

DEVICE AND METHOD FOR IMPLEMENTING A SCENE IN A SIMULATED ENVIRONMENT

TECHNICAL FIELD

[0001] Various aspects of this disclosure relate to devices and methods for implementing scenes in simulated environments.

BACKGROUND

[0002] In many applications users may create scenes in virtual 3D environments, e.g. scenes for a computer game or for virtual environments for testing, e.g. of a control policy for automated driving. The creation of a scene in a 3D environment typically involves defining the behaviour of a multiplicity of objects where multiple parameters need to be set.

[0003] A game engine like Unity, for example, allows a user to program scenes, e.g. game plot scenes, through a linear structure based on time series, called Timeline in case of Unity.

[0004] However, when using a linear structure based on time series like Unity Timeline to edit (in particular plot) scenes, this requires using complex functions and performing many repeated operations, which leads to low efficiency of the task.

[0005] Therefore, more efficient approaches for implementing a scene (including a sequence of events) in a simulated environment are desirable.

SUMMARY

[0006] Various embodiments concern a method for implementing a scene in a simulated environment, including generating, in accordance with user input, a graph representing a scene in a simulated environment, wherein the graph includes a plurality of nodes connected by directed connections specifying an order of the nodes, wherein the nodes include, for each simulated entity that acts in the scene, a node associated with the simulated entity which specifies an action of the simulated entity including animation of the simulated entity and sound output of the simulated entity and at least one camera perspective from which the simulated entity is to be shown while performing the action, converting the graph into a time-series based linear structure representing the scene according to the specified order of the nodes, wherein the time-series based linear structure includes a plurality of tracks including at

least one animation track for the simulated entity and at least one sound track and implementing the scene using the time-series based linear structure.

[0007] According to one embodiment, for at least one node, the simulated entity is a character, the sound output includes the speech of the simulated entity and the time-series linear structure includes a lip synchronisation control track for the simulated entity.

[0008] According to one embodiment, the at least one node includes a dialogue text string of the character.

[0009] According to one embodiment, the at least one node includes an identification of the character.

[0010] According to one embodiment, the at least one node includes an identification of the character.

[0011] According to one embodiment, the nodes include at least one background node specifying a background to be set at a time in the scene according to the position of the background node in the order of nodes.

[0012] According to one embodiment, the node specifies the action by reference to a group of animation tracks for the simulated entity.

[0013] According to one embodiment, the simulated entity is a character and the group of animation tracks includes a character body animation track, a character face animation track and a character lips animation track.

[0014] According to one embodiment, the time-series linear structure further includes a user interface control track.

[0015] According to one embodiment, the user input includes a configuration text describing the scene.

[0016] According to one embodiment, the method includes receiving the user input via a graphical user interface including a visual representation of the graph where nodes are represented as blocks showing information associated information and the connections are shown as lines connecting the blocks.

[0017] According to one embodiment, the method includes receiving the user input via menus which are activated when the user clicks the blocks.

[0018] According to one embodiment, the method includes implementing the scene using the time-series based linear structure using a simulation engine which accepts time-series based linear structures for implementing scenes in the simulated environment.

[0019] According to one embodiment, the simulated environment is a game world of a computer game and the simulation engine is a game engine.

[0020] According to one embodiment, the time-series linear structure is a time-series linear structure according to Unity Timeline.

[0021] According to one embodiment, a computing device is provided including a communication interface, a memory and a processing unit configured to perform the method of any one of the above embodiments.

[0022] According to one embodiment, a computer program element is provided including program instructions, which, when executed by one or more processors, cause the one or more processors to perform the method of any one of the above embodiments.

[0023] According to one embodiment, a computer-readable medium is provided including program instructions, which, when executed by one or more processors, cause the one or more processors to perform the method of any one of the above embodiments.

[0024] It should be noted that embodiments described in context of the method ... are analogously valid for the terminal device.

