

METHOD OF USING SCRIPTABLE OBJECTS TO INSERT AUDIO FEATURES INTO A PROGRAM

FIELD OF THE INVENTION

[0001] The invention pertains to a method of using scriptable objects to insert audio features into a program.

BACKGROUND

[0002] Audio middleware allows software developers to attach audio files into programs like a game environment. Audio middleware provides Cue Sheets and Cues for the software developers to use in their programs. Examples of audio middleware non-exhaustively include CRIWARE, Wwise and FMod.

[0003] Cues describe the triggers for the audio files. For example, Cue 1 can describe the following: “an attack action will trigger audio file A”. In another example, Cue 2 can describe the following: “a reload action will trigger audio file B”. A Cue Sheet is a collection of Cues and their corresponding audio files.

[0004] By referencing the Cue Sheets and Cues in the program code, software developers can incorporate the audio files into their programs.

[0005] However, the inefficiency is that each time the audio files are to be played, software developers will need to write corresponding lines of code at specific portions of the program code to explicitly code in the Cue Sheets and the Cues. This inefficiency is compounded if there are many instances where the Cue Sheets and Cues need to be coded in. This also results in a tight coupling between the loading logic of the Cue Sheets and the game logic, thereby restricting the flexibility of the resource management of the Cue Sheets.

[0006] What is required is a novel solution which allows Cue Sheets and their Cues to be incorporated into software programs in a more efficient manner. Furthermore, other desirable features and characteristics will become apparent from the subsequent detailed description and the appended claims, taken in conjunction with the accompanying drawings and this background of the disclosure.

SUMMARY OF INVENTION

[0007] According to an aspect of the invention, a method of using scriptable objects to insert audio features into a program is disclosed, the method comprising the steps of: obtaining a plurality of Cue Sheets from an audio middleware, each of the plurality of Cue Sheets comprising a Cue Sheet Name, a plurality of Cues and a plurality of audio files, and each of the plurality of Cues comprising a Cue Name and a description, the description illustrating a relationship between a unique action and at least one of the plurality of audio files; and scanning the plurality of Cue Sheets and the plurality of Cues to identify the plurality of Cue Sheets based on the Cue Sheet Name and the plurality of Cues based on the Cue Name.

[0008] The method further comprises the steps of traversing through each of the plurality of Cue Sheets to select each of the plurality of Cues; constructing a scriptable object based on each traversed Cue Sheet and each selected Cue, to form an association between the scriptable object and the traversed Cue Sheet and the selected Cue; and inserting the scriptable objects into the program code of the program.

[0009] Such that when the program is executed and when an action occurs, a corresponding scriptable object out of the scriptable objects will be referenced as a result of matching the action with the unique action as illustrated by the description of the associated Cue of the corresponding scriptable object; and a sound effect manager will be called using the corresponding scriptable object as a parameter which will result in the sound effect manager calling the Application Programming Interface (API) of the audio middleware to play the audio features of the at least one of the plurality of audio files as illustrated by the description of the associated Cue of the corresponding scriptable object.

[0010] Preferably, the audio middleware is any one of the following: CRIWARE, Wwise and FMod.

[0011] Preferably, the sound effect manager belongs to a Unity engine.

[0012] Preferably, the program is a computer game.

[0013] According to another aspect of the invention, a method of using scriptable objects to play audio features in a computer game is disclosed, the method comprising the steps of loading a plurality of Cue Sheets obtained from an audio middleware into memory, each of the plurality

of Cue Sheets comprising a Cue Sheet Name, a plurality of Cues and a plurality of audio files, and each of the plurality of Cues comprising a Cue Name and a description, the description illustrating a relationship between a unique action and at least one of the plurality of audio files; and loading a plurality of scriptable objects, each of the plurality of scriptable objects having an association with at least one of the plurality of Cue Sheets and at least one of the plurality of Cues.

