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

(19) World Intellectual Property Organization
International Bureau

(43) International Publication Date
29 November 2012 (29.11.2012)

WIPO | PCT

(10) International Publication Number
WO 2012/161102 A1

(51) International Patent Classification:
G06T 15/00 (2011.01) *A63F 13/12* (2006.01)
A63F 13/00 (2006.01)

(21) International Application Number:
PCT/JP2012/062720

(22) International Filing Date:
11 May 2012 (11.05.2012)

(25) Filing Language: English

(26) Publication Language: English

(30) Priority Data:
61/489,761 25 May 2011 (25.05.2011) US
2011-260976 29 November 2011 (29.11.2011) JP
13/453,212 23 April 2012 (23.04.2012) US

(71) Applicant *(for all designated States except US)*: **SQUARE ENIX HOLDINGS CO., LTD.** [JP/JP]; 3-22-7, Yoyogi, Shibuya-ku, Tokyo, 1518544 (JP).

(72) Inventor; and

(75) Inventor/Applicant *(for US only)*: **IWASAKI, Tetsuji** [JP/JP]; c/o SQUARE ENIX CO., LTD., 3-22-7 Yoyogi, Shibuya-ku, Tokyo, 1518544 (JP).

(74) Agents: **OHTSUKA, Yasunori** et al.; 7th FL., KIOICHO PARK BLDG., 3-6, KIOICHO, CHIYODA-KU, Tokyo, 1020094 (JP).

(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, CL, 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, 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, PE, PG, PH, PL, PT, QA, RO, RS, RU, RW, SC, SD, SE, SG, SK, SL, SM, ST, SV, SY, TH, TJ, TM, TN, TR, TT, TZ, UA, UG, US, UZ, VC, VN, ZA, ZM, ZW.

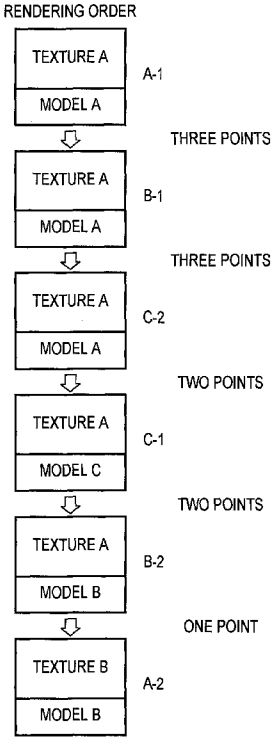
(84) Designated States *(unless otherwise indicated, for every kind of regional protection available)*: ARIPO (BW, GH,

[Continued on next page]

WO 2012/161102 A1

(54) Title: RENDERING CONTROL APPARATUS, CONTROL METHOD THEREOF, RECORDING MEDIUM, RENDERING SERVER, AND RENDERING SYSTEM

FIG. 10



(57) Abstract: For each of a plurality of rendering objects to be used to generate a screen to be provided for a client device, identification information and detailed information indicating data necessary for rendering are acquired. By referring to detailed information of each of the plurality of rendering objects, the rendering order of all the rendering objects is determined so as to allocate consecutive ordinal numbers to rendering objects having at least partial data indicated by detailed information in common. A rendering control apparatus transfers data, indicated by detailed information of a rendering object in accordance with the rendering order, to a GPU. In this process, among data indicated by detailed information of rendering objects which are continuous in the rendering order, only data which is not the same as data already transferred to the GPU is read out and transferred.

GM, KE, LR, LS, MW, MZ, NA, RW, SD, SL, SZ, TZ, UG, ZM, ZW), Eurasian (AM, AZ, BY, KG, KZ, RU, TJ, TM), European (AL, 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, RS, SE, SI, SK, SM, TR), OAPI (BF, BJ, CF, CG, CI, CM, GA, GN, GQ, GW, ML, MR, NE, SN, TD, TG).

Published:

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

DESCRIPTION

TITLE OF INVENTION

RENDERING CONTROL APPARATUS, CONTROL METHOD THEREOF, RECORDING MEDIUM, RENDERING SERVER, AND RENDERING SYSTEM

TECHNICAL FIELD

[0001] The present invention relates to a rendering control apparatus, a control method thereof, a recording medium, a rendering server, and a rendering system and, more particularly, to a technique of providing a rendered screen for a client device connected to a network.

BACKGROUND ART

[0002] Client devices such as personal computers (PCs) connectable to networks are widely used. With the spread of these devices, the network population of the Internet is increasing. Recently, various services using the Internet have been developed for network users, and entertainment services such as games are also provided.

[0003] One of these services for the network users is a massively multiplayer type network game such as an MMORPG (Massively Multiplayer Online Role-Playing Game). In this massively multiplayer type network game, a user can perform a match-up play or team play

with another user using a client device connected to a server for providing the game, by connecting his or her client device to the server.

[0004] In a general massively multiplayer type network game, a client device exchanges the data necessary to render the game with a server. The client device executes a rendering process by using received data necessary for rendering, and displays a generated game screen on a display device connected to the client device, thereby providing the user with the game screen. Also, information input by the user by operating an input interface is transmitted to the server, and used in calculation processing in the server, or transmitted to another client device connected to the server.

[0005] Unfortunately, some network games that perform a rendering process on a client device as described above require each user to use a PC having sufficient rendering performance or a dedicated game machine. Accordingly, the number of users of a network game (one content) depends on the number of users who own devices meeting the performance required by the content. That is, it is difficult to increase the number of users of a game requiring high rendering performance such as a game that provides beautiful graphics.

[0006] In contrast, as described in an

International Publication No. 2009/138878, a game that a user can play without depending on the processability such as the rendering performance of a client device has recently been provided.

[0007] In the game as described in the International Publication No. 2009/138878, a server acquires information of an operation performed on a client device, and provides a client device with a game screen obtained by executing a rendering process by using the information. That is, when performing the rendering process on the server in response to an operation performed on the client device, it is necessary to increase the response speed, that is, rapidly provide a game screen reflecting the operation, in order to allow a user to play the game without any stress.

[0008] Especially in a massively multiplayer type network game, a server simultaneously generates game screens to be provided for a plurality of client devices. Therefore, it is desirable to reduce the time required for the rendering process of the plurality of game screens. However, the International Publication No. 2009/138878 does not mention any practical method of increasing the efficiency of the game screen rendering process.

SUMMARY OF INVENTION

[0009] The present invention has been made in consideration of the problem of the prior art as described above. The present invention provides a method of performing an efficient rendering process having high responsiveness, in a rendering system that provides a game screen for one or more client devices.

[0010] The present invention in its first aspect provides a rendering control apparatus comprising: acquisition means for acquiring information of a plurality of rendering objects to be used to generate a screen to be provided for a client device, and store the information in storage means, wherein the information of each rendering object includes identification information of the rendering object, and detailed information indicating data necessary to render the rendering object; determination means for referring to detailed information of each of the plurality of rendering objects acquired by the acquisition means, and determine a rendering order of the plurality of rendering objects; and transfer means for acquiring identification information of a rendering object in accordance with the rendering order determined by the determination means, read out data indicated by detailed information of a rendering object corresponding to the identification information from data storage means, and transfer the data to rendering means for generating a screen by sequentially rendering

the plurality of rendering objects, wherein the determination means allocates consecutive ordinal numbers as a rendering order of the rendering objects, to the plurality of rendering objects, which have at least partial data indicated by detailed information in common, and when performing rendering in accordance with the rendering order, the transfer means reads out, from the data storage means, data which is not the same as data already transferred to the rendering means, among the data indicated by the detailed information of the rendering objects which are continuous in the rendering order, and transfers the readout data.

[0011] Further features of the present invention will become apparent from the following description of exemplary embodiments (with reference to the attached drawings).

BRIEF DESCRIPTION OF DRAWINGS

[0012] Fig. 1 is a view showing the system arrangement of a rendering system according to an embodiment of the present invention;

[0013] Fig. 2 is a block diagram showing the functional arrangement of a rendering server 100 according to the embodiment of the present invention;

[0014] Fig. 3 is a block diagram showing the functional arrangement of a center server 200 according to the embodiment of the present invention;

[0015] Fig. 4 is a flowchart showing an example of game processing of the center server 200 according to the embodiment of the present invention;

[0016] Figs. 5A and 5B are views showing examples of the data structures of operation input information and a rendering instruction according to the embodiment of the present invention;

[0017] Fig. 6 is a flowchart showing an example of a rendering process of the rendering server 100 according to the embodiment of the present invention;

[0018] Fig. 7 is a flowchart showing an example of a rendering order determination process of the rendering server 100 according to the embodiment of the present invention;

[0019] Fig. 8 is a view for explaining the rendering order determination process according to the embodiment of the present invention;

[0020] Fig. 9 is another view for explaining the rendering order determination process according to the embodiment of the present invention;

[0021] Fig. 10 is a view for explaining a rendering order determined by the rendering order determination process according to the embodiment of the present invention; and

[0022] Fig. 11 is a flowchart showing an example of the rendering order determination process of the rendering server 100 according to a modification of the

present invention.

DESCRIPTION OF EMBODIMENTS

[0023] [First Embodiment]

Exemplary embodiments of the present invention will be explained in detail below with reference to the accompanying renderings. Note that an embodiment to be explained below is an example of a rendering system in which the present invention is applied to a center server capable of accepting the connection of one or more client devices, and a rendering server capable of simultaneously generating screens to be provided for one or more client devices. However, the present invention is applicable to an arbitrary apparatus and arbitrary system capable of simultaneously forming screens to be provided for one or more client devices.

[0024] In this specification, "a service" provided for a client device by the center server is a massively multiplayer type network game as described above. In the rendering system, the rendering server generates a game screen to be provided for a client device, and the game screen is distributed to the client device via the center server. However, the rendering system need not always provide a service like this, and need only have an arrangement that renders and distributes a screen to a client device.

[0025] <Arrangement of Rendering System>

Fig. 1 is a view showing the system arrangement of a rendering system according to the embodiment of the present invention.

[0026] As shown in Fig. 1, client devices 300a to 300e for receiving a provided service and a center server 200 for providing the service are connected across a network 400 such as the Internet. Likewise, a rendering server 100 for rendering a screen to be provided for the client device 300 is connected to the center server 200 across the network 400. Note that in the following explanation, "the client device 300" indicates all the client devices 300a to 300e unless otherwise specified.

[0027] The client device 300 is not limited to, e.g., a PC, home game machine, or portable game machine, and may be an device such as a cell phone, PDF, or tablet. In the rendering system of this embodiment, the rendering server 100 generates a game screen corresponding to operation input caused on a client device, and the center server 200 distributes the screen to the client device 300. Accordingly, the client device 300 need not have any rendering function of generating a game screen. That is, the client device 300 need only be an apparatus including a user interface for detecting operation input and a display device for displaying a screen, or an apparatus connectable to the user interface and display device

and capable of displaying a received game screen on the display device. That is, since a rendering process of generating a game screen uses more hardware resources than those to be used by a process of decoding a video stream, the present invention provides a game independent of the rendering performance of a client device by transmitting a game screen generated by a server to the client device.

