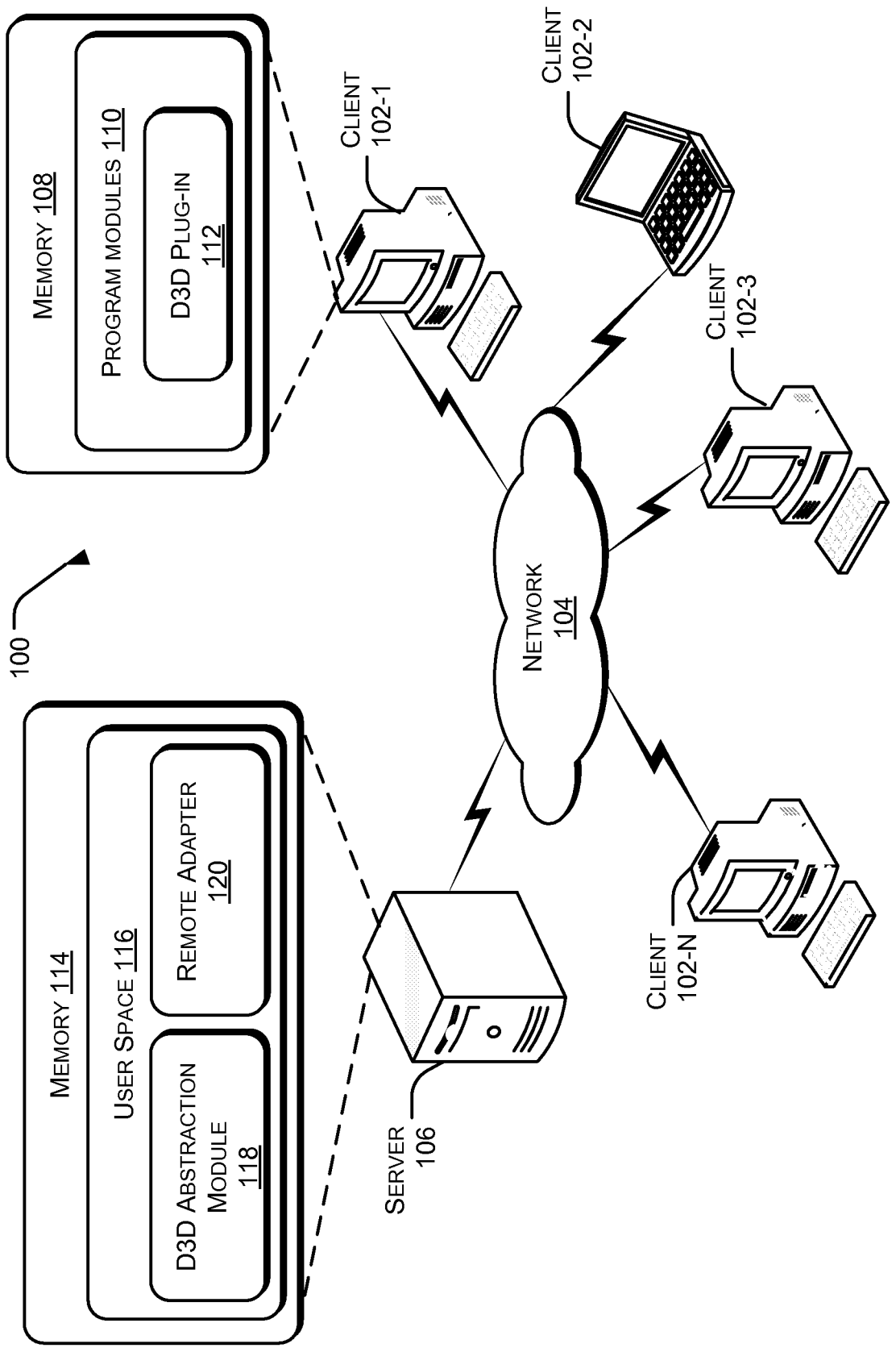

100
MEMORY 108
PROGRAM MODULES 110
D3D PLUG-IN 112
CLIENT 102-1
CLIENT 102-2
CLIENT 102-3
CLIENT 102-N
NETWORK 104
SERVER 106
MEMORY 114
USER SPACE 116
D3D ABSTRACTION MODULE 118
REMOTE ADAPTER 120