[0014] The method further comprises the steps of, in response to an action occurring during the computer game, referencing a corresponding scriptable object out of the plurality of scriptable objects as a result of matching the action with the unique action as illustrated by the description of the associated Cue of the corresponding scriptable object; and using the corresponding scriptable object as a parameter to call a sound effect manager, which results in the sound effect manager calling the Application Programming Interface (API) of an audio middleware to play the audio features of the at least one of the plurality of audio files as illustrated by the description of the associated Cue of the corresponding scriptable object.

[0015] Preferably, the audio middleware is any one of the following: CRIWARE, Wwise and FMod.

[0016] Preferably, the sound effect manager belongs to a Unity engine.

[0017] Preferably, a system comprising one or more computers and one or more storage devices storing computer-readable instructions that, when executed by the one or more computers, cause the one or more computers to perform the method.

[0018] Preferably, one or more non-transitory computer storage media storing instructions that, when executed by one or more computers, cause the one or more computers to perform the method.

[0019] Other aspects of the invention will become clearer from the following detailed description of some preferred embodiments thereof.

BRIEF DESCRIPTION OF THE DRAWINGS

[0020] The accompanying figures, where like reference numerals refer to identical or functionally similar elements throughout the separate views and which together with the detailed description below are incorporated in and form part of the specification, serve to

illustrate various embodiments, by way of example only, and to explain various principles and advantages in accordance with a present embodiment.

[0021] Figure 1 depicts a Cue Sheet comprising a plurality of Cues.

[0022] Figure 2 depicts the data structure of an Audio Reference object, in accordance with certain embodiments.

[0023] Figures 3(a) and 3(b) depicts exemplary instantiations of the Audio Reference object, in accordance with certain embodiments.

[0024] Figure 4 depicts exemplary pseudocode which can be used by a software developer to insert Audio Reference objects into the program code of the program.

[0025] Figure 5 depicts a method of using scriptable objects to insert audio features into a program, in accordance with certain embodiments.

[0026] Figure 6 depicts a method of using scriptable objects to play audio features in a computer game, in accordance with certain embodiments.

[0027] Skilled artisans will appreciate that elements in the figures are illustrated for simplicity and clarity and have not necessarily been depicted to scale. For example, the dimensions of some of the elements in the block diagrams or steps in the flowcharts may be exaggerated relative to other elements to help improve understanding of the present embodiment.

DETAILED DESCRIPTION

[0028] The following detailed description is merely exemplary in nature and is not intended to limit the invention or the application and uses of the invention. Furthermore, there is no intention to be bound by any theory presented in the preceding background of the invention or the following detailed description.

[0029] It is the intent of certain embodiments to teach a method of using scriptable objects to insert audio features into a program, the method comprising the steps of obtaining a plurality of Cue Sheets from an audio middleware, each of the plurality of Cue Sheets comprising a Cue Sheet Name, a plurality of Cues and a plurality of audio files, and each of the plurality of Cues comprising a Cue Name and a description, the description illustrating a relationship between a unique action and at least one of the audio files of the Cue Sheet; and scanning the plurality of

Cue Sheets and the plurality of Cues to identify the plurality of Cue Sheets based on the Cue Sheet Name and the plurality of Cues based on the Cue Name.

[0030] The method further comprises the steps of traversing through each one of the plurality of Cue Sheets to select each one of the plurality of Cues; and constructing a scriptable object based on each traversed Cue Sheet and each selected Cue, to form an association between the scriptable object and the traversed Cue Sheet and the selected Cue.

[0031] The method further comprises the step of inserting the scriptable objects into the program code of the program, such that when the program is executed and when an action occurs, a corresponding scriptable object out of the scriptable objects will be referenced as a result of matching the action with the unique action as illustrated by the description of the associated Cue of the corresponding scriptable object; and a sound effect manager will be called using the corresponding scriptable object as a parameter which will result in the sound effect manager calling the Application Programming Interface (API) of the audio middleware to play the audio features of the audio file as illustrated by the description of the associated Cue of the corresponding scriptable object.