[0028] The center server 200 executes and manages a game processing program, instructs the rendering server 100 to perform a rendering process, and exchanges data with the client device 300. More specifically, the center server 200 executes game processing of a massively multiplayer type network game as a service to be provided for the client device 300.

[0029] The center server 200 manages information such as the position and direction on a map of a character operated by the user of each client device, and an event to be provided for each character. The center server 200 causes the rendering server 100 to generate a game screen corresponding to the state of a character being managed. For example, when information of operation input caused by the user on each connected client device is input across the network 400, the center server 200 executes a process of reflecting this information on information of a character being managed. Then, the center server 200 determines a

rendering object to be rendered on the game screen based on the character information reflecting the information of operation input, and transmits a rendering instruction to the rendering server 100.

[0030] The rendering server 100 is a server for executing a rendering process, and includes four GPUs in this embodiment as will be described later. The rendering server 100 renders a game screen in accordance with the rendering instruction received from the center server 200, and outputs the generated game screen to the center server 200. Note that the rendering server 100 can simultaneously form a plurality of game screens. Based on information received from the center server 200 for each game screen, that is, identification information of rendering objects contained in the game screen, and detailed information indicating data necessary to render each rendering object, the rendering server 100 executes a game screen rendering process by a GPU.

[0031] In this embodiment, "detailed information indicating data necessary to render a rendering object" contains information indicating the following attribute data.

·Identification information for specifying model data

·Identification information for specifying texture data

·Specific information of a rendering program (for example, a shader) to be used

·Specific information of data for calculations (for example, the light source intensity, light source vector, and rotation matrix) to be used in the rendering program

[0032] Note that the detailed information containing the above-mentioned information is transmitted from the center server 200 to the rendering server 100 in this embodiment, but information to be contained in the detailed information is not limited to the above information. For example, the detailed information may contain at least one of the five pieces of information described above, and may also contain information to be used in the process of rendering a rendering object in addition to the above information. Note also that in the following explanation, the data for calculations contains numerical values such as a constant and variable to be used in a calculation in the rendering process.

[0033] The center server 200 distributes the game screen, which is received from the rendering server 100 in accordance with the transmitted rendering instruction including the rendering object identification information and detailed information, to a corresponding client device. Thus, the rendering system of this embodiment can generate a game screen

corresponding to operation input caused on a client device, and provide the game screen to the user via the display device of the client device.

[0034] Note that the rendering system of this embodiment includes one rendering server 100 and one center server 200, but the present invention is not limited to this arrangement. For example, it is also possible to allocate one rendering server 100 to a plurality of center servers 200, or allocate a plurality of rendering servers 100 to a plurality of center servers 200. The center server 200 can also designate a rendering server 100 or a GPU of a rendering server 100 to be used to execute a rendering instruction, in accordance with information indicating the number of game screens simultaneously generated by a rendering server 100 or each GPU of a rendering server 100.

[0035]<Arrangement of Rendering Server 100>

Fig. 2 is a block diagram showing the functional arrangement of the rendering server 100 according to the embodiment of the present invention.

[0036] A CPU 101 controls the operation of each block of the rendering server 100. More specifically, the CPU 101 reads out a rendering process operation program stored in a ROM 102 or recording medium 104, expands the program on a RAM 103, and executes the program, thereby controlling the operation of each

block.

[0037] The ROM 102 is, for example, a programmable nonvolatile memory. The ROM 102 stores the rendering process operation program, other operation programs, and information such as a constant required for the operation of each block of the rendering server 100.

[0038] The RAM 103 is a volatile memory. The RAM 103 is used not only as an operation program expansion area, but also as a storage area for temporarily storing, for example, intermediate data output during the operation of each block of the rendering server 100.

[0039] The recording medium 104 is, for example, a recording device such as an HDD detachable from the rendering server 100. In this embodiment, the recording medium 104 stores the following data to be used to generate a screen in the rendering process.

·Model data

·Texture data

·Rendering program

·Data for calculations to be used in the rendering program

[0040] A communication unit 113 is a communication interface of the rendering server 100. The communication unit 113 exchanges data with another apparatus, for example, the center server 200 connected across the network 400. When transmitting data, the

communication unit 113 converts the data into a data transmission format determined with respect to the network 400 or a transmission destination apparatus, and transmits the data to the transmission destination apparatus. When receiving data, the communication unit 113 converts data received across the network 400 into an arbitrary data format readable by the rendering server 100, and stores the data in, for example, the RAM 103.

[0041] First, second, third, and fourth GPUs 105, 106, 107, and 108 each generate a game screen to be provided for the client device 300 in the rendering process. Each GPU is connected to a video memory (one of first, second, third, and fourth VRAMs 109, 110, 111, and 112) as a game screen rendering area. Also, each GPU includes a cache memory. When performing rendering on the connected VRAM, each GPU expands a rendering object on the cache memory, and writes the mapped rendering object in the corresponding VRAM. Note that one video memory is connected to one GPU in this embodiment, but the present invention is not limited to this. That is, the number of video memories connected to the GPU can be any arbitrary number.

[0042]<Arrangement of Center Server 200>

The functional arrangement of the center server 200 of this embodiment will be explained below. Fig. 3 is a block diagram showing the functional arrangement

of the center server 200 according to the embodiment of the present invention.

[0043] A center CPU 201 controls the operation of each block of the center server 200. More specifically, the CPU 201 reads out a game processing program from, for example, a ROM 202 or center recording medium 204, expands the program on a center RAM 203, and executes the program, thereby controlling the operation of each block.

[0044] The center ROM 202 is, for example, a programmable nonvolatile memory. The center ROM 202 stores the game processing program, and may also store other programs. The center ROM 202 also stores information such as a constant required for the operation of each block of the center server 200.

[0045] The center RAM 203 is a volatile memory. The center RAM 203 is used not only as a game processing program expansion area, but also as a storage area for temporarily storing, for example, intermediate data output during the operation of each block of the center server 200.

[0046] The center recording medium 204 is, for example, a recording device such as an HDD detachable from the center server 200. In this embodiment, the center recording medium 204 is used as, for example, a database for managing users and client devices using the game, and a database for managing various kinds of

information on the game, which are required to generate a game screen to be provided for each connected client device.

[0047] The center communication unit 205 is a communication interface of the center server 200. The center communication unit 205 exchanges data with the rendering server 100 and client device 300 connected across the network 400. Note that the center communication unit 205 converts data into a data format complying with the communication specifications, like the communication unit 113.

[0048]<Game Processing>

Practical processing of basic game processing executed on the center server 200 of the rendering system of this embodiment having the arrangement as described above will be explained below with reference to a flowchart shown in Fig. 4. Processing corresponding to this flowchart can be implemented by the center CPU 201 by reading out a corresponding processing program stored in, for example, the center recording medium 204, expands the program on the center RAM 203, and executes the program.

[0049] Note that this game processing is started when, for example, the center server 200 is activated, and repetitively executed for each frame of a provided game. Note also that in the following explanation, the client devices 300a to 300e are already connected to

the center server 200. In this game processing, the center server 200 provides a game screen for each client device.

[0050] In step S401, the center CPU 201 determines whether information indicating operation input caused on each client device 300 is received. More specifically, the center CPU 201 determines whether information received by the center communication unit 205 and stored in the center RAM 203 is information indicating operation input caused on the client device 300.

[0051] The information indicating operation input caused on the client device 300 may have, for example, a data structure as shown in Fig. 5A. As shown in Fig. 5A, the operation input information of this embodiment contains, for example,

·identification information (for example, an IP address and user identification information) of a client device on which operation input is caused

·a moving amount and moving direction input by the operation

·identification information of a selected action

·rendering range of a game screen (for example, a camera parameter)

·the display setting (for example, the screen display resolution and the number of colors) of the display device of the client device

Note that the data structure of the information received by the center server 200 as operation input caused on each client device 300 is not limited to this, and may also be a data structure predetermined by the game.

[0052] In this embodiment, information obtained by converting an operation caused on the client device 300 by the user or the like into a numerical value or parameter in the game is transmitted to the center server 200. However, the present invention is not limited to this. That is, the operation input information may also be information indicating an input signal detected by a client device by operating a button or the like by the user. In this case, the process of conversion into a numerical value or parameter in the game need only be performed in accordance with the information indicating the input signal received by the center CPU 201, and the type of client device.

[0053] If the center CPU 201 determines that the information indicating operation input caused on the client device 300 is received, the center CPU 201 advances the process to step S402; if not, the center CPU 201 advances the process to step S403.

[0054] In step S402, the center CPU 201 updates game information based on the information indicating operation input on the client device 300, which is

received in step S401. More specifically, the center CPU 201 reads out, from the center recording medium 204, state information corresponding to a character in the game, which is an operation target of the user of the corresponding client device 300. The character state information is information of the action and appearance of the character that can be changed by a user's operation, for example, the position (coordinate information) of the character on the map, the gazing direction of the character, and a character's action.

[0055] The center CPU 201 refers to the received operation input information, and updates a parameter that changes by operation input, among parameters contained in the state information, by the received operation input information. Accordingly, the center CPU 201 can reflect, on the game, the information of operation input caused on the client device 300.

[0056] Note that of the information of operation input on the client device 300 received in step S401, the center CPU 201 associates information that is not used to update the character state information with the client device identification information, and records the information on the center recording medium 204.

[0057] In step S403, the center CPU 201 updates state information of a rendering object as a state management target in the game, except for the character as an operation target of the user of each client

device 300. Examples of the rendering object as a state management target in the game are a character called an NPC (Non Player Character) that is not a target of a user's operation, and a background object such as a landform. A rendering object as a state management target in the game changes with time or by the action of a character as a user's operation target. In step S403, therefore, the center CPU 201 updates the state information of a rendering object as a state management target in the game, in accordance with the elapse of time or the character state information updated in step S402.

[0058] In step S404, the center CPU 201 specifies rendering objects contained in the game screen provided for each client device 300. The rendering objects contained in the game screen are an above-described rendering object as a state management target in the game, and a character's rendering object as a user's operation target.

[0059] More specifically, the center CPU 201 selects each client device 300 presently connected to the center server 200, and reads out, from the center recording medium 204, information of the game screen rendering range with which the identification information of the client device 300 is associated. As described above, the information of the game screen rendering range contains camera parameters

corresponding to the game screen. The center CPU 201 refers to the camera parameters, and specifies, as a rendering object contained in the game screen, a rendering object such as a character whose state information contains coordinate information included in the camera's rendering range. Note that identification information is assigned to a rendering object, and the center CPU 201 associates the identification information of the specified rendering object contained in the game screen with the identification information of the client device, and records the information on the center recording medium 204.

