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

(19) World Intellectual Property Organization
International Bureau

(43) International Publication Date
11 January 2007 (11.01.2007)

PCT

(10) International Publication Number
WO 2007/004837 A1

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

(21) International Application Number:
PCT/KR2006/002592

(22) International Filing Date: 3 July 2006 (03.07.2006)

(25) Filing Language: Korean

(26) Publication Language: English

(30) Priority Data:
10-2005-0059231 1 July 2005 (01.07.2005) KR
10-2005-0059232 1 July 2005 (01.07.2005) KR

(71) Applicant *(for all designated States except US)*: **NHN CORPORATION** [KR/KR]; Bundang Venture Town, 25-1, Jeongja-dong, Bundang-gu, Seongnam-si, Kyunggi-do 463-844 (KR).

(72) Inventor; and

(75) Inventor/Applicant *(for US only)*: **KIM, Jeong Ju** [KR/KR]; No. 104-304, Sangnoksu Apt., Irwon-dong, Gangnam-gu, Seoul 135-230 (KR).

(74) Agent: **CHUN, Sung Jin**; MUHANN Patent & Law Firm, 5th Fl., Youngpoong Building, 142 Nonhyun-dong, Kangnam-gu, Seoul 135-749 (KR).

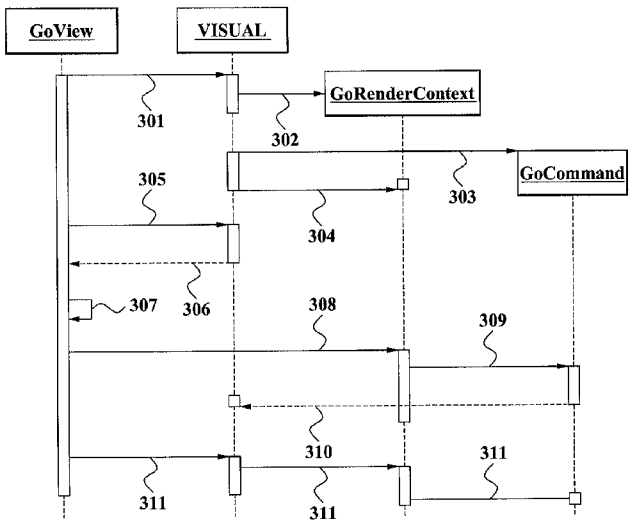
(81) Designated States *(unless otherwise indicated, for every kind of national protection available)*: AE, AG, AL, AM, AT, AU, AZ, BA, BB, BG, BR, BW, BY, BZ, CA, CH, CN, CO, CR, CU, CZ, DE, DK, DM, DZ, EC, EE, EG, ES, FI, GB, GD, GE, GH, GM, HN, HR, HU, ID, IL, IN, IS, JP, KE, KG, KM, KN, KP, KZ, LA, LC, LK, LR, LS, LT, LU, LV, LY, MA, MD, MG, MK, MN, MW, MX, MZ, NA, NG, NI, NO, NZ, OM, PG, PH, PL, PT, RO, RS, RU, SC, SD, SE, SG, SK, SL, SM, 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, HU, IE, IS, IT, LT, LU, LV, MC, NL, PL, PT, RO, SE, SI, SK, TR), OAPI (BF, BJ, CF, CG, CI, CM, GA, GN, GQ, GW, ML, MR, NE, SN, TD, TG).

Published:
— *with international search report*

[Continued on next page]

(54) **Title:** METHOD FOR RENDERING OBJECTS IN GAME ENGINE AND RECORDABLE MEDIA RECORDING PROGRAMS FOR ENABLING THE METHOD



(57) **Abstract:** A method of rendering objects in a game engine, in which at least one view requests at least one visual for RenderContext information, and the view registers the RenderContext information which has been received from the visual, sorts the registered RenderContext information according to a predetermined sorting algorithm, and renders the objects according to the sorted RenderContext information, is provided. Also, a method of rendering objects according to the present invention may display a multiview via a plurality of independent views and visuals corresponding to the independent views in a game engine. Accordingly, a method of rendering objects according to the present invention sorts and renders predetermined RenderContext information which is included in a visual according to a sorting rule which is provided in a game application in a view, and thereby may perform differentiated rendering according to a feature of a game which is provided in a respective game application.

WO 2007/004837 A1

For two-letter codes and other abbreviations, refer to the "Guidance Notes on Codes and Abbreviations" appearing at the beginning of each regular issue of the PCT Gazette.

METHOD FOR RENDERING OBJECTS IN GAME ENGINE AND RECORDABLE MEDIA RECORDING PROGRAMS FOR ENABLING THE METHOD

Technical Field

The present invention relates to a method of rendering of objects, which are provided in a game engine, to various types of game applications, and more particularly, to a method of rendering objects in a game engine, in which at least one view requests at least one visual for RenderContext information, and the view registers the RenderContext information which has been received from the visual, sorts the registered RenderContext information according to a predetermined sorting algorithm, and renders the objects according to the sorted RenderContext information.

Background Art

Currently, a computer terminal provides a user various types of three-dimensional (3D) games. In order to provide a realistic image to be displayed in a 3D game on a two-dimensional monitor, various types of rendering skills are required. A method of rendering which is currently performed in a game engine will be described in detail in FIG. 1.

FIG. 1 illustrates a configuration of a view 101 and a visual 102 performing rendering when receiving a rendering command from a game application according to a conventional art.

In order to depict a predetermined object, a game application 100 transfers the rendering command, with respect to the object, to a view 101. Also, the view 101 determines information of the object such as texture information, brightness level information, perspective information, and type of the object. The view 101 calls a visual 102 and transfers the rendering command according to the information of the object.

The visual 102 includes at least one RenderContext information 103. The RenderContext information 103 is a function to display the information of the object such as the texture, the brightness level, the perspective, and the type of the object. The RenderContext information 103 is included and maintained in the visual 102. The

visual 102 which has received the rendering command from the view 101 renders the object from the fixed RenderContext information 103. As an example, the view 101 for depicting a scene where 1) nature is set as a background, 2) the sun exists as a light source on a left side, and 3) a viewpoint is a first-person view in the game application 100, transfers a rendering command with respect to the 1), 2), and 3) described above. The visual 102 calls the RenderContext information 103 corresponding to the 1), 2), and 3) described above of the RenderContext information 103, and performs the rendering of the scene. The RenderContext information includes a pointer of a visual, state information including a light source and a rendering state, quality information, and transform information of a visual.

However, according to the conventional art, the view 101 calls the visual 102 and performs the rendering in a conventional game engine. Also, in the conventional method, the RenderContext information 103 included in the visual 102 is included to display a particular object. Accordingly, various types of game applications may not be supported by using a single game engine. Specifically, in order to interoperate with a particular game application, development of a specified game engine for a corresponding game is required. Also, a game engine which may be generally used in various types of games is required.

Also, when the visual 102 does not include the RenderContext information 103 for performing rendering of the view 101 which has been requested by the game application 100, a complex coding is needed. The complex coding is for performing operations such as adding, deleting, and modifying the RenderContext information 103 which is fixed (*and included?) in the visual 102. In this instance, the operations described above may cause an increase in maintenance and repair costs which are the largest part of a production cost for a program. Accordingly, a negative effect on a multimedia industry, including a game industry, may be caused.

