US 20110306397A1

(19) **United States**

(12) **Patent Application Publication**

Fleming et al.

(10) **Pub. No.: US 2011/0306397 A1**

(43) **Pub. Date: Dec. 15, 2011**

(54) **AUDIO AND ANIMATION BLENDING**

(75) Inventors: **James Fleming**, Brighton, MA (US); **Marc A. Flury**, Cambridge, MA (US); **Dean N. Tate**, Cambridge, MA (US); **Matthew C. Boch**, Somerville, MA (US); **Isaac Adams**, Revere, MA (US); **Riseon Kim**, Dorchester, MA (US); **Sachi Sato**, Belmont, MA (US)

(73) Assignee: **Harmonix Music Systems, Inc.**, Cambridge, MA (US)

(21) Appl. No.: **12/940,809**

(22) Filed: **Nov. 5, 2010**

Related U.S. Application Data

(60) Provisional application No. 61/354,073, filed on Jun. 11, 2010.

Publication Classification

(51) **Int. Cl.**
A63F 9/24 (2006.01)
A63F 13/00 (2006.01)

(52) **U.S. Cl.** **463/7**; 463/35; 463/31

(57) **ABSTRACT**

Presented herein are methods, apparatuses, programs, and systems for providing a smooth animation transition in a game. An event timeline is provided with event markers denoting points in time on the event timeline. Each event marker is associated with an animation segment from the number of animation segments. A first marker on the event timeline is provided, which indicates a first animation segment to be displayed on the display (at a point in time with respect to event timeline). A second marker on the event timeline is also provided, which indicates a second animation segment to be displayed on the display (at a second point in time with respect to event timeline). Then as the game progresses, and the second point time on the timeline is approaching, a set of animation segments that need to be blended together is determined, to provide a smooth transition from the first animation segment to the second animation segment. Once the set of animations have been determined, a blend is performed among the set of animation segments.

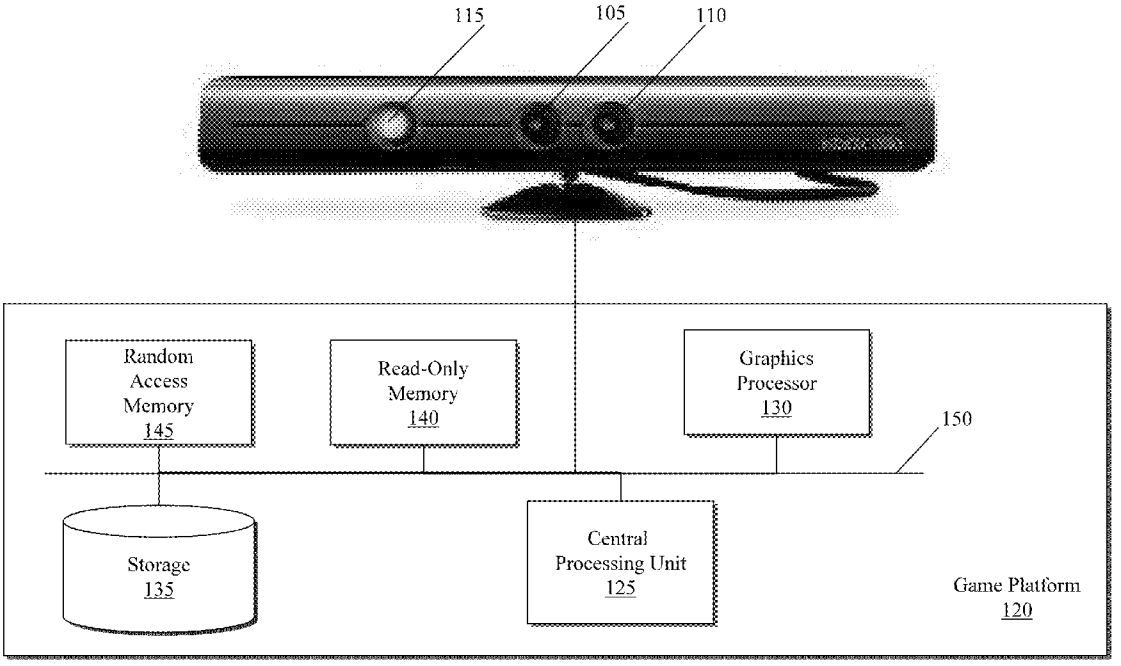


Fig. 1A

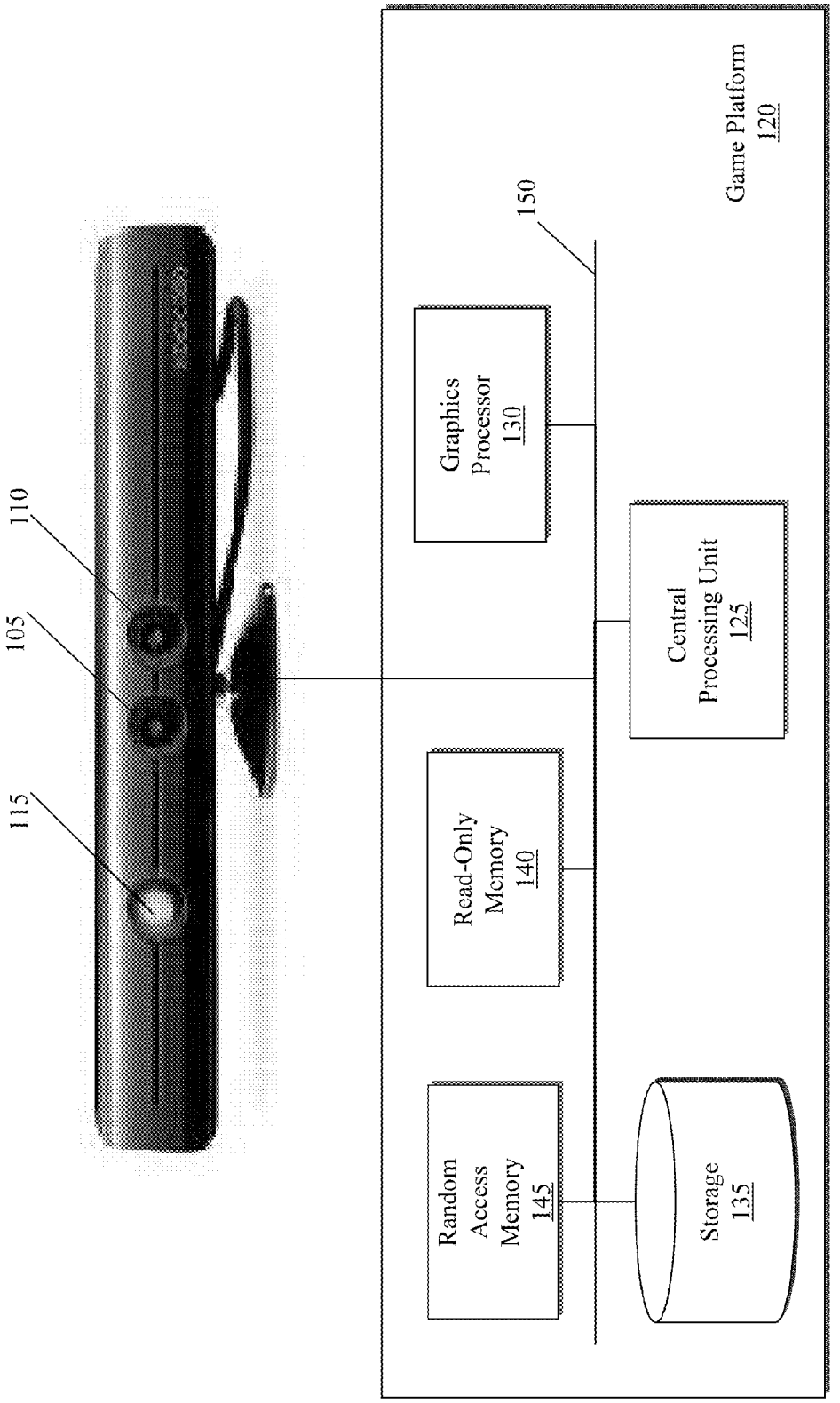

Game Platform
120
150
Graphics
Processor
130
Central
Processing Unit
125
Read-Only
Memory
140
Random
Access
Memory
145
Storage
135
110
105
115