[0060] In step S405, the center CPU 201 transmits, to the rendering server 100, an instruction to render game screens to be provided for all the client devices 300, and causes the rendering server 100 to execute the rendering process. More specifically, the center CPU 201 first reads out the identification information, which is recorded on the center recording medium 204 in association with the identification information of each client device 300, of the rendering object contained in the game screen to be provided for the client device 300. The center CPU 201 also reads out, from the center recording medium 204, the detailed information set for each rendering object contained in the game screen. Then, the center CPU 201 associates the identification information of the rendering objects

contained in the game screen to be provided for each client device 300, and the detailed information of each rendering object, with the identification information of the client device 300, and transmits the information as a rendering instruction to the center server 200 via the center communication unit 205. At the same time, the center CPU 201 reads out, from the center recording medium 204, the information of rendering range of the game screen and the display setting information of the display device of each client device, which are associated with the identification information of the client device, and transmits the readout information to the rendering server 100.

[0061] That is, as shown in Fig. 5B, the rendering instruction transmitted to the center server 200 in order to render a game screen to be provided for one client device contains

·identification information of each rendering object contained in the game screen

·detailed information of each rendering object contained in the game screen

·state information of each rendering object contained in the game screen

·information of the rendering range and display setting of the game screen

associated with the identification information of the client device. Note that if the game is, for example,

a game in which the camera position remains unchanged or a game having a fixed display resolution, information such as the camera parameters and display setting indicating the game screen rendering range need not be contained in the rendering instruction.

[0062] Note that in this embodiment, the rendering instruction is transmitted to the rendering server 100 for each client device 300. However, it will readily be understood that it is also possible to simultaneously transmit rendering instructions for a plurality or all of the client devices 300.

[0063] In step S406, the center CPU 201 determines whether the game screen generated by the rendering server 100 in accordance with the rendering instruction is received from the rendering server 100. More specifically, the center CPU 201 determines whether the center communication unit 205 has received the game screen generated and output by the rendering server 100 for each client device 300. If it is determined that the game screen is received, the center CPU 201 advances the process to step S407; if not, the center CPU 201 repeats the processing of this step. That is, the center CPU 201 waits in this step until the game screen of each client device 300 is received.

[0064] Note that the game screen received in this step is preferably a game screen generated based on the rendering instruction transmitted to the rendering

server 100 in step S404 as an immediately preceding step, but the game screen is not limited to this. That is, this game processing is executed for each frame. Therefore, if rendering of the game screen is not complete within one frame time defined by the frame rate of the provided game, the game screen received in this step may also be a screen generated in accordance with a rendering instruction transmitted in a frame preceding the present frame by a few frames.

[0065] In step S407, the center CPU 201 distributes the game screen received in step S406 to the corresponding client device 300. More specifically, the center CPU 201 refers to identification information of the client device 300 contained in, for example, the header of the game screen received by the center communication unit 205 and stored in the center RAM 203. Then, the center CPU 201 transmits the game screen to the client device 300 indicated by the identification information via the center communication unit 205, and completes this game processing.

[0066] As described above, the center server 200 can generate a game screen reflecting operation input caused on each client device 300, and distribute the obtained game screen to the client device 300, by transmitting the rendering instruction to the rendering server 100.

[0067] Depending on the number of client devices 300 or the number of rendering objects included in the rendering range, however, it takes a long time to generate a game screen reflecting operation input caused on each client device 300. This may give the user of the client device 300 an impression that the response of the game is low. In this embodiment, a method of shortening the time required for the rendering process in the rendering server 100 by determining the rendering object detailed information contained in the rendering instruction as follows will be explained below.

[0068] The rendering process is performed by reading out, for each rendering object, "identification information for specifying model data", "identification information for specifying texture data", "specific information of a rendering program", and "data for calculations to be used in the rendering program" contained in the detailed information. That is, when a plurality of rendering objects are contained in the game screen, each rendering object is selected, detailed information is read out, and rendering is performed in accordance with the detailed information. After that, rendering for the next rendering object is performed. That is, whenever rendering objects to be rendered are switched during the rendering process, the rendering server 100 executes processes pertaining to

initialization, for example, transfers model data and texture data to a GPU, converts the data, reads out a rendering program, and sets data for calculations. In the rendering process for one game screen, therefore, the processing time (switching cost) for initialization for rendering one rendering object is required for the number of rendering objects included in the rendering range of the game screen. The switching cost further increases when simultaneously generating game screens to be distributed to a plurality of client devices 300.

[0069] On the other hand, in the generation of a game screen, rendering objects having at least partial data indicated by the detailed information in common sometimes exist particularly in, for example, the background. For example, objects such as trees contained in the game screen are rendered by using one texture data for a few patterns of model data. Also, the same rendering program and the same data for calculations are used for, for example, objects having the same material or texture, or objects existing under the same light source. Furthermore, if the camera positions and directions on the map in the game are similar in game screens to be provided for different client devices, it is highly likely that common rendering objects exist in the rendering range and the same detailed information is used.

[0070] This embodiment reduces the switching cost

during the rendering process by allocating consecutive ordinal numbers as a rendering order to rendering objects having at least partial data indicated by detailed information in common as described above. That is, when switching rendering objects to be rendered, the time required for common processing is shortened by using the result of a calculation performed on a rendering object rendered in front or data read out for it, thereby reducing the switching cost.

[0071]<Rendering Process>

Practical processing of the rendering process of the rendering server 100 of this embodiment will be explained below with reference to a flowchart shown in Fig. 6. Processing corresponding to this flowchart can be implemented by the CPU 101 by reading out a corresponding processing program stored in, for example, the ROM 102, and executing the program by expanding it on the RAM 103. Assume that this rendering process is started when, for example, the rendering instruction from the center server 200 is received. Note that in the following explanation, the rendering server 100 generates game screens to be provided for the five client devices 300a to 300e in the same manner as in the above-described game processing.

[0072] In step S601, the CPU 101 classifies the

generation of game screens to be provided for the client devices 300a to 300e to the GPUs of the rendering server 100. That is, in this step, the CPU 101 determines a GPU to be used to process the received rendering instruction, from the first, second, third, and fourth GPUs 105, 106, 107, and 108.

[0073] Examples of the method of classifying client devices as game screen generation targets to be allocated to one GPU are as follows.

[0074] 1. Camera Parameter (Rendering Range) Defining Game Screen

The game screen rendering range (visual field) is defined by a camera parameter for a game screen to be provided for each client device 300. The visual field of one camera is normally defined by a quadrangular pyramid or truncated quadrangular pyramid. Game screens in which the visual fields overlap each other presumably contain the same rendering object. Therefore, the efficiency of the rendering process can be increased by classifying client devices as game screen generation targets to be allocated to the same GPU, by taking account of the overlap of the visual fields. More specifically, a triangle obtained by projecting a quadrangular pyramid defined by the visual field onto a two-dimensional plane is calculated in accordance with the coordinates and direction of a camera defining a game screen to be provided for each

client device, game screens in which the triangular ranges overlap each other by a predetermined ratio or more are classified as one group, and the rendering process of the group is allocated to the same GPU.

[0075] Simply, client devices having close camera coordinates defining game screens are expected to have similar game screen rendering contents. Accordingly, game screens in each of which the distance between camera coordinates defining the game screen falls within a predetermined distance may be classified into one group, and the rendering process of the group may be allocated to the same GPU.

[0076] Also, an area using the same background object such as a mountain, wood, or sea exists on the map in a game. That is, when the camera coordinates defining a game screen exist in a specific area in which this background object is arranged, or when the camera direction is the direction of this specific area, generated game screens contain a common background object. Therefore, the efficiency of the rendering process can be increased by classifying game screens containing a common background object into one group, and allocating the rendering process of the group to the same GPU.

[0077] Furthermore, the same rendering object may be used as a background in various areas on the map in a game. That is, even when the camera coordinates and

directions defining game screens are different, the same background object may be included in the rendering range. Accordingly, the camera positions or the like in game screens to be generated by one GPU need not always be close.

[0078] As information of a GPU to be allocated, information predetermining a GPU for performing the rendering process may be recorded as a lookup table on the recording medium 104, for a combination of, for example, the camera coordinates and direction information. As the information of a GPU to be allocated, information of a GPU for performing processing may also be obtained by, for example, a function using the values of the camera position and direction as arguments.

[0079] 2. Combination of Detailed Information of Rendering Objects Included in Rendering Range

The switching cost in the rendering process reduces as the coincidence of detailed information of rendering objects having consecutive ordinal numbers as a rendering order increases. Therefore, the CPU 201 refers to the detailed information of all rendering objects contained in each of game screens to be provided for all the client devices 300, and calculates the coincidence of rendering objects between the game screens on a detailed information level. The CPU 201 then classifies the generation of game screens having a

high coincidence into one group, and allocates the rendering process of the group to the same GPU.

[0080] Note that the coincidence of detailed information can also be obtained as the similarity of a distribution formed for rendering objects contained in each game screen, based on a combination of, for example, texture data and model data in detailed information. It is also possible to classify rendering objects contained in each game screen based on specific attribute data indicated by detailed information, and, for only classification in which the number of rendering objects belonging to each set is equal to or larger than a threshold value, compare detailed information in a round robin manner between game screens, thereby calculating the coincidence of the detailed information.

[0081] 3. Presence/absence of Participation of Character as User's Operation Target in Event

When a specific condition is met in a game, for example, when an object exists in a specific point on the map in a game, an event may occur and the game screen may forcedly be changed to a screen different from a normal one. For example, in a combat event against an enemy character or the like, a character as a user's operation target fights the enemy character in a dedicated battlefield. Since events like this may simultaneously occur in arbitrary positions on the map,

the switching cost can be reduced by classifying game screens to be provided for client devices 300 on which events of the same kind have occurred into one group, and allocating the rendering process of the group to the same GPU.

[0082] 4. Progress Status of Game

For a game of a kind in which the progress of a story is determined, the types of objects to be rendered may be limited in accordance with the progress status of the game. For example, in a story in which the field changes like the inside of a building → an open space → a grassy plain → a forest, it is possible to classify groups whenever the field changes, and allocate game screen rendering to different GPUs.

[0083] As described above, rendering objects contained in a game screen are classified into a plurality of groups having similar rendering contents by using the similarity of the rendering objects as a determination criterion, and the rendering of game screens of each group is allocated to the same GPU. Then, a rendering order determination process (to be described later) is executed. This allows each GPU to generate a game screen more efficiently. Note that the aforementioned method of determining a GPU for generating a game screen to be provided for the client device 300 is merely an example. Accordingly, it is also possible to use a method capable of increasing the

efficiency of the rendering process by allocating the rendering of game screens regarded as having similar rendering contents to the same CPU.