Fig. 1

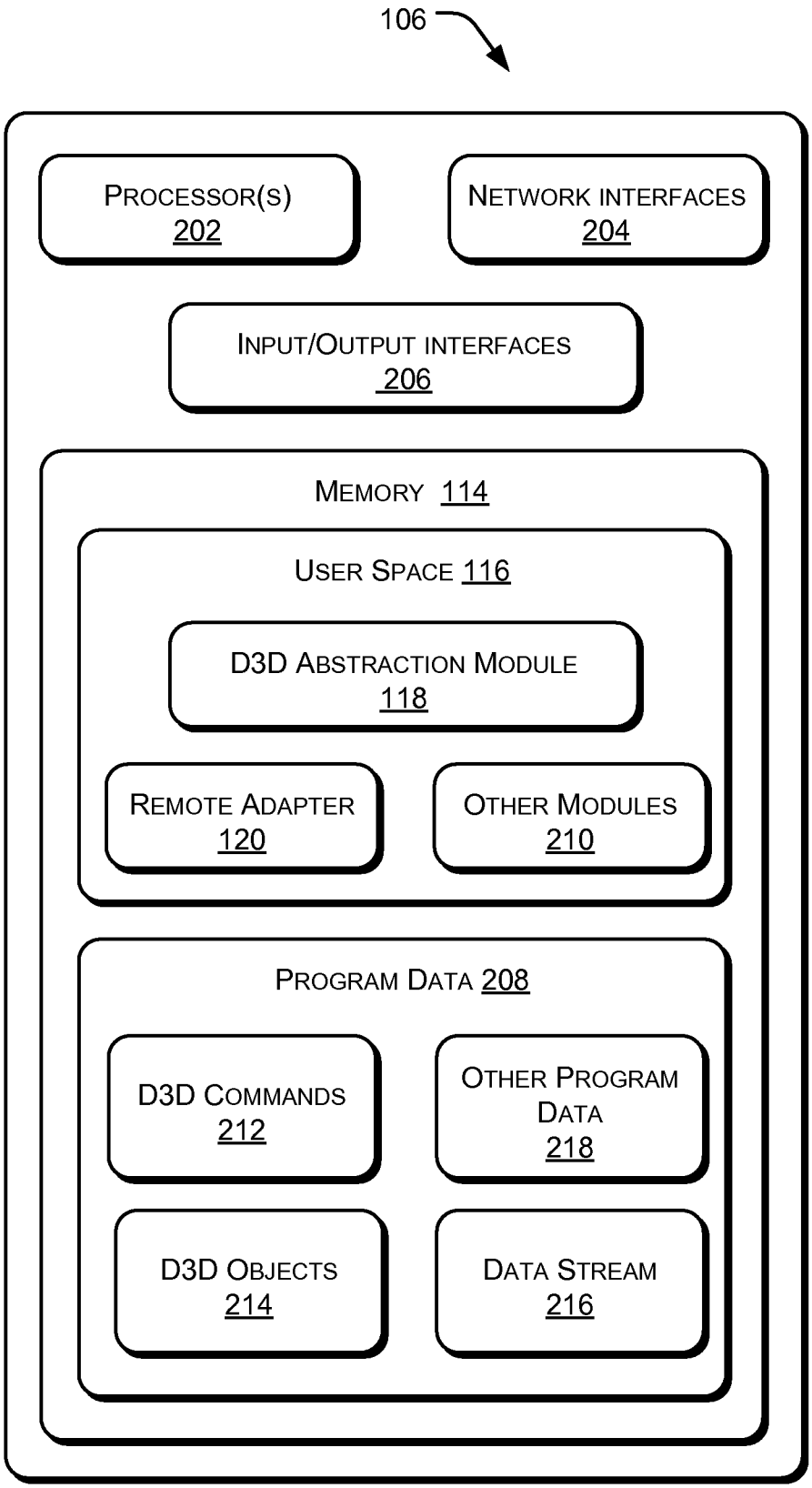

106
PROCESSOR(S) 202
NETWORK INTERFACES 204
INPUT/OUTPUT INTERFACES 206
MEMORY 114
USER SPACE 116
D3D ABSTRACTION MODULE 118
REMOTE ADAPTER 120
OTHER MODULES 210
PROGRAM DATA 208
D3D COMMANDS 212
OTHER PROGRAM DATA 218
D3D OBJECTS 214
DATA STREAM 216