Fig. 1B

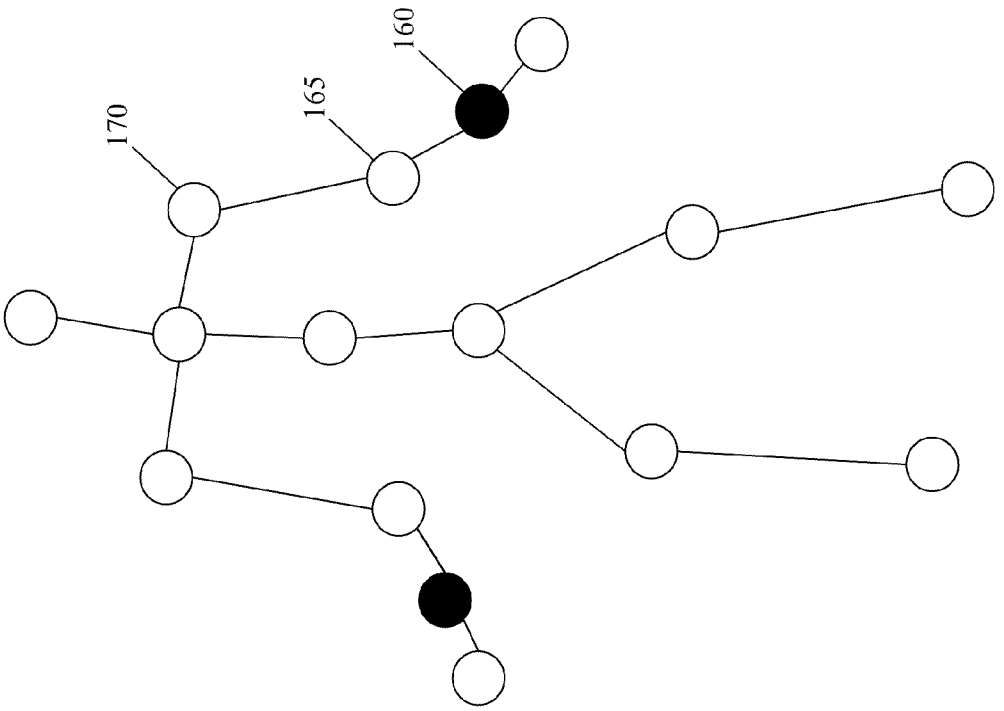


Fig. 1C

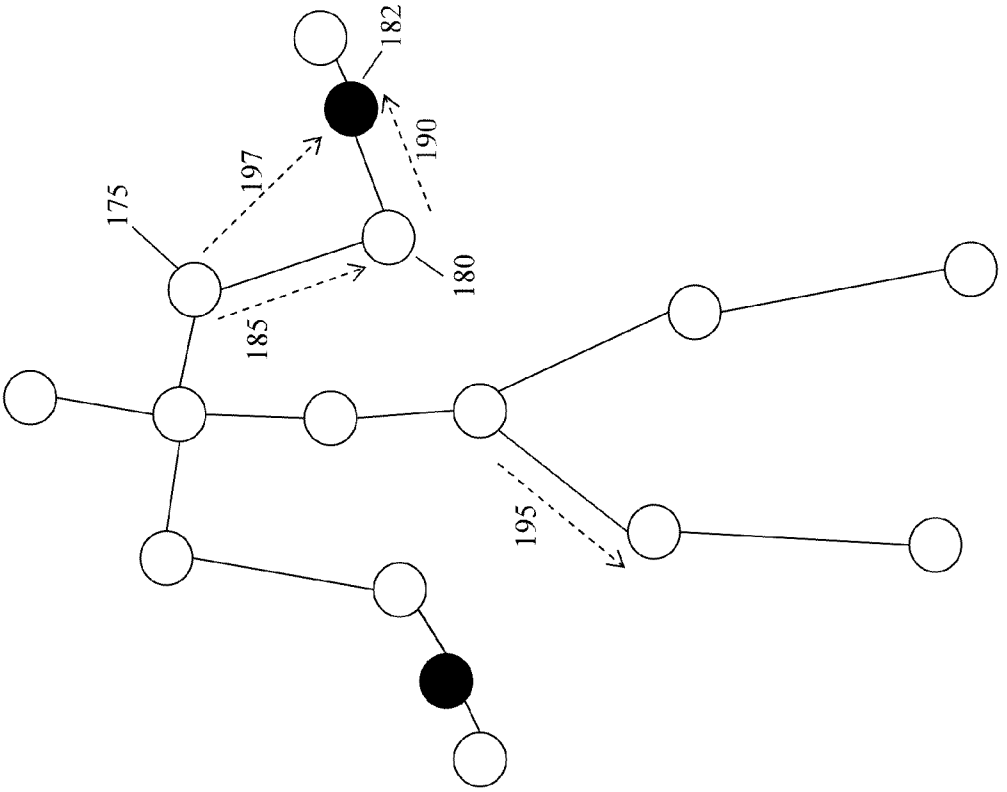


Fig. 2A

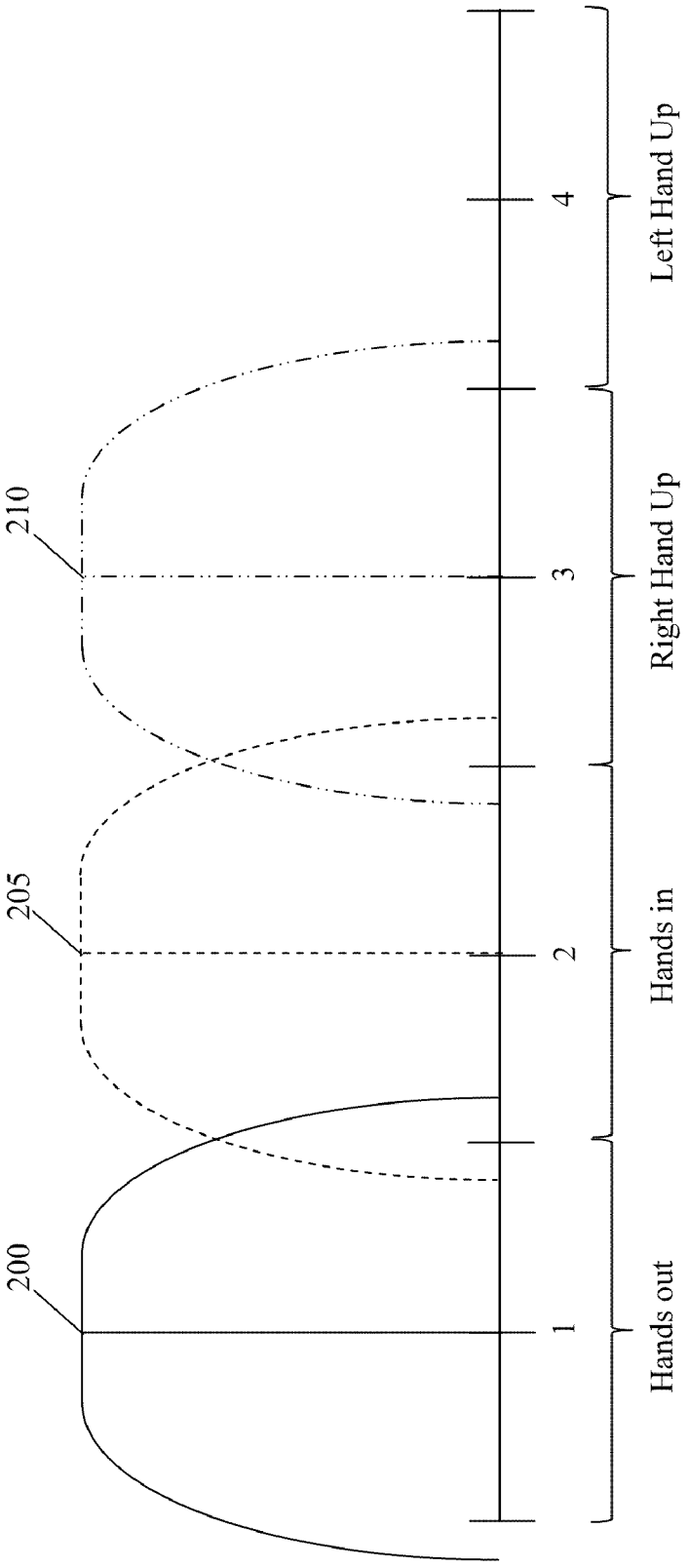


Fig. 2B

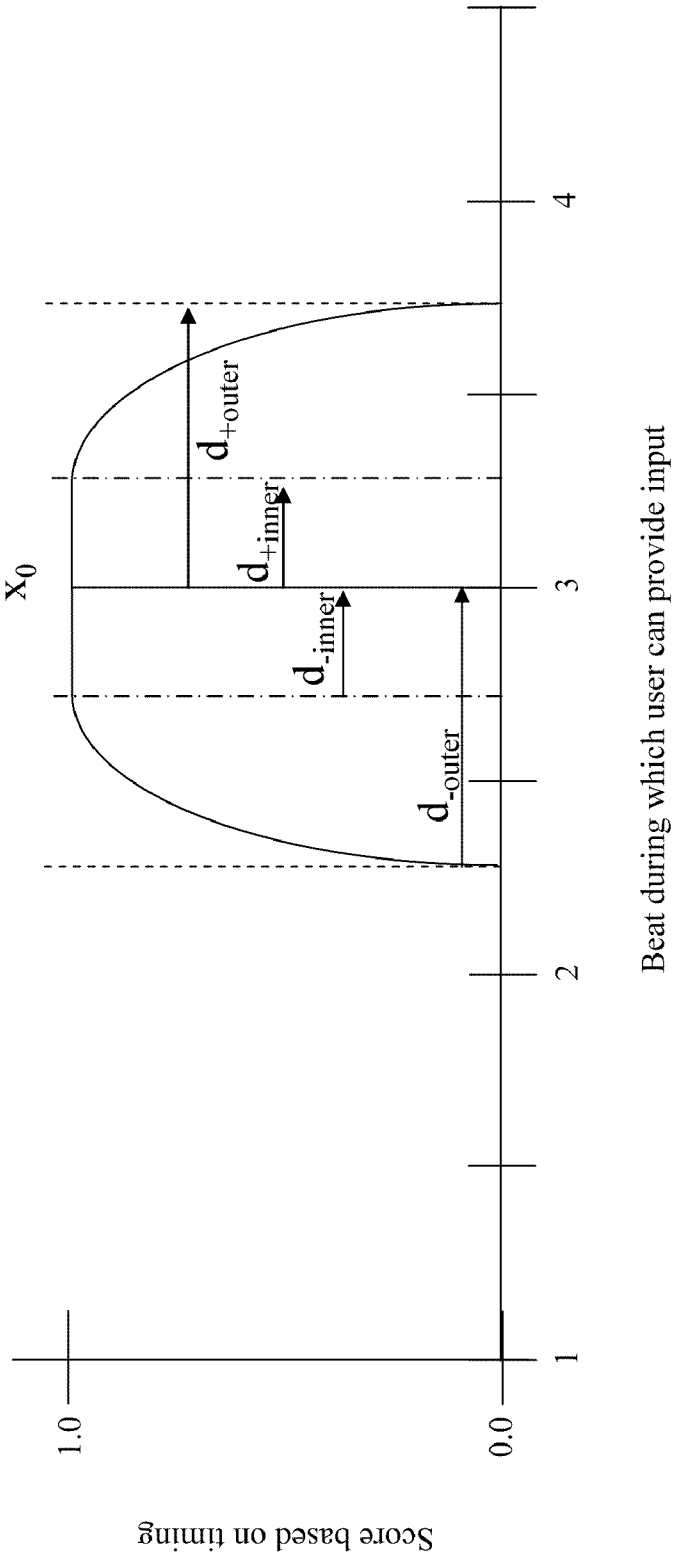

Score based on timing
1.0
0.0
1
2
3
4
Beat during which user can provide input
x_0
d_{+outer}
d_{+inner}
d_{-inner}
d_{-outer}

Fig. 3A

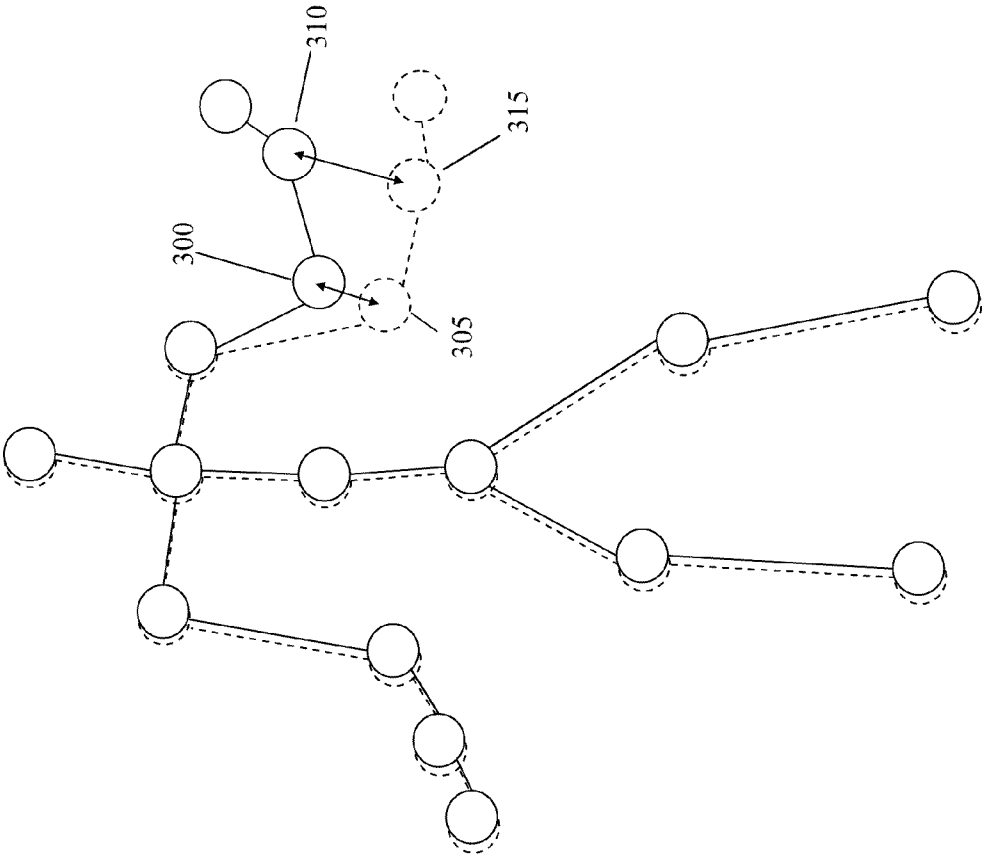


Fig. 3B

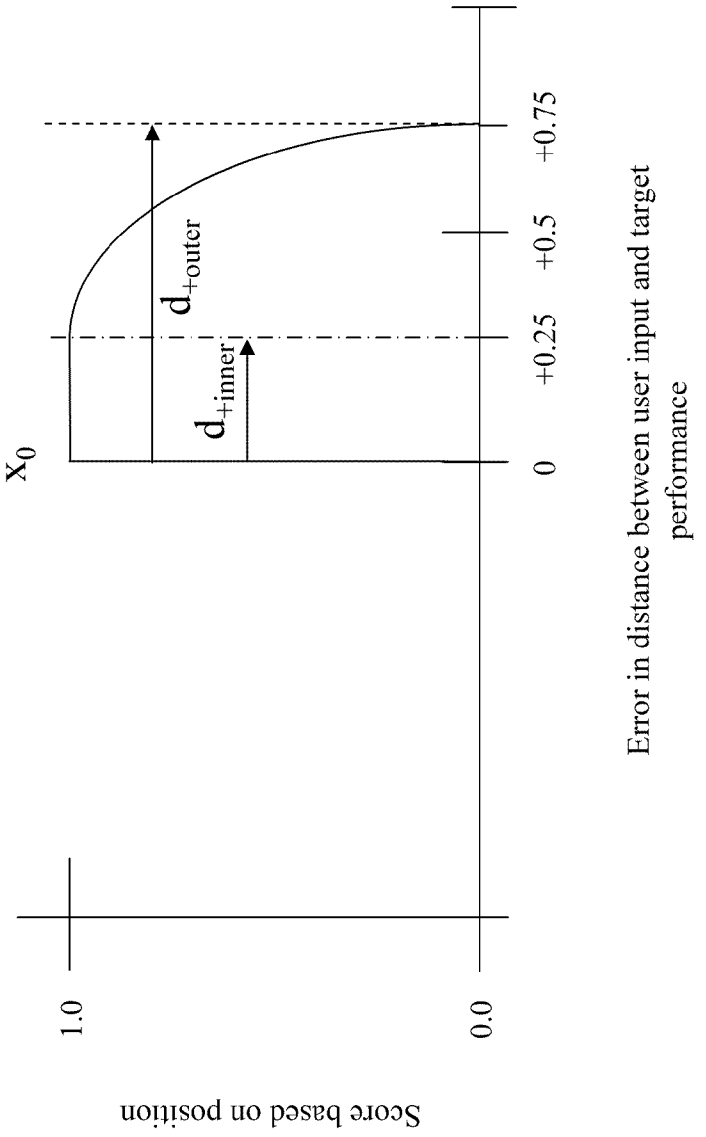
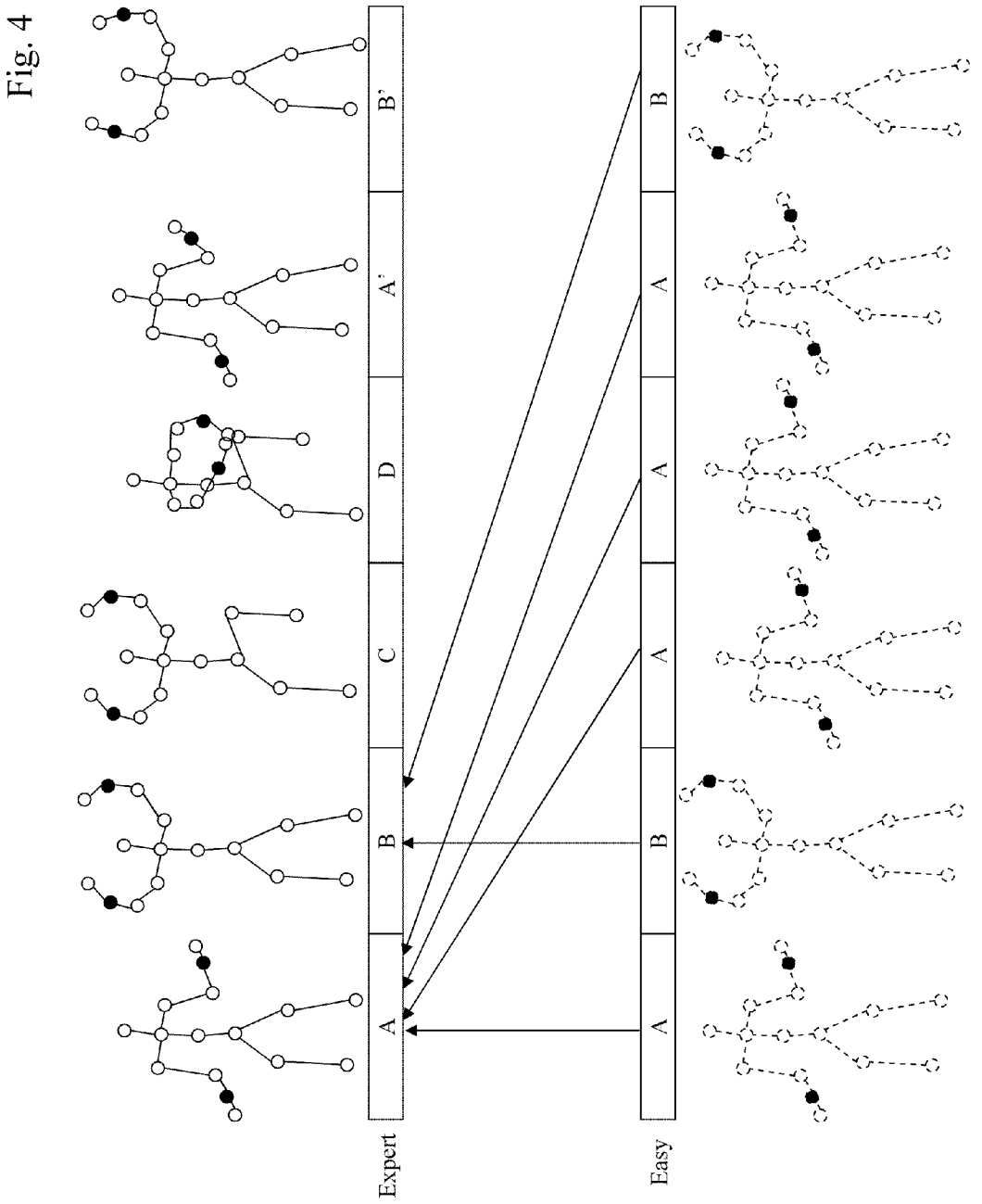