Also, according to the conventional art, a configuration that a single view 101 is displayed by performing rendering via RenderContext information included in the visual 102 may not depict a plurality of scenes in a single screen. In the conventional art, it is possible to display the view 101 which is facing forward in a car running forward. However, it may be very difficult to simultaneously display another scene (*another view?) in a rear view mirror which displays a rear. In order to

simultaneously display the view 101 and another view 101 in a particular game engine, a subroutine of a complex coding, which calls the view 101, should be newly added. Also, when the game engine is used in another game, the subroutine is no longer used. Specifically, a method to display a plurality of views according to the conventional art may not provide a generalized game engine. Also, the method according to the conventional art may cause a decrease in game speed due to an addition of unnecessary subroutines.

Accordingly, development of a generalized game engine which can reflect features of various types of game applications is required. Also, a method of rendering which can reduce coding process which is added when adding, deleting, and modifying the RenderContext information 103, and provide a flexible multiview is highly required.

Disclosure of Invention

Technical Goals

The present invention provides a method of rendering objects, in which a view sorts and renders predetermined RenderContext information included in a visual according to a predetermined sorting algorithm which is provided in a game application, and thereby, the view can perform various rendering according to a feature of a game which is provided in each of various types of game applications.

The present invention also provides a method of rendering objects, which can change RenderContext information included in a visual by referring to a BaseRenderContext, and thereby can change the RenderContext information more flexibly.

The present invention also provides a generalized multiview in which a view does not refer to another view via a subroutine to display a multiview, and a plurality of views are maintained in parallel, and a visual and BaseRenderContext associated with each of the plurality of views are also maintained.

Technical solutions

According to an aspect of the present invention, there is provided a method of rendering an object in a game engine, the method including: transmitting a RenderContext (RC) query signal which requests a RenderContext information

registration which is necessary for rendering the object to a visual in a view; receiving a response signal with respect to the RC query signal from the visual and registering RenderContext information corresponding to the response signal in the view; requesting a predetermined game application to sort the registered RenderContext information in the view; receiving a result of the sorting of the registered RenderContext information from the game application in the view; and calling a predetermined rendering command and controlling the view to render the object according to the sorted RenderContext information in the view, wherein the game application comprises a predetermined sorting algorithm for sorting the RenderContext information.

According to another aspect of the present invention, there is provided a method of rendering an object in a plurality of views in a game engine, the method comprising: transmitting a first RC query signal which requests a RenderContext information registration which is necessary for rendering the object to a first visual in a first view; transmitting a second RC query signal which requests a RenderContext information registration which is necessary for rendering the object to a second visual in a second view; receiving a response signal with respect to the first RC query signal from the first visual in the first view, receiving a response signal with respect to the second RC query signal from the second visual in the second view, and registering the RenderContext information corresponding to each of the response signal in the first view and the second view; and calling a predetermined rendering command in the first view and the second view, and controlling the first view and the second view to render the object according to the RenderContext information, wherein the game application comprises a predetermined sorting algorithm for sorting the RenderContext information.

Brief Description of Drawings

FIG. 1 illustrates a configuration of a view and a visual performing rendering when receiving a rendering command from a game application according to a conventional art;

FIG. 2 illustrates a class diagram of RenderContext information in a game engine according to an embodiment of the present invention;

FIG. 3 is illustrates a sequence diagram in which a view receives RenderContext information from a visual, and sorts and renders the RenderContext

information according to an embodiment of the present invention;

FIG. 4 illustrates an example of sorting RenderContext information according to an embodiment of the present invention;

FIG. 5 illustrates a class diagram of RenderContext information in a game engine which supports a multiview according to an embodiment of the present invention; and

FIGS. 6 and 7 illustrate examples of scenes which are embodied in a multiview according to an embodiment of the present invention.

Best Mode for Carrying Out the Invention

Hereinafter, an indirect rendering method in a game engine according to the present invention, and an indirect rendering method in a game engine which supports a multiview according to the present invention will be described in detail with reference to the accompanying drawings.

FIG. 2 illustrates a class diagram of RenderContext information in a game engine according to an embodiment of the present invention.

Referring to FIG. 2, a view 202 receives a rendering command with respect to a predetermined object from a game application. The object is depicted in the view 202. For example, the object includes a description of objects such as an apple, a tree, a person, the sun, a background, a shadow, a desk, and the like. Each of the objects has its own features including texture information, brightness level information, perspective information, and a type of an object. The view 202 confirms the features of the objects. The view 202 requests the RenderContext information corresponding to the features of the objects to the visual 203 in order to perform rendering of the objects. As an example, the view 202 requests the RenderContext information, which is "What kind of RenderContext information is required, when an object having which type of particular features should be rendered?" in operation 204.

The visual 203 confirms the RenderContext information which is currently included in the visual 203, and provides the view 202 including predetermined RenderContext information as a response signal with respect to operation 204 received from the view 202 in operation 205. Also, the view 202 registers the RenderContext information, and sorts the RenderContext information according to a sorting algorithm

of RenderContext information provided in the game application. The view 202 calls a RenderCommand, and renders the objects according to the sorted RenderContext information. The objects which have been rendered are displayed as a single scene.

When the visual 203 does not include the RenderContext information with respect to operation 204, the visual 203 may update the RenderContext information from BaseRenderContext. Specifically, the visual 203 searches the RenderContext information which has been requested from the view 202. Also, when the RenderContext information is not currently included in the visual 203, the visual 203 requests a BaseRenderContext 201 to send the RenderContext information, and updates the RenderContext information.

The BaseRenderContext 201 includes a rendering state information, shading, a RenderCommand, quality information, or a transform information. Also, the BaseRenderContext 201 discloses information of rendering to a view. The rendering state includes values corresponding to each state for rendering the objects, such as a transparent state, an opaque state, a state of gradually changing from a transparent state to an opaque state, or a state of gradually changing from an opaque state to a transparent state. Specifically, the rendering state includes the opaque state for rendering the objects such as the desk, the apple, the person, and the tree, the transparent state for rendering the object such as a glass or water, and the opaque state or a translucent state for rendering the objects such as fog or smoke. In this instance, the translucent state is a state which the transparent state and the opaque state are suitably mixed.

The quality includes brightness level, texture, and the like. Specifically, the quality specifying the brightness level of the object according to an irradiated amount of light which is reflected on the object, and the texture of the object according to a state of a surface of the object, surfaces of fabrics, clothes, skin, or a road.

The RenderCommand is a function which is called so that the view 202 executes a command to start rendering the objects by using the sorted RenderContext information. The RenderCommand may be equally applied to most game applications.

The transform includes information for depicting perspective. For example, scenes positioned in an order of nearest to farthest or of farthest to nearest may be depicted through the transform. Accordingly, the BaseRenderContext 201 indicates information associated with rendering of the visual 203. In this instance, the visual

203 is an object for depicting a scene. Also, the BaseRenderContext 201 participates in processing the rendering of the view, and enables an indirect rendering process with respect to the object.

The BaseRenderContext 201 may be previously generated by a predetermined visual designer. The game engine according to the present invention may process rendering with respect to the visual 203 by using RenderContext 206 which has been first generated by the visual writer. Also, the RenderContext 206 includes configuration information (T_Type) 207 as a specified RenderContext. The configuration information (T_Type) 207 enables an embodiment which is not influenced by a specific configuration as a part of a template class of C++ language.