Fig. 2

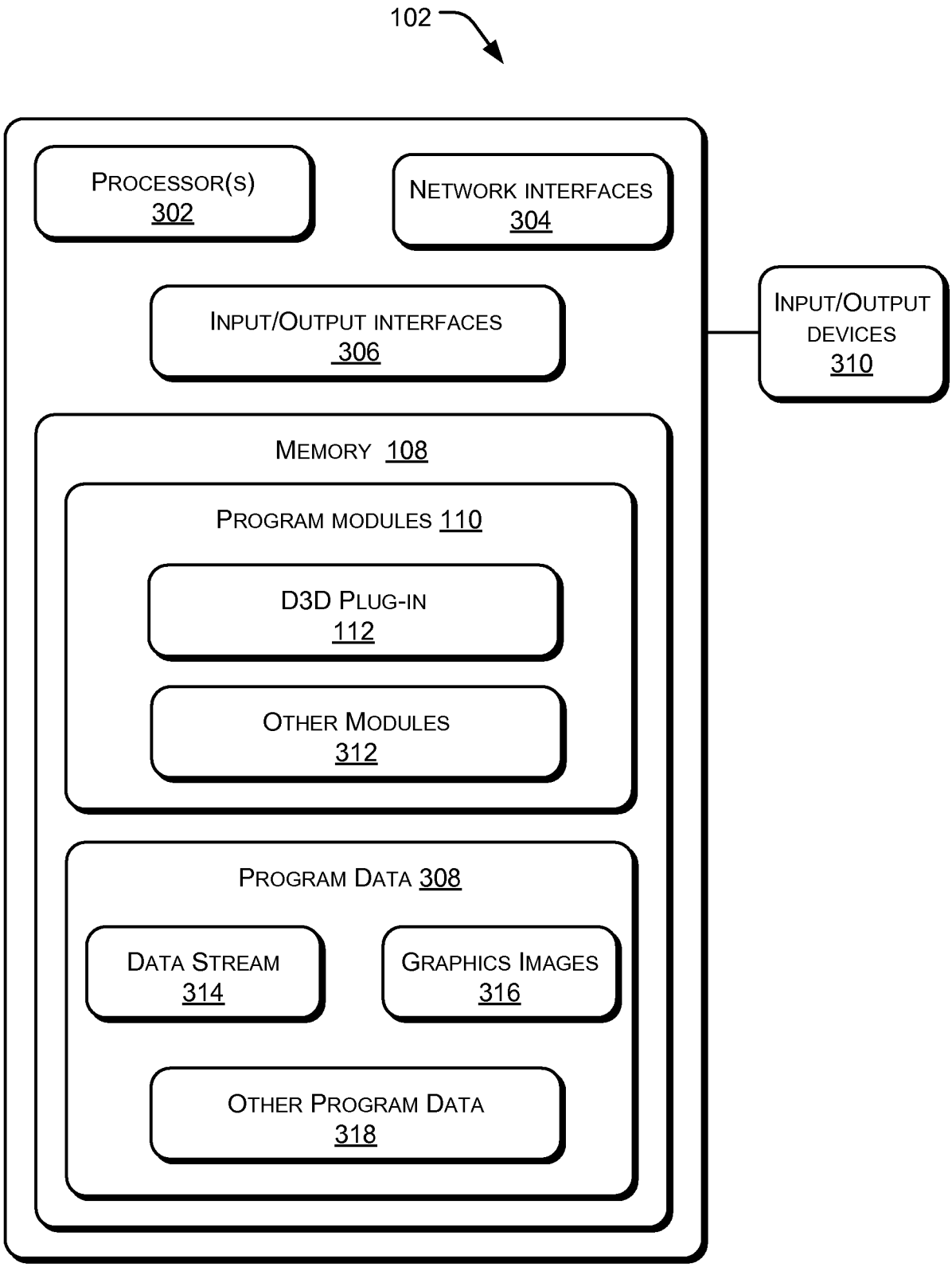

102
PROCESSOR(S)
302
NETWORK INTERFACES
304
INPUT/OUTPUT INTERFACES
306
INPUT/OUTPUT
DEVICES
310
MEMORY 108
PROGRAM MODULES 110
D3D PLUG-IN
112
OTHER MODULES
312
PROGRAM DATA 308
DATA STREAM
314
GRAPHICS IMAGES
316
OTHER PROGRAM DATA
318