Score based on position
1.0
0.0
x_0
d_{+inner}
d_{+outer}
0
+0.25
+0.5
+0.75
Error in distance between user input and target performance

Fig. 4



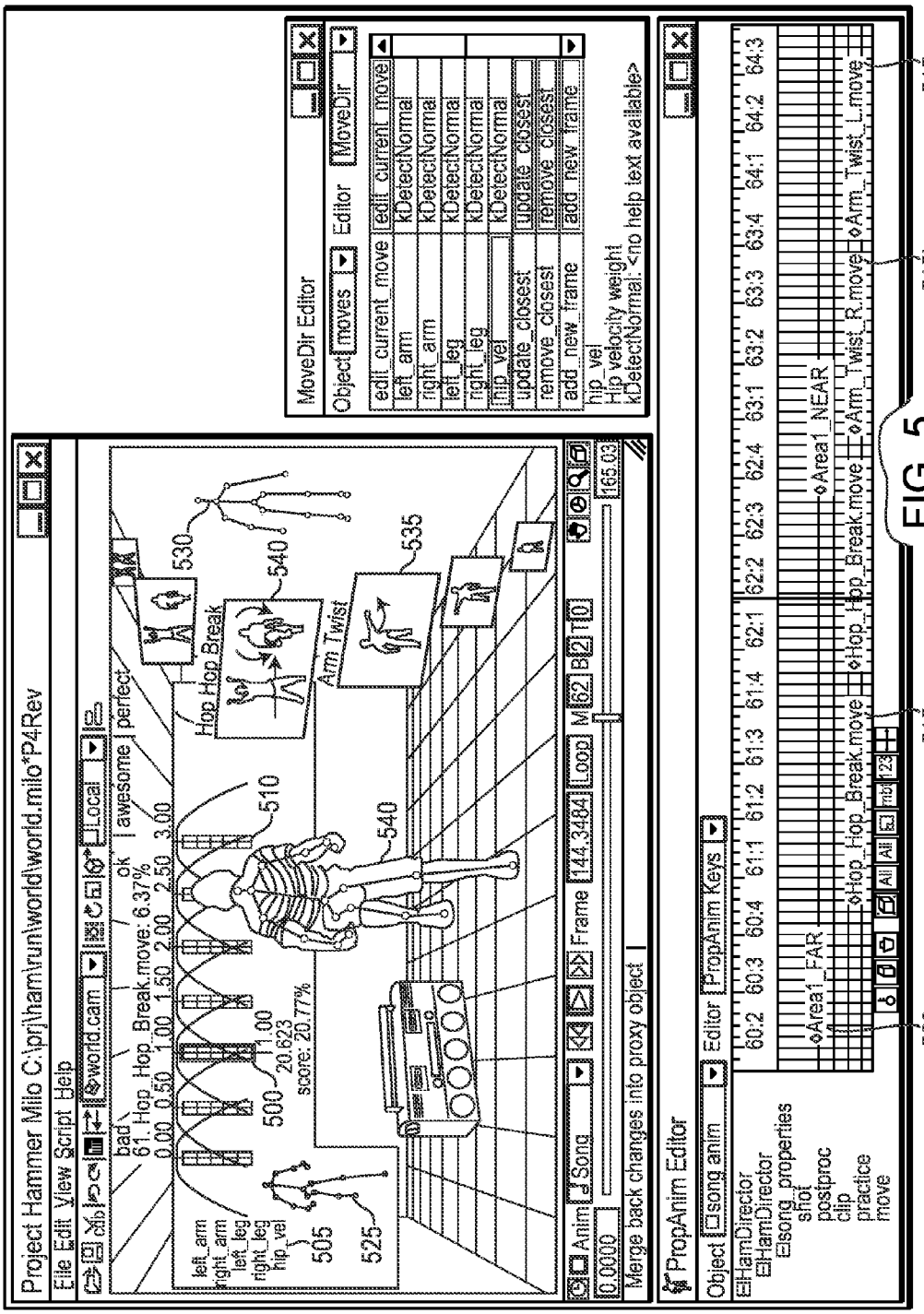

Project Hammer Milo C:\pr\hamrun\world\world.milo*P4Rev
File Edit View Script Help
world.cam
Local
bad
ok
awesome
perfect
61 Hop_Hop_Break.move: 6.37%
0.00 0.50 1.00 1.50 2.00 2.50 3.00
Hop Hop Break
Arm Twist
left_arm
right_arm
left_leg
right_leg
hip_vel
505
525
500
510
520
530
535
540
11.00
20.623
score: 20.77%
Anim
Song
Frame
144.3484
Loop
M62
B2 10
165.03
0.0000
Merge back changes into proxy object
PropAnim Editor
Object song anim
Editor
PropAnim Keys
HamDirector
song_properties
shot
postproc
clip
practice
move
60:2 60:3 60:4 61:1 61:2 61:3 61:4 62:1 62:2 62:3 62:4 63:1 63:2 63:3 63:4 64:1 64:2 64:3
Area1_FAR
Area1_NEAR
Hop_Hop_Break.move
Arm_Twist_R.move
Arm_Twist_L.move
515a
515b
515c
MoveDir Editor
Object moves
Editor MoveDir
edit_current_move
left_arm
right_arm
left_leg
right_leg
hip_vel
update_closest
remove_closest
add_new_frame
kDetectNormal
Hip velocity weight
kDetectNormal: <no help text available>

FIG. 5

AUDIO AND ANIMATION BLENDING

CROSS REFERENCE TO RELATED APPLICATION

[0001] This application claims benefit of priority to Application No. 61/354,073, filed Jun. 11, 2010 and entitled "Dance Game and Tutorial" by Flury et al., the disclosure of which is incorporated herein by reference in its entirety.

FIELD OF THE INVENTION

[0002] The present invention relates generally to gesture-based video games and, more specifically, to dance video games based on positional input from a user.

BACKGROUND

[0003] Although video games and video game consoles are prevalent in many homes, game controllers, with their myriad of buttons and joysticks, are still intimidating and confusing to people that do not often play video games. For these people, using a game controller to interact with the game is an obstacle to enjoying it. Also, where the game is a dance game, often an additional controller is required in the form of a dance mat or dance pad. These dance mats have specific input sections (similar to buttons on a traditional controller) that react to pressure from the user's feet. But these mats take up a lot of space and are often single use controllers—they are used just for dance games and must be rolled up and stored when not in use.

[0004] To increase a user's feeling of immersion in the game, as well as to overcome the cumbersome nature of game controllers or dance mats for users not familiar with them, some game platforms forego the use of traditional controllers and utilize cameras instead. The cameras detect a user's physical movements, e.g., the waving of his arm or leg, and then interpret those movements as input to the video game. This allows the user to use a more natural-feeling input mechanism he is already familiar with, namely the movement of his body, and removes the barrier-to-entry caused by the many-buttoned controller.

[0005] One example of a camera-based controller is the EyeToy camera developed by Logitech and used with the Sony PlayStation 2 game console. The EyeToy, and similar cameras, typically include a camera and a microphone. The EyeToy sends a 640×480 pixel video stream to the PlayStation, and the game executing on the PlayStation parses the frames of the video, e.g., calculating gradations of color between pixels in the frame, to determine what in the camera's field-of-view is the user ("player") and what is the background ("not player"). Then, differences in the stream over time are used to determine and recognize the user's movements, which in turn drive the user's interaction with the game console.

[0006] Other cameras used by game platforms include the DreamEye for the Sega Dreamcast, The PlayStation Eye (a successor to the EyeToy) for Sony's PlayStation 3, and the Xbox Live Vision for Microsoft's Xbox 360. These cameras all provide a typical single-input camera that can stream video or take still photographs, and some, such as the PlayStation Eye, additionally provide a microphone for audio input.

[0007] Microsoft is currently developing a depth-aware camera system in the form of Project Natal. A Natal system provides an RGB camera, a depth sensor, a multi-array microphone, and software that processes the inputs from the camera, depth sensor, and microphone. Beneficially, the Natal software provides, based on the input, a three-dimensional skeleton that roughly maps to the user's body. Specifically, rather than just determining a difference between "player" and "not player" like prior game cameras, Natal determines what is the user's right hand, left hand, head, torso, right leg, and left leg. This skeleton is preserved as a user moves his body in the camera's field of view, allowing for the tracking of specific limbs. This skeleton framework, however, is the extent of what Natal provides. Namely, no user interface is provided by Natal, and users must still use a game controller to interact with a game or menu system.

[0008] Other systems, based on non-camera technologies, have also been developed that attempt to track a user's movements. For example, the Nintendo Wii provides players with an infrared transmitter "Wii remote" that the user holds in his hand. The Wii remote is used as pointing device and has a built-in accelerometer to track changes in the Wii remote's position. The Wii remote is often paired with a "nunchuk" (which also has an accelerometer) that is held in the player's other hand, allowing the Wii to, in a sense, track the movements—or at least changes in the movements—of the user's hands. Another technology based on a hand-held controller is sixense, which is demonstrated at http://www.sixense.com

[0009] High-end motion capture ("mocap") systems have also been used to track a user's movements. Typically mocap systems involve the user wearing a body suit that has dozens of white spheres located at relevant locations. The mocap cameras detect these spheres and use them to infer positional information about the user's body. Mocap systems, however, are expensive and not practical for the average user.

SUMMARY OF THE INVENTION

[0010] The invention includes methods, systems, computer program products and means for providing a dance video game that, by utilizing a camera-based system, obviates the need for, or use of, a typical game controller or dance mat for input. Though Natal is used as an example herein, the invention is not limited to a Natal implementation.

[0011] In one embodiment, there is a filter system for capturing and scoring what a user is doing. The user's input is normalized to a reference framework and compared against key frames of a target performance, which has also been normalized to the reference framework. The closer the user's input is to the correct move at the correct time, the better the rating awarded to the user.

[0012] Advantageously, the game and its filters behave similarly for a short person and a tall person relative to their own bodies. In one embodiment of the invention, appendage and body position determinations are made based on, and relative to, the skeleton of the person interpreted by the system, not on an absolute coordinate system within the camera's field of view. Other embodiments can utilize an absolute coordinate system to infer information about the user's body to create a skeleton for use with the invention.

[0013] Typically, ranges are used to determine if a user has successfully performed a move because motion-tracking input is inherently noisy. Determining precisely where a user's appendages are is difficult due to the natural movement of the user over time and the lag between receiving camera input and processing it. This is complicated when the user is trying to perform a particular dance move at a particular time—he may start or end the move too early or too late, or some appendages may be positionally inaccurate, or a com-

bination of these. Therefore, the filter system allows for variation in both timing and position when scoring the user.

[0014] The invention can also be used to teach a user how to dance. In some implementations, there is a means for teaching a specific move or series of moves to a user using audible cues and repetition. To facilitate this functionality, an additional aspect of the invention is an animation blending technique that uses animation transitions from an idle state into a move, and from the move into an idle state, along with the animation of the move in the context of the entire dance, to allow the teaching avatar to demonstrate and repeat a single move.

[0015] There are also scripted transitions, or "bridge animation segments" that allow for seamless reuse of portions of a motion capture performance, that, in the performance, are not actually adjacent. Beneficially, these bridge animation segments can be used in a variety of contexts. For example, a difficult dance routine with many different moves can be simplified into a lower difficulty routine by repeating a subset of the moves, which requires jumping to non-adjacent animation segments. Also, the bridge animation segments can be used in a practice mode to repeat moves until the player has successfully performed them. And, bridge animation segments can be used to extend the motion capture performance in a multiplayer mode by looping a segment of the motion capture performance.

[0016] The invention also provides seamless audio track transition playback during multi-player modes. It is more fun for users to trade off dancing during a song than it is for each user to play all the way through while the other waits. But songs are often written for a single play-through and do not facilitate smooth "rewinding" from an audible standpoint. Specifically, the music at time t_1 (later in the song) does not usually lend itself to a smooth transition to t_0 (earlier in the song). The invention solves this by providing segments of transition audio to use between different parts of the song, selectively muting the original audio and unmuting the appropriate transition segment when a transition to a different part of the song is necessary. As with bridge animations, these bridge audio segments can be used in a variety of contexts. For example, the bridge audio segments can be used in a practice mode to repeat sections of the song the player is practicing until the player has successfully performed them. And, bridge audio segments can be used to extend the song in a multiplayer mode by looping a segment of the song audio.

[0017] In one embodiment, there is a method, executed on a game platform, for scoring a player performance that includes one or more poses in a dance-based video game based on input received via a sensor. The method and the components it interacts with can also be expressed as a system, in the form of a computer program product, or as an apparatus with means for accomplishing the interaction, where the structures correspond to a game platform and a sensor (e.g., a camera) in communication with the game platform. In cases where results are displayed, a display in communication with the game platform may also be used. The method includes receiving a performance 3D skeleton indicating a pose of the player, then calculating a score by comparing a position, a timing, or both, associated with one or more joints of the performance 3D skeleton to a position, a timing, or both, associated with one or more joints of a target pose; and then altering one or more characteristics of the dance-based video game based on the score.

[0018] In another embodiment, there is also a method, executed on a game platform, for evaluating a player performance based on input from a sensor. As described above, the method and the components it interacts with can also be expressed as a system, in the form of a computer program product, or as an apparatus with means for accomplishing the interaction, where the structures correspond to a game platform and a sensor (e.g., a camera) in communication with the game platform. In cases where results are displayed, a display in communication with the game platform may also be used. The method includes receiving a performance 3D skeleton indicating a portion of the player performance, providing a target 3D skeleton indicating a portion of a target performance, defining a per joint error function, calculating an error using the per joint error function based on the performance 3D skeleton and the target 3D skeleton, and producing, with the game platform, an audio or visual indication of the error.

[0019] In one implementation, there is a method, executed on a game platform, for scoring a player performance that includes one or more poses in a dance-based video game based on input received via a sensor. As described above, the method and the components it interacts with can also be expressed as a system, in the form of a computer program product, or as an apparatus with means for accomplishing the interaction, where the structures correspond to a game platform and a sensor (e.g., a camera) in communication with the game platform. In cases where results are displayed, a display in communication with the game platform may also be used. The method includes providing a target 3D skeleton indicating a target pose and receiving a number of 3D performance skeletons indicating player poses. Then an overall score for a particular 3D performance skeleton is generated by comparing a position associated with one or more joints of the 3D performance skeleton to a corresponding position associated with one or more joints of the target 3D skeleton. Then the score generating step is repeated for each of the of 3D performance skeletons that fall within a predetermined temporal range. This generates a number of overall scores, and an audio or visual indication of the accuracy of the performance, that is based on one or more of the overall scores, is displayed on the display.