FIG. 3 is illustrates a sequence diagram in which a view receives RenderContext information from a visual, and sorts and renders the RenderContext information according to an embodiment of the present invention.

In operation 301, an initialization of a device is performed to use a visual including RenderContext information. In operation 302, the visual initializes the RenderContext information which is currently included in the visual. The initialization includes a general initialization process such as an initialization declaration of variables which are included in each of the objects.

The RenderContext information includes at least one of rendering level information, rendering texture information, rendering brightness level information, and rendering perspective information with respect to the visual. Specifically, the visual maintains the RenderContext information to render a predetermined object.

In operation 303, the visual generates a RenderCommand (GoCommand) with reference to a BaseRenderContext. In operation 304, the visual sets the generated RenderCommand (GoCommand) in a RenderContext (GoRenderContext). Accordingly, the RenderContext (GoRenderContext) is ready to perform rendering of an object according to the RenderContext information.

Then, in operation 305, a view (GoView) transmits a RenderContext (RC) query signal which requests the RenderContext information registration which is necessary for rendering the object to a visual. The view (GoView) receives information of the object which should be rendered from a game application, and analyzes the information of the object. Then, the view (GoView) transmits the RC

query signal corresponding to the analyzed information of the object to the visual. Also, the view awaits receipt of the RenderContext information as a response signal from the visual. As an example, when the game application transmits information on a command that renders a person who throws a ball as an object to the view (GoView), the view (GoView) provides the visual the information as the RC query signal, and awaits the RenderContext information with respect to the RC query signal as the response signal.

While receiving the response signal with respect to the RC query signal from the visual in the view (GoView), when the RenderContext information, which has received the request for registration from the view (GoView), is not included in the visual, the visual collects the RenderContext information from the BaseRenderContext. In this instance, the BaseRenderContext is recording at least one RenderContext information. Accordingly, modification of coding may be simplified, when inputting or modifying the RenderContext information. Also, when an update of the RenderContext information in the BaseRenderContext is needed, the RenderContext information may be updated by further registering additional RenderContext information via the game application or a predetermined external device script. Accordingly, a method of rendering according to the present invention may be flexibly applied to various types of game applications.

In operation 306, the view (GoView) receives the response signal with respect to the RC query signal from the visual, and registers the RenderContext information corresponding to the response signal. As an example, when the visual receives a RC query signal to describe a person who throws a ball from the view (GoView), the visual transmits the RenderContext information for depicting a texture of the ball, a location of a light source, a route of the ball including viewing perspective, and a background, as the response signal with respect to the RC query signal. The view (GoView) which has received the RenderContext information registers the RenderContext information, and prepares for rendering of the object.

In operation 307, a predetermined game application is requested to sort the registered RenderContext information in the view, and a result of the sorting of the registered RenderContext information is received from the game application.

A game application maintains a rendering order of the RenderContext

information caused by a characteristic of a game. This characteristic is caused (*generated?) by technicians associated with game production such as a game designer. Since the rendering sequence highly affects depiction of a scene, a game engine is mostly changed based on the game application.

However, through operation 307, the game engine may not be changed based on the game application. Specifically, the game application determines an order of the sorting, which is based on the operation of the RenderContext information, according to the rendering sequence which has been previously generated. Also, the RenderContext information is sorted according to the order of the sorting. The sorting of the RenderContext information will be described in detail referring to FIG. 4.

FIG. 4 illustrates an example of sorting RenderContext information according to an embodiment of the present invention.

Referring to FIG. 4, the RenderContext information 401, which has been registered from a visual, has a fixed order of RC1, RC2, ..., RCn. The RenderContext information 401 is loaded in a stack or a queue, and is operated based on a loading sequence such as first-in first-out (FIFO) or first-in last-out (FILO). However, the RenderContext information 401 for a description should not be performed according to the loading sequence. Also, the RenderContext information 401 should be newly added to the stack or the queue according to a performing sequence of the RenderContext information of the game application. For this, the game application newly sorts the RenderContext information which was received from a view (GoView) according to a predetermined sorting rule, and transfers the sorted RenderContext information 402 to the view (GoView). The sorting rule is based on a rendering state, a shade, or a transform. Also, the sorting rule may be determined by game producers including game designers. As an example, a plurality of objects, which have a same state, shade or transform, included in a single view, is distinguished from the objects, which respectively have a different state, shade, or transform, included in the single view. Then, the sorting rule may be determined.

The sorted RenderContext information 402 is loaded in an order of RC3, RC2, RC7...as RenderContext information in the stack or the queue, and transferred to the view (GoView). Specifically, an arrangement of the RenderContext information which determines the rendering sequence is performed in the game application. Accordingly,

the game engine according to the present invention may be flexibly used in various types of game applications. Also, a complex coding for rendering objects according to a feature for each game may be simplified.

Referring again to FIG. 3, a predetermined rendering command is called in the view (GoView) in operation 308, the RenderCommand (GoCommand) is performed to render the object according to the sorted RenderContext information in operation 309. In operation 310, the RenderCommand (GoCommand) calls the sorted RenderContext information, and performs rendering of the object. Specifically, through the operations described above in FIG.3, coding which is added when adding, deleting, and modifying the RenderContext information may be reduced. In this instance, the RenderContext information is for rendering the object in the game engine according to the game application. Also, the rendering sequence of the RenderContext information may be determined for individual features of each of the game applications. Accordingly, a differentiated description may be available according to the game application, without separately modifying coding. Thus, various effects may be embodied.

In operation 311, when the rendering the object is completed, a temporary pause command of a device object is generated from the view (GoView). An establishment among the view (GoView), the visual, and BaseRenderContext (GoRenderCommand, GoCommand) stops, and a standby status is maintained.

The method of rendering according to another embodiment of the present invention may be used in a game engine which supports a multiview. The method of rendering with respect to the game engine which supports the multiview will be described in detail below.

FIG. 5 illustrates a class diagram of RenderContext information in a game engine which supports a multiview according to an embodiment of the present invention.

Referring to FIG. 5, views 520 receives a rendering command with respect to a predetermined object from a game application. The view 520 according to the present invention includes a plurality of views including a first view 521, a second view 522 through an n^{th} view 523. Each of the views refers to a visual 530. Accordingly, the first view 521, the second view 522 through the n^{th} view 523 may respectively describe an independent scene.

In order to render a particular object, the first view 521 requests the visual 203

to send the RenderContext information corresponding to a feature of the object. The visual 530 includes a first visual through an n^{th} visual corresponding to each of the views 520. Also, each of the views 520 may maintain an independent visual. The first visual through the n^{th} visual is called the visual 530 in this specification.