[0084] This embodiment uses method 1 described above by which a GPU for performing the rendering process is determined in accordance with the camera position and coordinate information defining a game screen. In the following description, the generation of game screens to be provided for the five client devices 300a to 300e is classified into one group and executed by the first GPU 105. Note that the number of game screens (the number of client devices) classified into one group is determined by the processability of the GPU and the capacity of the VRAM. Note also that information of client devices classified into each group can be managed by, for example, adding identification information of the client devices to a table having one group ID, and storing the table in the RAM 103.

[0085] In step S602, the CPU 101 executes the rendering order determination process of determining the rendering order of rendering objects contained in all game screens to be rendered, for each group allocated to each GPU.

[0086] (Rendering Order Determination Process)

The rendering order determination process of this embodiment will be explained in detail below with

reference to a flowchart shown in Fig. 7. Note that a process of determining the rendering order of rendering objects contained in game screens of one group to be rendered by the first GPU 105 will be explained below, but it will readily be understood that the same process is applicable to other GPUs.

[0087] In step S701, the CPU 101 forms pairs of all combinations each including two rendering objects, for all rendering objects contained in game screens of one group to be rendered by the first GPU 105.

[0088] In step S702, the CPU 101 scores the coincidence of detailed information of two rendering objects of all the pairs formed in step S701, by using scores obtained by unbalanced allocation by arranging attributes of detailed information in descending order of priority. In this embodiment, the attribute of each data indicated by detailed information is given priority by the following order.

1. Identification information for specifying texture data
2. Identification information for specifying model data
3. Specific information of a rendering program to be used
4. Specific information of data for calculations to be used in the rendering program

[0089] The priority of the attribute of each data

indicated by detailed information is determined in accordance with the data size on a cache memory when the data having the attribute is read out and expanded on the memory. Data having a high-order attribute has a large data size and requires a high switching cost, that is, when this data is switched to different data, it takes a long time to, for example, expand the data on the cache memory, read out the data from the cache memory, and set parameters. That is, the switching cost can efficiently be reduced by performing rendering by a rendering order that decreases the switching frequency (increases the reuse frequency) of data having a high-priority attribute.

[0090] More specifically, scoring in this step gives a high score to a pair of rendering objects having a high detailed information coincidence, that is, to a pair of rendering objects by which the switching cost is reduced by increasing the reuse frequency for attribute data having a high priority.

[0091] In step S703, the CPU 101 refers to the coincidence scores of all the pairs, and determines the rendering order of all rendering objects contained in game screens of one group to be rendered by the first GPU 105. More specifically, the CPU 101 determines a rendering order that maximizes the total sum of scores for a combination of continuous rendering objects when all the rendering objects are arranged in the rendering

order.

[0092] A practical rendering order determination procedure will be explained in detail below with reference to an accompanying rendering. To simplify the explanation, a method of determining the rendering order by taking account of the coincidence of texture data and model data, among the four attributes of detailed information described above.

[0093] Fig. 8 shows rendering objects contained in a game screen to be provided for each client device, which are used in an example of the rendering object rendering order determination procedure in step S703. In this example shown in Fig. 8, the same GPU generates game screens to be provided for three client devices, and the GPU renders two rendering objects for each game screen, that is, renders a total of six rendering objects.

[0094] When the CPU 101 generates pairs of possible combinations of all objects, the number is ${}_6C_2 = 15$. That is, the CPU 101 performs coincidence scoring for all the fifteen pairs. In this embodiment, when attributes of detailed information coincide, the score is two points for the coincidence of texture data, and one point for the coincidence of model data. In this case, the scores of these pairs are as shown in Fig. 9.

[0095] The CPU 101 first notes pairs having a

highest score of 3. In this example, three pairs (A-1, B-1), (A-1, C-2), and (B-1, C-2) have a highest score of 3. In this case, the number of pairs having the highest score is equal to the number of rendering objects (three-point objects) forming the pairs. This demonstrates that all the pairs are formed by three-point objects having the same detailed information. The CPU 101 then refers to the scores of other pairs including each of the three-point objects, and determines the rendering order of the three-point objects.

[0096] As shown in Fig. 9, the score of another pair including the three-point objects is two as the second highest score or score 0 of a pair with A-2. That is, when allocating a rendering object other than A-2 as a rendering object following the three-point objects in the rendering order, the switching cost remains the same regardless of the rendering object. Therefore, the CPU 101 determines the order based on, for example, the sorting order of identification information, without particularly taking account of the rendering order of the three three-point objects described above.

[0097] Note that if there is only one pair having the second highest score, a rendering object included in the pair, among the three three-point objects, is determined as the third, that is, the last in the

three-point object rendering order, and the other rendering object included in the pair is allocated as the fourth.

[0098] Similarly, the CPU 101 determines the rendering order of rendering objects (two-point objects B-2 and C-1), other than the three-point objects, included in a pair having the second highest score. As shown in Fig. 9, a pair with rendering object A-2 is only the pair including a two-point object and having the second highest score or less. As shown in Fig. 9, only the score of a pair with two-point object B-2 is 1 for rendering object A-2. Of the two-point objects, therefore, rendering object B-2 is determined as the fifth, that is, the last of the two-point objects in the rendering order, and rendering object A-2 is determined as the sixth.

[0099] By repeating determination as described above, the CPU 101 can determine the rendering order of all rendering objects contained in game screens of one group. In this example, the CPU 101 finally determines A-1 → B-1 → C-2 → C-1 → B-2 → A-2 as the rendering order as shown in Fig. 10.

[0100] In this way, ordinal numbers of the rendering order that reduces the switching cost can be allocated to all rendering objects contained in game screens classified into one group. Note that the ordinal numbers of the rendering order can be allocated

by, for example, associating identification information of rendering objects with the rendering order, and storing the result as a list in the RAM 103 or the like.

[0101] After the rendering order determination process in step S602 is complete, the CPU 101 renders rendering objects in accordance with the determined rendering order in step S603, thereby generating a game screen to be provided for each client device 300. More specifically, the CPU 101 refers to detailed information of rendering objects in the rendering order, reads out data specified by the detailed information from the recording medium 104, and transfers the data to a GPU for performing rendering. In this process, the CPU 101 reads out and transfers only data different from data already expanded in an expansion area; the CPU 101 does not read out the same data as the already expanded data, and uses the already expanded data instead.

[0102] Note that a game screen is generated by using state information containing coordinate information and action information of each rendering object, and game screen rendering range/display setting information including camera parameters. Since the use of these pieces of information is well known as processing contents, an explanation thereof will be omitted.

[0103] In step S604, the CPU 101 adds, to the header of a game screen generated by each GPU, identification information of a client device for which the game screen is to be provided, and transmits the game screen to the center server 200 via the communication unit 113.

[0104] Since the rendering process can be performed by thus reducing the switching cost, a game screen having high responsiveness to user's operation input can be provided for a user operating a client device.

[0105] Note that in this embodiment, "a game screen" is simply transmitted from the center server 200 to the client device 300. Although "a game screen" may be simple still image data, "a game screen" may also be transmitted as frame image data of moving image data to which an encoding process such as interframe prediction that further decreases the data size is applied. The encoding process can, of course, be performed by either the rendering server 100 or center server 200.

[0106] Also, in this embodiment, the present invention is explained by taking a massively multiplayer type network game as an example. As described previously, however, the present invention is applicable to an arbitrary apparatus and arbitrary system capable of simultaneously generating screens to

be provided for one or more client devices, and is not limited to the above embodiment. For example, when game contents corresponding to a single user are independently provided for each client device in a rendering system, similar game screens may be output to a plurality of client devices. That is, common rendering objects are perhaps contained in game screens provided when the positions or directions on the map are similar, when the statuses of progress of the game are similar, or when the same action is executed. Therefore, the efficiency of the rendering process can be increased by applying the present invention.

[0107] Furthermore, the rendering server 100 and center server 200 are separated in the rendering system of this embodiment, but the present invention is not limited to this. That is, the present invention may also be carried out by one server having the functions of both the rendering server 100 and center server 200.

[0108] As explained above, the rendering control apparatus of this embodiment can perform an efficient rendering process having high responsiveness in a rendering system that provides a game screen to one or more client devices. More specifically, the rendering control apparatus acquires identification information and detailed information indicating data necessary for rendering, for each of a plurality of rendering objects to be used to generate a screen to be provided for a

client device. By referring to detailed information of each of the plurality of rendering objects, the rendering control apparatus determines the rendering order of all the rendering objects so as to allocate consecutive ordinal numbers to rendering objects having at least partial data indicated by the detailed information in common. The rendering control apparatus reads out the data indicated by the detailed information of the rendering objects in accordance with the rendering order, and transfers the data to a GPU. In this case, of the data indicated by the detailed information of rendering objects that are continuous in the rendering order, the rendering control apparatus reads out and transfers only data that are not the same as data already transferred to the GPU.

[0109] The processing as described above makes it possible to reduce the time required for the game screen rendering process, and efficiently perform rendering.

[0110] [Modification]

In the above-described embodiment, the rendering order determination process of determining the rendering order by making pairs of all combinations of all rendering objects contained in game screens classified into one group and performing scoring on these pairs is explained. In this modification, a method of simply performing scoring by sorting and

classifying rendering objects by arranging attributes of detailed information in descending order of priority will be explained.

[0111] (Rendering Order Determination Process)

The rendering order determination process of this modification will be explained in detail below with reference to a flowchart shown in Fig. 11. Note that a process of determining the rendering order of rendering objects contained in game screens of one group to be rendered by the first GPU 105 will be explained below, but it will readily be understood that the same process is applicable to other GPUs.

[0112] In step S1101, for all rendering objects contained in game screens of one group to be rendered by the first GPU 105, the CPU 101 classifies by arranging attributes indicated by detailed information in descending order of priority, based on attribute data having the highest priority. The attribute of each data indicated by detailed information is given priority by the following order in this modification as well.

1. Identification information for specifying texture data

2. Identification information for specifying model data

3. Specific information of a rendering program to be used

4. Specific information of data for calculations to be used in the rendering program

That is, in this step, the CPU 101 classifies all rendering objects by identification information for specifying texture data.

[0113] In step S1102, the CPU 101 refers to the number of members belonging to each set of rendering objects using the same texture data, which is generated by the classification in step S1101, and specifies sets in which the number of members is 1. Then, the CPU 101 integrates the sets in which the number of members is 1 into one set (a texture non-coincidence set).

[0114] In step S1103, the CPU 101 classifies rendering objects in each of the sets of rendering objects using the same texture data and the texture non-coincidence set, by using identification information for specifying model data having the second highest priority. That is, in this step, rendering objects belonging to each of the sets of rendering objects using the same texture data and the texture non-coincidence set are classified into child sets of rendering objects using the same model data.

[0115] In step S1104, the CPU 101 refers to the number of members belonging to each set of rendering objects using the same model data, which is generated by the classification in step S1103, and specifies sets in which the number of members is 1. Then, for each of

the sets of rendering objects using the same texture data and the texture non-coincidence set, the CPU 101 integrates rendering objects in sets in which the number of members is 1 into one set (a model non-coincidence set).