[0032] The method as described is advantageous as the scriptable objects function as “data containers” to house each Cue Sheet and Cue pair, and software developers can simply insert the scriptable objects into the program code, instead of writing multiple lines of code to code in the Cue Sheets and Cues, while maintaining access to the functionality of the Cues and the audio files in the Cue Sheets. This greatly simplifies the task for software developers. As one can imagine, there would be numerous scenarios and portions of the program code where the Cue Sheets and Cues would be required. Therefore, by using the scriptable objects to substitute multiple lines of code while retaining access to the Cue Sheets and Cues, this greatly increases the efficiency for the software developers during the coding process.

[0033] Furthermore, the use of scriptable objects to “represent” each Cue Sheet and Cue pair advantageously decouples the loading logic of the Cue Sheets and the Cues from the game logic.

[0034] Figure 1 depicts an exemplary Cue Sheet 110 that can be obtained from an audio middleware. The audio middleware can be for example: CRIWARE, Wwise and FMod. Cue Sheet 110 can have a Cue Sheet Name. As shown in Figure 1, the Cue Sheet Name of Cue

Sheet 110 is “Player”. Cue Sheet 110 comprises a plurality of Cues 120. Cue Sheet 110 also comprises a plurality of audio files.

[0035] Each Cue 120 has a Cue Name. As shown in Figure 1, Cue Sheet 110 comprises eleven Cues 120 and their Cue Names are respectively : “SFX_Player_Attack”, “SFX_Player_Skill”, “SFX_Player_Ultimate”, “SFX_Player_Reload”, “SFX_Player_Attack03”, “SFX_Player_Attack04”, “SFX_Player_Hit”, “SFX_Player_Death”, “SFX_Player_skill01”, “SFX_Player_Passive” and “SFX_Player_ultimate_start”.

[0036] Each Cue 120 also has a description. This description illustrates a relationship between an action and one or more of the audio files of Cue Sheet 110. For example, Cue Sheet 110 can comprise the audio files: Audio File A (which plays the audio feature “da-da-da”) and Audio File B (which plays the audio feature “tik”). Cue 120 having Cue Name “SFX_Player_Attack” can have a description that illustrates the relationship between an “attack” action and Audio File A. Cue 120 having Cue Name “SFX_Player_Reload” can have a description that illustrates the relationship between a “reloading” action and Audio File B.

[0037] Therefore, based on the description of Cue 120 having Cue Name “SFX_Player_Attack”, when an “attack” action occurs, this triggers the playing of Audio File A. In a similar vein, based on the description of Cue 120 having Cue Name “SFX_Player_Reload”, when a “reload” action occurs, this triggers the playing of Audio File B.

[0038] In embodiments of the invention, a scriptable object is constructed for each Cue Sheet 110 and Cue 120 pair. This scriptable object is henceforth referred to as Audio Reference object 210. The data structure of Audio Reference object 210 is shown in Figure 2, and comprises the Cue Sheet Name of the Cue Sheet 110 and the Cue Name of the Cue 120.

[0039] Figure 3(a) shows an instantiation of Audio Reference object 210 depicted as Audio Reference object 1 310. Audio Reference object 1 310 comprises Cue Sheet “Player” 320, and Cue “SFX_Player_Attack” 330. Therefore, an association is formed between Audio Reference object 1 310 and Cue Sheet “Player” 320 and Cue “SFX_Player_Attack” 330. Cue Sheet “Player” 320 can comprise Audio File A, and Cue “SFX_Player_Attack” 330 can have a description that illustrates the relationship between an “attack” action and Audio File A.

[0040] Figure 3(b) shows another instantiation of Audio Reference object 210 depicted as Audio Reference object 2 311. Audio Reference object 2 311 comprises Cue Sheet “Player”

320, and Cue “SFX_Player_Reload” 331. Therefore, an association is formed between Audio Reference object 2 311 and Cue Sheet “Player” 320 and Cue “SFX_Player_Reload” 331. Cue Sheet “Player” 320 can comprise Audio File B, and Cue “SFX_Player_Reload” 331 can have a description that illustrates the relationship between a “reload” action and Audio File B.

