Title: THE PROCESSING METHOD OF DYNAMIC EVENT AND THE APPLICATION SYSTEM THEREOF IN ON LINE GAME ENVIRONMENT

Abstract: The present invention relates to a processing method of a dynamic event applied to an on-line game and an application system using the same in on-line game environment, and more particularly to a combo event that successively moves a block by a character and a block event that explodes a moved block and an adjacent block or giving a damage depending on superiority or inferiority of the characters by using correlation of characters. Such combo event or block event changes an attribute of an object composed of character and items, and also arouses interest and level change according to weight of each level in the processing method and the one-line game system using the same.
[Invention Title]

THE PROCESSING METHOD OF DYNAMIC EVENT AND THE APPLICATION SYSTEM THEREOF IN ONLINE GAME ENVIRONMENT

[Technical Field]

The present invention relates to a processing method of a dynamic event applied to an online game having simulation or the like and an application system using the same in online game environment, and more particularly to a processing method of a dynamic event that changes attributes of an object composed of a character and items and induces interest and level changes according to weight of each level, and an application system using the same in online game environment.

[Background Art]

A conventional online game market is mainly composed of MUD (Multi User Dungeon Game) in which rooms where events happen in a text format are arranged such that players cope with the events to raise their levels, and MUG (Multi User Graphical game) in which events are exhibited more visually. However, along with the development of computer graphic and propagation of equipments with excellent specification, the online game market has also been advanced, so various kinds and formats of online games are provided dependent on game genre, scale and pattern.

Generally, the online game may be classified based on genre into arcade game, MMORPG, strategic simulation, adventure game and so on. The genre such as adventure game mostly allows a user to enjoy solely, so its position is gradually decreased in the online game market, currently substantially not existing except for the package game market. In particular, MMORPG (Massively Multiplayer Online Role Playing Game) developed from the role playing game is a massive online game based on a plurality of clients and servers, which employs a message code system to minimize packet traffics with a main server and a user server. MMORPG is considered as the main stream in aspect of increased consumers and profits.
In addition, an on-line game is classified into a status-based on-line game, a skill-based on-line game and a mixed on-line game, depending on a point distribution pattern according to level up.

First, the status-based on-line game allows increase of status points such as strength, health, mental force and luck, and possibilities of increasing the abilities are different dependent on a character type. For example, an aggressive character increases strength, a defensive character increases health, a magician character increases a mental force, and an exploring character increases luck.

Also, the skill-based on-line game gives differential skills to a weapon used in the game such that the skill is mastered at suitable level and condition. Such skills may be learned when type, level and condition of a character are suitable.

[Disclosure]

[Technical Problem]

Such an on-line game gives an action event by manipulating an input device or utilizing a special key, and a representative action event is a combo event.

Such an action event gives a splendor to a stylish action and also induces interest to a gamer by asking a strategic idea to the gamer. This action event initiates an action with successive motions under a certain condition, beyond a pattern of one action satisfying one input condition. There are various kinds of combo events are included in on-line games, but such combo events are implemented just in a monotonous action that is repeated over and over.

In order to solve this problem, the present invention provides a new dynamic event with a different pattern from combo events currently suggested.

[Technical Solution]

In order to accomplish the above object, a processing method of a dynamic event in an on-line game environment according to the present invention is a processing method of a block event in an on-line game, which
occurs when a character moves a block, wherein a block is moved by the
canctter, and wherein, in case the block stops, it is checked whether the
stopped block has the same attribute as an adjacent block adjacent thereto,
and then a process for occurring a block event according to the checking
result is executed.

In another aspect of the present invention, a processing method of a
dynamic event in an on-line game environment according to the present
invention is a processing method of a block event in an on-line game, which
occurs when a character moves a block to cause explosion between adjacent
blocks, wherein an attribute is endowed to the character who moves the block
to induce explosion between blocks, and wherein, in case the character whose
attribute is maintained collides with another character, a process of
determining superiority and inferiority of attributes of the characters and
then giving a damage to an inferior character is executed.

In another aspect of the present invention, a processing method of a
dynamic event in an on-line game environment according to the present
invention is a processing method of a block event in an on-line game, which
occurs when a character moves a block to cause explosion between adjacent
blocks, wherein the explosion between adjacent blocks generates an explosion
object with a predetermined attribute and a predetermined life time, and
wherein, in case a character collides with an explosion area formed around
the explosion object, a process for determining superiority and inferiority of
attributes of the explosion object and the character and then giving a damage
to an inferior character is executed.

In another aspect of the present invention, a processing method of a
dynamic event in an on-line game environment according to the present
invention is a processing method of a combo event in an on-line game, which
conducts successive movement of a block using a single event, wherein, in
case a combo event signal for successive movement of a block is input and the successive movement is available as a result of checking availability of the successive movement of the block according to the input signal, a process for finding a block for the movement to set a combo value thereto according to the input signal and then moving the block according to the combo value is executed.

<17>

In another aspect of the present invention, an on-line game system for processing a dynamic event according to the present invention is a system in which a server and at least one client are connected through a communication network, wherein any one of the processing methods of a dynamic event explained above is executed in the server.