[0116] In step S1105, the CPU 101 determines whether there is an attribute not used in set classification. If there is an attribute not used in set classification, the CPU 101 selects an attribute not used in set classification in priority order, and returns the process to step S1103. That is, the CPU 101 then classifies rendering objects in each of the sets of rendering objects using the same model data and the model non-coincidence set, by using rendering program designating information.

[0117] By thus performing steps S1101 and step S1102, and the repetitive processing from steps S1103 to step S1105, it is possible to classify rendering objects for each attribute data indicated by detailed information, and finally classify, into a set, rendering objects having at least partial data of data indicated by detailed information in common.

[0118] Note that if the CPU 101 determines in this step that there is no attribute not used in set classification, the CPU 101 advances the process to step S1106.

[0119] In step S1106, the CPU 101 determines the

rendering order of all rendering objects contained in game screens of one group to be rendered by the first GPU 105. That is, the CPU 101 determines a rendering order that reduces the switching cost, in accordance with information of rendering object classification performed in step S1101 to step S1105.

[0120] More specifically, the CPU 101 first selects one set, other than the texture non-coincidence set, from sets classified for "identification information for specifying texture data" having the highest priority. Then, the CPU 101 sequentially selects child sets included in the set, and allocates ordinal numbers of the rendering order to rendering objects in the child sets. In this process, the child sets can be selected by, for example, representing each set by using a binary number, and selecting sets in descending order of binary value. For example, scoring is performed by adding 2^3 to a set in which rendering objects have the attribute having the second highest priority in common, 2^2 to a set in which rendering objects have the attribute having the third highest priority in common, 2^1 to a set in which rendering objects have the attribute having the fourth highest priority in common, and 2^0 to a set in which rendering objects have the attribute having the lowest priority in common, and the rendering order is determined in descending order of score.

[0121] Also, when the rendering order of rendering objects is determined for one set classified for "identification information for specifying texture data", the rendering order is similarly determined for the next set using the same texture data. Furthermore, when the rendering orders of rendering objects are determined for all sets other than the texture non-coincidence set, the rendering order is similarly determined for rendering objects included in the texture non-coincidence set.

[0122] Accordingly, rendering orders that reduce the switching cost can be allocated to all rendering objects.

CLAIMS

1. A rendering control apparatus comprising:

acquisition means for acquiring information of a plurality of rendering objects to be used to generate a screen to be provided for a client device, and store the information in storage means, wherein the information of each rendering object includes identification information of the rendering object, and detailed information indicating data necessary to render the rendering object;

determination means for referring to detailed information of each of the plurality of rendering objects acquired by said acquisition means, and determine a rendering order of the plurality of rendering objects; and

transfer means for acquiring identification information of a rendering object in accordance with the rendering order determined by said determination means, read out data indicated by detailed information of a rendering object corresponding to the identification information from data storage means, and transfer the data to rendering means for generating a screen by sequentially rendering the plurality of rendering objects,

wherein said determination means allocates consecutive ordinal numbers as a rendering order of the rendering objects, to the plurality of rendering

objects, which have at least partial data indicated by detailed information in common, and

when performing rendering in accordance with the rendering order, said transfer means reads out, from said data storage means, data which is not the same as data already transferred to said rendering means, among the data indicated by the detailed information of the rendering objects which are continuous in the rendering order, and transfers the readout data.

2. The apparatus according to claim 1, wherein

detailed information of a rendering object indicates a plurality of attribute data necessary to render the rendering object, and

said determination means preferentially allocates ordinal numbers as a rendering order of the rendering objects, to the plurality of rendering objects, which have high-priority attribute data indicated by detailed information in common.

3. The apparatus according to claim 2, wherein the priority is set such that when a plurality of attribute data indicated by detailed information are read out from said data storage means and expanded in an expansion area, a high priority is given to an attribute whose data occupies a large area.

4. The apparatus according to claim 2 or 3, wherein said determination means preferentially allocates ordinal numbers as a rendering order of the rendering

objects, to the plurality of rendering objects, in which the number of common attribute data indicated by the detailed information is large.

5. The apparatus according to any one of claims 1 to 4, wherein the detailed information of a rendering object indicates at least one of texture data, model data, a rendering program, and data for rendering calculations.

6. The apparatus according to any one of claims 1 to 5, wherein

said acquisition means acquires information of the rendering objects to be used to generate screens to be provided for a plurality of client devices, and

said determination means determines a rendering order of all rendering objects to be used to generate screens to be provided for the plurality of client devices.

7. A control method of a rendering control apparatus, comprising:

an acquisition step of acquiring information of a plurality of rendering objects to be used to generate a screen to be provided for a client device, and storing the information in storage means, wherein the information of each rendering object includes identification information of the rendering object, and detailed information indicating data necessary to render the rendering object;

a determination step of referring to detailed information of each of the plurality of rendering objects acquired in the acquisition step, and determining a rendering order of the plurality of rendering objects; and

a transfer step of acquiring identification information of a rendering object in accordance with the rendering order determined in the determination step, reading out data indicated by detailed information of a rendering object corresponding to the identification information from data storage means, and transferring the data to rendering means for generating a screen by sequentially rendering the plurality of rendering objects,

wherein in the determination step, consecutive ordinal numbers as a rendering order are allocated to the rendering objects, among the plurality of rendering objects, which have at least partial data indicated by detailed information in common, and

in the transfer step, when performing rendering in accordance with the rendering order, data which is not the same as data already transferred to the rendering means, among the data indicated by the detailed information of the rendering objects which are continuous in the rendering order, is read out from the data storage means and transferred.

8. A recording medium recording a program which

causes a computer to function as each means of a rendering control apparatus cited in any one of claims 1 to 6.

9. A computer-readable storage medium wherein a program is stored which causes a computer to function as each means of a rendering control apparatus cited in any one of claims 1 to 6.

10. A rendering server comprising a rendering control apparatus cited in any one of claims 1 to 6.

11. A rendering system which provides a screen generated by a rendering server for each of one or more client devices connected to a center server,

said center server comprising:

reception means for receiving input data from said one or more client devices;

specifying means for specifying, for each of said one or more client devices, a rendering object contained in a screen to be provided for the client device, in accordance with the input data received by said reception means;

transmission means for transmitting, to said rendering server, information of a plurality of rendering objects contained in a screen to be provided for said one or more client devices, which are specified by said specifying means, wherein information of each rendering object includes identification information of the rendering object, and detailed

information indicating data necessary to render the rendering object; and

distribution means for receiving a screen to be provided for each of said one or more client devices, which is rendered by said rendering server, and distribute the screen to a corresponding client device, and

said rendering server comprising:

acquisition means for acquiring information of the plurality of rendering objects transmitted by said transmission means, and store the information in storage means;

rendering means for rendering the plurality of rendering objects in order, and generate a screen to be provided for each of said one or more client devices;

determination means referring to detailed information of each of the plurality of rendering objects acquired by said acquisition means, and determine a rendering order of the plurality of rendering objects; and

transfer means for acquiring identification information of a rendering object in accordance with the rendering order determined by said determination means, read out data indicated by detailed information of a rendering object corresponding to the identification information from data storage means, and transfer the data to said rendering means,

wherein said determination means allocates consecutive ordinal numbers as a rendering order of the rendering objects, to the plurality of rendering objects, which have at least partial data indicated by detailed information in common, and

when performing rendering in accordance with the rendering order, said transfer means reads out, from said data storage means, data which is not the same as data already transferred to said rendering means, among the data indicated by the detailed information of the rendering objects which are continuous in the rendering order, and transfers the readout data.

12. The system according to claim 11, wherein

detailed information of a rendering object indicates a plurality of attribute data necessary to render the rendering object, and

said determination means preferentially allocates ordinal numbers as a rendering order of the rendering objects, to the plurality of rendering objects, which have high-priority attribute data indicated by detailed information in common.

13. The system according to claim 12, wherein the priority is set such that when a plurality of attribute data indicated by detailed information are read out from said data storage means and expanded in an expansion area, a high priority is given to an attribute whose data occupies a large area.

14. The system according to claim 12 or 13, wherein said determination means preferentially allocates ordinal numbers as a rendering order of the rendering objects, to the plurality of rendering objects, in which the number of common attribute data indicated by the detailed information is large.

15. The system according to any one of claims 11 to 14, wherein the detailed information of a rendering object indicates at least one of texture data, model data, a rendering program, and data for rendering calculations.

16. The system according to any one of claims 11 to 15, wherein said specifying means refers to a camera parameter which changes in accordance with the input data received by said reception means and defines a screen of each of said one or more client devices, classifies client devices for which rendering ranges specified by the camera parameter overlap each other by a predetermined amount into one group, and specifies the rendering objects contained in screens to be provided for the client devices in the group.

17. The system according to any one of claims 11 to 15, wherein said specifying means refers to camera coordinates which change in accordance with the input data received by said reception means and define a screen of each of said one or more client devices, classifies client devices for which the camera

coordinates fall within a predetermined distance into one group, and specifies the rendering objects contained in screens to be provided for the client devices in the group.

18. The system according to any one of claims 11 to 17, wherein

said acquisition means acquires the information of the rendering objects to be used to generate screens to be provided for a plurality of client devices,

said determination means determines a rendering order of all rendering objects to be used to generate screens to be provided for the plurality of client devices, and

said rendering means simultaneously renders screens to be provided for the plurality of client devices.