BRIEF DESCRIPTION OF THE DRAWINGS

[0025] The invention will be better understood with reference to the detailed description when considered in conjunction with the non-limiting examples and the accompanying drawings, in which:

- FIG. 1 shows a data processing system.
- FIG. 2 shows a visual editing window for a character emotion group according to an embodiment.
- FIG. 3 shows a preview window for a character model corresponding to the linear structure of the character emotion group.
- FIG. 4 shows a role emotion group linear structure serialization data browsing window according to an embodiment.
- FIG. 5 shows a selection window for plot configuration test asset to graph conversion according to an embodiment.
- FIG. 6 shows an example of a text configuration table.
- FIG. 7 shows a visual scene editing window based on a graph data structure according to an embodiment.

- FIG. 8 shows a visual scene editing window including, in this example, background editing, based on a graph data structure, according to an embodiment.
- FIG. 9 illustrates time-based linear structure data.
- FIG. 10 shows a user interface of a real-time playback module for time-based linear structure data.
- FIG. 11 shows a flow diagram illustrating a method for implementing a scene in a simulated environment according to an embodiment.

DETAILED DESCRIPTION

[0026] The following detailed description refers to the accompanying drawings that show, by way of illustration, specific details and embodiments in which the disclosure may be practiced. These embodiments are described in sufficient detail to enable those skilled in the art to practice the disclosure. Other embodiments may be utilized and structural, and logical changes may be made without departing from the scope of the disclosure. The various embodiments are not necessarily mutually exclusive, as some embodiments can be combined with one or more other embodiments to form new embodiments.

[0027] Embodiments described in the context of one of the devices or methods are analogously valid for the other devices or methods. Similarly, embodiments described in the context of a device are analogously valid for a vehicle or a method, and vice-versa.

[0028] Features that are described in the context of an embodiment may correspondingly be applicable to the same or similar features in the other embodiments. Features that are described in the context of an embodiment may correspondingly be applicable to the other embodiments, even if not explicitly described in these other embodiments. Furthermore, additions and/or combinations and/or alternatives as described for a feature in the context of an embodiment may correspondingly be applicable to the same or similar feature in the other embodiments.

[0029] In the context of various embodiments, the articles “a”, “an” and “the” as used with regard to a feature or element include a reference to one or more of the features or elements.

[0030] As used herein, the term “and/or” includes any and all combinations of one or more of the associated listed items.

[0031] In the following, embodiments will be described in detail.

[0032] FIG. 1 shows a data processing device 100 like a computer or smartphone.

[0033] The data processing device 100 has a CPU (central processing unit) 101 and a random access memory (RAM) 102. The RAM 102 is used to load program code, for example from a hard disk 103, and the CPU 101 executes the program code. The data processing device 100 further includes a screen 113.

[0034] In the present example, it is assumed that a user intends to use the data processing device 100 to create a scene for a simulation, like for example a plot scene for a computer game.

[0035] To do so, the user executes a development environment 104 on the CPU 101.

[0036] The development environment 104 enables the user to create a definition of a scene 105. The scene may be for a simulation program (e.g. a computer game) intended for various devices 106, i.e., target hardware. The user of the data processing device 100 may also create a movie clip from the scene 105 which may be distributed to the devices 106. For example, the devices 106 are handheld devices, such as smartphones or tablets or dedicated handheld gaming devices.

[0037] The user of the data processing device 100 may distribute the scene 105 (e.g. in the form of a movie clip, the whole simulation program, e.g. game, or content for the simulation program) to appropriate devices 106 via a communication network 107. For example, the user uploads the created scene or movie clip or the whole simulation program to a server 108 (connected to the communication network 107) from which users of the other devices 106 may download it.