[0020] In one embodiment, there is a method, executed on a game platform, for scoring a player performance that includes one or more poses in a dance-based video game based on input received via a sensor. As described above, the method and the components it interacts with can also be expressed as a system, in the form of a computer program product, or as an apparatus with means for accomplishing the interaction, where the structures correspond to a game platform and a sensor (e.g., a camera) in communication with the game platform. In cases where results are displayed, a display in communication with the game platform may also be used. The method begins by receiving a performance 3D skeleton indicating a pose of the player. Then a score is calculated by comparing a measurement of one or more reference points, e.g., joints, bones, or derivations of either, of the performance 3D skeleton to a measurement of one or more reference points, e.g., joints, bones, or derivations of either, of a target pose. Some examples of measurements are displacement, velocity, acceleration, but other measurements would be understood by one of skill in the art. Then, one or more characteristics of the dance-based video game are altered based on the score.

[0021] Any of the above embodiments may enjoy one or more of the following benefits. In some versions, the target pose, expressed as a 3D skeleton, is generated based on motion capture data. This is then provided during the game for weighting, comparison against the player's performance, and/or scoring. And in some instances, the target pose is selected based on its importance to the target performance, e.g., a transitional movement is not important, but a particular orientation of the player's body at a particular time is for the movement to be recognized as a particular dance move.

[0022] Also, in some implementations, the position associated with the one or more joints of the 3D skeleton and the position associated with the one or more joints of the target pose are based on a normalization of the spatial position of the one or more joints of the 3D skeleton and the one or more joints of the target pose, respectively. For example, normalizing the spatial position of a joint can include computing a unit vector reflecting an offset of the joint relative to an origin joint. It can further involve defining a coordinate system originated at the origin joint; and translating the unit vector into the coordinate system. Examples of potential origin points are the left shoulder, the right shoulder, the left hip, or the right hip.

[0023] Alternatively, normalizing the spatial position of a joint could be based on computing a first vector reflecting the offset of the joint relative to an origin joint, computing a second vector reflecting the offset of an intermediate joint relative to the origin joint, and then computing a third vector reflecting the offset of the joint relative to the intermediate joint. Then the first vector is divided by the sum of the magnitudes of the second and third unit vectors.

[0024] In some implementations, a second target 3D skeleton is provided indicating a second target pose and a second number or group of 3D performance skeletons are received indicating player poses. An overall score is generated for a second particular 3D performance skeleton by comparing a position associated with one or more joints of the second 3D performance skeleton to a corresponding one or more joints of the second target 3D skeleton. Then, the score generating step is repeated for each of the second group of 3D performance skeletons that fall within a second predetermined temporal range to generate a second number of overall scores. Lastly, a second audio or visual indication—based on one or more of the first and second plurality of overall scores—is produced on the display. The overall score can be based on a positional score and/or a timing-based score. Additionally or alternatively, the overall score can be based on a displacement score, a velocity score, or an acceleration score.

[0025] In some implementations, the target pose is associated with a target performance timing, and a timing-based score varies inversely as the difference between the target performance timing and a timing of performance of the player pose. In some implementations, the timing-based score is based on a first constant if the difference is less than a first predetermined threshold. In some, it varies inversely as the difference if the difference is between the first predetermined threshold and a second predetermined threshold. And in some implementation, the score is a second constant if the difference is greater than the second predetermined threshold. And in some implementations, computing the timing-based score uses a combination of the foregoing techniques.

[0026] In some embodiments, the positional score can include one or more per-joint scores, where the contribution of each per-joint score to the overall score is weighted. Additionally or alternatively, the positional score can include one or more body part scores, with each body part score having one or more per-joint scores. And as above, the contribution of each body part score to the overall score can be weighted.

[0027] In some versions, comparing the joints of the 3D performance skeleton to the joints of the target 3D skeleton includes calculating a Euclidean distance between the position associated with the more joints of the 3D performance skeleton and the corresponding position associated with the joints of the target 3D skeleton to generate a per-joint score. And this per-joint score can vary inversely as the Euclidean distance. In some embodiments, for example, the per-joint score can be a first constant if the Euclidean distance is less than a first predetermined threshold. In some embodiments, it varies inversely as the Euclidean distance if the Euclidean distance is between the first predetermined threshold and a second predetermined threshold. And in some embodiments, it is a second constant if the Euclidean distance is greater than the second predetermined threshold. And in some implementations, computing the per-joint score uses a combination of the foregoing techniques.

[0028] Additionally or alternatively, calculating the Euclidean distance can include weighting the contributions of the x, y, and z axes to the Euclidean distance, with the orientations of the x, y, and z axes are relative to each body zone. For example, the x, y, and z axes of an arm body zone can originate at a shoulder connected to the arm and are based on a first vector from the shoulder to an opposite shoulder, a second vector in the direction of gravity, and a third vector that is a cross product of the first and second vectors. Alternatively, the x, y, and z axes of a leg body zone can originate at a hip connected to the leg and are based on a first vector from the hip to an opposite hip, a second vector in the direction of gravity, and a third vector that is a cross product of the first and second vectors. And, the orientations of the x, y, and z axes can also be relative to the target 3D skeleton. That is, the x, y, and z axes are rotated based on an angle between a vector normal to a plane of the target 3D skeleton and a vector normal to a view plane.

[0029] In some embodiments, the audio or visual indication comprises a visual indication of a score for a body part, such as flashing the body part green or red for success or poor performance respectively. Or, the audio or visual indication comprises an audible indication of the score for a body part, e.g., "nice arms!" or "you need to move your feet more!" The audio or visual indication can also include one more graphics, with the number of graphics being based on the one or more overall scores. The audio or visual indication can also be a simulated crowd, where one or more characteristics of the simulated crowd varies based on the one or more of the plurality of overall scores, e.g., the number of people in the crowd can grow or shrink. The audio or visual indication can also be a simulated environment, where one or more characteristics of the simulated environment varies based on the one or more of the plurality of overall scores, e.g., there can be fireworks or flashing lights if the player is performing well and destruction or dimming of the environment if the player is performing poorly.

[0030] There is also a method of providing a smooth animation transition in a game. As described above, the method and the components it interacts with can also be expressed as a system, in the form of a computer program product, or as an apparatus with means for accomplishing the interaction, where the structures correspond to a game platform and a

display in communication with the game platform. The method includes, during gameplay, providing an event timeline with event markers denoting points in time on the event timeline. Each event marker is associated with an animation segment from the number of animation segments. The first marker on the event timeline is provided, which indicates a first animation segment to be displayed on the display (at a point in time with respect to event timeline). A second marker on the event timeline is also provided, which indicates a second animation segment to be displayed on the display (at a second point in time with respect to event timeline). Then as the game progresses, and the second point time on the timeline is approaching, a set of animation segments that need to be blended together is determined, to provide a smooth transition from the first animation segment to the second animation segment. Once the set of animations have been determined, a blend is performed among the set of animation segments. For example, a blend between the first animation segment and the second animation segment may be all that is needed and portions of each are blended together. More specifically, a number of beats at the end of the first animation segment may be blended with a number of beats at the beginning of the second animation. Alternatively, a smoother result may be achieved using a bridge animation segment, where a portion of the first animation segment is blended with the bridge animation segment and then a portion of the bridge animation segment is blended with the second animation segment. In some versions, the existence of a bridge animation segment is determined by the presence of a bridge animation segment or a reference to a bridge animation segment listed in a lookup table, keyed off the animations that are being blended from and to. If the entry in the lookup table exists, the bridge animation segment is used. If not, no bridge animation segment is used, and the first animation segment is blended directly into the second animation segment.

[0031] These blending techniques are useful for multiplayer gameplay as well, where a first player is scored before the blend occurs and a second player is scored after the blend. Beneficially, the second animation can be based on the difficulty of the game, so if the first player is very good and is performing a complex move, the transition to the second player's move, which is easier, will not be jarring because of the transition. Or, where a series of animations are recorded and then "cut up" and reused to form progressively easier routines (with animation sequences from the hardest difficulty being arranged for each difficulty level), the transitions can be used to create a dance routine using non-contiguous or looping animation sequences. The technique can also be used to blend an animation with itself, that is, blend the ending position of the dancer with the starting position of the dancer. This can be used for repeated moves.

[0032] Another technique is to blend audio segments. In one embodiment, there is a method, executed on a game platform, for providing smooth audio transitions in a song. The method includes providing, during play of a game, an event timeline with event markers denoting points in time on the event timeline. The event timeline is associated with an audio track played during play of the game. The audio track has a first audio segment and a second audio segment. Then, a first marker on the event timeline is provided to indicate a first point in time (that is associated with the first audio segment). Also, a second marker on the event timeline is provided that indicates a second point in time with respect to the event timeline (and is associated with the second audio segment). During game play it is determined that the second time is approaching and a set of audio segments to be blended is determined that will ensure a smooth transition from the first audio segment to the second audio segment. Then, based on the set of audio segments that will be used for blending, a blend is applied among the set of audio segments based on the prior determination made. For example, the set of audio segments that are used for the blend could be just the first audio segment and the second audio segment, and the blend could be accomplished by crossfading between the two. Or, a bridge audio segment may also be necessary, and the blend can be applied to a portion of the first audio segment and blended with a portion of the bridge audio segment, and then a blend can be applied from a portion of the bridge audio segment to a portion of the second animation. As above, a blend can be a crossfade, but can also be an attenuation, e.g., the first audio segment can be reduced in volume or silenced completely (muted) while the bridge audio segment is played and then the bridge audio segment can be reduced in volume or stopped completely and the second audio would be played. For example, the first audio segment can be muted for its final beat, the second audio segment can be muted for its first two beats, and the bridge audio segment is played in the three beat-long "hole", i.e., during the muted last beat of the first audio segment and muted first two beats of the muted second audio segment.

[0033] In some implementations, the first audio segment and the second audio segment are the same audio segment, as would be the case of a repeated section of audio. The bridge audio segment or a reference to a bridge audio segment can also be stored in a lookup table that is keyed based on the audio segment that is being transitioned from and the audio segment that is being transitioned to.

[0034] Other aspects and advantages of the present invention will become apparent from the following detailed description, taken in conjunction with the accompanying drawings, illustrating the principles of the invention by way of example only.

BRIEF DESCRIPTION OF THE DRAWINGS

[0035] The foregoing and other objects, features, and advantages of the present invention, as well as the invention itself, will be more fully understood from the following description of various embodiments, when read together with the accompanying drawings, in which:

[0036] FIG. **1**A depicts a game platform with a Project Natal camera system;

[0037] FIG. **1**B depicts an example of a skeleton provided by Project Natal;

[0038] FIG. **1**C depicts an example of a skeleton that includes vectors used in determining normalized joint position;

[0039] FIG. **2**A shows a series of movements spread over four beats that begin a representative dance move;

[0040] FIG. **2**B shows a representative window to determine a user's timing error in performing a move;

[0041] FIG. **3**A shows a distance calculation between the target performance skeleton (shown in outline) and the user's input (shown solid);

[0042] FIG. **3**B shows a window of acceptable error for position when performing a move;

[0043] FIG. **4** depicts how a mocap for a dance routine may be refactored to create a dance routine of an easier difficulty;

[0044] FIG. **5** depicts one embodiment of an authoring system for the dance game.

DETAILED DESCRIPTION

[0045] One embodiment of the present invention is based on the Project Natal framework developed by Microsoft Corporation of Redmond, Wash. As indicated in FIG. **1**A, the Project Natal system includes an RGB camera **105**, a depth sensor **110**, a multi-array microphone **115**, and a processor (not shown). The RGB camera **105** delivers a three-color (Red, Green, Blue) video stream to the game console, enabling facial recognition and full-body tracking. The depth sensor **110** is an infrared projector combined with a monochrome CMOS sensor. This allows a game console **120** utilizing Natal to recognize objects in the camera's field of view in three dimensions instead of forcing the game console to parse a two-dimensional video-stream. The multi-array microphone **115** parses voices and sound input, while simultaneously extracting and nullifying ambient noise. Project Natal also features a processor with proprietary software that coordinates the inputs of the Natal system and provides a three-dimensional, skeleton-based system to game developers. Developers can use this system to utilize three-dimensional position information of the joints in the user's body to interact with the game platform.

[0046] Although Project Natal provides a framework for determining positional information of a user's body, it does not provide a means for grading a dance performance or teaching a user to dance. While in some embodiments, a camera-based system is used to determine positional information about the user's body in three dimensions to produce a skeleton model, in other embodiments, transducers attached to the user's body are used to detect the positions of the user's limbs and produce a skeleton model. Other embodiments use infrared pointing devices or other motion tracking peripherals. All that is required is a system than can parse movement in two dimensions to produce a skeleton model; adding dimension information from a third dimension, typically depth, simply makes the invention easier to implement due to the additional information provided to the system. In embodiments where the system is already provided a skeleton, such as Natal, relative body scale mapping is easier to accomplish.

[0047] Also shown in FIG. **1**A is an exemplary game platform **120**. The game platform typically includes a Central Processing Unit (CPU) **125**, a graphics processor **130**, storage component **135** such as a hard drive, Read Only Memory (ROM) **140**, Random Access Memory (RAM) **145**, all in signal communication via a bus **150**. The bus **150** also connects to an input for the Project Natal System. In some embodiments, the Natal system connects to the game platform **120**, e.g., an Xbox 360, via a Universal Serial Bus (USB) connection.

[0048] As used herein, the terms "joint", "bone", and "skeleton" are intended to have the meaning one of skill in the art of motion capture and animation would ascribe to them. For example, a skeleton can comprise bones, but the number of bones and their positions are a function of the motion capture equipment and the animation rig and do not necessarily correlate to the number and positions of bones in a human skeleton. Similarly, a joint can be at the distal endpoint of a single bone (e.g., a fingertip or the head), and need not be at a point where two bones come together. An example of the Natal skeleton is shown in FIG. **1**B. The skeleton provided by the Natal system provides a framework for the dance game, and allows for tracking of not only limbs generally, but specific joints as well. For example, the wrist joint **160** on the right arm is treated separately from the right elbow **165**, which is treated differently than the right shoulder **170**. Additional portions of the body are also recognized, such as the pelvis, middle of the torso, the head, the neck, and the knees and feet.

[0049] One of the benefits provided by the skeleton-based system is that the skeletal model can be used to calculate scale vectors based on two or more joints. This provides a spatially relative system, i.e., what is the positional distance from body part X to body part Y compared to the positional distance from body part X to body part Z, instead of an absolute coordinate system.

[0050] A "filter" as used herein, is in effect a test, e.g., is the user's right hand in a particular position at time t_n? Although typically a producing a Boolean outcome, e.g., if the condition is true, the filter is satisfied and registers a success, and if not, then the filter is not satisfied. Filters may also output a contiguous score indicating the degree to which the condition is being satisfied spatially or temporally. Advantageously, multiple filters can be checked simultaneously, e.g., is the user's right hand in position x and is his left foot in position y? These filters can then be combined to determine if a user has successfully completed a pose. But pose-matching, in and of itself, is not a complete solution to scoring a sequence of dance moves.