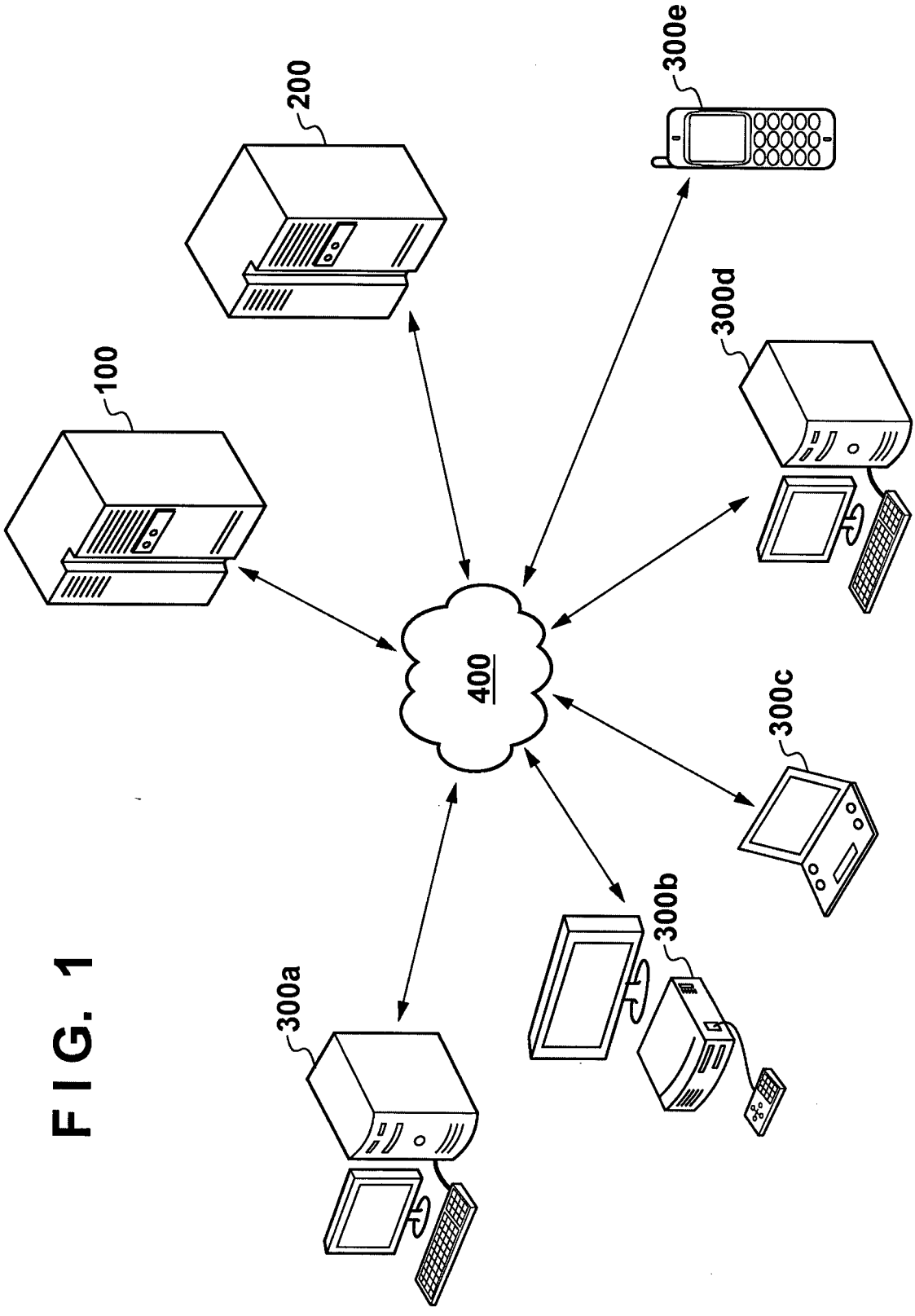


FIG. 1

FIG. 2

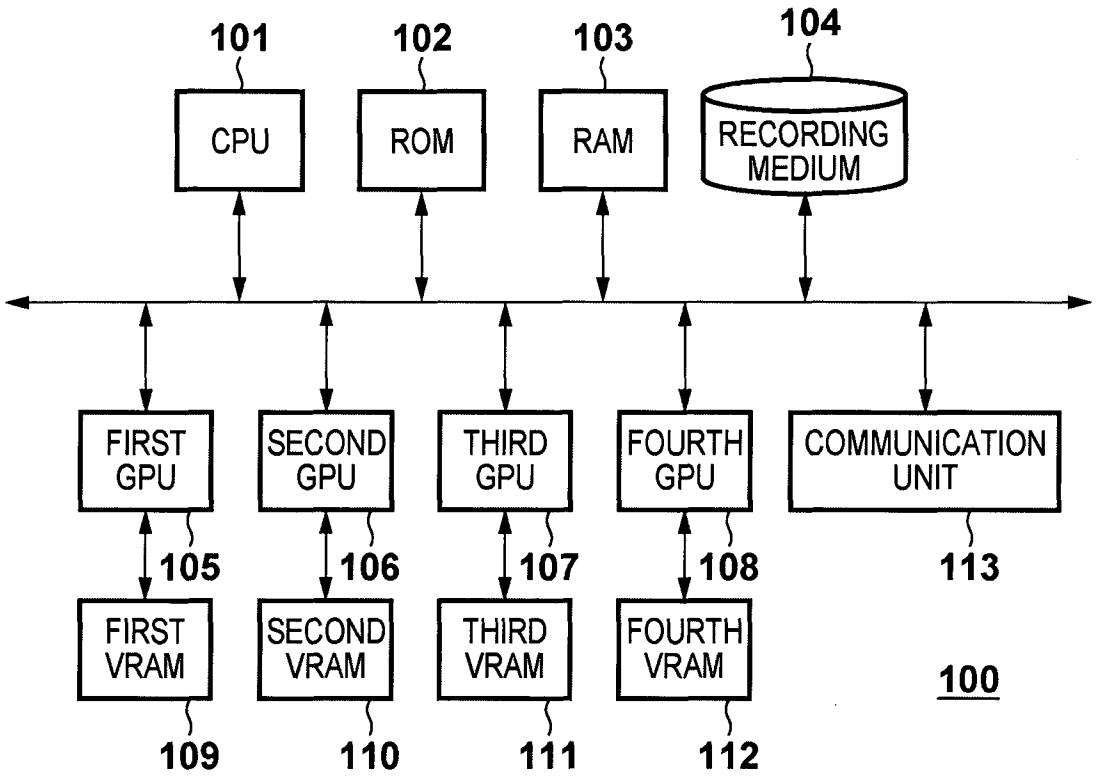

101
102
103
104
CPU
ROM
RAM
RECORDING MEDIUM
FIRST GPU
SECOND GPU
THIRD GPU
FOURTH GPU
COMMUNICATION UNIT
105
106
107
108
113
FIRST VRAM
SECOND VRAM
THIRD VRAM
FOURTH VRAM
100
109
110
111
112

FIG. 3

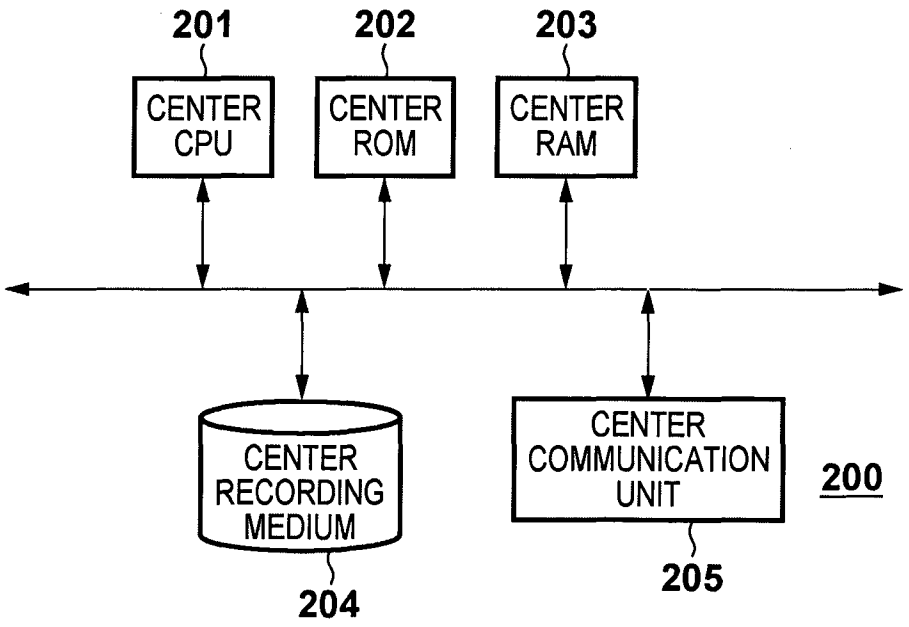

201
202
203
CENTER CPU
CENTER ROM
CENTER RAM
CENTER RECORDING MEDIUM
CENTER COMMUNICATION UNIT
200
204
205

FIG. 4

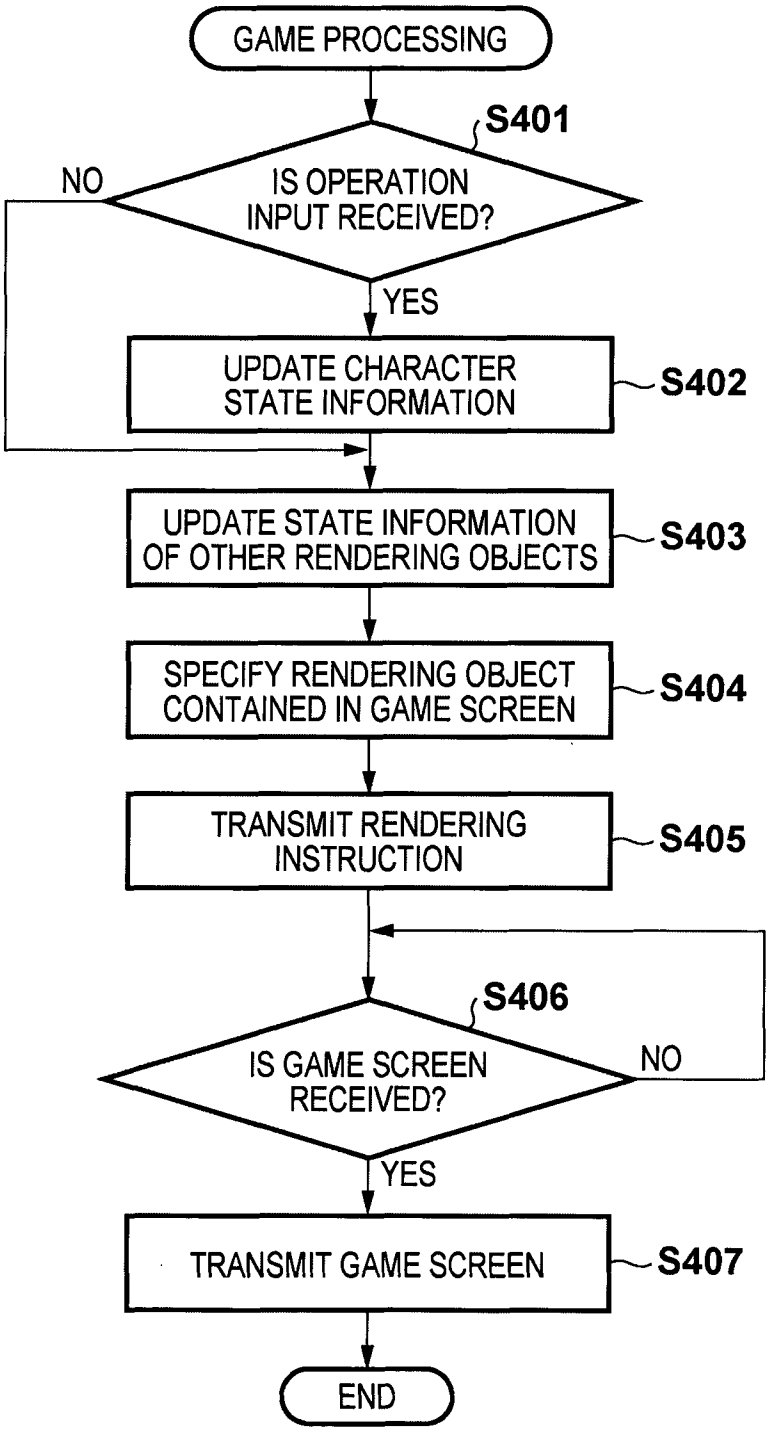

GAME PROCESSING
S401
IS OPERATION INPUT RECEIVED?
NO
YES
UPDATE CHARACTER STATE INFORMATION
S402
UPDATE STATE INFORMATION OF OTHER RENDERING OBJECTS
S403
SPECIFY RENDERING OBJECT CONTAINED IN GAME SCREEN
S404
TRANSMIT RENDERING INSTRUCTION
S405
S406
IS GAME SCREEN RECEIVED?
NO
YES
TRANSMIT GAME SCREEN
S407
END