<18>

In another aspect of the present invention, a processing method of a dynamic event in an on-line game environment according to the present invention is a processing method of a combo event in an on-line game, which conducts successive movement of a block using a single event, the processing method comprising: a start signal inputting step for inputting a combo event start signal for the purpose of successive movement of a block; a combo input availability checking step for checking whether the input of a combo event signal for successive movement of the block is available; a combo inputting step for inputting the combo event signal; and a validity checking step for checking whether the block is capable of moving according to the input combo event signal.

<19>

In addition, an on-line game system for processing a dynamic event according to the present invention is a system in which a server and at least one client are connected through a communication network, wherein any one of the processing methods of a dynamic event, explained above, is executed in the client.

[Advantageous Effects]
According to the present invention, it is possible to provide a processing method of a dynamic event in a new pattern different from conventional on-line games.

In addition, a user may generate a character, enter a game and play the game on a board zone with a theme to acquire ability and experience, and in this way, the user may grow the character and change attributes of character and parts wearable on the character.

Also, the present invention induces competition among the same level users included in the same level by configuring level-based channels, and also may arouse motives for competition by providing user ranking and various rankings of characters.

In addition, by introducing a regional matching system for activation of regional communities, the present invention may give service such as user revitalization in the same region.

[Description of Drawings]

FIG. 1 is a flowchart illustrating a block event that is a dynamic event according to the present invention.

FIG. 2 is a flowchart illustrating event processing between characters according to endowed attributes.

FIG. 3 is a flowchart illustrating event processing between characters according to block attributes.

FIG. 4 is a flowchart illustrating a combo event that is a dynamic event according to the present invention.

FIG. 5 is a flowchart illustrating a combination checking process according to the present invention.

FIG. 6 is a flowchart illustrating a character attribute superiority determining process according to the present invention.

FIG. 7 is a flowchart illustrating a block attribute superiority determining process according to the present invention.

FIG. 8 is an online game image shot to which the dynamic event of the present invention is applied.
FIG. 9 is an online game image shot to which the dynamic event of the present invention is applied.

FIG. 10 is an online game image shot to which the dynamic event of the present invention is applied.

FIG. 11 is an online game image shot to which the dynamic event of the present invention is applied.

FIG. 12 is an online game image shot to which the dynamic event of the present invention is applied.

FIG. 13 is an online game image shot to which the dynamic event of the present invention is applied.

FIG. 14 is an online game image shot to which the dynamic event of the present invention is applied.

FIG. 15 is an online game image shot to which the dynamic event of the present invention is applied.

FIG. 16 is an online game image shot to which the dynamic event of the present invention is applied.

[Best Mode]

Hereinafter, a processing method of a dynamic event and an application system using the same in an on-line game environment according to a preferred embodiment of the present invention to accomplish the objects and effects will be described in detail with reference to the accompanying drawings.

FIG. 1 is a flowchart illustrating a block event that is a dynamic event. Referring to FIG. 1, the present invention may be applied to a game using blocks as follows, as one embodiment.

If a start step (AO) in which a character provided from a server is selected and generated by means of selection of a subscriber user and enters the game is conducted such that the subscriber user accesses a game environment, parts wearable to the character and change of attribute values are set in consideration of point and level of the subscriber, and then a game is progressed on a board zone that gives various themes.

In order to move a block by a character operated by user control, a
block moving step (Al) is conducted on the program according to recognition of keyboard input value information.

For the block moved through the moving step (Al), it is determined in a block stop checking step (A2) whether the movement of the block is stopped due to any obstacle or a set value. In case a stop condition is not satisfied, the process returns to the block moving step (Al) such that the block moving step (Al) is progressed again. If the stop condition set in the system is satisfied to stop the block, an attribute combination checking process executing step (AP) is executed, and then the event is ended (A3).

The attribute combination checking process (AP) conducts a process as shown in FIG. 5. That is to say, in case a block stopping its movement has same attribute as at least one block that contacts with the stopped block in an east, west, south or north direction, explosion is made. The attribute explosion may be conducted in a chain within a predetermined area such that any block contacted with an exploding block and having the same attribute as the exploding block is also exploded in a chain. The attributes may adopt paper, stone and scissors whose superiority and inferiority are already regularly promised, as an example. In the embodiment of the present invention, attributes are endowed with water, fire and ice (water > fire > ice > water). Each block is set to have any one of the above attributes in advance.

The above process may be implemented in systematic steps as follows.

The start step (A0) of the attribute combination checking process is executed. Then, the step API for checking an attribute of a block stopping its movement in the block event is executed to check an attribute of the corresponding stopped block. And then, an attribute sameness checking step (AP2) for checking whether at least one of blocks adjacent to the stopped block in east, west, south and north directions (namely, in four directions) has the same attribute as the stopped block is executed.

If there is no block with the same attribute in the attribute sameness checking step (AP2), the corresponding process is ended (AP7). If at least
one of the adjacent blocks has the same attribute as the stopped block, the step AP3 for exploding the stopped block and the corresponding adjacent block is executed.

In addition, an exploded block attribute sameness checking step (AP4) for checking whether the exploded adjacent block has the same attribute as another adjacent block (namely, adjacent blocks in remaining three directions except for the exploded stopped block) is executed. If there is an adjacent block with the same attribute, the step AP5 for exploding the corresponding adjacent block in a chain is executed.