As an example of the request for the RenderContext information, the first view 521 transmits a first RC query signal 540 to the visual 530. In this instance, the first RC query signal 540 requests the RenderContext information, which indicates that "What kind of RenderContext information is required, when an object having which type of particular features should be rendered?". The visual 530 confirms the RenderContext information which is currently included in the visual 530, and provides the first view 521 with RenderContext information as a response signal with respect to the first RC query signal 540 received from the first view 521 in operation 550. Also, the first view 521 registers the RenderContext information, and sorts the RenderContext information according to a sorting algorithm of RenderContext information provided in the game application. Then, the first view 521 calls a RenderCommand, and renders the object according to the sorted RenderContext information. In order to describe the object, the second view 522, which has received a command of rendering a particular object from the game application, performs an operation in the same way as the first view 521. Specifically, the second view 522 transmits a second RC query signal 540 to the visual 530. In this instance, the first RC query signal 540 requests the RenderContext information. The visual 530 provides the second view 522 predetermined RenderContext information as a response signal with respect to the second RC query signal 550. Accordingly, a plurality of views 202 may be included in a single scene. As an example of the view 520, in a battle scene, an image in which a soldier wearing night vision goggles looks around and an image in which a background of the soldier is depicted may be provided all together. Accordingly, the object which has been rendered is displayed as a single scene.

When the RenderContext information is not included in the visual 530, with respect to the request 540 which has been transmitted to the visual 530 in the first view 521 or the second view 522, the visual 530 may update the RenderContext information from BaseRenderContext. The update of the RenderContext information with respect to a plurality of visuals refers to a method of updating RenderContext information

which has been described with FIG. 2.

Accordingly, a user may reduce an additional amount of work of coding is required for performing rendering in a particular game application. Also, compatibility with the game application may be improved. A complex coding in order to describe a plurality of views in a single scene, which adds views by increasing an amount of subroutines, may be simplified.

As described above, the sequence diagram is based on FIG. 3. In this instance, the sequence diagram illustrates the multiview receiving the RenderContext information from the visual, and sorts and renders the RenderContext information. Specifically, the first view and the second view respectively generate an individual RC query signal, receive the response signals with respect to each of the RC query signals from the visual, and register the RenderContext information corresponding to the response signals.

In this instance, the rendering of the first view and the second view may sequentially proceed due to respective independence. Also, the rendering of the first view and the second view may proceed in parallel.

As an example, when the first view (GoView) is a scene depicting a forward view where a car moves forward, the first visual transmits the RenderContext information for rendering the object depicting the forward view to the first view (GoView). Also, when the second view (GoView) is a scene depicting a rear view of the car in a rear view mirror, the second visual transmits the RenderContext information for rendering the object depicting the rear view to the second view (GoView). An embodiment of the multiview will be described in detail referring to FIGS. 6 and 7.

FIG. 6 illustrates examples of scenes which are embodied in a multiview according to an embodiment of the present invention.

A game application transmits a rendering command to a game engine according to the present invention. In this instance, the rendering command indicates that a direction of a car is a first view 601. In this instance, the first view transmits a first RC query signal to a first visual so as to render objects such as trees and a road with respect to the direction of the car. The first visual transmits predetermined RenderContext information to the first view 601 as a response signal with respect to the first RC query signal. Also, the first view 601, which has received the RenderContext information, requests the game application to sort the RenderContext information.

Also, the game application transmits a rendering command to a second view 602 so as to depict a rear view while the car moves forward. In this instance, the rendering command is to render the objects such as trees and a road which are located in a back area of the car. The second view 602 transmits a second RC query signal to a second visual. Also, the second visual transmits predetermined RenderContext information to the second view 602 as a response signal with respect to the second RC query signal. The second view 602 requests the game application to sort the RenderContext information. Accordingly, the first view 601 and the second view 602 are rendered, and the game engine displays scenes on a screen according to locations of the first view 601 and the second view 602. Also, when the rendering is completed, the first view 601 and the second view 602 stands by for depicting another scene. Accordingly, various scenes may be flexibly depicted via a plurality of views.

FIG. 7 illustrates examples of scenes which are embodied in a multiview according to an embodiment of the present invention.

The FIG. 7 illustrates a first view 701 depicting a scene where a gun is pointed at a soldier 700 that is an object, and a multiview including a perspective view 702, a top side view 703, and a front view 704 of a face of the soldier 700.

First, the game application transmits object information to describe the first view 701, and commands a rendering. In this instance, the first view 701 is one of a plurality of views which the game engine, according to the present invention, maintains. Then, the first view 701 transfers the object information to a first visual, and receives RenderContext information. Also, the game application transmits object information to describe the second view 702, and issues a command to perform rendering. In this instance, the second view 702 is one of the plurality of views except for the first view 701. Then, the second view 702 transfers the object information to a second visual, and receives RenderContext information. In the same way described above, the first view 701, the second view 702, a third view 703, and a fourth view 704 maintain RenderContext information of object. Also, the first view 701, the second view 702, the third view 703, and the fourth view 704 transmit the RenderContext information to the game application, and request the game application to sort the RenderContext information. The game application sorts the RenderContext information which has been received by a sorting rule which has been previously set. Then, the game

application transmits the sorted RenderContext information to the first view 701, the second view 702, the third view 703, and the fourth view 704. Then, the game engine displays the first view 701, the second view 702, the third view 703, and the fourth view 704 on a screen. In this instance, the views are displayed at locations where the game application designates. Thus, the game engine may provide a plurality of views more impartially by using a plurality of multiviews which are currently maintained, without a complex coding or a jump to subroutines.

In an embodiment of FIGS. 6 and 7, each view receives and uses the individual RenderContext information from a predetermined visual to embody an independent object. Conversely, a game engine according to another embodiment of the present invention adds RenderContext information which is related to a predetermined effect with respect to a first visual, and may generate a second visual. As an example, when the first view describes an ordinary battle scene, and the second view describes a scene where the battle scene is seen through night vision goggles, the second visual uses the RenderContext information which is included in the first visual by copying the RenderContext information. In this instance, the second visual may set a predetermined effect in the RenderContext information. In order to depict the battle scene, the second visual may set effects, such as brightness or a negative, in the RenderContext information which is included in the first visual.

As illustrated in FIG. 7, multiview technique in a game screen may be used in novel ways in order to display various effects. A multivew way according to the present invention, which is specified for each game, may have a configurational limit. Specifically, the multiview method may be used only in a corresponding game. Also, in the present invention, an indirect rendering technique may be generalized via the RenderContext information.

The above-described embodiment of the present invention may be recorded in computer-readable media including program instructions to implement various operations embodied by a computer. The media may also include, alone or in combination with the program instructions, data files, data structures, and the like. The media and program instructions may be those specially designed and constructed for the purposes of the present invention, or they may be of the kind well known and available to those having skill in the computer software arts. Examples of computer-readable

media include magnetic media such as hard disks, floppy disks, and magnetic tape; optical media such as CD ROM disks and DVD; magneto-optical media such as optical disks; and hardware devices that are specially configured to store and perform program instructions, such as read-only memory (ROM), random access memory (RAM), flash memory, and the like. The media may also be a transmission medium such as optical or metallic lines, wave guides, etc. including a carrier wave transmitting signals specifying the program instructions, data structures, etc. Examples of program instructions include both machine code, such as produced by a compiler, and files containing higher level code that may be executed by the computer using an interpreter. The described hardware devices may be configured to act as one or more software modules in order to perform the operations of the above-described embodiments of the present invention.

Although a few embodiments of the present invention have been shown and described, the present invention is not limited to the described embodiments. Instead, it would be appreciated by those skilled in the art that changes may be made to these embodiments without departing from the principles and spirit of the invention, the scope of which is defined by the claims and their equivalents.