Creating a Target Representation

[0051] The process of one implementation begins by using motion capture technology (known in the art as "mocap") to create a three-dimensional model of a target performance of a dance or part of a dance. Motion capture is a recording of human actor which can be used by a computer to reproduce the actor's performance. When the mocap session is recorded, sensors at various points on the actor's body provide the recording computer with information such as joint and limb position data over time. In the case of a dance game, the mocap is typically a recording of a dancer performing a particular dance move, or series of movements that makes up a dance move, and in one implementation, the mocap is a recording of an entire dance routine for a song. The mocap performance becomes a representation of the dance in a form usable by the game system (i.e., a "target performance"). Beneficially, the positional information received during mocap is similar to the positional information received by a camera-based game system when a user is playing a game. This similarity can be exploited to grade a user on how well he is dancing at a particular time by comparing a user's performance (the input performance) to a keyframe of the target performance. Also beneficially, the mocap data can be used to drive on-screen animations of avatars, thus demonstrating to the user the exact movements he must perform to maximize his score.

[0052] At least one notable problem arises though that prevents a direct comparison between the user's performance and the target performance: because the user and the mocap actor could have different heights and appendage lengths, or have different body types, a direct comparison of positional information of the input performance and the target performance could result in the user scoring poorly, even if he is performing the moves correctly. For example, the actor in the target performance could have an arm fully extended which, based on the dimensions of the actor's body, positions the actor's wrist two and a half feet in front of his shoulder. The

user's input, also reflecting a fully extended arm, could have the (shorter-in-stature) user's wrist positioned two feet in front of his shoulder. In a purely comparative system, the user has not satisfied a test of "is the user's wrist in the same position as the wrist of target performance actor?" because the user's wrist, even though his arm is fully extended, is still a half foot closer to the reference point, i.e., the shoulder. Therefore, it is advantageous to express both the target performance and the user's performance in the same frame of reference.

Normalizing the Input Performance and Target Performance

[0053] To create a consistent frame of reference, the mocap data, which is expressed in its own representation (in some implementations even its own skeleton), and the user's input are both normalized, creating a normalized target performance and a normalized input performance. In one implementation, normalization of each joint is achieved by deriving unit vectors reflecting offsets of one specific joint relative to another specific joint.

[0054] In one embodiment, there are four different player-normalized coordinate systems: left arm, right arm, left leg, and right leg. The left arm coordinate system's origin is at the left shoulder, the up vector is away from gravity (in Natal systems, based on Natal's accelerometer). The right vector is from the left shoulder to the right shoulder, the forward vector is the cross product of the up vector and the right vector. The right arm coordinate system is just the mirror of this. The left leg coordinate system's origin is the left hip, the up vector is gravity, the right vector is from the left hip to the right hip, and the forward vector is the cross product of the up vector and the right vector. The right leg coordinate system is the mirror of this.

[0055] As an example, referring to FIG. **1**C, the normalized position of joints on the left arm can be determined as follows. The left shoulder joint **175** is treated as the origin of the vector **185** from the shoulder to the elbow **180** and that vector **185** is transformed from the skeleton's coordinate system into the left arm coordinate system. The vector is then normalized by dividing it by its magnitude. The resulting vector is a "normalized elbow position." A similar process is applied to the input skeleton to determine a normalized elbow position for the user. This method can be used for other joints as well, e.g., the wrist position can be normalized by determining the vector **190** from the elbow **180** to the wrist **182**, transforming that vector from the skeleton's coordinate system into the left arm coordinate system, and dividing it by the magnitude of that vector **190**. A knee's position can be normalized based on the vector **195** between the hip and the knee, transformed from the skeleton's coordinate system into the appropriate-side leg coordinate system, and divided by the magnitude of that vector. An ankle's position can be determined based on the vector from the knee to the ankle, and so forth. Other joints such as hips are usable as well: foot raises are determined as a "squish" from foot to waist where the foot's position is drawn in towards the waist. In one embodiment, the normalized joint positions in the entire skeleton are computed, using the joint more proximal to the body core as the reference joint. In other embodiments, only a subset of the joints that have a correspondence in both skeletons are normalized, and normalization occurs on a limb-by-limb basis. In either embodiment, the normalization of the target performance can be carried out in advance of gameplay, or can be carried out during gameplay.

[0056] There are several options for normalizing joints that are not directly connected to an origin joint. Continuing the previous example with the shoulder **175** being the origin joint, the wrist's position could be normalized by determining the vector **197** from the shoulder **175** to the wrist joint **182**, transforming the vector **197** from the skeleton's coordinate system into the left arm coordinate system, and dividing the resulting vector by the sum of the magnitude of the vector **185** from the shoulder to the elbow and the magnitude of the vector **190** from the elbow to the wrist. Alternatively, the vector **197** from the shoulder to the wrist could be determined, transformed, and divided by the magnitude of that vector **197**. For legs, an ankle position could be based on foot position, transformed from the skeleton's coordinate system into the appropriate-side leg coordinate system, and divided by the sum of the magnitudes of the vector from the hip to the knee and from the knee to the ankle

[0057] Typically, normalizing the target performance and the input performance yields positional information analogous to both, e.g., both have elbow position representations, both have wrist position representations, etc. Where data is not available in the mocap data or the user input for a particular joint though, in some embodiments, the game interpolates between two joints to create a "pseudo-joint" that maps to a joint in the other skeleton. For example, if the mocap skeleton has a left hip joint and a right hip joint, but a user skeleton only has a mid-pelvis joint, a mid-pelvis pseudo-joint can be synthesized for the mocap skeleton at the midpoint of the two hip joints, and used in further normalization and scoring. Alternatively, pseudo-joints could be interpolated from both data sets/skeletons to map to a third idealized skeleton. Additionally, where the input camera system is a Project Natal system, adjustments are typically made to conform the mocap skeleton to the Natal skeleton, or vice versa, e.g., dropping the hips, adjusting the shoulder height, and others. In some embodiments, the game creates a "pseudo-joint" even when data is available in both the mocap data and the user input, in order to provide a reference point or measurement that is more stable than a joint in the existing skeleton.

Comparing the Input Performance to the Target Performance

[0058] In one embodiment of the invention, every "frame" of the input performance is compared with the corresponding frame of the target performance to produce a score for that frame. This strategy, however, does not allow the game to account for inaccuracies in the user's timing, such as dancing a move with perfect position but slightly late or early. In another embodiment, the invention addresses this issue by scoring each frame of the input performance against the corresponding frame of the target performance and a range of adjacent frames. The scoring process incorporates positional and temporal score using a technique described below. For a given target frame, a score is determined by finding the maximum score of all input frames scored against that target frame.

[0059] This approach, however, can be prohibitively expensive computation-wise on some game consoles. To alleviate this, in some embodiments, only a fraction of the input frames are compared with target frames (e.g., half of the input frames). The specific frames in the input performance that are chosen for comparison can be regularly spaced, or the frames can be chosen randomly with a probability matching that fraction.

[0060] This approach, however, does not capture the intent behind a dance move where certain intermediate poses are more important and the transition movements into or out of those poses are less important. In a preferred embodiment, the input frames should be compared to the target frames most important to the dance itself.

[0061] In one embodiment, each frame of the target performance is assigned a weight (e.g., in the range 0.0 to 1.0). As stated above, each target frame receives a score based on the maximum score of all input frames scored against that target frame. In this embodiment, that score is multiplied by the weight to produce a weighted score for each target frame. The score for a move is determined by combining the weighted scores using a sum or average.

[0062] In one embodiment, each frame of the target performance is assigned a weight (e.g., in the range 0.0 to 1.0) that is computed based on the target performance. The weight for a frame of the target performance may be computed based on any number of neighboring frames of the target performance. The computation determines which target frames are the most important to the dance by detecting inflections in direction of parts of the target skeleton, or inflections in distance between parts of the target skeleton.

[0063] For example, the initial weight for a frame may be 0.0. A velocity vector can be computed for each joint in a target frame by subtracting its position in the previous frame from its position in the current frame. Whenever any joint's velocity experiences a derivative of zero with respect to time, along the x, y, or z axis in the camera-based coordinate system, or along the x, y, or z axis in the skeleton-based coordinate system (see below for a technique for computing a skeleton-based coordinate system), that frame's weight is increased. For example, if the weight of the target frame before considering the joint was w**0**, the new weight might be (1+w**0**)/2, or it may be set to a predetermined "one joint zero-derivative" value such as 0.5. If another joint's velocity simultaneously experiences a derivative of zero, the frame's weight is increased by substituting the previous weight into (1+w**0**)/2 again, or it may be set to a predetermined "two joint zero-derivative" value such as 0.75. Likewise, additional joints that experience simultaneous derivatives of zero make the current frame have a higher weight using the formula or a lookup table that references number of contributing joints to a weight value between 0.0 and 1.0.

[0064] Although derivatives of joint positions can be used to determine the weight for a frame of the target performance, other measurements can also contribute to the weight. For example, distances between specific joints can be computed for each frame and tracked across frames, and zero-derivative measurements can contribute to the weight. For example, the distance between wrist joints may be measured for each frame. Frames in which the distance experiences a zero derivative would increase the frame's weight by substituting its previous weight into (1+w**0**)/2 or looking up a value from a table as above.

[0065] Other measurements can also contribute to the weight, such as zero-derivative measurements of the overall bounding rectangle of the skeleton along x, y, or z axes in a camera-centered coordinate system or x, y, or z axes in a skeleton-based coordinate system.

[0066] However the target weight is computed, the final weight assigned to each target frame is used in the same way as described previously.

[0067] In a preferred implementation, a subset of the frames of the target performance are marked as keyframes, each keyframe representing a specific frame in the target performance with which the input performance should be compared. The target performance—comprising an entire dance routine—is aligned with a timeline, the performance being divided into moves, each move having a start time and an end time relative to the beginning of the dance, measured in units of measures/beats/ticks. Alternatively, each move can have a start time and a duration.

[0068] All times and durations are typically measured in units of measures, beats, and ticks, but alternatively can be measured in units of seconds. Times are measured relative to the beginning of the dance, but alternative reference points are possible, such as the end of the dance, the start of the previous move, the end of the previous move, or any other moment in time within the timeline.

[0069] Each keyframe includes a time offset relative to the beginning of the move. In addition to timing information, each keyframe can include weighting information for x, y, and z axes relative to the camera (explained below). Additionally or alternatively, each keyframe can include weighting information for x, y, and z axes relative to the entire skeleton in the target performance, or weighting information for x, y, and z axes relative to each "body zone" (limb-centered coordinate systems) in the target performance (explained below). In one implementation, relaxing the scoring is achieved by unevenly weighting the contributions of the x, y, and z axes to the Euclidean distance measurement above, where x, y, and z are taken to be in the left arm coordinate systems, right arm coordinate system, left leg coordinate system, or left leg coordinate system.

[0070] In addition to weighting information for the axes, the keyframe also includes weights for different bone groups themselves to emphasize performing a particular motion, e.g., moving the user's arms during the "shopping cart," or deemphasizing other motions one, e.g., ignoring or forgiving poor leg position during "the shopping cart".

[0071] Keyframes are placed wherever necessary on the timeline to capture the most important poses in the dance sequence. Often, keyframes are placed at eighth-note boundaries, but they may be spaced irregularly depending on the dance or move to be tested.

[0072] In a preferred embodiment, the target performance is expressed as mocap data associated with a Milo file. The Milo file contains a timeline and allows for events, tags, or labels to trigger events in the game. Advantageously, the target performance is aligned to the timeline. The Milo file is also typically associated with a music track, which is also aligned to the timeline. This allows the developer to assign events to certain portions of the music track. The Milo file also has instructional timelines for providing audio cues to the user (explained below). Another benefit of using the Milo file is the ability to mark parts of the timeline, and therefore parts of the target performance, as keyframes. Keyframes are placed at specific measures or beats on the timeline and represent times to test user input.

[0073] Comparing the input performance to the target performance input at a particular keyframe may be accomplished in several ways. In one embodiment, each keyframe has a time window associated with it, beginning before the keyframe and extending beyond it. The time window is typically symmetrical around the time of the keyframe, but may be adjusted for a longer intro if a move is difficult to get into or

a longer outro if the move is harder to get out of. The time window is typically of a fixed width in seconds. Alternatively, the time window can be expressed as fixed width in a variable unit of time such as beats, so that the window expands and contracts as the dance tempo slows down or speeds up, respectively.

[0074] FIG. **2**A provides an illustrative example. FIG. **2**A shows a series of movements spread over four beats that begin a move called "Push It." The first beat is a move marked "hands out", the second is a move marked "hands in," the third is a "right hand up", and the fourth is "left hand up" move. In FIG. **2**A, three keyframe windows are displayed, each centering on a beat: the first keyframe **200** is for the "Hands out" move at beat **1**, the second keyframe **205** is for the "Hands in" move on beat **2**, and the third **210** is for the "Right hand up" move on beat **3**. The user's input, sampled a certain number of times per second, e.g., 30, is examined to determine if it matches the target performance. For example, on beat **1** (and for a period before and after beat **1** illustrated by the umbrella around **200**) the user's input is sampled to determine if, in this case, the user's hands are stretched out in front of him in a way that matches the target input which is based on the mocap performance. Then, on beat **2** (and before and after), the user's input is sampled to determine if it matches the target performance where the user's hands are pulled back in. The windows around each keyframe are to allow for variation in time for the user to complete the move. Variation is allowed for in both time and positional displacement because rarely will the user have their limbs exactly in the expected position at exactly the right time. Additionally, as stated above, some leeway is provided because the camera is an inherently noisy input.

Allowing for Variation in Time

[0075] Referring to FIG. **2**B, if any of the user's inputs match the target performance within a certain inner time window around the keyframe, e.g., in the range to d_{-inner} to d_{+inner}, the user is given full score for performing that portion of the move that aligns with that keyframe (+/− to allow for the user to reach the move early or late, and the allowances either before or after are not necessarily symmetrical). This is accomplished by examining each frame of input during the window and selecting the closest match.

[0076] Between an inner time window and an outer time window, e.g., in the range d_{-outer} to d_{-inner} and the range d_{+inner} to d_{+outer}, a score is still given for performing the move, but the score for that performance is reduced as the temporal "distance" outside the inner window increases. Outside the outer windows, i.e., before d_{-outer} and after d_{+outer}, respectively, no score (or a score of zero) is given for performing the move because the user is just too early or too late. The fall off function for the score during the periods of d_{-outer} to d_{-inner} and d_{+inner} to d_{+outer} is typically a variation of $1-x^2$. This yields a parabolic shape that starts from 0 and builds to 1 between d_{-outer} and d_{-inner}, and then falls from 1 to 0 between d_{+inner} to d_{+outer}. More specifically, in one embodiment, the scoring curve is assembled piecewise:

For frames before d_{-outer}, $y(x)=0$.