After that, the process returns to the exploded block attribute sameness checking step (AP4) for checking whether the finally exploded block has the same attribute as adjacent blocks in remaining three directions, thereby checking whether or not to conduct the chain explosion again. This chain explosion may be conducted within a predetermined area, and in this case, a step of determining an explosion area may be executed before the exploded block attribute sameness checking step.

If there is no adjacent block with the same attribute as the finally exploded block, the step AP6 for endowing subsequent or complex character attributes to the character who moves the block is applied according to history of a chain of the exploded blocks, and then the corresponding process is ended (AP7).

The attribute endowing step (AP6) gives a value of exploded attribute (for example, water, fire or ice) to the character during a limited time as a weight endowed to the character who induces the explosion. In this case, in case explosion occurs in a chain to give chain attribute, the attribute values may be endowed subsequently in order to the explosions for a predetermined time, respectively. In addition, in case the exploded blocks have different attributes, different attributes may be endowed to the character.

Such endowment of attribute may be used for giving a damage to the character or decrease level/point (level/point is endowed by the system as a
compensation for victory, and it generally means any tool used for upgrading the character or purchasing separate parts wearable by the character) according to superiority or inferiority of attributes when the character comes in contact with another character during the game.

FIG. 2 is a flowchart illustrating event processing between characters according to endowed attributes. It is a processing flow for determining win or lose between a character who is endowed with an attribute in the character attribute endowing step (AP6) as shown in FIG. 5 and a character with no attribute or between characters endowed with different attributes, by means of the attribute combination checking process executing step (AP) according to block movement in FIG. 1. That is to say, the process is progressed when characters of different users moving in the game space contact with each other.

A start step (BO) is progressed to a step (Bl) for endowing an attribute to a character for n seconds. After that, an attribute retaining time expiration checking step (B2) is repeatedly conducted until a condition is satisfied. If the attribute retaining time expires (a state value: F), the process goes to an attribute canceling step (B6) to end the corresponding flow (B7). If an attribute retaining time remains (a stats value: T), a collision checking step (B3) is executed to determine whether collision between characters occurs.

In the collision checking step (B3), if there is no collision (a state value: N), the process returns to the attribute retaining time expiration checking step (B2) again to execute the attribute retaining time expiration checking step (B2) and the collision checking step (B3) repeatedly. In case there occurs collision (or, contact) between characters (a state value: Y), an attribute superiority determining process executing step (BP) is performed.

After that, a damage calculating step (B4) and a damage applying step (B5) are conducted, and then the corresponding flow ends (B7) after the
attribute canceling step (B6) is executed.

In the damage calculating step (B4), a damage is calculated in a time-based manner that counts a time from the point that the attribute is endowed to the character for n seconds in the attribute endowing step (B3) and then sets the damage to be weakened as the counted time is passing, or in a set value-based calculating manner that calculates intensity of a damage according to a kind of endowed attribute (complex attribute) using a set value previously given. In one embodiment, a damage is obtained in a way of calculating a difference between a basic attribute attacking damage and a victim shield value (damage = basic attribute damage - shield value of victim).

In addition, the damage applying step (B5) applies a damage of a victim character, which decreases a shield value (on occasions, decrease point or level together) and applies animation and effect on a client display to visually show the damage. In case a set value for survival (for example, HP) becomes zero, it is determined that the character cannot continue the game, so the character is processed as killed and then loses the game.

Also, if the damage relation between characters contacting with each other is applied as mentioned above, the attribute canceling step (B6) for canceling all attributes applied to the collided character is executed.

The attribute superiority determining process (BP) conducts the process as shown in FIG. 6, which determines superiority and inferiority between characters for damage calculation conducted later.

After a start step (BP0), a remaining attribute checking step (BP1) is executed to check whether any attribute is endowed to a character. In case characters with no attribute collide with each other, it is impossible to conduct a separate damage calculation, so the process is ended (BP6).

In case an endowed attribute (or, a remaining attribute) exists (a state value: Yes), a step (BP2) for inputting (and storing) attributes of characters with the remaining attribute to the server (or, an attribute
superiority determining unit) is executed. After that, a superior attribute checking step (BP3) is conducted such that, if an attribute of a character input earlier is superior to an attribute of a character input later (a state value: T), an increase flag is endowed to the earlier input character to input (and store) the increase flag (BP4), but if the attribute of the character input earlier is inferior to the attribute of the character input later (a state value: F), an increase flag is endowed to the later input character or a decrease flag is endowed to the earlier input character to input (and store) the flag (BP5). Then, the corresponding process is ended (BP6).

The increase or decrease flag is utilized as an operation data for damage calculation of FIG. 2, after the process is ended. The increase or decrease flag preferably includes information of the corresponding character and numerical information of increased or decreased value.

FIG. 3 is a flowchart illustrating event processing between characters according to block attributes. If blocks with the same attribute are exploded by executing the block movement and attribute combination checking process (AP) of FIG. 1, the explosion is maintained for a predetermined time, which shows the following flows.