Industrial Applicability

According to the present invention, a method of rendering objects sorts and renders predetermined RenderContext information included in a visual according to a predetermined sorting algorithm which is provided in a game application, and thereby may perform various rendering according to a feature of a game which is provided in each of various types of game applications.

Also, according to the present invention, a method of rendering objects can change RenderContext information included in a visual by referring to a BaseRenderContext, and thereby can change the RenderContext information more flexibly.

Also, according to the present invention, a generalized multiview may be provided by maintaining a plurality of views in parallel, and a visual and BaseRenderContext associated with each of the plurality of views in order to display a multiview, without referring to another view via a subroutine in a single view.

CLAIMS

1. A method of rendering an object in a game engine, the method comprising:

transmitting a RenderContext (RC) query signal which requests a RenderContext information registration which is necessary for rendering the object to a visual in a view;

receiving a response signal with respect to the RC query signal from the visual and registering RenderContext information corresponding to the response signal in the view;

requesting a predetermined game application to sort the registered RenderContext information in the view;

receiving a result of the sorting of the registered RenderContext information from the game application in the view; and

calling a predetermined rendering command and controlling the view to render the object according to the sorted RenderContext information in the view,

wherein the game application comprises a predetermined sorting algorithm for sorting the RenderContext information.

2. The method of claim 1, wherein the RenderContext information comprises at least one of rendering level information, rendering texture information, rendering brightness level information, and rendering perspective information with respect to the visual.

3. The method of claim 1, wherein, in the receiving of the response signal, the visual collects the RenderContext information from a BaseRenderContext which is recording at least one RenderContext information, when the RenderContext information which has received the request for registration from the view is omitted from the visual.

4. The method of claim 3, wherein the BaseRenderContext further registers additional RenderContext information via the game application or a predetermined external script means.

5. A method of rendering an object in a plurality of views in a game engine, the

method comprising:

transmitting a first RC query signal which requests a RenderContext information registration which is necessary for rendering the object to a first visual in a first view;

transmitting a second RC query signal which requests a RenderContext information registration which is necessary for rendering the object to a second visual in a second view;

receiving a response signal with respect to the first RC query signal from the first visual in the first view, receiving a response signal with respect to the second RC query signal from the second visual in the second view, and registering the RenderContext information corresponding to each of the response signal in the first view and the second view; and

calling a predetermined rendering command in the first view and the second view, and controlling the first view and the second view to render the object according to the RenderContext information,

wherein the game application comprises a predetermined sorting algorithm for sorting the RenderContext information.

6. The method of claim 5, further comprising:

requesting a predetermined game application to sort the registered RenderContext information in the first view and the second view; and

receiving a result of the sorting of the registered RenderContext information from the game application in the first view and the second view,

wherein the controlling calls the predetermined rendering command in the first view and the second view, and renders the object with reference to the sorted RenderContext information.

7. The method of claim 5, wherein the game engine adds the RenderContext information associated with a predetermined effect to the first visual, and thereby, generates the second visual.

8. The method of claim 7, wherein the second visual copies and maintains the

RenderContext information which is included in the first visual.

9. The method of claim 5, wherein the RenderContext information comprises at least one of rendering level information, rendering texture information, rendering brightness level information, and rendering perspective information with respect to the first visual or the second visual.

10. The method of claim 5, wherein, in the receiving of the response signal with respect to the first RC query signal, the first visual collects the RenderContext information from a BaseRenderContext which records at least one RenderContext information, when the RenderContext information which has received the request for registration from the first view is omitted from the first visual

11. The method of claim 10, wherein the BaseRenderContext further registers additional RenderContext information via the game application or a predetermined external device script.

12. A computer-readable recording medium storing a program for implementing the method according to any one of claims 1 through 11.