For frames between d_{-outer} and d_{-inner}:

$$y(x) = 1 - \left(\frac{x - x_0 + d_{-inner}}{d_{-outer} - d_{-inner}}\right)^2.$$

For frames between d_{-inner} and d_{+inner} (including x_0), $y(x)=1$.

For frames between d_{+inner} and d_{+outer}:

$$y(x) = 1 - \left(\frac{x - x_0 - d_{+inner}}{d_{+outer} - d_{+inner}}\right)^2$$

For frames after d_{+outer}: $y(x)=0$.

[0077] But other variations are possible as well, e.g., a linear function, a constant, a parabolic function, a square-root, $1/x$, $1/(x^n)$ (e.g., inverse square, inverse cube, etc.), polynomial, exponential, logarithmic, hyperbolic, Gaussian, sine, cosine, tangent, or any combination or piecewise combination thereof.

[0078] Beneficially, in some embodiments, as shown in FIG. **2**A, the windows for keyframes can overlap, e.g., keyframe **205** overlaps **200**. In these cases, an input frame in the overlapping area is scored against both keyframes. The maximum score of all input frames that are scored against a given keyframe is assigned as the score for that keyframe. Any keyframe that the user can match, i.e., that his input falls within an umbrella for, is considered an "active keyframe" for that input frame.

Allowing for Variation in Position

[0079] As discussed above, the user's positional success is determined based on comparing the normalized input performance to the normalized target performance. When comparing the input performance to a keyframe (again, preferably done for each sampling of the input performance), the aggregate distance is taken between the two to determine how close the normalized input performance is to the normalized target performance of the keyframe. This can be done for the whole skeleton of the target performance or can be done on a limb by limb basis. Distances are calculated as the Euclidean distance between the normalized input performance's joint position in the input frame and the normalized target performance's joint position in the keyframe.

[0080] FIG. **3**A shows a distance determination between the target performance skeleton (shown in outline) and the user's input (shown solid). The distance between the user's elbow joint **300** and the target performance skeleton's elbow **305** is determined, reflecting the error the user is committing in terms of positioning his limb. If a filter is just testing elbow position, the analysis stops with comparing **300** and **305**. If the filter also tests wrist position, the distance is determined between the user's wrist position **310** and the target performance skeleton's wrist position **315**. As shown in FIG. **3**A, the user's elbow position is only slightly off the target performance's elbow, whereas the user's wrist significantly out of position. These differences are then used to determine how well the user is satisfying the filter. Although arms are shown in FIG. **3**A, differences between the user's leg and the target performance's leg are determined similarly.

[0081] For hips, hip velocity is a vector from the hip position in the previous keyframe to the hip position in the current keyframe. The vector is divided by the amount of time elapsed between the keyframes. To normalize the hip velocity, the

velocity vector is then divided by the length of the spine. Then the resulting vector is then used for Euclidean comparison similar to that described with respect to arms and legs. Advantageously, dividing by the length of the spine normalizes the velocity measurement to account for the size of the user, e.g., a child needs to displace his hips a smaller amount than a taller adult, in order to receive the same score.

[0082] In some embodiments, the total skeleton score is an aggregate (e.g., sum) of five different scores, i.e., left arm score, right arm score, left leg score, right leg score, and hip velocity score. These are each made up of score calculations themselves for the individual joints and represent how well the user performed the move for each "body zone". For example, the left arm score is an aggregate of the wrist score and elbow score, and the leg score is an aggregate of the knee score and ankle score. Beneficially, displacement of the body, measured by hip velocity, may also be incorporated into the score calculation. Also beneficially, contributions to the aggregate skeleton score by the aggregate body zone score may be weighted per keyframe to enhance the contribution from zones that are more important to executing the keyframe pose. For example, if the left arm is most important to a particular pose, the weight of its contribution to the score can be increased, or contributions of other body zones' scores can be decreased, or some combination thereof. Beneficially, contributions to aggregate body zone score by individual joint score may be weighted per keyframe, to enhance contribution from individual joint positions that are more important to executing the keyframe pose. For example, the elbow is more important than the wrist for the "Funky Chicken" pose, so the weight of the elbow joint's score can be increased, or the weight of the wrist joint score can be decreased, or some combination thereof. Typically though, if a user's joint or body zone is in the correct position, the user will be given full credit for the correct position and the weight of that limb's contribution will not be decreased.

[0083] Referring now to FIG. **3**B, like timing, there is a window of acceptable error for position. The error for position is determined based on the distance between the normalized input joint position and the normalized target joint position. If the distance is below a threshold (using the same convention as timing: d_{+inner}), e.g., 0.25 or less, the error is considered zero for that joint, so input frame receives a 100% score. If the distance is greater than the d_{+inner}, the score will fall off quickly as the distance increases to some outer boundary, d_{+outer}. Between d_{+inner} and d_{+outer}, the input frame still receives some score, but the further the scored limb or joint is from the target position, i.e., the closer it is to d_{+outer}, the less score the user receives. Once the joint's position is so far off position that the distance falls outside d_{+outer}, the user receives no score (or zero score) for that frame. Unlike timing errors, which may represent times before or after the keyframe and may therefore be positive or negative, distances are always positive.

[0084] The score of an input from for a particular keyframe is determined aggregating the positional score and the timing score. In a preferred embodiment, the positional score for an input frame compared against a particular keyframe is then multiplied by the timing score for that input frame to produce an overall score for the input frame for that keyframe. If the score for an particular input frame is greater than the score of any other input frame for a particular keyframe, i.e., that input frame is the "closest" to the keyframe in terms of the combination of weighted timing and position scores, that score is the assigned score for that keyframe and is used to determine the player's overall score for the move. When the user has satisfied a certain percentage of the filters for the bar, e.g., 80%, the user is considered to have successfully performed the entire move for that bar (because it is unlikely that a user will satisfy 100% of the filters). In implementations with graduated feedback (discussed below), completing 80% may be "Perfect," 60% may be "Good," 40% may be "Fair," and 20% may be "Poor."

Compensating for the Limits of the Camera and User

[0085] The present invention overcomes one limitation of the user's ability to parse input presented on the display. Certain movements of the on-screen dancer along the z axis (into and out of the screen) are difficult for the user to parse precisely. For example, when the avatar's arm is held out directly in front of its body, and the wrist is then moved closer to or further from the avatar's body along the z axis, the degree of that motion is hard to see from the user's perspective. This is problematic for a dance game because the game may require the user to replicate this movement, and the user cannot easily judge the distance well enough to execute the movement well.

[0086] In one implementation of the present invention, this is overcome by unevenly weighting the contributions of the x, y, and z axes to the Euclidean distance measurement above. This has the effect of "flattening" the error space in a dimension if that dimension is difficult to detect visually. This is typically expressed as a front-to-back relaxing of the scoring along the z axis, because movements in a camera-based system towards the camera (forward) or away from the camera (back) are the ones being compensated for. The relaxation of scoring along an axis is automatically provided by the invention by reducing the contribution along that axis by a coefficient in the Euclidean distance calculation. The developer may also specify, for a given keyframe, coefficients for one or more axis to reduce or enhance the contribution of error along that axis to the final score.

[0087] The present invention also overcomes the limitation caused by occlusion that is inherent to any camera-based input. When a dance move requires one or more parts of the body to be moved behind other parts of the body, the occlusion of the joints makes it very difficult to determine their positions with accuracy. This is problematic because joints can be occluded in normal dance moves, such as when an arm goes behind the back, or when a move requires the user to turn sideways to the camera.

[0088] The present invention additionally overcomes a limitation with a user attempting to reproduce the target performance when the mocap for the target performance was executed by a professional dancer who is very flexible. This is problematic because a professional dancer can place his body in positions that cannot be achieved by a casual user, and therefore the user cannot score well on the move. For example, a professional dancer can touch his elbows together behind his back, but it would be unfair to penalize a typical user for this lack of flexibility, so the scoring for these moves can be relaxed.

[0089] In one implementation of the present invention, relaxing the scoring is achieved by unevenly weighting the contributions of the x, y, and z axes to the Euclidean distance measurement above, where x, y, and z are taken to be in the mocap performer's frame of reference. The frame of reference of the mocap skeleton is computed per-frame as a rota-

tion about the z axis of the camera's frame of reference. The angle of rotation can be computed by finding the plane created by the shoulders and the center of the pelvis, finding the forward-facing normal, and rotating the frame of reference through the angle from the view plane normal to the forward-facing normal. Alternatively, the frame of reference of the mocap skeleton can be computed by starting with the plane created by both hips and the head.

[0090] In one implementation, relaxing the scoring is achieved by unevenly weighting the contributions of the x, y, and z axes to the Euclidean distance measurement above, where x, y, and z are taken to be in the left arm coordinate systems, right arm coordinate system, left leg coordinate system, or left leg coordinate system.

[0091] One the frame of reference has been rotated, relaxing scoring along an axis has the effect of "flattening" the error space in a dimension. For example, if a move requires the elbows to be pulled back very far, relaxing scoring along the z axis in the frame of reference of the mocap performer will reduce the distance the elbows need to be pulled back in order to achieve a good score. The relaxation of scoring along an axis is specified with the keyframe information as coefficients for the Euclidean distance calculation.

[0092] Beneficially, the game developer can manually weight certain moves to be more forgiving along any axis simply because a move is hard to perform.

[0093] In some implementations, weighting is based on the "confidence" that the camera system may provide for detecting a joint's position. For example, in some versions of Project Natal, the camera system provides "tracked" positional information in the form of a position for a joint and a confidence level that the position is correct. When the joint is off-screen, Natal also provides an "inferred" position. When a joint's position is inferred, e.g., when the joint is clipped or occluded, neighboring joints can be examined to better assess where the inferred joint is. For example, if an elbow is raised above the user's ear, there are only a few possible locations of the user's wrist, e.g., straight up above the elbow, down near the user's chin, or somewhere in between. In these scenarios, because the object of the game is to be fun, the maximum positional window, e.g., 0 to d_{+outer}, is widened so that the filtering is looser to allow for greater variation in positional differences. Additionally, the inner window of "perfect" position, zero to d_{+inner}, may also be widened.

[0094] In some embodiments, the invention will suspend the game if too much of the skeleton is occluded or off-screen for more than a threshold amount of time, e.g., 10 second, or 6 beats, rather than continuing to reward the user for incorrect positioning.

[0095] To assist the user in completing moves correctly, per-limb feedback is given to the user when performing a move. In some embodiments, if the user is not satisfying a filter for a limb, the game renders a red outline around the on-screen dancer's corresponding limb to demonstrate to the user where they need to make an adjustment. In some embodiments, the per-limb feedback is on the mirror-image limb from the limb that is not satisfying the filter. For example, if the user is satisfying the filter for both feet, the hips, and the left arm, but not satisfying the filter for the right arm, the game renders a red outline around the on-screen dancer's left arm. This indicates to the user that his right arm is not correct, since the user is facing the on-screen dancer and mimicking the on-screen dancer in mirror image.

[0096] Other per-limb feedback is also possible. In some embodiments, an indicator such as a "phantom" limb is drawn in the target location. Alternatively or additionally, an indicator is anchored on the errant limb and its direction and length are based on the direction and degree of error in the user's limb position. For example, if the user's wrist is below the target location, the game draws an arrow starting from where the user's wrist is located in the input performance and ending where the on-screen dancer's wrist is in the target performance. Alternatively, in embodiments where a representation of what the user is doing is displayed on-screen, the arrow is drawn starting from the user representation's wrist. In some embodiments, the indicator persists until the user satisfies the filters for the target performance's arms. In some embodiments, the intensity, geometry, material, or color characteristic of the indicator may be changed based on the degree of error for that limb. For example, the color of the indicator may become a more saturated red if the error for a limb becomes greater. Other highlighting may also be used, as may verbal cues such as "get your <limbs> movin'" where <limbs> is any body zone that is not satisfying the filter.

[0097] In some embodiments, there is an additional indicator showing how well the user is cumulatively satisfying all filters in a move, such as a ring of concentric circles under the on-screen dancer's feet. If the user has satisfied a certain percentage of the filters, e.g., 20%, the inner ring of circles is illuminated. When the user successfully performs the next threshold percentage of filters, e.g., 40%, the next set of rings is illuminated. This is repeated such that when the user has successfully performed the entire move, the outermost set of rings is illuminated. A notable side effect is that as the user is satisfying filters, the ring grows under the on-screen dancer's feet. In some embodiments, the success indicator moves with the on-screen dancer, e.g., is based on the position of the mid-point of the pelvis of the skeleton of the target performance, so that the user does not have to look at a different part of the screen to determine how well he is performing. While described in terms of discrete rings, the effect can occur continuously. Also, other shapes or graphical effects may be used, e.g., a meter indicating how many filters are satisfied, and bigger and bigger explosions or fireworks may be displayed to indicate the user satisfying more and more filters. Beneficially, in some embodiments, a qualitative evaluation is also displayed, e.g., good!, great!, or awesome!

[0098] Beneficially, the setting of the game may react to changes in the user's performance. For example, as the user is satisfying filters, a crowd of spectators may begin to circle or gather near the on-screen dancer. Or the venue in which the on-screen dancer is performing may become brighter, more colorful, or transform into a more spectacular, stimulating, or elegant venue. Correspondingly, if the user is performing poorly, on screen crowds may dissolve and walk away or the venue may become darker, less colorful, or transform into a less spectacular, stimulating, or elegant venue. Changes in venue and setting can based on the consecutive number of moves completed, e.g., after five successful moves the venue and dancers on screen change to an "improved mode." After ten successful moves the venue and dancers may change to a "more improved mode" and so forth. Changes in venue and setting can also be based on the overall score of the input performance, or on the overall score of the input performance as compared to an average performance.

Dance Training

[0099] In some implementations, there is a trainer mode to assist the user in learning a dance. In trainer mode, a dance

move is demonstrated using the on-screen dancer and audible cues and no score is kept. The user is then expected to mimic the on-screen dancer's movements. If the user performs the move correctly, an indicator indicates he has performed the move correctly, the next move is demonstrated, and the user may continue practicing. If the user does not perform the move correctly, the move is repeated and the user must keep trying to perform the move before he is allowed to continue.