That is to say, explosion between adjacent blocks with the same attribute is initiated (C0), and an explosion object generating step (C1) for generating an explosion object with a life time of n seconds is performed. After that, an explosion area is formed over a predetermined area around the explosion, and an object life checking step (C2) for checking whether the predetermined time of n seconds expires is executed. In case the life time of the object expires (a state value: F), an attribute explosion object life time expiring step (C6) for retrieving or terminating the generated object is executed, and then the corresponding process is ended (C7). If the object has a remaining life time (a state value: T), a collision checking step (C3) for checking collision between the explosion area and an approaching (or,
entering) character around the explosion area is executed. In case there is no collision, an object life time checking step (C2) for checking whether the predetermined time of \( n \) seconds expires and a collision checking step (C3) for checking collision between the explosion area and a character are repeatedly executed.

In case collision of a character in the explosion area is detected according to approach of the character thereto in the collision checking step (C3) (a state value: Yes), an attribute superiority determining process executing step (CP), a damage calculating step (C4), a damage applying step (C5) and an attribute endowing step (C6) are conducted. After that, the corresponding flow is ended (C7).

In the damage calculating step (C4), a damage is calculated in a time-based manner that counts a time from the explosion object generating step (C1) for generating an explosion object with a life time of \( n \) seconds and then sets the damage to be weakened as the counted time is passing, or in a set value-based calculating manner that calculates intensity of a damage according to a kind of endowed attribute (complex attribute) using a set value previously given. In one embodiment, a damage is obtained in a way of calculating a difference between an attacking force (or, an explosion force) of the object generated by explosion and a victim shield value (damage = attacking force of the explosion object - shield value of victim).

The attribute superiority determining process (CP) executes a process as shown in FIG. 7, which determines superiority and inferiority between an explosion object and a character for damage calculation conducted later.

After a start step (CPO), a remaining attribute checking step (CP1) is executed to check whether an attribute is endowed to a block. If the block is not an explosion object, a separate damage calculation is impossible, so the process is set to go to a step (CP6) for ending the process. In case an attribute (or, a remaining attribute) endowed to the block is an explosion object (a state value: Yes), an inputting (and storing) step (CP2) for inputting attributes of characters adjacent thereto to the server (or, an
attribute superiority determining unit) and a superior attribute checking step (CP3) are executed. In case the explosion attribute has an attribute superior to a character input later (a state value: T), an increase flag inputting (and storing) step (CP4) for endowing a decrease flag to the character input later or endowing an increase flag to the explosion object and then inputting flag information of the character adjacent to the explosion object is executed. On the contrary, in case the explosion object input earlier has an attribute inferior to the character input later, (a state value: F), a decrease flag inputting (and storing) step (CP5) is conducted by endowing an increase flag to the superior character input later or endowing a decrease flag to the explosion object input earlier, and then the corresponding process is ended (CP6).

After the process is ended, the increase or decrease flag is utilized as an operation data for damage calculation of FIG. 2, so the flag preferably includes information of the corresponding explosion object and the character and numerical information of increased or decreased value.

As a specific example of determining superiority in the attribute superiority determining process (BP), attributes may be classified into water, fire and ice, whose superiority relations are defined as water > fire > ice > water.

FIG. 4 is a flowchart illustrating a processing method of a combo event that is a dynamic event according to the present invention. A block used in the combo event is a term with an inclusive concept of a moving object, not limited to a hexahedral box, and the box is just an example.

The combo event initiated by a user may be applied to the aforementioned block event that causes a chain explosion among adjacent blocks by transmitting a keyboard input value for movement of a block by a character and thus executing the block moving and attribute combination checking process executing step (AP).

For the combo event, a combo data for direction change by a user is
input in the client, and the server receives the combo data to conduct the combo event.

A start step (SO) is executed in which a user prepares for combo input in the client. The start step (SO) may be conducted in a way that the user presses a specific key (for example, a spacebar key) promised for starting the combo input. After that, the process goes to a combo input availability checking step (Sl), and the combo input availability checking step (Sl) checks whether a stopped block exists within a predetermined distance in a direction faced by the character in a stop state.

In the combo input availability checking step (Sl), in case there is no block within a predetermined distance (a state value: No), the process goes to an ending step (or, a standby step) (S12) without giving any influence on the combo event. On the while, in case a block exists within a predetermined distance (a state value: Yes), the client executes a combo input collecting (and storing) step (S2) for collecting (and storing) n combo inputs composed of a combination of input values with respect to four directions, namely in east, west, south and north directions.

After that, a validity checking step (S3) for checking possibility of movement and whether there is a space for an object to move according to an input direction key is executed.

The combo input process at the client side may be implemented in a way that a user presses different direction keys predetermined times (generally, less than 5 times) successively while a specific key (for example, a spacebar key) of a keyboard is being pressed.

If the validity checking of the client is passed, the server executes a combo input receiving step (S4) for receiving the combo input signal and then executes a secondary validity checking step (S5).

In the combo input receiving step (S4), a specific object ID (ObjectID) having a 32 bit integer value for a target block and information about a combo list related to moving direction are input and received from the client (or, the user) to the server.
The secondary validity checking by the server may include (1) checking whether a block exists in a direction faced by the character, (2) checking whether a distance from the character is suitable for combo input, and (3) checking whether the block is available for moving.