[0041] Audio Reference objects 210 can be inserted into the program code of the program by the software developer. For example, Figure 4 depicts exemplary pseudocode 410 which can be used by the software developer to code Audio Reference object 1 310 into the program code of the program.

[0042] When the program is executed, Audio Reference objects 210 and Cue Sheets 110 will be loaded into memory.

[0043] When an “attack” action occurs, Audio Reference object 1 310 will be referenced based on the matching of the “attack” action as illustrated by the description of Cue “SFX_Player_Attack” 330.

[0044] Using Audio Reference object 1 310 as the parameter, a sound effect manager will be called. The sound effect manager can belong to a game engine such as the Unity engine.

[0045] The sound effect manager determines that Audio Reference object 1 310 is associated with Cue Sheet “Player” 320 and Cue “SFX_Player_Attack” 330.

[0046] The sound effect manager then calls the API of the audio middleware to play the audio file as illustrated by the description of Cue “SFX_Player_Attack” 330, which is Audio File A. The audio features of Audio File A are then played.

[0047] In a similar fashion, when a “reload” action occurs, Audio Reference object 2 311 will be referenced based on the matching of the “reload” action as illustrated by the description of Cue “SFX_Player_Reload” 331.

[0048] Using Audio Reference object 2 311 as the parameter, the sound effect manager will be called.

[0049] The sound effect manager determines that Audio Reference object 2 311 is associated with Cue Sheet “Player” 320 and Cue “SFX_Player_Reload” 331.

[0050] The sound effect manager then calls the API of the audio middleware to play the audio file as illustrated by the description of Cue “SFX_Player_Reload” 331, which is Audio File B. The audio features of Audio File B are then played.

[0051] In embodiments, each Audio Reference object 210 can also comprise a reference counter. When the Audio Reference object 210 is being referenced, the reference counter of the Audio Reference object 210 will be updated to “1”. When the Audio Reference object 210 is no longer being referenced, the reference counter of the Audio Reference object 210 will be updated to “0”.

[0052] If the program pertains to a computer game, Audio Reference objects 210 can be utilized in various scenarios of the computer game. For example, Audio Reference objects 210 can be utilized at the start or in the “opening scene” of the computer game. Audio Reference objects 210 can also be utilized for the user interface interactive sound effects in the computer game. Audio Reference objects 210 can also be utilized for the background sound effects in the computer game. Audio Reference objects 210 can also be utilized for the in-game sound effects, the in-game character dubbing, the in-game narrator dubbing, the in-game skill sound effects, and the in-game plot sound effects.

[0053] Figure 5 depicts a method of using scriptable objects to insert audio features into a program, in accordance with certain embodiments.

[0054] In step 502, a plurality of Cue Sheets 110 are obtained from an audio middleware. Each Cue Sheet 110 comprises a Cue Sheet Name, a plurality of Cues 120 and a plurality of audio files. Each Cue 120 comprises a Cue Name and a description, the description illustrating a relationship between a unique action and one of the audio files of Cue Sheet 110.

[0055] In step 504, Cue Sheets 110 and Cues 120 are scanned to identify Cue Sheets 110 based on the Cue Sheet Name and Cues 120 based on the Cue Name. This step can be automated.

[0056] In step 506, each Cue Sheet 110 is traversed to select each Cue 120 of Cue Sheet 110.

[0057] In step 508, a scriptable object (i.e. Audio Reference object 210) is constructed for each traversed Cue Sheet 110 and selected Cue 120 pair. This step can be automated. Each Audio Reference object 210 will comprise the Cue Sheet Name of the traversed Cue Sheet 110

and the Cue Name of the selected Cue 120 (as per Figure 2). Therefore, an association is formed between each Audio Reference object 210 and its traversed Cue Sheet 110 and selected Cue 120 pair.

[0058] For example, Cue Sheet “Player” 320 is traversed and Cue “SFX_Player_Attack” 330 is selected. Audio Reference object 1 310 is thus constructed for Cue Sheet “Player” 320 and Cue “SFX_Player_Attack” 330. Therefore, an association is formed between Audio Reference object 1 310 and Cue Sheet “Player” 320 and Cue “SFX_Player_Attack” 330.