[0100] When the user does not perform the movement correctly, additional instruction is provided. In some embodiments, a verb timeline, normal_instructions, runs simultaneously with the target performance, and has multiple verb labels indicated on it. The verb labels refer to pre-recorded audio samples that have both waveform data and offsets. The offset indicates where the stress—or important accent—is located in the waveform data. For example, if the wave form data represents the spoken word "together," the offset indicates the first "e" sound such that playback of "together" begins before the point of the verb label on the timeline and the playback of the "e" sound aligns with the point of the verb label on the timeline. This allows the developer to specify which point on the timeline a particular syllable of the audible cue falls on. As the target performance is displayed, the waveform data is played back according to the positions of the verb labels and the offsets to provide instruction to the user that is synchronized with the movement of the on-screen dancer.

[0101] In some embodiments, a second verb timeline, slow_instructions, runs simultaneously with the target performance and may have a different or more detailed set of verb labels indicated on it. These verb labels also refer to pre-recorded audio samples with waveform data and offsets, similar to those described above. When the user cannot successfully perform a particular move after a threshold number of attempts, the game slows down and the slow_instructions timeline is used to provide additional, more detailed instruction to the user. For example, on the normal_instructions timeline, there may be a verb label that refers to an audio cue of "step and clap." On the slow_instructions timeline, this may be represented by three labels, "left foot out," "right foot together," and "clap." When the game is slowed down, rather than referencing verb labels on the normal_instructions timeline to trigger audio cues, the game references the verb labels on slow_instructions timeline. Beneficially, when the game is slowed down, there is enough time between body movements that the additional instructions can be played. In some implementations, the slowed down audible cues are stored in a different file or a different audio track than the normal speed audible cues. When the user has successfully reproduced the move, the game is sped back up and the normal_instructions timeline is used, or alternatively, the additional instructions are muted or not played.

Fitness Mode

[0102] In some embodiments, there is a calorie counter displayed on the display during the dance game to encourage users to dance. As the user dances, the calorie counter is incremented based on the Metabolic Equivalent of Task ("MET", and generally equivalent to one kcal/kg/hour) value of what the user is doing. As an example, sitting on the couch has a MET value of 1. Dancing and most low impact aerobics have a MET value of approximately 5. High impact aerobics has a MET value of 7. To determine the MET for a frame of input skeleton data, the joint velocities for all joints on the user's input skeleton are summed. To determine a joint's velocity, the joint's position (in three dimensional space) in the previous frame is subtracted from its position in the current frame. This yields a vector. The vector is divided by the elapsed time between the previous frame and the current frame. The length of the resulting vector is the velocity of that joint.

[0103] Once the sum is determined, it is exponentially smoothed to reduce transient noise. The result is a mapped to a MET scale of 1 to 7 with, in some embodiments, a sum of 0 mapping to 1 and a sum of 40 mapping to 7, with 1 representing no movement and 7 being a large or vigorous movement. Beneficially, any sum less than five can map to 1 to account for the noise inherent in the input. The mapping can be linear, piecewise linear, or any interpolation function. Using the MET value, and knowing the user's body weight (which can be input via a menu, or can be inferred based on the camera's input and a body/mass calculation), calories burned can be estimated.

[0104] METs are converted to calories-consumed-per-second using the equation of (METs*body weight in kilograms)/seconds in an hour=calories/second. This value can then be displayed on the screen, or summed over time to produce a value displayed on the screen for total calories. The value for calories/second or total calories can stored as a "high score" and, in some embodiments, can be used to increase or decrease the tempo of a song or the difficulty of a series of moves. Advantageously, this allows the user to track total calories burned, average rate burned, and other statistics over time.

Reusing Elements of a Mocap Performance

[0105] In some embodiments of the dance game, the most difficult or complex target performance is recorded as one linear mocap session and only parts of the recorded performance are used to simulate easier versions of the performance. For example, in FIG. **4**, the most difficult or "expert" dance routine comprises a series of movements following pattern of A, B, C, D, A, B, D, C. In some embodiments, these moves are marked on the expert timeline using "move labels," which each denote the name of a move animation and where in the timeline the move animation begins. In other embodiments, these moves are marked on a timeline that parallels the expert timeline, called "anim_clip_annotations." Rather than capture multiple target performances for each difficulty level, e.g., a dance with the previous pattern for "expert," and progressively simpler sequences for "hard," "medium," and "easy," the game can re-use the motion capture recorded for expert to simulate a pattern for any of these difficulty levels by referring to the move labels on the expert timeline. For example, given the expert sequence above, the easy sequence might be A, B, A, A, A, B, A, A. In other words, for the easy routine, a repetition of the A move replaces both the C and D moves.

[0106] The easier routines can be created programmatically, e.g., the game determines how often to repeat a movement based on a difficulty value for the move, favoring easier moves for easier difficulty levels. The easier routines can also be authored by the game developer by creating an "easy" timeline and referencing the move labels on expert track. An example of this is the "easy" track in FIG. **4**, where the A sections reference the A move in the expert track and the B sections reference the B move. C and D sections, that involve a more complicated knee raise (C) and knee slap (D), are

omitted from "Easy" so the user only needs to repeat the "arms out" move of A or "arms up" move of B.

[0107] Reusing moves allows space savings on the storage medium (only one target performance needs to be stored) and it allows the game developer to later change the performances of the other difficulties after the game is released if it is later determined that the performance for a difficulty setting is too hard or too easy or is boring. Since the expert performance is linear, each A section in expert will be slightly different because the mocap actor likely did not have his limbs in the exact same position every time. Examples of this are A' and B' where the skeletons are similar to A and B respectively, but the arm positions are slightly different. To make an easier difficulty target performance, the A move that is repeated in the easier difficulties can be A or it can be A', or some combination. In some embodiments, a move that is repeated in an easier difficulty uses the most recent version of that move in the timeline. In some embodiments, a move that is repeated in an easier difficulty uses the earliest version of that move that appeared in the routine. Beneficially, the animations from the expert track can also be reused when creating the "easy" performance.

[0108] A sequence of moves for an easier routine may correspond to a sequence of moves in the original expert linear mocap such that a specific pattern of moves is present in both (although they may not correspond on the timeline). In this case, the sequence of moves may be copied from the expert performance into the desired position in the easier routine's timeline. But if a sequence of moves for an easier routine does not correspond to a sequence of moves in the original expert linear mocap, individual moves may be separately copied from the expert performance into the desired position in the easier routine's timeline. Beneficially, copying larger sequences of moves from the linear mocap produces sequences with fewer animation artifacts.

Animation Blending

[0109] When moves or sequences of moves are used in easier difficulties, the moves can abut other moves that were not adjacent in the linear mocap. The transitions in the move animations between these moves can be jarring, since the skeleton in the last frame of one move can be in a completely different pose than the first frame of the next move, which would produce a sudden, nonlinear animation. Animation blending can be used to transition smoothly from the end of one move to the beginning of the next move in the sequence, if the two moves were not adjacent in the linear mocap. Using the example above of an expert performance following the pattern of A, B, C, D, A, B, D, C, when creating the easier difficulty performance, there may be a pattern of A, A that is not part of the linear mocap. Animation blending is used to transition from the end of the first A animation to the beginning of the same A animation to produce an A, A pattern. In one embodiment, the last beat of the move before an animation transition is blended with the beat before the beginning of the next move. In the example of the A, A pattern, the last beat of the A move is blended with the beat before the A move for the duration of one beat. Then the animation continues with the first beat of the second A move.

[0110] In some cases, the animation blending technique described above produces animations that are still jarring. This is often due to the large differences between the pose at the end of one move and the pose at the beginning of the next move, that can't be overcome through simple blending. In these cases, the animation can appear to jerk from one position to another during the transition, or to move in a way that's physically impossible. In some embodiments, additional mocap is recorded to produce bridge animation segments. A bridge animation segment is designed to make the transition between two other animations smooth. For example, using the example above, if the end of the A move was a very different pose than the beginning of the A move, a simple animation blend might produce a poor result. An A, A bridge animation segment would be recorded, wherein the actor would actually perform the transition from the end of the A move to the beginning of the A move. In one embodiment, the bridge animation segment is three beats long. The next-to-last beat of the first A move is blended with the first beat of the bridge animation segment in such a way that contribution from the bridge animation segment is interpolated linearly over the course of the beat from 0% to 100%. The second beat of the bridge animation segment is played without blending, then the first beat of the second A move is blended with the third beat of the bridge animation segment in such a way that the contribution from the bridge animation segment is interpolated linearly over the course of the beat from 100% to 0%. The bridge animation segment may be any number of beats long, for example two beats, and the blending can also be done over the course of any number of beats, for example two beats. The interpolation may be done in a way that is not linear, such as parabolic, inverse-squared, etc.

[0111] In some embodiments, a table is provided that is keyed by the start and end move labels associated with two animations that may abut. If a bridge animation segment is required to produce a smooth transition between the associated animations, the table will contain an entry indicating the bridge animation segment that should be used. This table is consulted for all pairs of animations that are displayed.

[0112] Beneficially, the move animations and the results of the animation blending, e.g., from A to A, or from prior move to first A or from second A to next move, can be used as the target performance, and can therefore be scored similarly to the normal gameplay performance. This provides a fluid game experience and rewards users that accurately mimic the dancer on the screen.

[0113] In a training mode, it is often necessary to isolate and repeat a move or series of moves, with a gap in between the repetitions. For example, when demonstrating the A move, it is useful for the game to count in the beat while the animation is in an idling state, then execute the move animation, then return to an idle animation. This can be accomplished in a way that is similar to the bridge animation segments described for gameplay above. In one embodiment, a three beat bridge animation segment of the transition from an idle state to the first beat of a move is recorded as mocap data. This is blended with the idle animation and move animation as described above.

[0114] FIG. **5** shows one embodiment of an authoring system for the dance game. In FIG. **5**, the keyframes **500** are depicted with their respective timing umbrellas. Each body zone being tested **505** is shown as having a corresponding portion of the filter to be satisfied (each square in the rectangle **510**). The move is completely satisfied when all body zone filters are satisfied (although in some difficulty settings, only a percentage of the body zone filters need to be satisfied). The labels **515***a*, **515***b*, **515***c* (Hip_Hop_Break.move, Arm_Twist_R.move, and Arm_Twist_L.move, respectively) applied to each move are shown on the timeline **520**. As stated

above, these labels can be reused to create easier dance routines based on the mocap recording. The mocap skeleton **525** shows the desired joint movements, and the input skeleton **530** shows what the user is currently inputting. Look-ahead icons show the user what move is coming next, e.g., Arm Twist, and the current move icon **535** is displayed prominently. The dancer **540** on screen is a representation of what the user is supposed to input and the skeleton of the on-screen dancer **540** resembles that of the mocap skeleton **525**.

Determining an Active Player with Multiple Skeletons Available

[0115] When more than one player is within the field of view of the camera, the system must determine which player is the active player, and which player is the inactive player, for the purposes of shell navigation and gameplay.

[0116] For this discussion of determining the active player, it is useful to define two terms. A skeleton is considered "valid" if it is not sitting and it is facing the camera. Also, "queuing a skeleton for activation" means setting a timer to go off at particular time, at which point the active skeleton is set to be inactive and the queued skeleton is set to be active.

[0117] In some embodiments, queuing a skeleton for activation does not set a timer if that skeleton is already queued for activation. In some embodiments, queuing a skeleton for activation does not set a timer if any skeleton is already queued for activation. In some embodiments, the timer is always set for 1 second in the future.

[0118] In some embodiments, determining the active player begins when a frame of skeleton data is received by the system. In some embodiments, a frame of skeleton data is received and processed every thirtieth of a second. In each frame, there may be any number of distinct skeletons in the skeleton data. At any time, one of the skeletons in the skeleton data is considered active, and the rest, if any, are considered inactive.

[0119] In some embodiments, if the active skeleton is behind—further from the camera than—an inactive skeleton, or the active skeleton is near the edge of the camera's view, then the system can search for an inactive skeleton to activate. In some embodiments, the active skeleton is considered near the edge of the camera's view if its centerline is in the left or right fifth of the camera's view. If there is an inactive skeleton nearer to the center of the camera's view than the active skeleton, the inactive skeleton can be queued for activation.

[0120] In some embodiments, if an inactive skeleton that is queued for activation is not present in the current frame, or is not valid, or is crossing its arms, or is behind the active skeleton, the queued activation of that skeleton is cancelled. In some of these embodiments, the queued activation of the inactive skeleton is not cancelled if the active skeleton is near the edge of the camera's view.

[0121] In some embodiments, if the active skeleton is not in the frame, or if the active skeleton is invalid, but there is at least one inactive skeleton, the system immediately activates one of the inactive skeletons.

[0122] In some embodiments, if an inactive skeleton's hand is raised and the active skeleton's hand is not raised, the inactive skeleton is queued for activation or scoring for dancing. Beneficially, this allows a user to express intent to control the shell or have their performance be the one that is graded by raising their hand.

Multi-Player Modes—Animation

[0123] A dance game can be more satisfying if it provides multi-player competitive or cooperative game modes. One difficulty that arises is that the original song and the choreography for the song may not be balanced such that two players can have equal opportunities to contribute to their competing or combined scores (for competitive and cooperative modes, respectively). In addition, the song may be too short to give either player sufficient opportunity to perform for a satisfying duration.

[0124] In one embodiment, the invention addresses these shortcomings by artificially extending the song and its choreography by looping back to previous parts of the song to give multiple players an opportunity to dance the same section. Beneficially, this provides the same potential scoring for all players in a multi-player mode. Although animation blending in this context is primarily intended for looping back to previous parts of a song, the mechanism applies equally well to any non-contiguous jump between points in the song, or jumps between jumps points in more than one song.

[0125] In one embodiment, a section that is to be repeated in multi-player mode is indicated in a MIDI file, in a track called multiplayer_markers, aligned with the audio timeline. Alternatively, the markers can be located in the same MIDI track as other MIDI data, or can be indicated across multiple MIDI files, in respective tracks called multiplayer_markers, or can be located in the same MIDI track as other MIDI data, spread across multiple MIDI files. The section indicators are special multiplayer text events, MP_START and MP_END. During gameplay, when the game time reaches the time of the MP_END text event the first time, the game time jumps to MP_START and the other player begins play. When the game time approaches the time of MP_END the second time, it continues without jumping.

[0126] In one embodiment, when the game jumps to a non-contiguous point in the song, for example to the point designated by MP_END, animation blending can be used, as described above for creating easier difficulties, to make the transition less jarring. For example, if it is determined that a single section should be repeated, the animation of the last beat of the section can be blended with the animation the beat before the beginning of the first beat of the section. The animation blending can take place over two beats, or it can extend over multiple beats. In all cases, the animation for the end of the section is blended with the animation before the beginning of the section such that the blend begins with 100% contribution from the end of the section and ends with 100% contribution from before the beginning of the section. The interpolation can be linear, or can use any other interpolating function such as polynomial.