Fig. 3

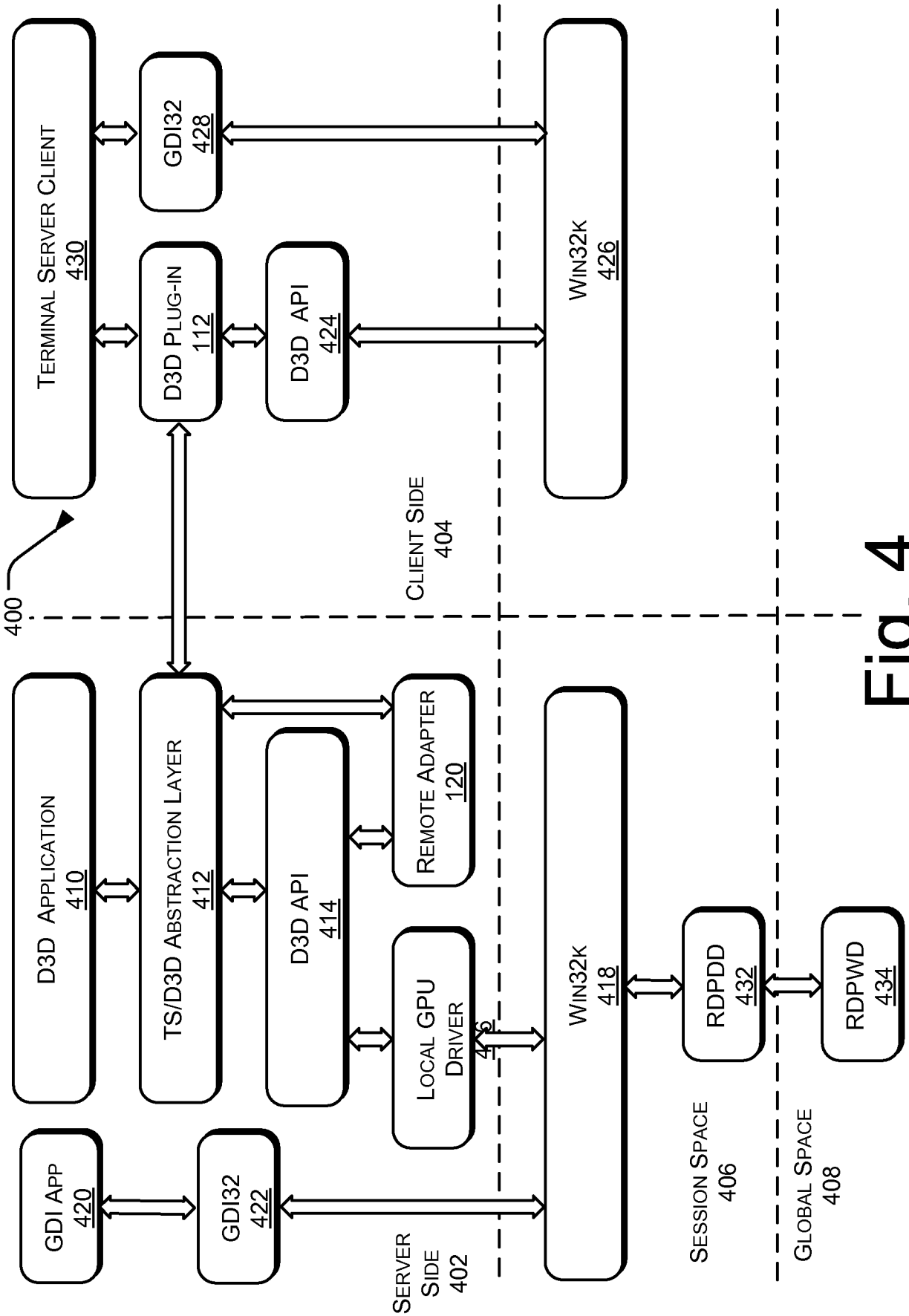

400
TERMINAL SERVER CLIENT 430
GDI32 428
D3D PLUG-IN 112
D3D API 424
WIN32K 426
CLIENT SIDE 404
D3D APPLICATION 410
TS/D3D ABSTRACTION LAYER 412
D3D API 414
REMOTE ADAPTER 120
LOCAL GPU DRIVER 416
WIN32K 418
GDI APP 420
GDI32 422
SERVER SIDE 402
SESSION SPACE 406
GLOBAL SPACE 408
RDPDD 432
RDPWD 434