[0038] According to various embodiments, to allow the user of the data processing device 100 to efficiently define a scene (such that the definition can be interpreted by the simulation program, i.e. be converted into a scene in a simulated 3D (or possibly also 2D) environment, the development environment 105 includes a visual graph node management tool 109 which allows managing scene (e.g. plot) elements like characters and other objects that perform actions in the scene by visually representing them in the form of a graph on the screen 113. This tool is easier to use for the user than a scene definition tool requiring interacting with a linear structure based on time series like Unity Timeline. So, a developer (e.g. the user of the data processing device 101) can quickly learn and make scenes. The development environment 105 includes a transformation function 110 implementing an algorithm to convert a scene representation in form of a graph created by the graph node management tool

109 into a linear structure based on time series for a tool 111 which uses such a time-series based linear structure for scene definition, like Unity Timeline and creates the scene 105. So, the user can efficiently define a scene using a visual graph representation of the scene and create a suitable representation (i.e. the time-series based linear structure) for a simulation software (e.g. Unity) from it. The visual graph node management tool 109 and the transformation function 110 may thus for example be used on top of a simulation software (e.g. game engine) like Unity. So, the visualization tool 109 is for example implemented as a visualization UI extension in the Unity engine to allow easy node asset management based on graphs. The transformation function 110 in this example implements a conversion algorithm to convert plot assets based on graph nodes into plot assets based on time series that Unity can recognize and run.

[0039] By avoiding the need to directly operate on the time-series based linear structure time can be saved and the number of scenes (e.g. story or plot scenes in a game) created by the user (or multiple users using multiple data processing devices 100 in this manner) may be increased.

[0040] The development environment 104 may further include a tool for real-time preview of plot assets 112. This tool may operate on the time-series based linear structure (created by the transformation function 110, i.e. conversion tool). The tool for real-time preview may implement an algorithm that can load and instantiate other (e.g. edited versions of) assets and implement a mapping relationship (e.g. to original assets) to generate a preview.

[0041] According to various embodiments, the visual graph node management tool 109 includes the following modules:

1. A role emotion editing module providing:
 - 1.1 A role emotion group linear structure visual editing window
 - 1.2 A preview window of role model corresponding to linear structure of role emotion group
 - 1.3 A role emotion group linear structure serialization data management window
2. A scene configuration conversion module for converting text assets into graph data structure assets
3. A (e.g. plot) scene graph data structure visualization module providing:
 - 3.1 A visual plot editing window based on graph data structure

3.2 A browse window of serialized file list based on graph data structure

4. A dialogue background editing module providing:

- 4.1. A dialogue background asset management window
- 4.2. A dialogue background camera parameter editing module
- 4.3. A dialogue background role position editing module

[0042] The transformation function 110 may be provided by a conversion module to convert a scene graph data structure assets into time-based linear structure data. The time-based linear structure data may be used by a the real-time playback module 112 for time-based linear structure data which loads of simulation (e.g. game) resources required for real-time playback, including for example characters, special effects, scenes, and virtual locations and provides a real-time playback function.

[0043] In the following, the modules mentioned above are described in more detail and how a user may use them.

[0044] The role emotion editing module is mainly to be used for asset management of character scene (e.g. plot) actions based on a linear structure, including finding, adding and modifying.

[0045] Various embodiments use the emotion group as a basic concept. It is a combination of multiple action assets of a role arranged in time series. A typical emotion group may be composed of a character's body movement data, the character's mouth movements, the character's micro expressions, such as smiling, blinking and other actions.

[0046] FIG. 2 shows the visual editing window 200 (displayed, e.g., on screen 113, like all the windows described in the following) of the emotion group (i.e. a role emotion group linear structure asset visualization view window), which provides the addition and deletion of the emotion group track, the addition and deletion of the emotion animation clip, and the transition functions of the two emotion group animation clips. After its modification, it is saved as a TimelineAsset to harddisk 103 through a serialization function.

[0047] FIG. 3 shows a preview window for a character model corresponding to the linear structure of the character emotion group.