FIG. 1

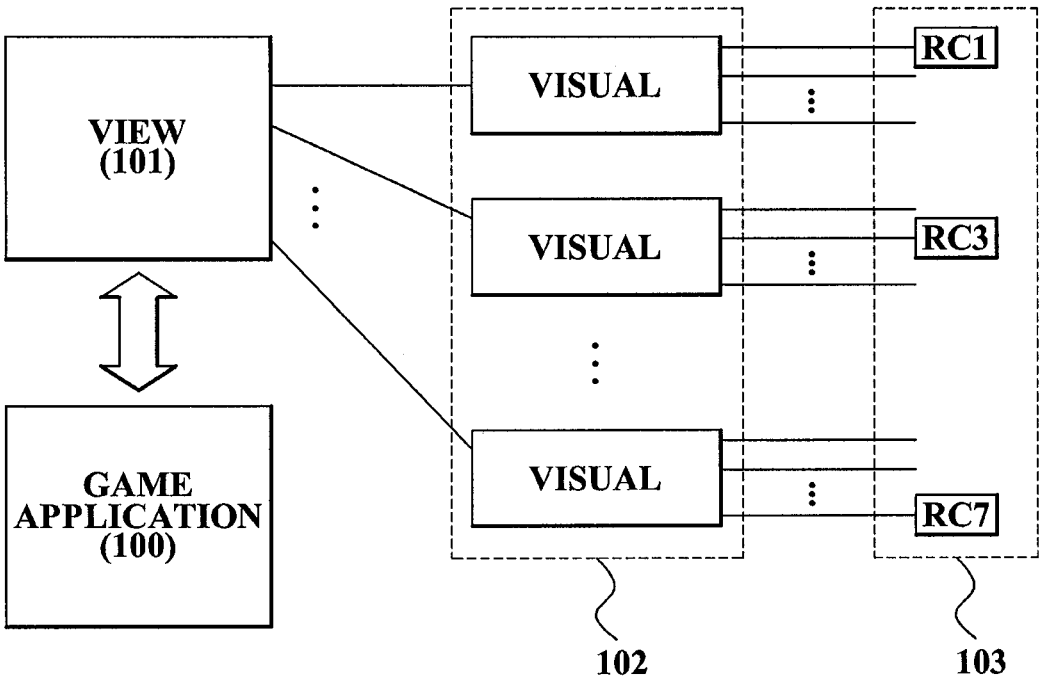

VIEW
(101)
GAME
APPLICATION
(100)
VISUAL
VISUAL
VISUAL
RC1
RC3
RC7
102
103

FIG. 2

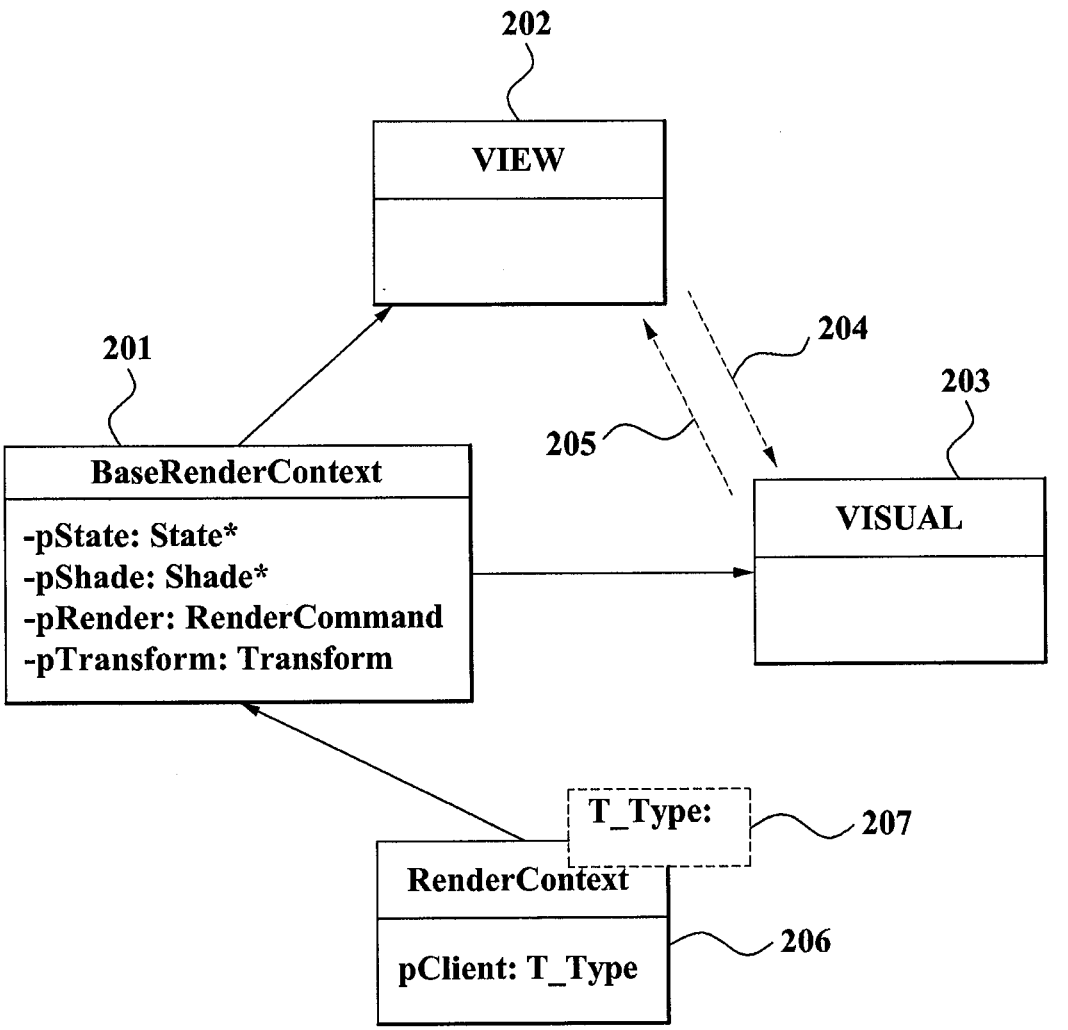

202
VIEW
201
204
203
205
BaseRenderContext
-pState: State*
-pShade: Shade*
-pRender: RenderCommand
-pTransform: Transform
VISUAL
T_Type:
207
RenderContext
pClient: T_Type
206

FIG. 3

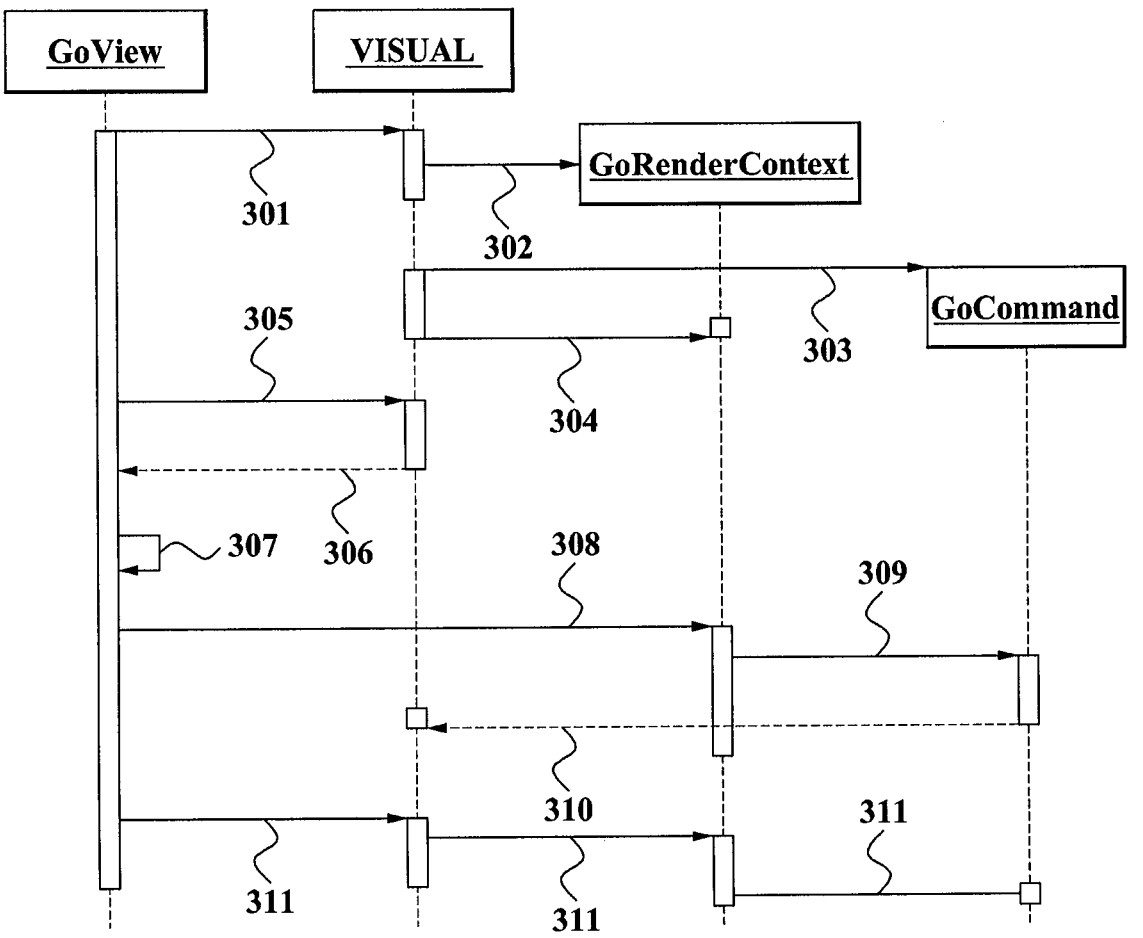

GoView
VISUAL
GoRenderContext
GoCommand
301
302
303
304
305
306
307
308
309
310
311
311
311