FIG. 5A

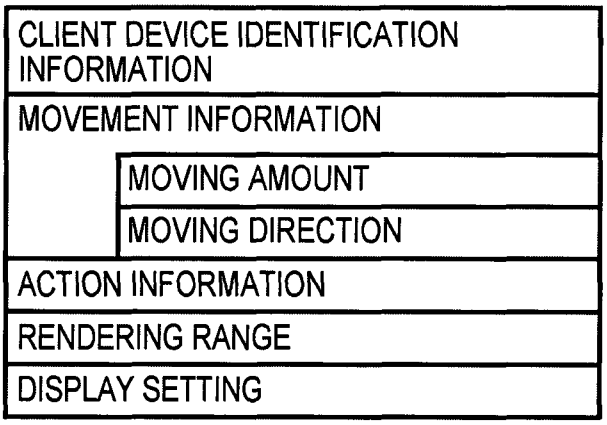

CLIENT DEVICE IDENTIFICATION INFORMATION
MOVEMENT INFORMATION
MOVING AMOUNT
MOVING DIRECTION
ACTION INFORMATION
RENDERING RANGE
DISPLAY SETTING

FIG. 5B

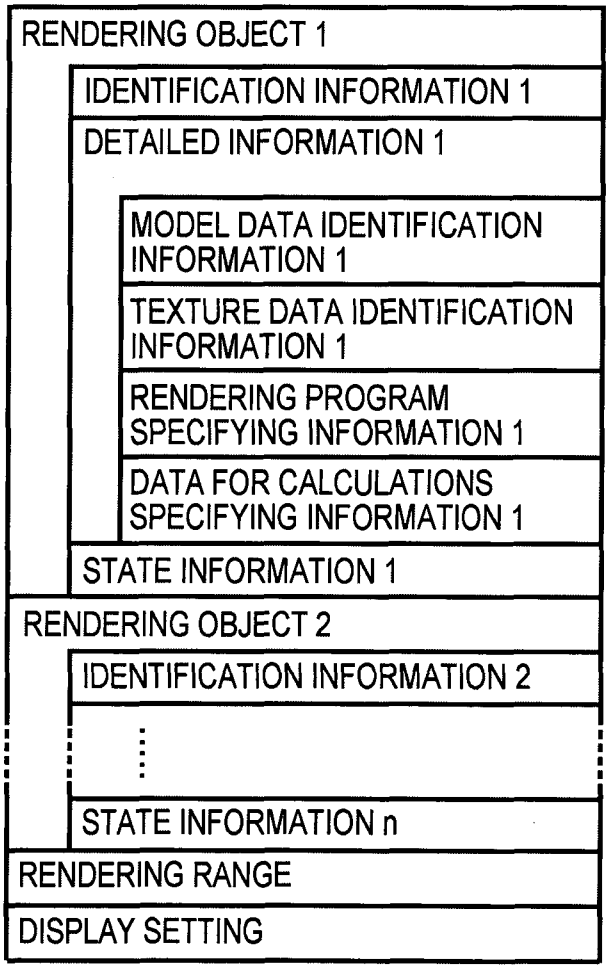

RENDERING OBJECT 1
IDENTIFICATION INFORMATION 1
DETAILED INFORMATION 1
MODEL DATA IDENTIFICATION INFORMATION 1
TEXTURE DATA IDENTIFICATION INFORMATION 1
RENDERING PROGRAM SPECIFYING INFORMATION 1
DATA FOR CALCULATIONS SPECIFYING INFORMATION 1
STATE INFORMATION 1
RENDERING OBJECT 2
IDENTIFICATION INFORMATION 2
STATE INFORMATION n
RENDERING RANGE
DISPLAY SETTING

FIG. 6

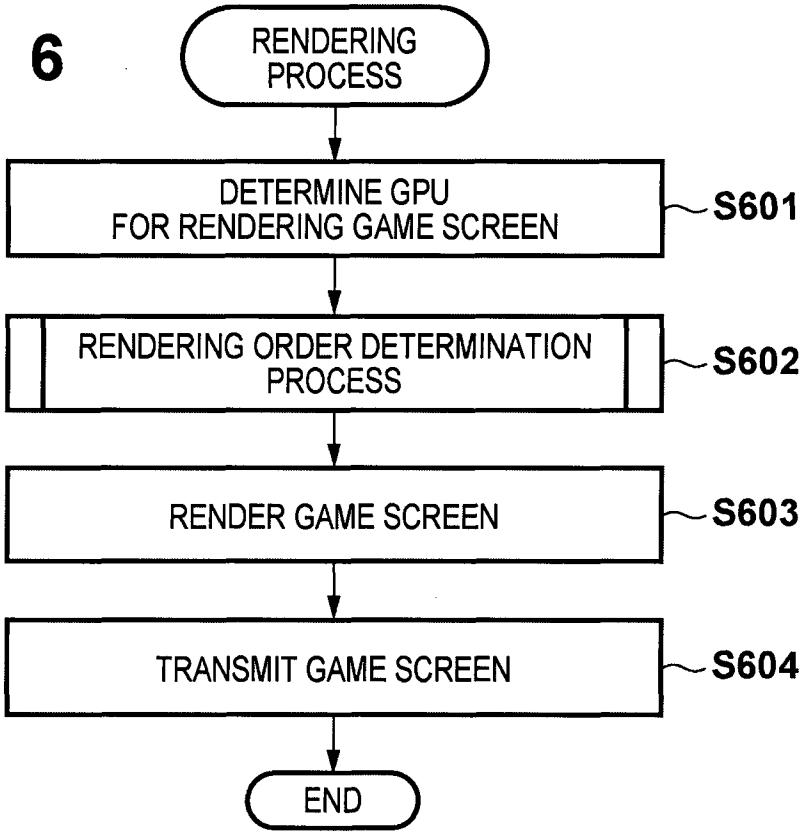

RENDERING PROCESS
DETERMINE GPU FOR RENDERING GAME SCREEN
S601
RENDERING ORDER DETERMINATION PROCESS
S602
RENDER GAME SCREEN
S603
TRANSMIT GAME SCREEN
S604
END

FIG. 7

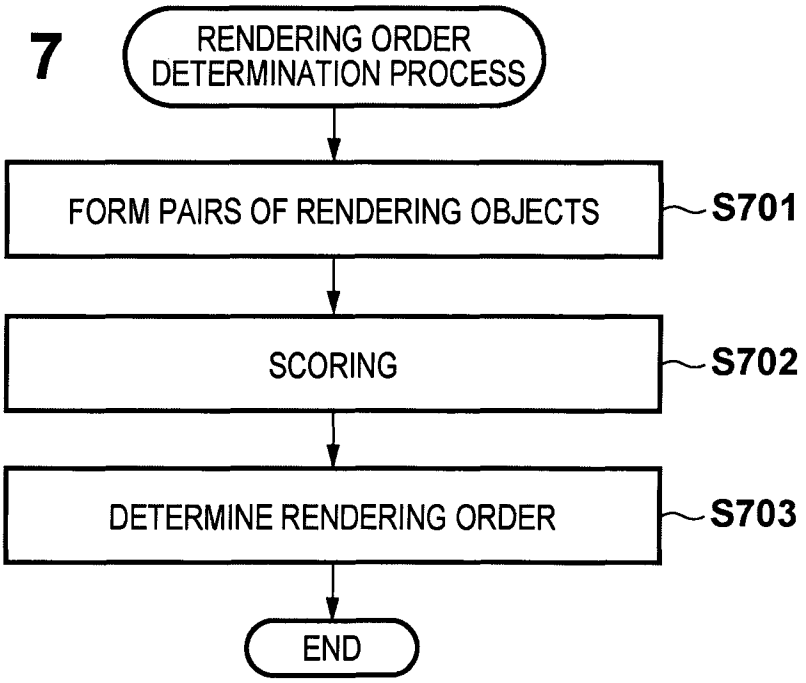

RENDERING ORDER DETERMINATION PROCESS
FORM PAIRS OF RENDERING OBJECTS
S701
SCORING
S702
DETERMINE RENDERING ORDER
S703
END