[0059] In another example, Cue Sheet “Player” 320 is traversed and Cue “SFX_Player_Reload” 331 is selected. Audio Reference object 2 311 is thus constructed for Cue Sheet “Player” 320, and Cue “SFX_Player_Reload” 331. Therefore, an association is formed between Audio Reference object 2 311 and Cue Sheet “Player” 320 and Cue “SFX_Player_Reload” 331.

[0060] In Step 510, Audio Reference objects 210 are inserted into the program code of the program. Audio Reference objects 210 can be inserted into the program code of the program by the software developer. For example, Figure 4 depicts exemplary pseudocode 410 which can be used by the software developer to insert Audio Reference objects 210 into the program code of the program.

[0061] The use of Audio Reference objects 210 is advantageous as Audio Reference objects 210 make it easier for software developers to insert the necessary code into the program code. For instance, instead of writing lines of code to reference Cue Sheet 110 and Cue 120, the software developer simply writes a single line of code to reference the Audio Reference object 210 (i.e. “*Load Audio Reference Object 1*”).

[0062] The usefulness and utility of Audio Reference objects 210 is further amplified when one considers that audio middleware typically exports many Cue Sheets 110, each containing many Cues 120. Therefore, the software developer can easily insert references to the Audio Reference objects 210 in various parts of the program code, and yet maintaining access to the functionality of Cues 120 and the audio files in Cue Sheets 110.

[0063] Furthermore, the “encapsulation” of a unique Cue Sheet 110 and Cue 120 pair in Audio Reference object 210 forms a coupling between that particular Cue Sheet 110 and that particular Cue 120. Therefore, software developers will not need to ascertain which Cue 120

belongs to which Cue Sheet 110 and can just simply rely on Audio Reference objects 210 which already encodes this relationship. Further, due to said coupling in Audio Reference object 210, software developers will be less error prone when compared to the traditional way of entering code to refer to the Cue Sheet 110 and Cue 120, as in the latter, software developers can make mistakes like referring to a Cue Sheet 110 that does not contain a particular Cue 120.

[0064] Figure 6 depicts a method of using scriptable objects to play audio features in a program, the program being a computer game, in accordance with certain embodiments.

[0065] In step 602, when the computer game is launched, Audio Reference objects 210 and Cue Sheets 110 are loaded into memory. Each Cue Sheet 110 comprises a Cue Sheet Name, a plurality of Cues 120 and a plurality of audio files. Each Cue 120 comprises a Cue Name and a description, the description illustrating a relationship between a unique action and one of the audio files of Cue Sheet 110. Each Audio Reference object 210 is associated with a unique Cue Sheet 110 Cue 120 pair.

[0066] In step 604, when a specific action occurs during the computer game, the Audio Reference object 210 whose associated Cue 120 has a description that matches the specific action is referenced.

[0067] For example, if the specific action is an “attack” action, Audio Reference object 1 310 is referenced based on the matching of the “attack” action as illustrated by the description of Cue “SFX_Player_Attack” 330.

[0068] In another example, if the specific action is a “reload” action, Audio Reference object 2 311 is referenced based on the matching of the “reload” action as illustrated by the description of Cue “SFX_Player_Reload” 331.

[0069] In step 606, using the referenced Audio Reference object 210 as the parameter, a sound effect manager of the game engine is called. The game engine can be the Unity engine.

[0070] For example, Audio Reference object 1 310 is used as a parameter to call the sound effect manager of the game engine. In another example, Audio Reference object 2 311 is used as a parameter to call the sound effect manager of the game engine.

[0071] In step 608, the sound effect manager determines that the referenced Audio Reference object 210 is associated with a specific Cue Sheet 110 and Cue 120 pair.

[0072] For example, the sound effect manager determines that Audio Reference object 1 310 is associated with Cue Sheet “Player” 320 and Cue “SFX_Player_Attack” 330. In another example, the sound effect manager determines that Audio Reference object 2 311 is associated with Cue Sheet “Player” 320 and Cue “SFX_Player_Reload” 331.