[0048] A scenario asset designer can browse the completed emotion group assets through the preview window 300 of the character model corresponding to the linear structure of the respective character emotion group. This window 300 provides the model rendering of the

character to be previewed, reads the serialized emotion group assets, and plays back the animation clip in the emotion group according to the occurrence sequence of time.

[0049] FIG. 4 shows a role emotion group linear structure serialization data browsing window 400.

[0050] It is a visual management window for managing serialized emotion group assets. This window 400 contains the following functions:

1. Add a new emotion group configuration object and serialize it to harddisk 103. It includes specifying the role ID of the new emotion group, specifying the alias, generating the unique ID for identifying the asset, and the asset index of the emotion group timeline asset (e.g. Unity's TimelineAsset).
2. Emotion group preview button. By clicking this button, the user can open the preview window 300 of the role model corresponding to the emotion group in FIG. 3.
3. Emotion group detail editing function. When the user wants to change the name of the emotion group, a new name can be specified through the rename function. With the option to disable automatic blinking, the user can add timeline clip information (e.g. Unity's TimelineClip) information that disables blinking to the subsequent time-based linear structure assets.

[0051] The linear structure of the role emotion group corresponds to the preview window 300 of the role model, as shown in FIG. 3. Through this window, the user can preview the action when the emotion group is applied to the character. The user may click an action playback button 301 to repeatedly view the results of the data of the emotion group acting on the role.

[0052] In addition, the user can specify the camera position and camera rotation parameters to adjust the position of the preview character in the window 300.

[0053] The scene configuration conversion module serves for converting text assets into graph data structure assets.

[0054] FIG. 5 shows a selection window 500 for plot configuration test asset to graph conversion.

[0055] Through the window 500, the user can convert the plot configuration text assets into graph data structure assets.

[0056] FIG. 6 shows an example of a text configuration table 600.

[0057] The result of the conversion (i.e. graph) is saved to hard disk 103, e.g. in a file format that can be recognized by Unity. The asset can be subsequently edited and modified through a visual editing window as illustrated in figures 7 and 8.

[0058] FIG. 7 shows a visual scene editing window 700 based on a graph data structure.

[0059] FIG. 8 shows a visual scene editing window 800 including, in this example, background editing, based on a graph data structure.

[0060] The visual scene editing windows 700, 800 are provided by the scene graph data structure visualization module of the visual graph node management tool 109.

[0061] The scene graph data structure visualization module allows the user to configure of the basic elements of the scene through the visual scene editing window 700, 800.

[0062] The windows include character information involved in the scene (e.g. plot), the ID of each character participating in the scene and the initial location information of the character as well as environmental information required for the scene, for example a 2D background board, and dynamic elements of the environment.

[0063] Further, scene graph data structure visualization module allows the user to configure

[0064] This module realizes the configuration of scene flow through the visual editing window by providing the following functions:

1. Editing dialogue nodes. Editing functions include

- 1.1 Creation of new dialogue nodes. The user can create a new dialogue node by (e.g.) right-clicking a pop-up menu button and selecting “dialogue”.

- 1.2 Definition of node connections: by connecting dialogue nodes, the user can define the sequence relationship of dialogue events in a time series (which is visually expressed in the visual scene editing window 700, 800 by corresponding (directed) connections 703, 704 between the dialogue nodes 702);

- 1.3 Setting of node parameters.

2. Editing background nodes. Editing functions include:

- 2.1 Creation of new background nodes. The user can create a new background node by right-clicking the pop-up menu 701 and selecting “background”.

2.2 Modifying a background node 801. The user may click the background name button to pop up a menu and select the background to which the user wants to switch.

2.3 Definition of node connections. The user may connect a background node 801 to another node (in particular a dialogue node 802). For example, like illustrated in FIG. 8, the user may connect a first background node to the dialog node via a first (directed) connection 803, and then connect the dialog node 801 to a second background via a second (directed) connection 804 to define a scene where a first background (corresponding to the first background node) is displayed, the character's emotion group data is played (and e.g. corresponding text lines are displayed), and then a second background (corresponding to the second background node) is displayed.