Fig. 4

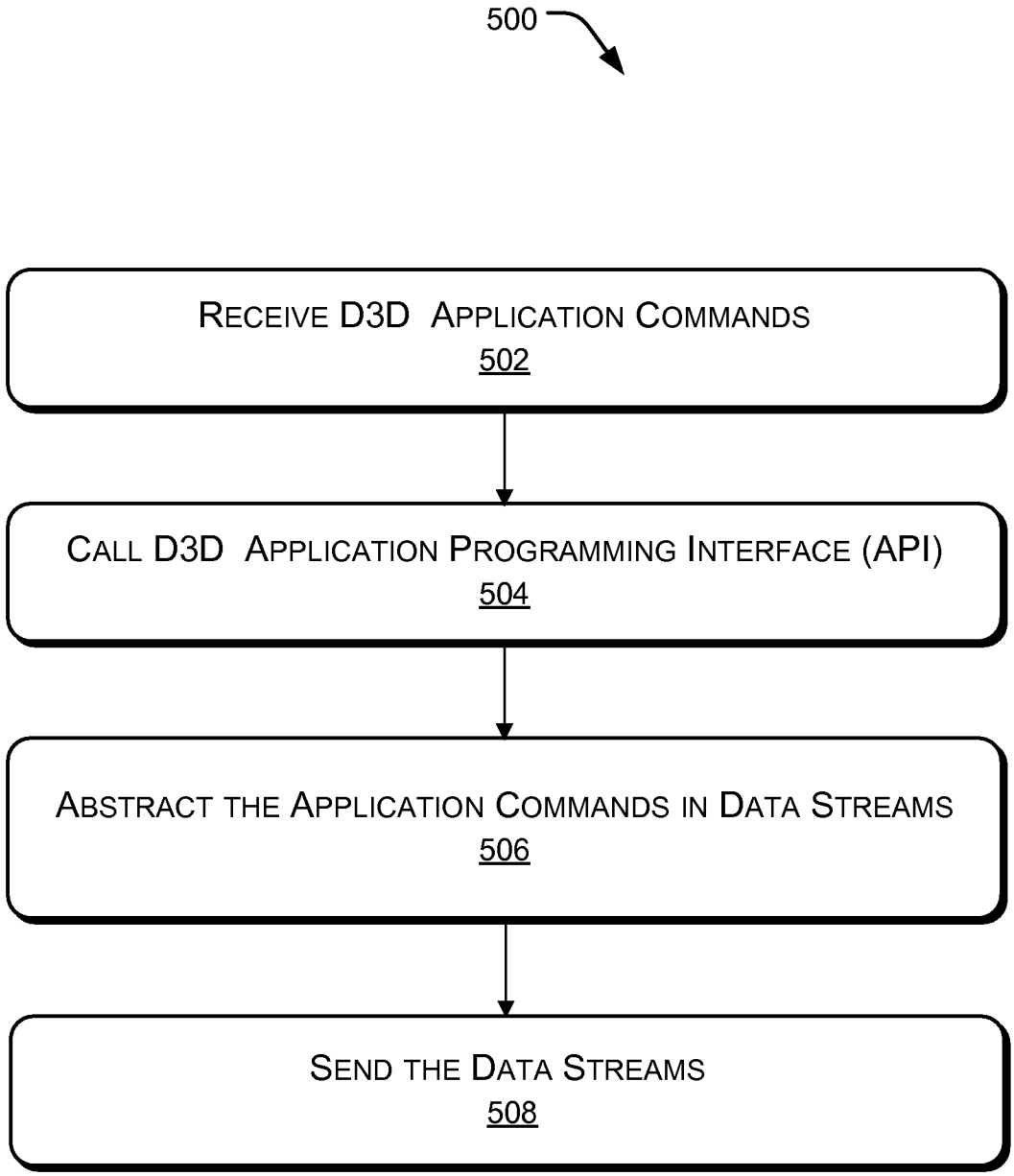

500
RECEIVE D3D APPLICATION COMMANDS
502
CALL D3D APPLICATION PROGRAMMING INTERFACE (API)
504
ABSTRACT THE APPLICATION COMMANDS IN DATA STREAMS
506
SEND THE DATA STREAMS
508