FIG. 4

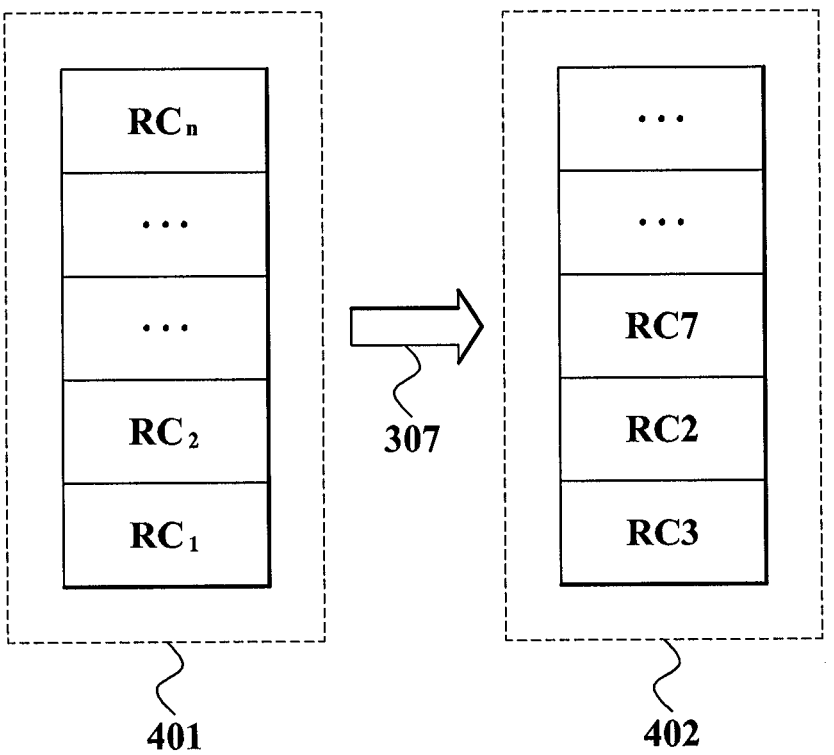


FIG. 5

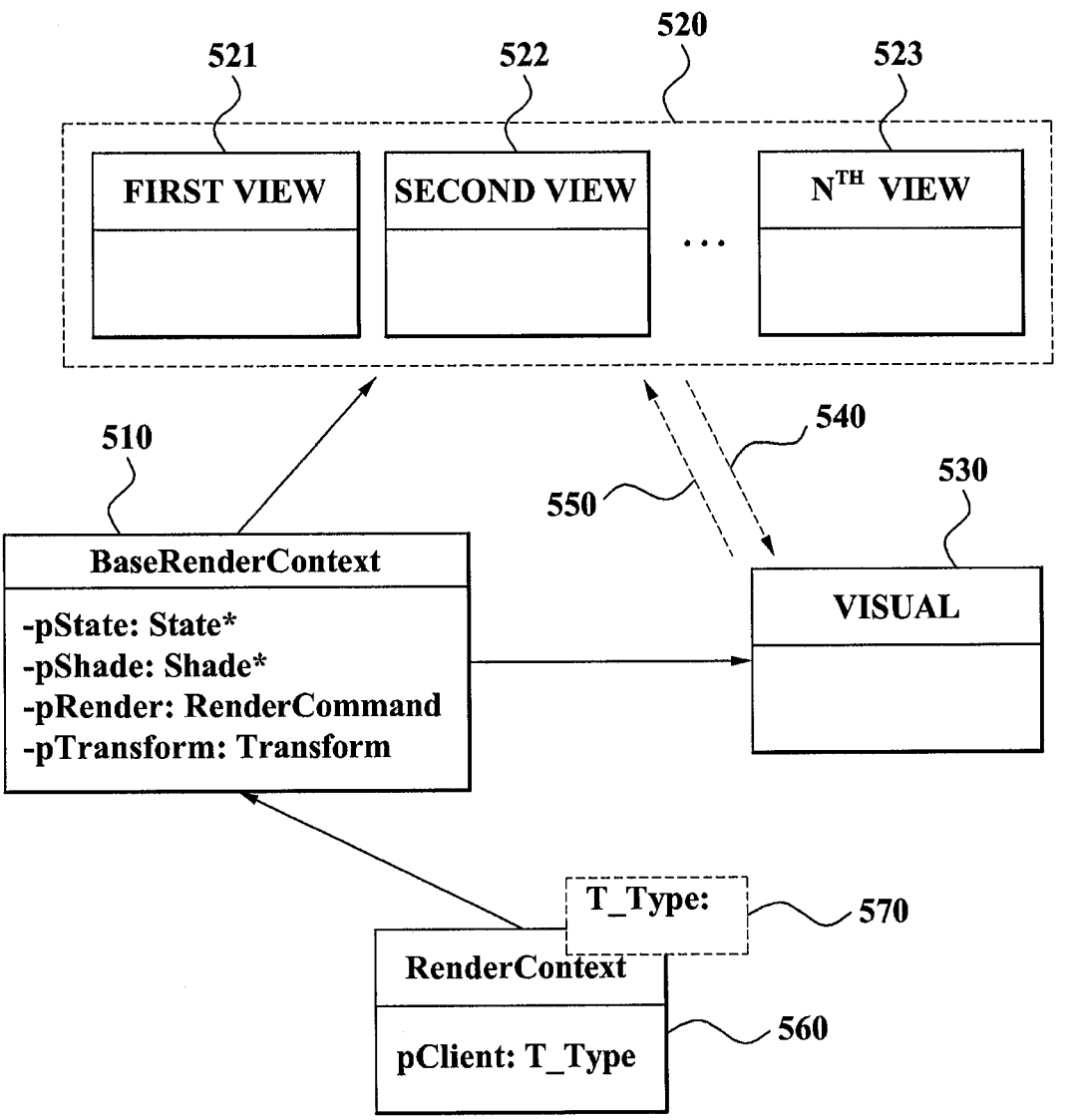

520
521
522
523
FIRST VIEW
SECOND VIEW
N^TH VIEW
510
540
550
530
BaseRenderContext
-pState: State*
-pShade: Shade*
-pRender: RenderCommand
-pTransform: Transform
VISUAL
T_Type:
570
RenderContext
pClient: T_Type
560