[0065] The scene configuration conversion module implements the transformation function 110. It converts scene graph data structure assets generated by the user via the visual editing window 700, 800 into time-based linear structure data. To trigger the conversion of the graph data structure displayed in the visual editing window 700, 800, the user may click a generate timeline button 705, 805.

[0066] FIG. 9 shows the time-based linear structure data as it is displayed by a time-based visual editing tool for linear data series (e.g. Unity's Timeline).

[0067] The real-time playback module 112 for time-based linear structure data allows data loading and display, including the corresponding role model and the background resource corresponding to the specified background ID.

[0068] FIG. 10 shows the user interface of the real-time playback (i.e. visualization) module (or tool) 112 for time-based linear structure data. It thus allows the user to preview the results of the time-based linear structure data in real time.

[0069] In summary, according to various embodiments, a method is provided as illustrated in FIG. 11.

[0070] FIG. 11 shows a flow diagram 1100 illustrating a method for implementing a scene in a simulated environment.

[0071] In 1101, a graph representing a scene in a simulated environment is generated in accordance with user input.

[0072] The graph includes a plurality of nodes connected by directed connections specifying an order of the nodes, wherein the nodes include, for each simulated entity that acts in the scene, a node associated with the simulated entity which specifies an action of the simulated entity including animation of the simulated entity and sound output of the simulated entity and at least one camera perspective from which the simulated entity is to be shown while performing the action.

[0073] In 1102, the graph is converted into a time-series based linear structure representing the scene according to the specified order of the nodes, wherein the time-series based linear structure includes a plurality of tracks including at least one animation track for the simulated entity and at least one sound track.

[0074] In 1103, the scene is implemented using (i.e. based on, i.e. by processing) the time-series based linear structure.

[0075] According to various embodiments, in other words, a scene is defined by a user based on a graph structure which is then converted to a time-series based linear structure for further processing by the respective simulation software (wherein this may involve a further conversion process from the time-series based linear structure to a format that can be understood by the simulation software, e.g. a game engine). So, according to various embodiments the structure (in particular the temporal structure of a scene), e.g. a plot structure based is managed based on a graph and converted it into a scene (e.g. plot) asset based on time series through a generation (i.e. conversion) algorithm.

[0076] The approach of FIG. 11 may be used to define scenes for various simulated environments, e.g. environments for testing control policies (e.g. traffic environments including a certain behaviour of vehicles or pedestrians for testing autonomous driving), or other environments for testing any technical devices including machines etc. and computer games.

[0077] For example, according to various embodiments, the approach of FIG. 11 is applied to game development based on the Unity engine. This means that a method for generating and managing game scenario assets that can be used in Unity may be provided which includes providing a visual editing UI system of nodes based on a graph data structure. Through the visual operation based on the graph data structure (management operations of adding different types of nodes, connections between nodes, and node deletion), the text, role

actions, special effects, background and environment of the game plot can be efficiently managed by a user (game developer in this example).

[0078] Further, a generation mechanism may be provided through which the plot assets based on graph data structure can be transformed into time-based linear plot assets according to Unity Timeline. In addition, a mechanism for real-time preview of time-based linear plot assets may be provided which allows users to preview results of editing a scene in a separate environment in real-time. This allows providing a development environment which, compared to using a timeline development process directly using Unity Timeline, is less difficult to learn for the user, reduces repeated mechanical work of users and thus allows improving development efficiency while maintaining or even increasing quality.

[0079] The method of FIG. 11 is for example carried out by a computing device, e.g. corresponding to the data processing device 100 of FIG. 1. The computing device may include a communication interface (e.g. configured to communicate with a server distributing the created content). The computing device further includes a processing unit and a memory. The memory may be used by the processing unit to store, for example, program data for the various development tool and data representing the created content.