Fig. 5

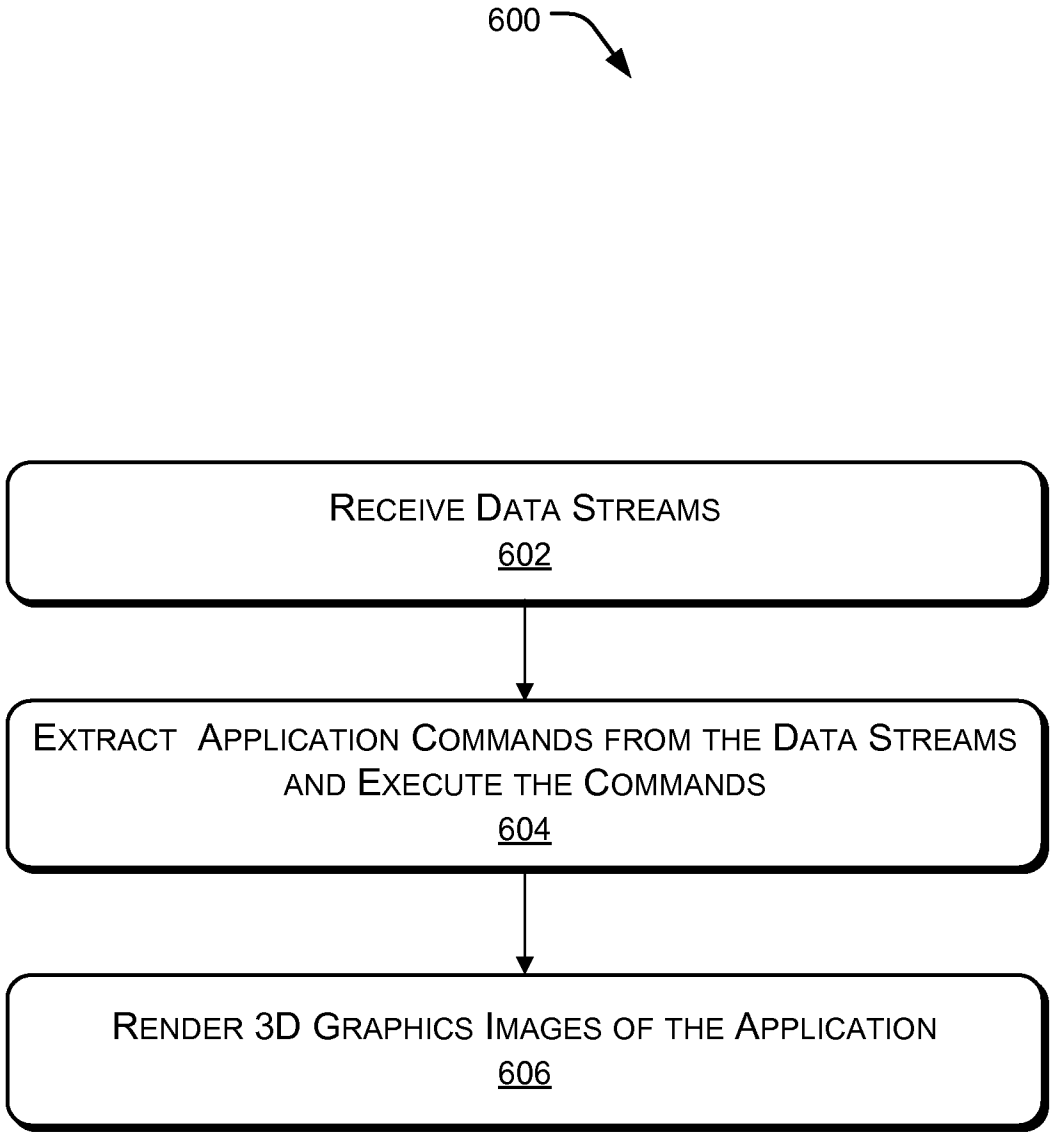

600
RECEIVE DATA STREAMS
602
EXTRACT APPLICATION COMMANDS FROM THE DATA STREAMS AND EXECUTE THE COMMANDS
604
RENDER 3D GRAPHICS IMAGES OF THE APPLICATION
606

Fig. 6

INTERNATIONAL SEARCH REPORT

International application No.
PCT/US2009/030201

A. CLASSIFICATION OF SUBJECT MATTER

G06T 1/00(2006.01)i, G06T 15/00(2006.01)i

According to International Patent Classification (IPC) or to both national classification and IPC

B. FIELDS SEARCHED

Minimum documentation searched (classification system followed by classification symbols)

IPC G06T, G06F

Documentation searched other than minimum documentation to the extent that such documents are included in the fields searched

Korean Utility models and applications for Utility models since 1975

Japanese Utility models and applications for Utility models since 1975

Electronic data base consulted during the international search (name of data base and, where practicable, search terms used)

eKIPASS, SEARCH TERMS: REMOTE, GRAPHIC, USER SPACE, and similar terms

C. DOCUMENTS CONSIDERED TO BE RELEVANT

Category*	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
X	KR 10-2006-0067248 A (ETRI), 19 June 2006 See abstract, page4, lines2-23	1, 10, 18
A		2-9, 11-17, 19-22
A	JP 2002-063601 A (SHIMOJO SHINJI et al.), 28 February 2002 See abstract, paragraphs[0049-0063]	1-22
A	WO 98-58478 A2 (IRONSIDE TECHNOLOGIES INC.), 23 December 1998 See abstract	1-22
A	US 5983218 A (TANVEER F. SYEDA-MAHMOOD), 09 November 1999 See abstract	1-22

☐ Further documents are listed in the continuation of Box C.

☒ See patent family annex.

* Special categories of cited documents:

"A" document defining the general state of the art which is not considered to be of particular relevance

"E" earlier application or patent but published on or after the international filing date