If the combo value is not effective in the first validity checking step (S3) by the client and the second validity checking step (S5) by the server, the process goes to a block stopping step (S11) to end the combo event (S12) without giving any influence on the corresponding block. In the block stopping step (S11), the client sets a velocity of the block into 0 and removes moving sound or effects.

In addition, in case the above condition is satisfied in the second validity checking (a state value: Yes), a corresponding block is found to execute a combo value setting step (S6) so as to move the block according to the combo event.

As one scheme for the combo value setting, it is possible to query an object storage containing information of blocks and then call SetComboList that is a kind of setting method, thereby setting a combo value to the block corresponding to the object ID (Object ID).

If the combo value is set as mentioned above, a block moving step (S7) is executed, and then a block stop condition checking step (S8) is performed. In the block stop condition checking step (S8), it is checked whether the block should be stopped. In case the stop condition is not satisfied (a state value: F), the process goes to the block moving step (S7) again to repeat the block moving process. If the block stop condition is satisfied (a state value: T), a combo queue value consumption checking step (S9) is executed.

The block stop condition may include (1) a case that a block reaches an outer area of NPC (Non Player Character: not accessible by a user character but accessible only by a system character), (2) a case that a block collides (or, contacts) with another block, and (3) a case that a block collides (or, contacts) with other obstacles.
The combo queue value stores information about the number of combos input in the combo value setting step (S6) in a queue format, which has a FIFO (First-In First-Out) structure. Thus, storage information of a queue corresponding to one direction value is subsequently terminated by block movement. Thus, in case a queue value remains (a state value: Q>0), a combo direction changing step (S10) is executed to change the moving direction of the block into a secondly input direction, and then the block moving step (S7) is executed. After that, the block stop condition checking step (S8) is executed to check whether the block should be stopped. If the stop condition is not satisfied (a state value: F), the process goes to the block moving step (S7) again to repeat the block moving process. In case the block stop condition is satisfied (a state value: T), the combo queue value consumption checking step (S9) is performed.

If the queue value becomes 0 (zero) in the combo queue value consumption checking step (S9) by repeating the above process (a state value: Q=0, on occasions, a condition of Q0) may be additionally included to prevent deadlock), it is considered that there is no further movement by the combo event. Thus, a block stopping step (S11) is executed, and the combo event is ended (S12).

[Mode for Invention]

FIGs. 8 to 16 are on-line game image shots obtained by applying a dynamic event to an on-line game system, under an on-line game environment according to the present invention. Now, one embodiment of an on-line game implemented by the present invention will be explained with reference to FIGs. 8 to 16.

A player controls his/her character to push a block in the game, which gives damage to an opponent or combines and explodes at least two blocks with the same attribute to give damage to an opponent around the explosion. In the game, three attributes of fire, water and ice are given such that blocks with the same attribute are exploded when being combined. A character that explodes blocks obtains the corresponding attribute for a predetermined time
such that the character gives damage to an opponent character when being contacted with the character according to correlation among the attributes (fire > ice > water > fire). In this way, the character struggles for making the opponent character extinct by such damage in the end.

FIG. 8 is an image shot showing that a character positioned at a right upper end is driven to an NPC area, which is an outermost region that a character may reach, by attacking of another character using blocks, and thus the character comes to a crisis of leaving out of the game. FIG. 9 is an image shot showing that a fire attribute (left upper portion) and a water attribute (left lower portion) are endowed to characters that move blocks and thus cause explosion.

In addition, FIG. 10 shows that three characters (lower center) are confined in a corner by blocks and topography by an opponent with the use of blocks. As mentioned above, by using the dynamic event composed of combo event and block event according to the present invention, a user may play the game by combining or disturbing attribute explosion among blocks and also giving more damage to an opponent using superior attribute.

FIG. 11 shows a simple pulling action by a character, and FIG. 12 shows a 5-stage combo event that successively makes movement and direction change of a block. The combo event of the present invention may allow a user to input a combo instruction using a direction key while pressing a spacebar as shown in FIG. 16.

FIG. 13 is an image shot showing that some of various forms of characters and parts used for each character are assembled. FIGS. 14 and The combo event of the present invention may allow a user to input a combo instruction using a direction key while pressing a spacebar as shown in FIG. 16.
FIG. 13 is an image shot showing that some of various forms of characters and parts used for each character are assembled. FIGs. 14 and 15 are image shots showing a map that may be designed directly by a player, a moving tile that may make an instant movement, and blocks generated at random every time, which allows a user to play an on-line game with fresh feeling every time.

[Industrial Applicability]

The present invention may be applied for using a combo event or an action event in an on-line game, and also may be applied to a desktop game or a mobile equipment game in addition to the on-line game.
[CLAIMS]

[Claim 1]

A processing method of a block event in an online game, which occurs when a character moves a block,

wherein a block is moved by the character, and

wherein, in case the block stops, it is checked whether the stopped block has the same attribute as an adjacent block adjacent thereto, and then a process for occurring a block event according to the checking result is executed.

[Claim 2]

The processing method of a block event in an online game according to claim 1, wherein the process includes:

- a block moving step in which the block is moved by the character; and
- a block stop checking step for checking a stop state of the block,

wherein an attribute combination checking process for processing a block event according to correlation with respect to the stopped block.