[0080] The methods described herein may be performed and the various processing or computation units and the devices and computing entities described herein may be implemented by one or more circuits. In an embodiment, a "circuit" may be understood as any kind of a logic implementing entity, which may be hardware, software, firmware, or any combination thereof. Thus, in an embodiment, a "circuit" may be a hard-wired logic circuit or a programmable logic circuit such as a programmable processor, e.g. a microprocessor. A "circuit" may also be software being implemented or executed by a processor, e.g. any kind of computer program, e.g. a computer program using a virtual machine code. Any other kind of implementation of the respective functions which are described herein may also be understood as a "circuit" in accordance with an alternative embodiment.

[0081] While the disclosure has been particularly shown and described with reference to specific embodiments, it should be understood by those skilled in the art that various changes in form and detail may be made therein without departing from the spirit and scope of the invention as defined by the appended claims. The scope of the invention is thus indicated by the appended claims and all changes which come within the meaning and range of equivalency of the claims are therefore intended to be embraced.

CLAIMS

1. A method for implementing a scene in a simulated environment, comprising:
Generating, in accordance with user input, a graph representing a scene in a simulated environment,
wherein the graph includes a plurality of nodes connected by directed connections specifying a order of the nodes;
wherein the nodes comprise, for each simulated entity that acts in the scene, a node associated with the simulated entity which specifies an action of the simulated entity including animation of the simulated entity and sound output of the simulated entity and at least one camera perspective from which the simulated entity is to be shown while performing the action;
converting the graph into a time-series based linear structure representing the scene according to the specified order of the nodes, wherein the time-series based linear structure comprises a plurality of tracks including at least one animation track for the simulated entity and at least one sound track; and
implementing the scene using the time-series based linear structure.
2. The method of claim 1, wherein, for at least one node, the simulated entity is a character, the sound output includes speech of the simulated entity and the time-series linear structure comprises a lip synchronisation control track for the simulated entity.
3. The method of claim 2, wherein the at least one node includes a dialogue text string of the character.
4. The method of claim 2 or 3, wherein the at least one node includes an identification of the character.
5. The method of any one of claims 2 to 4, wherein the at least one node includes an identification of the character.

6. The method of any one of claims 1 to 5, wherein the nodes include at least one background node specifying a background to be set at a time in the scene according to the position of the background node in the order of nodes.
7. The method of any one of claims 1 to 6, wherein the node specifies the action by reference to a group of animation tracks for the simulated entity.
8. The method of claim 7, wherein the simulated entity is a character and the group of animation tracks comprises a character body animation track, a character face animation track and a character lips animation track.
9. The method of any one of claims 1 to 8, wherein the time-series linear structure further comprises a user interface control track.
10. The method of any one of claims 1 to 9, wherein the user input includes a configuration text describing the scene.
11. The method of any one of claims 1 to 10, comprising receiving the user input via a graphical user interface comprising a visual representation of the graph where nodes are represented as blocks showing information associated information and the connections are shown as lines connecting the blocks.
12. The method of claim 11, comprising receiving the user input via menus which are activated when the user clicks the blocks.
13. The method of any one of claims 1 to 12, comprising implementing the scene using the time-series based linear structure using a simulation engine which accepts time-series based linear structures for implementing scenes in the simulated environment.
14. The method of claim 13, wherein the simulated environment is a game world of a computer game and the simulation engine is a game engine.

15. The method of any one of claims 1 to 14, wherein the time-series linear structure is a time-series linear structure according to Unity Timeline.
16. A computing device comprising a communication interface, a memory and a processing unit configured to perform the method of any one of claims 1 to 15.
17. A computer program element comprising program instructions, which, when executed by one or more processors, cause the one or more processors to perform the method of any one of claims 1 to 15.
18. A computer-readable medium comprising program instructions, which, when executed by one or more processors, cause the one or more processors to perform the method of any one of claims 1 to 15.