"L" document which may throw doubts on priority claim(s) or which is cited to establish the publication date of citation or other special reason (as specified)

"O" document referring to an oral disclosure, use, exhibition or other means

"P" document published prior to the international filing date but later than the priority date claimed

"T" later document published after the international filing date or priority date and not in conflict with the application but cited to understand the principle or theory underlying the invention

"X" document of particular relevance; the claimed invention cannot be considered novel or cannot be considered to involve an inventive step when the document is taken alone

"Y" document of particular relevance; the claimed invention cannot be considered to involve an inventive step when the document is combined with one or more other such documents,such combination being obvious to a person skilled in the art

"&" document member of the same patent family

Date of the actual completion of the international search	Date of mailing of the international search report
08 JUNE 2009 (08.06.2009)	**08 JUNE 2009 (08.06.2009)**
Name and mailing address of the ISA/KR Korean Intellectual Property Office Government Complex-Daejeon, 139 Seonsa-ro, Seo-gu, Daejeon 302-701, Republic of Korea Facsimile No. 82-42-472-7140	Authorized officer SHIN, Jae Chul Telephone No. 82-42-481-8215

Form PCT/ISA/210 (second sheet) (July 2008)

INTERNATIONAL SEARCH REPORT

Information on patent family members

International application No.

PCT/US2009/030201

Patent document cited in search report	Publication date	Patent family member(s)	Publication date
KR 10-2006-0067248 A	19.06.2006	NONE	
JP 2002-063601 A	28.02.2002	NONE	
WO 98-58478 A2	23.12.1998	AU 8005798 A	04.01.1999
		EP 0988739 A2	29.03.2000
		GB 9929428 D0	09.02.2000
		JP 2002-505776 A	19.02.2002
US 5983218 A	09.11.1999	JP 11-066106 A	09.03.1999