[Claim 3]

The processing method of a block event in an online game according to claim 2, wherein the block stop checking step includes:

- determining whether the movement of the block should be stopped due to an obstacle or a set value, then continuously moving the block in case the stop condition is not satisfied or executing the attribute combination checking process in case the stop condition and thus the movement of the block is stopped.

[Claim 4]

The processing method of a block event in an online game according to claim 2, wherein the attribute combination checking process includes:

- an explosion and attribute endowing step in which, in case the stopped
block and an adjacent block adjacent thereto have the same attribute, the stopped and the adjacent block are exploded, and the attribute of the exploded block is endowed to the character who moved the stopped block.

[Claim 5]

The processing method of a block event in an on-line game according to claim 2, wherein the attribute combination checking process includes:

- checking an attribute of the stopped block;
- an adjacent block attribute sameness checking step for checking whether the stop block and an adjacent block adjacent thereto have the same attribute; and
- exploding the blocks with the same attribute in case the stopped block and the adjacent block have the same attribute.

[Claim 6]

The processing method of a block event in an on-line game according to claim 5, wherein the attribute combination checking process includes:

- an exploding block attribute sameness checking step for checking whether the adjacent block to be exploded has the same attribute as an additional adjacent block adjacent thereto; and
- in case the adjacent block and the additional adjacent block have the same attribute, making the blocks with the same attribute be exploded in a chain.

[Claim 7]

The processing method of a block event in an on-line game according to claim 4,

wherein the explosion and attribute endowing step endows the same attribute as the exploded block to the character during a predetermined time such that, in case the explosion occurs in a chain with a chain attribute, the attributes are endowed subsequently during a predetermined time.
respectively.

<137>

[Claim 8]

<i38> The processing method of a block event in an on-line game according to claim 4,

<i39> wherein, in the explosion and attribute endowing step, in case the exploded blocks have different attributes, the different attributes are endowed to the character.

<i40>

[Claim 9]

<i41> A processing method of a block event in an on-line game, which occurs when a character moves a block to cause explosion between adjacent blocks,

<i42> wherein an attribute is endowed to the character who moves the block to induce explosion between blocks, and

<i43> wherein, in case the character whose attribute is maintained collides with another character, a process of determining superiority and inferiority of attributes of the characters and then giving a damage to an inferior character is executed.

<i44>

[Claim 10]

<i45> The processing method of a block event in an on-line game according to claim 9, wherein the process includes:

<i46> an attribute endowing step for endowing an attribute during n seconds to the character who moves the block to induce explosion between blocks;

<i47> an attribute retaining time expiration checking step for checking whether the attribute endowed to the character is maintained; and

<i48> a collision checking step for checking whether the attribute-endowed character collides with another character,

<i49> wherein, in case the collision between characters occurs, an attribute superiority determining process for determining superiority and inferiority of the attributes of the characters is executed.
[Claim 11]
The processing method of a block event in an on-line game according to claim 9, wherein the process includes:

- a damage calculating step for calculating a damage to be applied to the inferior character determined by the superiority and inferiority of attributes; and
- a damage applying step for applying the calculated damage to the inferior character.

[Claim 12]
The processing method of a block event in an on-line game according to claim 11,

wherein the damage calculating step adopts a time-based calculating manner, which counts a time from the point that the attribute is endowed in the attribute endowing step and then sets the damage to be weakened as the counted time is passing.

[Claim 13]
The processing method of a block event in an on-line game according to claim 11,

wherein, in case the attribute endowed in the attribute endowing step is complex, the damage calculating step adopts a set value-based calculating manner, which sets and calculates intensity of the damage according to the kind of attribute in advance.

[Claim 14]
The processing method of a block event in an on-line game according to claim 9, wherein the process includes:

- an attribute canceling step for canceling the endowed attributes of the characters when a damage is endowed between the collided characters.
[Claim 15]
The processing method of a block event in an on-line game according to claim 10, wherein the attribute superiority determining process includes:

- a remaining attribute checking step for checking whether a collided character has a remaining attribute;
- a character attribute input step for inputting the remaining attribute of the character to an attribute superiority determining unit in case the character has the remaining attribute;
- a superiority attribute checking step for determining superiority and inferiority of the input attributes of characters;
- a flag inputting and storing step for, in case the attribute of the character input earlier is superior to the attribute of the character input later, endowing an increase flag to the earlier input character; and
- a flag inputting and storing step for, in case the attribute of the character input earlier is inferior to the attribute of the character input later, endowing an increase flag to the later input character or endowing a decrease flag to the earlier input character.

[Claim 16]
The processing method of a block event in an on-line game according to claim 15,

wherein the increase or decrease flag stores information of the corresponding character and numerical information of increased or decreased value.

[Claim 17]
The processing method of a block event in an on-line game according to claim 10,

wherein the attribute superiority determining process has attribute kinds of water, fire and ice such that the attributes have superiority orders
of water > fire > ice > water.

[Claim 18]

A processing method of a block event in an on-line game, which occurs when a character moves a block to cause explosion between adjacent blocks, wherein the explosion between adjacent blocks generates an explosion object with a predetermined attribute and a predetermined life time, and wherein, in case a character collides with an explosion area formed around the explosion object, a process for determining superiority and inferiority of attributes of the explosion object and the character and then giving a damage to an inferior character is executed.