FIG. 8

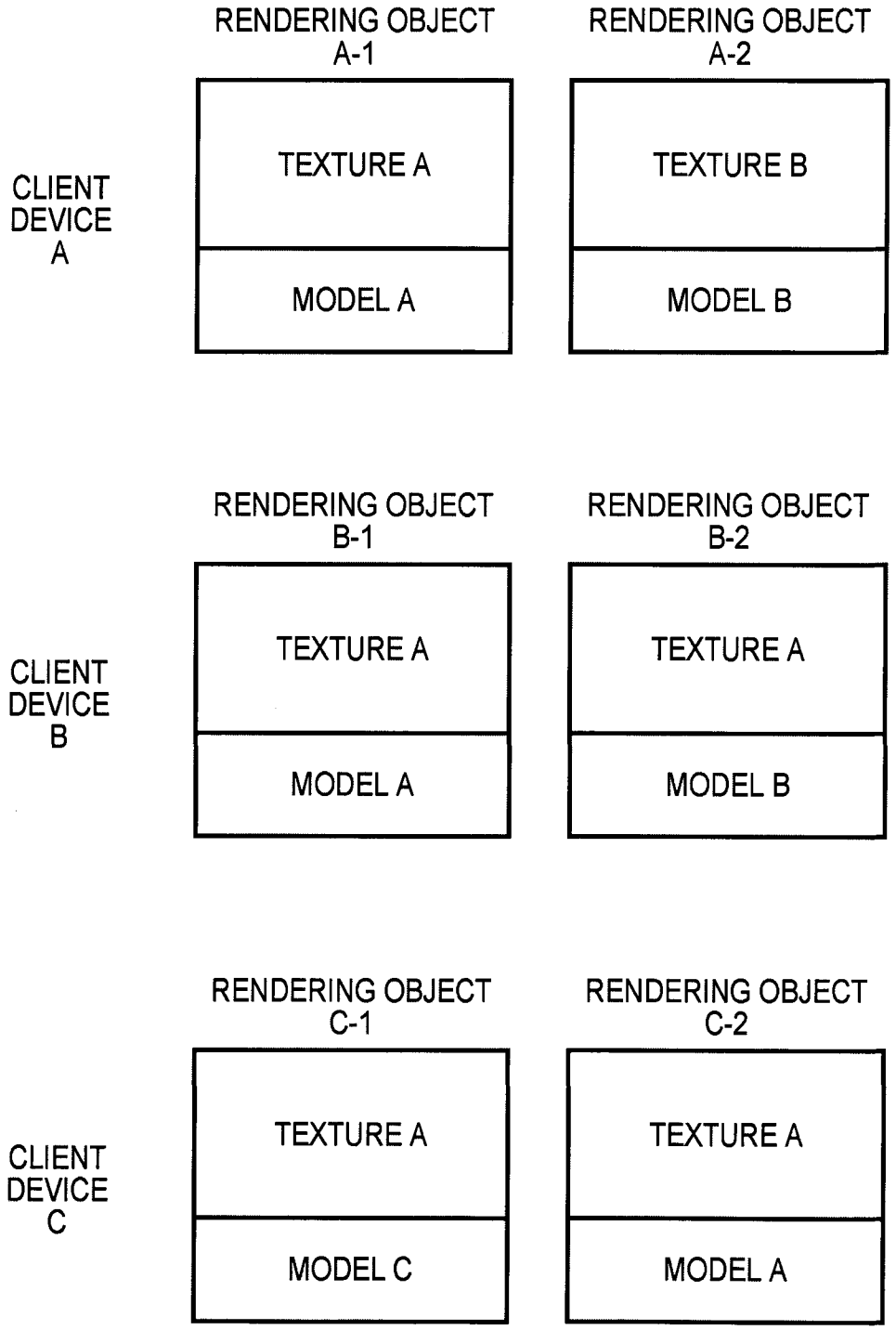

RENDERING OBJECT A-1
RENDERING OBJECT A-2
CLIENT DEVICE A
TEXTURE A
TEXTURE B
MODEL A
MODEL B
RENDERING OBJECT B-1
RENDERING OBJECT B-2
CLIENT DEVICE B
TEXTURE A
TEXTURE A
MODEL A
MODEL B
RENDERING OBJECT C-1
RENDERING OBJECT C-2
CLIENT DEVICE C
TEXTURE A
TEXTURE A
MODEL C
MODEL A

FIG. 9

	A-2	B-1	B-2	C-1	C-2
A-1	0	3	2	2	3
	A-2	0	1	0	0
		B-1	2	2	3
			B-2	2	2
				C-2	2

FIG. 10

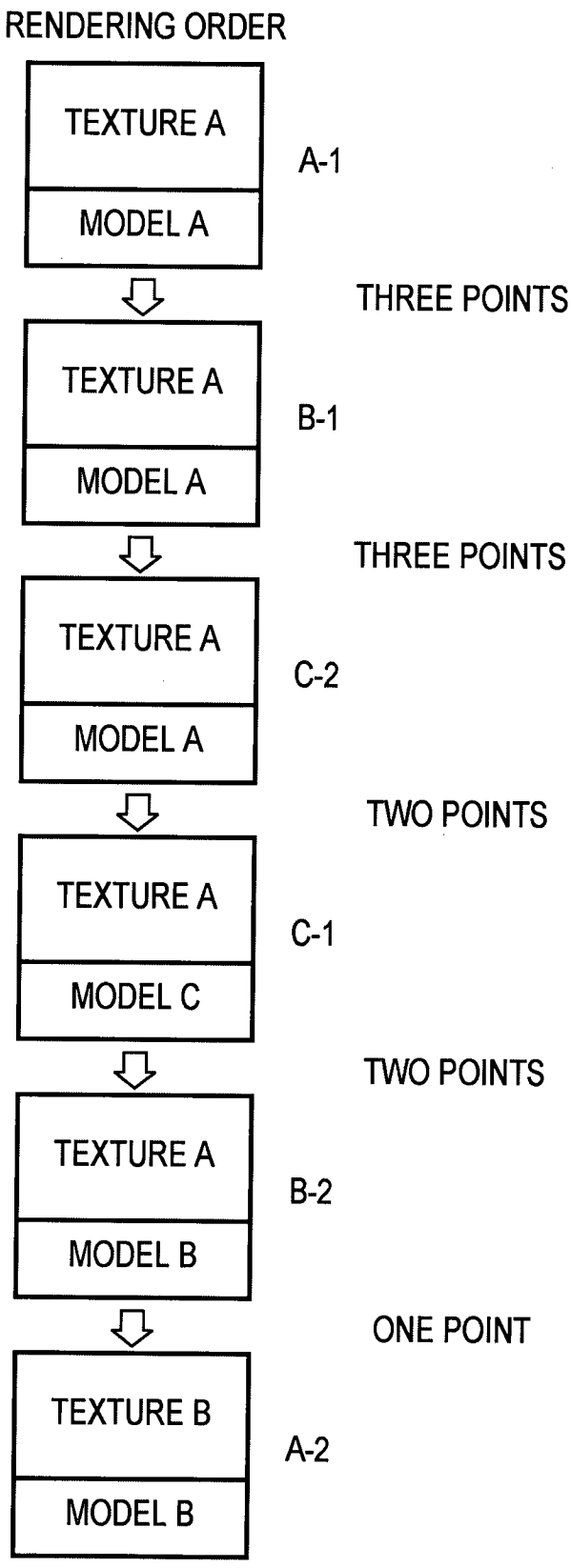
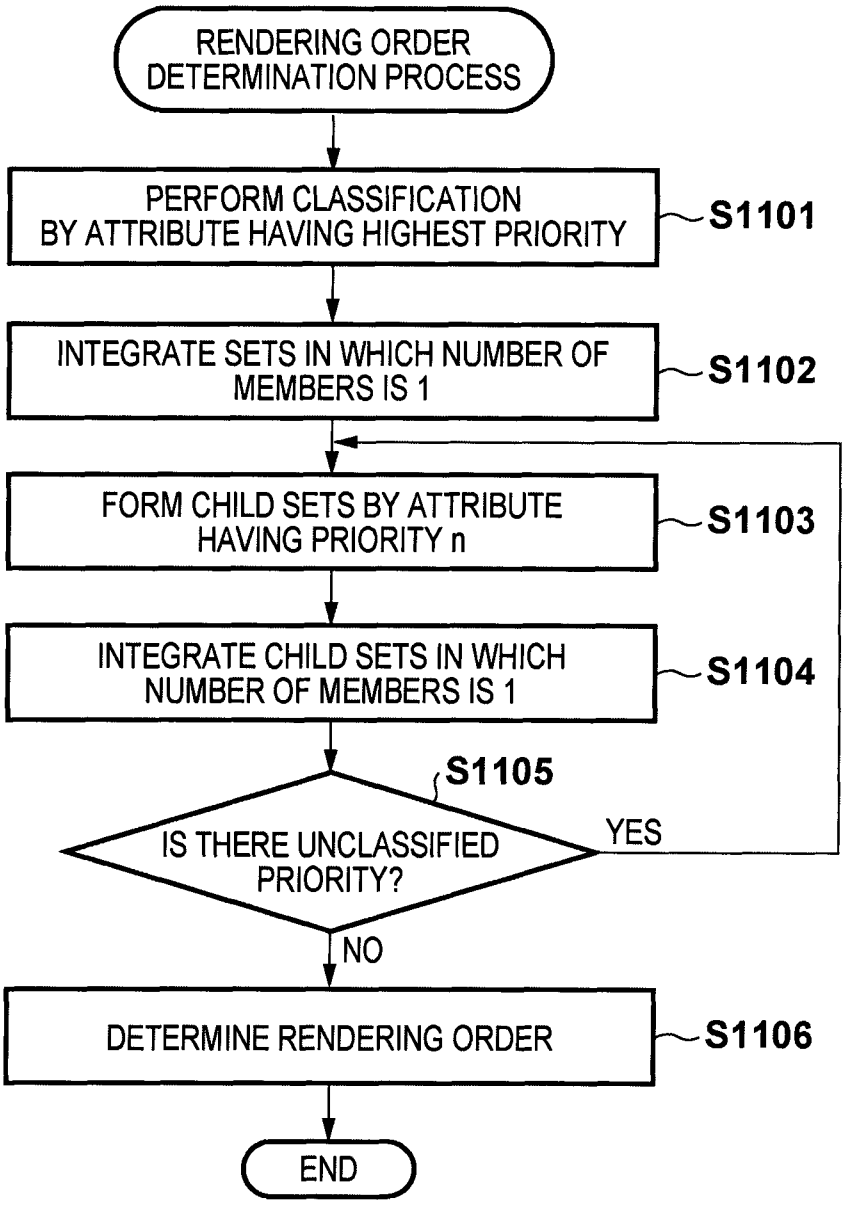

RENDERING ORDER
TEXTURE A
MODEL A
A-1
THREE POINTS
TEXTURE A
MODEL A
B-1
THREE POINTS
TEXTURE A
MODEL A
C-2
TWO POINTS
TEXTURE A
MODEL C
C-1
TWO POINTS
TEXTURE A
MODEL B
B-2
ONE POINT
TEXTURE B
MODEL B
A-2

FIG. 11


RENDERING ORDER DETERMINATION PROCESS
PERFORM CLASSIFICATION BY ATTRIBUTE HAVING HIGHEST PRIORITY
S1101
INTEGRATE SETS IN WHICH NUMBER OF MEMBERS IS 1
S1102
FORM CHILD SETS BY ATTRIBUTE HAVING PRIORITY n
S1103
INTEGRATE CHILD SETS IN WHICH NUMBER OF MEMBERS IS 1
S1104
S1105
IS THERE UNCLASSIFIED PRIORITY?
YES
NO
DETERMINE RENDERING ORDER
S1106
END

INTERNATIONAL SEARCH REPORT

International application No.
PCT/JP2012/062720

A. CLASSIFICATION OF SUBJECT MATTER

Int.Cl. G06T15/00(2011.01)i, A63F13/00(2006.01)i, A63F13/12(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)

Int.Cl. G06T15/00-15/87, A63F13/00-13/12

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

Published examined utility model applications of Japan 1922-1996
Published unexamined utility model applications of Japan 1971-2012
Registered utility model specifications of Japan 1996-2012
Published registered utility model applications of Japan 1994-2012

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

C. DOCUMENTS CONSIDERED TO BE RELEVANT

Category*	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
A	JP 2000-285255 A (SQUARE CO., LTD.) 2000.10.13, paragraphs [0049]-[0052] (No Family)	1-18
A	JP 2006-252292 A (NAMCO BANDAI GAMES INC.,) 2006.09.21, paragraph [0058] (No Family)	1-18

☐ 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 another 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
05.06.2012	12.06.2012

Name and mailing address of the ISA/JP	Authorized officer
Japan Patent Office 3-4-3, Kasumigaseki, Chiyoda-ku, Tokyo 100-8915, Japan	TAKASHI MURAMATSU 5H 9854 Telephone No. +81-3-3581-1101 Ext. 3531