FIG. 6

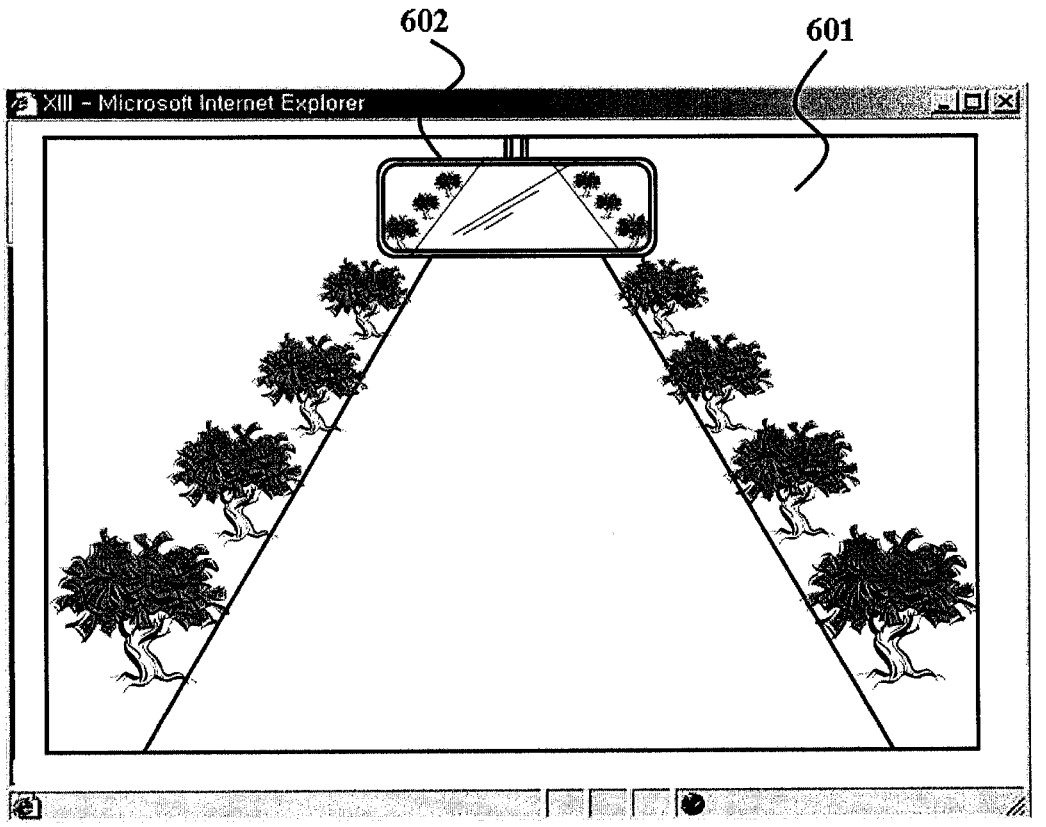
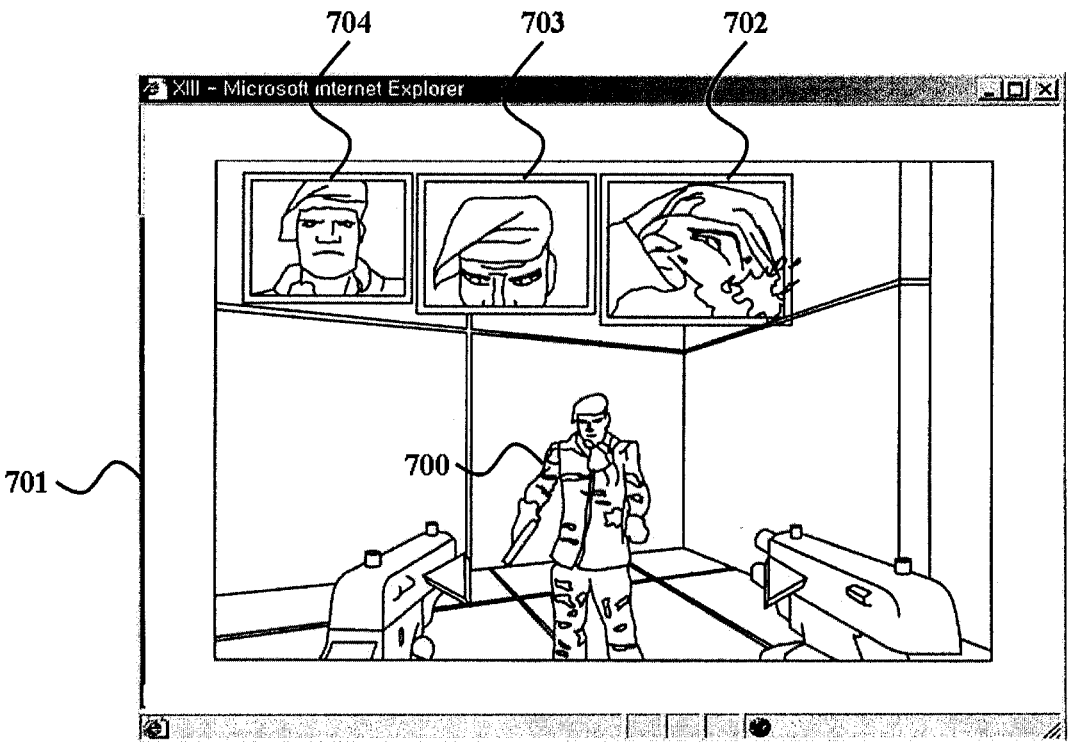


FIG. 7



INTERNATIONAL SEARCH REPORT

International application No.
PCT/KR2006/002592

A. CLASSIFICATION OF SUBJECT MATTER

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 8 G06T 15/00, G09G 5/00, G06T 15/40, G06T 13/00

Documentation searched other than minimum documentation to the extent that such documents are included in the fields searched
Korean Patents and applications for inventions since 1975
Korean Utility models and applications for Utility Models since 1975

Electronic data base consulted during the intertnational search (name of data base and, where practicable, search terms used)
eKIPASS(KIPO internal) "rendering, object, sorting "

C. DOCUMENTS CONSIDERED TO BE RELEVANT

Category*	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
A	US 6570565 B1(Woo Chan Park et al.) 27 May 2003 See claim 3, Fig. 6	1-12
A	US 2004/0189668 A1(Joseph S. Beda et al.) 30 September 2004 See claim 1, Fig. 3	1-12
A	KR 2001-50769 A(Sega Corporation.) 25 June 2001 See claims 1-5, 10, Fig. 3	1-12
A	KR 2005-59253 A(Nokia Corporation) 17 June 2005 See pages 1-2, Figs 1-3	1-12
A	KR 2005-61249 A(NHN Corportion) 22 June 2005 See page 2, claim 1, Fig. 2	1-12

☐ 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
13 OCTOBER 2006 (13.10.2006)	**13 OCTOBER 2006 (13.10.2006)**

Name and mailing address of the ISA/KR	Authorized officer
Korean Intellectual Property Office 920 Dunsan-dong, Seo-gu, Daejeon 302-701, Republic of Korea	KIM, Sung Hee
Facsimile No. 82-42-472-7140	Telephone No. 82-42-481-5728

Form PCT/ISA/210 (second sheet) (April 2005)

INTERNATIONAL SEARCH REPORT

Information on patent family members

International application No.
PCT/KR2006/002592

Patent document cited in search report	Publication date	Patent family member(s)	Publication date
US06570565 B	27.05.2003	KR1020020001072	09.01.2002
		KR2002001072A	09.01.2002
		US6570565B1	27.05.2003
		US6570565BA	27.05.2003
US20040189668A1	30.09.2004	AU2003204006A1	14.10.2004
		AU2003204006AA	14.10.2004
		AU2004279179A1	24.11.2005
		AU2004279179AA	24.11.2005
		BR200302161A	03.11.2004
		CA2428814AA	27.09.2004
		CA2428814A1	27.09.2004
		CN1534511A	06.10.2004
		EP01462936A2	29.09.2004
		EP1462936A2	29.09.2004
		HR20030390A2	28.02.2006
		HU200301191A0	28.07.2003
		HU200301191AB	28.10.2004
		IL155927A0	23.12.2003
		JP16295858	21.10.2004
		JP2004295858A2	21.10.2004
		KR1020040086043	08.10.2004
		MXPA03004412A	29.09.2004
		NO20032112A0	12.05.2003
		NO20032112A	28.09.2004
		NZ525666A	25.06.2004
		TR200300695A2	21.12.2004
		US2004189645A1	30.09.2004
		US2004189645AA	30.09.2004
		US2004189668AA	30.09.2004
		ZA200303507A	22.04.2004
KR1020010050769 A	25.06.2001	JP13101440	13.04.2001
		JP2001101440A2	13.04.2001
		US06690376	10.02.2004
		US6690376B1	10.02.2004
		US6690376BA	10.02.2004
KR1020050059253 A	17.06.2005	AU2002350540AA	04.05.2004
		EP1559074A1	03.08.2005
		JP18503355	26.01.2006
		JP2006503355T2	26.01.2006
		US20060146049A1	06.07.2006
		US2006146049AA	06.07.2006
		WO2004036504A1	29.04.2004
KR1020050061249 A	22.06.2005	None	