[Claim 19]

The processing method of a block event in an on-line game according to claim 18, wherein the process includes:

- an explosion object generating step for generating an explosion object with a predetermined attribute and a life time of n seconds when explosion between adjacent blocks occurs;
- an object life checking step for forming an explosion area around the explosion spot and checking whether the life time of n seconds passes; and
- a collision checking step for checking whether a character collides with the explosion area,

wherein, in case a character collides with the explosion area, an attribute superiority determining process for determining superiority and inferiority of attributes of the explosion area and the character and then giving a damage to an inferiority character is executed.

[Claim 20]

The processing method of a block event in an on-line game according to claim 18, wherein the process includes:

- a damage calculating step for calculating a damage to be applied to the
inferior character according to the superiority and inferiority of attributes; and

applying the calculated damage to the inferior character.

[Claim 21]

The processing method of a block event in an on-line game according to claim 20,

wherein the damage calculating step adopts a time-based calculating manner, which counts a life time of the explosion object and then sets the damage to be weakened as the counted time is passing.

[Claim 22]

The processing method of a block event in an on-line game according to claim 20,

wherein, in case the attribute of the explosion object is complex, the damage calculating step adopts a set value-based calculating manner, which sets and calculates intensity of the damage according to the kind of attribute in advance.

[Claim 23]

The processing method of a block event in an on-line game according to claim 19, wherein the attribute superiority determining process includes:

a remaining attribute checking step for checking whether the explosion object has a remaining attribute;

an attribute input step for inputting the attributes of the explosion object and the collided character to an attribute superiority determining unit in case the explosion object has the remaining attribute;

a superiority attribute checking step for determining superiority and inferiority of the input attributes of the explosion object and the character;

a flag inputting and storing step for, in case the attribute of the
explosion object input earlier is superior to the attribute of the character input later, endowing an increase flag to the earlier input explosion object; and

a flag inputting and storing step for, in case the attribute of the explosion object input earlier is inferior to the attribute of the character input later, endowing an increase flag to the later input character or endowing a decrease flag to the earlier input explosion object.

[Claim 24]

The processing method of a block event in an on-line game according to claim 23,

wherein the increase or decrease flag stores information of the corresponding character and numerical information of increased or decreased value.

[Claim 25]

The processing method of a block event in an on-line game according to claim 19,

wherein the attribute superiority determining process has attribute kinds of water, fire and ice such that the attributes have superiority orders of water > fire > ice > water.

[Claim 26]

A processing method of a combo event in an on-line game, which conducts successive movement of a block using a single event,

wherein, in case a combo event signal for successive movement of a block is input and the successive movement is available as a result of checking availability of the successive movement of the block according to the input signal, a process for finding a block for the movement to set a combo value thereto according to the input signal and then moving the block according to the combo value is executed.
[Claim 27]
The processing method of a combo event in an online game according to claim 26, wherein the process includes:

- a combo input receiving step for receiving the input of a combo event signal for successive movement of a block;
- a validity checking step for checking according to the input signal whether the block is available for the successive movement;
- a combo value setting step for setting a combo value to the block according to the input signal in case the block is available for the successive movement; and
- a block moving step for moving the block according to the combo value.

[Claim 28]
The processing method of a combo event in an online game according to claim 26, wherein the process includes:

- a stop condition checking step for checking whether a block stop condition is satisfied, in case a block moves in the block moving step; and
- a combo queue value consumption checking step for checking whether or not a combo value is consumed, using combo value information set for determination of a direction change, in case the block stop condition is satisfied.

[Claim 29]
The processing method of a combo event in an online game according to claim 27, wherein the validity checking step includes at least one of:

- checking whether a block exists in a direction faced by the character;
- checking whether a distance between the character and the block is suitable for combo input; and
- checking whether the block is capable of moving.
The processing method of a combo event in an on-line game according to claim 27, wherein, in the combo input receiving step, combo input information related to a combo list having information about a peculiar object ID (ObjectID) with 32 bit integer value and a moving direction of the block to be moved is received.

The processing method of a combo event in an on-line game according to claim 27, wherein, in the combo value setting step, a combo value for the block to be moved is set using a method called for moving the corresponding block by querying a peculiar object ID (ObjectID) to an object storage that includes information of blocks.

The processing method of a combo event in an on-line game according to claim 28, wherein, as the stop condition, the block stop condition checking step includes at least one of the following conditions:

1. In case a block reaches an outer area of NPC (Non Player Character: not accessible by a user character but accessible only by a system character);
2. In case a block collides (or, contacts) with another block; and
3. In case a block collides (or, contacts) with other obstacles.

The processing method of a combo event in an on-line game according to claim 28, wherein the combo queue value is obtained by storing information about
the number of combos input in the combo value setting step in a queue format, and

wherein, in the combo queue value consumption checking step, in case a queue value remains in a FIFO (First-In First-Out) structure (a state value: Q>0), a combo direction changing step is executed to change a direction of the block, and then the block moving step is executed.

[Claim 34]

A processing method of a combo event in an on-line game, which conducts successive movement of a block using a single event, the processing method comprising:

- a start signal inputting step for inputting a combo event start signal for the purpose of successive movement of a block;
- a combo input availability checking step for checking whether the input of a combo event signal for successive movement of the block is available;
- a combo inputting step for inputting the combo event signal; and
- a validity checking step for checking whether the block is capable of moving according to the input combo event signal.