[0073] In step 610, the sound effect manager calls the API of the audio middleware to play the audio features of the audio file as illustrated by the description of the Cue 120 associated with the referenced Audio Reference object 210.

[0074] For example, the sound effect manager calls the API of the audio middleware to play the audio features of Audio File A as illustrated by the description of Cue “SFX_Player_Attack” 330. In another example, the sound effect manager calls the API of the audio middleware to play the audio features of Audio File B as illustrated by the description of Cue “SFX_Player_Reload” 331.

[0075] The present invention may be implemented in any personal computer, processor or hardware platform suitable for performing the methodologies described herein. Computer-readable storage media refer to any medium or media that participate in providing instructions to a CPU for execution. Such media can take many forms, such as but not limited to, non-volatile and volatile media including optical or magnetic disks and dynamic memory, respectively. Common forms of computer-readable media include, for example, a floppy disk, a flexible disk, a hard disk, magnetic tape, any other magnetic medium, a CD- ROM disk, digital video disk (DVD), any other optical medium, RAM, PROM, EPROM, a FLASHEPROM, any other memory chip or cartridge.

[0076] Various forms of transmission media may be involved in carrying one or more sequences of one or more instructions to a CPU for execution. A bus carries the data to system RAM, from which a CPU retrieves and executes the instructions. The instructions received by system RAM can optionally be stored on a fixed disk either before or after execution by a CPU.

[0077] Unless specifically stated otherwise, and as apparent from the following, it will be appreciated that throughout the present specification, discussions utilizing terms such as “receiving”, “identifying”, “initiating”, “tagging”, “transmitting”, “running”, “incrementing”, “determining”, “assigning”, “approving”, “selecting”, “sending”, “calculating”, “determining”, “replacing”, “generating”, “initializing”, “outputting”, or the like, refer to the action and processes of a computer system, or similar electronic device, that manipulates and transforms

data represented as physical quantities within the computer system into other data similarly represented as physical quantities within the computer system or other information storage, transmission or display devices.

[0078] In the application, unless specified otherwise, the terms "comprising", "comprise", and grammatical variants thereof, intended to represent "open" or "inclusive" language such that they include recited elements but also permit inclusion of additional, non-explicitly recited elements.

[0079] It will be apparent that various other modifications and adaptations of the application will be apparent to the person skilled in the art after reading the foregoing disclosure without departing from the spirit and scope of the application and it is intended that all such modifications and adaptations come within the scope of the appended claims.

CLAIMS

1. A method of using scriptable objects to insert audio features into a program, the method comprising the steps of:

obtaining a plurality of cue sheets from an audio middleware, each of the plurality of cue sheets comprising a cue sheet name, a plurality of cues and a plurality of audio files, and each of the plurality of cues comprising a cue name and a description, the description illustrating a relationship between a unique action and at least one of the plurality of audio files;

scanning the plurality of cue sheets and the plurality of cues to identify the plurality of cue sheets based on the cue sheet name and the plurality of cues based on the cue name;

traversing through each of the plurality of cue sheets to select each of the plurality of cues;

constructing a scriptable object based on each traversed cue sheet and each selected cue, to form an association between the scriptable object and the traversed cue sheet and the selected cue; and

inserting the scriptable objects into the program code of the program;

such that when the program is executed and when an action occurs, a corresponding scriptable object out of the scriptable objects will be referenced as a result of matching the action with the unique action as illustrated by the description of the associated cue of the corresponding scriptable object; and a sound effect manager will be called using the corresponding scriptable object as a parameter which will result in the sound effect manager calling the Application Programming Interface (API) of the audio middleware to play the audio features of the at least one of the plurality of audio files as illustrated by the description of the associated cue of the corresponding scriptable object.

2. The method of claim 1 wherein the audio middleware is any one of the following: CRIWARE, Wwise and FMod.

3. The method of claim 1 or claim 2 wherein the sound effect manager belongs to a Unity engine.

4. The method of any one of the preceding claims wherein the program is a computer game.