[Claim 35]

The processing method of a combo event in an on-line game according to claim 34,

wherein the combo input availability checking step checks whether a stopped block exists within a predetermined distance in a direction faced by the stopped character, and

wherein the validity checking step checks whether a movable block exists and whether there exists any direction in which the block is movable in an input direction list.

[Claim 36]

A system in which a server and at least one client are connected
through a communication network,

wherein the processing method of a dynamic event, defined in any one of the claims 1, 9, 18 and 26, is executed in the server.

[Claim 37]

A system in which a server and at least one client are connected through a communication network,

wherein the processing method of a dynamic event, defined in the claim 34, is executed in the client.
<server side>
Figure 2

<server side>

B0 → START

B1 → ENDOW ATTRIBUTE TO CHARACTER FOR N SECONDS

B2 → ATTRIBUTE RETAINING TIME EXPIRATION CHECKING

F →

T → B3

No → COLLISION CHECKING

Yes → EXECUTE ATTRIBUTE SUPERIORITY DETERMINING PROCESS

BP → DAMAGE CALCULATION

B4 → APPLY DAMAGE

B5 → CANCEL ATTRIBUTE

B6 → CANCEL ATTRIBUTE

B7 → END
[Figure 3]  

<server side>  

START \(\rightarrow\) C0  

GENERATE EXPLOSION OBJECT WITH LIFE TIME OF N SECOND  

OBJECT LIFE TIME CHECKING  

C2  

\(T\) \(\rightarrow\) C3  

COLLISION CHECKING  

No \(\rightarrow\) \(F\) \(\rightarrow\) C1  

Yes  

EXECUTE ATTRIBUTE SUPERIORITY DETERMINING PROCESS  

\(\rightarrow\) CP  

DAMAGE CALCULATION  

\(\rightarrow\) C4  

APPLY DAMAGE  

\(\rightarrow\) C5  

ATTRIBUTE EXPLOSION OBJECT LIFE TIME EXPIRATION  

\(\rightarrow\) C6  

END  

\(\rightarrow\) C7  

<267>
<client side> <server side>

START \( S_0 \)

COMBO INPUT AVAILABILITY CHECKING

\( S_1 \)

No \( \rightarrow \)

Yes \( \rightarrow \)

COLLECT N COMBO INPUTS \( S_2 \)

VALIDITY CHECKING \( S_3 \)

No \( \rightarrow \)

Yes \( \rightarrow \)

RECEIVE COMBO INPUT \( S_4 \)

VALIDITY CHECKING \( S_5 \)

No \( \rightarrow \)

Yes \( \rightarrow \)

FIND CORRESPONDING BLOCK AND SET COMBO VALUE \( S_6 \)

BLOCK MOVEMENT \( S_7 \)

BLOCK STOP CONDITION CHECKING \( S_8 \)

T \( \rightarrow \)

CHANGE COMBO DIRECTION \( S_10 \)

Q > 0 \( \rightarrow \)

COMBO QUEUE VALUE CONSUMPTION CHECKING \( S_9 \)

Q ≤ 0 \( \rightarrow \)

BLOCK STOP \( S_11 \)

END \( S_{12} \)
<server side>

START \(\rightarrow\) AP0

STOP BLOCK ATTRIBUTE CHECKING \(\rightarrow\) AP1

AP2

ADJACENT BLOCK ATTRIBUTE SAMENESS CHECKING

Yes \(\rightarrow\) AP3

ADJACENT BLOCK EXPLOSION

No \(\rightarrow\) AP4

EXPLOSION BLOCK ATTRIBUTE SAMENESS CHECKING

Yes \(\rightarrow\) AP5

ADJACENT BLOCK EXPLOSION

No \(\rightarrow\) AP6

ENDOW ATTRIBUTE TO CHARACTER

END \(\rightarrow\) AP7
<server side>

[Figure 6]

START → BP0

<table>
<thead>
<tr>
<th>No</th>
<th>REMAINING ATTRIBUTE CHECKING</th>
</tr>
</thead>
</table>
| Yes| CHARACTER ATTRIBUTE INPUTTING → BP5

<table>
<thead>
<tr>
<th>Yes</th>
<th>INPUT DECREASE FLAG</th>
</tr>
</thead>
<tbody>
<tr>
<td>T</td>
<td>INPUT INCREASE FLAG</td>
</tr>
</tbody>
</table>

END → BP6

<270>
<server side>

Start (CPO)

No

Remaining Attribute Checking (CP1)

Yes

Block Attribute Inputting (CP2)

Superior Attribute Checking (CP3)

F

Input Decrease Flag (CP5)

T

Input Increase Flag (CP4)

End (CP6)

<271>

[Figure 8]
[Figure 12]

[Figure 13]

COMBINATION OF PARTS
[Figure 16]

Input a direction key while pressing a spacebar.

Combo is activated at the instant of releasing the spacebar.
(Combo may be input up to 5 times, and the character keeps moving in a facing direction if combo is not input.)

Combo may be input using a direction key while pressing a spacebar.