[0127] As in animation blending for easier difficulties, the blend from the end of a section to the beginning of the section can produce an unrealistic movement. In this case, bridge animation segments can be used, as discussed above regarding producing an easy difficulty.

Multi-Player Modes—Audio

[0128] Extending a song by looping back to previous sections brings with it some inherent difficulties in animation. The invention addresses these difficulties using animation blending and bridge animations. Non-contiguous jumps in the timeline of the song, or jumps between songs, also cause difficulties with continuity of the audio track. As with animation, the audio for the end of a section does not always merge smoothly into the audio for a section that is not adjacent in the song's timeline. Jarring discontinuities in the audio track can interfere with the users' enjoyment of multi-player modes.

The invention provides seamless audio track transition playback during multi-player modes to address this difficulty. For example, if the audio follows the sequence of sections A, B, C, it may be desirable in a multiplayer mode to loop from the end of the B section back to the beginning of the B section. The invention allows this extension to happen seamlessly.

[0129] In one embodiment, a section that is to be repeated in multi-player mode is indicated in a MIDI file in a track called multiplayer_markers, with MP_START and MP_END text events, as described above. In the example above, an MP_START text event in the MIDI file would be aligned with the beginning of the B section, and an MP_END text event would be aligned with the end of the B section, indicating that the entire B section is to be repeated in multi-player mode. Alternatively, a section that is to be repeated in multi-player mode can be indicated across multiple MIDI files, in respective tracks called multiplayer_markers, or can be located in the same MIDI track as other MIDI data, spread across multiple MIDI file.

[0130] In one embodiment, when there will be a transition from one part of the song to a non-adjacent part of the song, the audio track for a period of time before the origin of the transition is blended with the audio track for the same duration before the target of the transition, or the audio track for a period of time after the origin of the transition is blended with the audio track for the same duration after the target of the transition, or some combination. This is similar to how animations are blended when producing an easy difficulty. For example, one beat worth of audio before the MP_END event could be blended with one beat worth of audio before the MP_START event, then one beat worth of audio after the MP_END event could be blended with one beat worth of audio after the MP_START event. The blending is done such that at the beginning of the blend, the contribution from the audio before the MP_END event is 100%, and at the end of the blend, the contribution of the audio from after MP_START is 100%. This can be a linear crossfade, or it can use any other interpolating function, such as polynomial.

[0131] In some cases, as with animation blending, the result of audio blending is still jarring. This is often due to the discontinuity in the harmonic progression of the song when moving to a different place in the music, or presence or absence of vocal or instrument parts before or after the transition. In some embodiments, as with bridge animation segments, additional audio is recorded to produce waveform data for a bridge audio segment. The bridge audio segment is designed to make the audio transition between two non-adjacent parts of the song sound smooth. Using the example above with sections A, B, and C, if the game will repeat section B, a bridge audio segment can be provided that smoothly transitions from the last part of section B into the first part of section B.

[0132] In one embodiment, the waveform data for bridge audio segments are included in one or more additional bridge audio tracks in the multi-track audio data, and the bridge audio tracks are muted unless non-sequential looping is taking place. However, each bridge audio segment could be located in its own file referenced by the game authoring, or all bridge audio segments could be located in a single file, and the offset and duration of each segment of bridge audio in the single file would be stored as unique text events in the MIDI file.

[0133] In some embodiments, all bridge audio segments are of a fixed duration in beats, with a fixed number of beats before the transition. In these embodiments, the original song audio is played until a fixed amount of time in beats before the end of the transition. Then the original song audio track or tracks are muted, and the bridge audio segment is played until the transition point. Then the "current time" is moved to the target of the transition and the remainder of the bridge audio segment is played. At this point, the bridge audio track is muted and the original song audio track or tracks are unmuted. For example, all bridge audio segments might be three beats long, with one beat before the transition. Using the example above with sections A, B, and C, if the game will repeat section B, a 3-beat-long bridge audio segment from the end of B to the beginning of B may be provided. One beat before end of B, the original audio tracks are muted and the B-to-B bridge audio segment is played. When the end of B is reached, the current time is moved to the beginning of B, and the bridge audio segment continues playing for two more beats. After two beats, the bridge audio track is muted and the original tracks are unmuted. Beneficially, aligning the audio and changing the current time in this way allows for a single, consistent timeline for audio playback, animation, and other aspects of gameplay. Alternatively, the current time may be changed at the end of the bridge audio segment's playback, and moved directly to two beats after the beginning of B section. This example discusses bridge audio segments that are all 3 beats long, which start playing one beat before the transition, but other embodiments may have bridge audio segments that are all longer or shorter, or that all begin earlier or later with respect to the transition.

[0134] In some embodiments, the song audio and bridge audio segments may be muted and unmuted, as described. Alternatively, the song audio and bridge audio segments may be mixed, such as by lowering the normal song audio volume to 10% and playing the bridge audio segment at 90%. It is also possible to cross-fade the song audio and bridge audio segments. For example, the last beat of the B section would start with 100% of the song audio and end with 100% of the bridge audio segment, then the bridge audio segment would play at 100%, then the second beat of the B section would start with 100% of the bridge audio segment and end with 100% of the second beat of the song audio. The interpolation can be linear, but it can also use any other interpolating function, such as polynomial.

[0135] In some embodiments, as described above, the bridge audio segments can be of a fixed duration in beats or seconds. In other embodiments, each bridge audio segments can be of different durations. Beneficially, the ability to specify bridge audio segments of different durations makes it easier to provide a musically seamless transition, using more time if necessary, to achieve the proper harmonic and orchestration transitions, and less if possible, so that the playback departs as little as possible from the original music.

[0136] In one embodiment, all the waveform data for bridge audio segments is located on a single bridge audio track, bridge_audio, in the multi-track audio data file. The bridge audio waveform data for a given transition is divided into the sub-segment before the transition and the sub-segment after the transition. The sub-segment before the transition is positioned in the bridge_audio track so that it ends exactly at the transition point, corresponding to the MP_END text event in the associated MIDI file. The sub-segment after the transition is positioned in the bridge_audio track such that it begins exactly at the target of the transition, corresponding to the MP_START text event in the associated MIDI file.

[0137] In some embodiments, where the bridge audio segments are of a fixed duration, the beginning and end of the bridge audio segments is implicit in the fixed duration and the fixed amount of time before the transition, as described above.

[0138] In the preferred embodiment, the specification of the beginning and end of bridge audio segments is provided in a MIDI file, in the multiplayer_markers track, although the beginning and end of the bridge audio segments could be in their own MIDI track, or in their own MIDI file whose timeline is aligned with the audio timeline. In the multiplayer_markers track, special multiplayer text events, MP_BRIDGE_START and MP_BRIDGE_END, denote the beginning and end of a bridge audio segment. As the game is played in a multi-player mode, when an MP_BRIDGE_START text event is encountered on the timeline of multiplayer_markers, the original audio track or tracks are muted and the bridge_audio track is unmuted. As described above, attenuation of the original track or crossfading with the bridge audio track can be used instead of muting and unmuting. Playback continues until the transition point itself, which is indicated by the MP_END text event. At this point, the "current time" is set to the target of the transition, marked by the MP_START text event, and the bridge audio track continues. When the MIDI MP_BRIDGE_END event is encountered, the original audio track or tracks are unmuted, and the bridge audio track is muted. Note that when the transition is backwards in time, the MP_BRIDGE_END event occurs earlier on the timeline than the MP_BRIDGE_START event, since the current time is modified between them. Beneficially, dividing the bridge audio segments and modifying the current time at the transition point as described allows there to be a single concept of current time for the audio, animation, and gameplay. In other embodiments, the current time is modified only after the playback of the bridge audio segment is complete, and at that point it is set to the location of MP_START plus the length of the second sub-segment of the bridge audio segment. As described above, a section that is to be repeated in multi-player mode also can be indicated across multiple MIDI files, in respective tracks called multiplayer_markers, or can be located in the same MIDI track as other MIDI data, spread across multiple MIDI file.

Additional Variations

[0139] The examples given herein of a user satisfying a filter by completing a series of moves can be adapted to satisfy a "mirror mode" as well, where the user provides input that mirrors the target performance, e.g., providing input using a right hand when the target performance uses a left hand, providing right leg input when the target performance uses a left leg, and so forth.

[0140] Additionally, where a target performance skeleton is provided, it can be generated beforehand, or can be generated during execution of the game based on the motion capture data.

[0141] Any system that can detect movement can be used as long as positions of the scored joints can be determined in either two-dimensional space or three-dimensional space to create or simulate a skeleton. For two-dimensional implementations, scoring is typically adjusted to compare the projection of the target performance and the projection of the input performance onto a plane parallel to the screen. Although the system and technology has been described in terms of a camera input system like Natal, camera systems that utilizes sensors on the user's body, e.g., PLAYSTATION® Move, or systems that use sensors held in the user's hand, e.g., the NINTENDO® Wii, may also be utilized. In those implementations where only hand held sensors are utilized by the user, testing for leg input is ignored or not performed.

[0142] Although the embodiments described herein use dancing as an example, and the performance is typically accompanied by a song, the performance can also be movements that occur on a timeline with no musical accompaniment, e.g., a series of yoga poses, movements in a martial arts kata, or the like.

[0143] In some implementations, the mocap data is mapped to a skeleton similar to that used to reflect the user's input. Thus, the mocap data is used to generate an ideal skeleton that represents a performance of the dance routine in a format that is directly comparable to the skeleton representing the user's input. Then, during the game, as the user provides input, the user's skeleton is compared to the ideal skeleton, in effect normalizing the target input (the target performance) and actual inputs (the user's performance) to the same frame of reference, i.e., both performances are expressed in terms of the same skeleton-based technology.

[0144] In some embodiments, rather than matching position necessarily within a time window as described above, filter types are predefined and used to test user input. For example, proximity filters tests if a joint in a particular position, or close to a particular other joint, e.g., "are the left wrist and right wrist less than, greater than, or within a delta of a certain distance of one another. Another filter is a displacement filter which tests if a joint has moved a certain distance between times t_0 and t_n. Another example is the angle filter, which tests if a joint is at a particular angle from the origin. One or more of these filters is then hand-inserted (or "authored") into the timeline and bound to joints such that at a particular time, the condition is tested, e.g., "has the RIGHT WRIST moved from x_0 to x_n since I began tracking it?" would be a displacement filter. If the user's wrist had, the filter would be satisfied. Yet another filter is an acceleration filter which tests if a joint or bone has accelerated or decelerated between times t_0 and t_n. An acceleration filter can also test whether the magnitude of the acceleration matches a predetermined value.

[0145] In these embodiments, multiple filters can be overlaid on the timeline, and tested, in effect, simultaneously. An overall score for the frame is determined based on contributions from all of the active filters during a given frame. The filters can output a Boolean, and the score is computed from those. Or—in some implementations—the outputs are continuous, and the aggregate score is computed from those. Similar to the system described above, contributions from each active filter can be weighted differently in their contributions to the score. For Boolean filters, successfully completing 3 out of 5 filters gives the user a score of 0.6. In some implementations, each keyframe comparison gives a percentage credit for the move as a whole being correct. The user's score may be adjusted based on the aggregate score for a keyframe. Or the aggregate score for a keyframe may be quantized into groups, each group being compared to one or more thresholds, each group associated with a score that is added to the user's score. In any of these, if the user achieves a threshold score for a move, where if the user meets or exceeds the threshold, e.g., 80%, the user is considered to have successfully performed the move.

[0146] In some embodiments, execution of game software limits the game platform **120** to a particular purpose, e.g., playing the particular game. In these scenarios, the game platform **120** combined with the software, in effect, becomes a particular machine while the software is executing. In some embodiments, though other tasks may be performed while the software is running, execution of the software still limits the game platform **120** and may negatively impact performance of the other tasks. While the game software is executing, the game platform directs output related to the execution of the game software to a display, thereby controlling the operation of the display. The game platform **120** also can receive inputs provided by one or more users, perform operations and calculations on those inputs, and direct the display to depict a representation of the inputs received and other data such as results from the operations and calculations, thereby transforming the input received from the users into a visual representation of the input and/or the visual representation of an effect caused by the user.

[0147] The above-described techniques can be implemented in digital electronic circuitry, or in computer hardware, firmware, software, or in combinations of them. The implementation can be as a computer program product, i.e., a computer program tangibly embodied in a machine-readable storage device, for execution by, or to control the operation of, data processing apparatus, e.g., a programmable processor, a computer, a game console, or multiple computers or game consoles. A computer program can be written in any form of programming language, including compiled or interpreted languages, and it can be deployed in any form, including as a stand-alone program or as a module, component, subroutine, or other unit suitable for use in a computing environment. A computer program can be deployed to be executed on one computer or game console or on multiple computers or game consoles at one site or distributed across multiple sites and interconnected by a communication network.

[0148] Method steps can be performed by one or more programmable processors executing a computer or game program to perform functions of the invention by operating on input data and generating output. Method steps can also be performed by, and apparatus can be implemented as, a game platform such as a dedicated game console, e.g., PLAYSTATION® 2, PLAYSTATION® 3, or PSP® manufactured by Sony Corporation; NINTENDO WII™, NINTENDO DS®, NINTENDO DSi™, or NINTENDO DS LITE™ manufactured by Nintendo Corp.; or XBOX® or XBOX 360® manufactured by Microsoft Corp. or special purpose logic circuitry, e.g., an FPGA (field programmable gate array) or an ASIC (application-specific integrated circuit) or other specialized circuit. Modules can refer to portions of the computer or game program and/or the processor/special circuitry that implements that functionality.

[0149] Processors suitable for the execution of a computer program include, by way of example, special purpose microprocessors, and any one or more processors of any kind of digital computer or game console. Generally, a processor receives instructions and data from a read-only memory or a random access memory or both. The essential elements of a computer or game console are a processor for executing instructions and one or more memory devices for storing instructions and data. Generally, a computer or game console also includes, or be operatively coupled to receive data from or transfer data to, or both, one or more mass storage devices for storing data, e.g., magnetic, magneto-optical disks, or optical disks. Data transmission and instructions can also occur over a communications network. Information carriers suitable for embodying computer program instructions and data include all forms of non-volatile memory, including by way of example semiconductor memory devices, e.g., EPROM, EEPROM, and flash memory devices; magnetic disks, e.g., internal hard disks or removable disks; magneto-optical disks; and CD-ROM and DVD-ROM disks. The processor and the memory can be supplemented by, or incorporated in special purpose logic circuitry.

[0150] To provide for interaction with a user, the above described techniques can be implemented on a computer or game console having a display device, e.g., a CRT (cathode ray tube) or LCD (liquid crystal display) monitor, a television, or an integrated display, e.g., the display of a PSP®, or Nintendo DS. The display can in some instances also be an input device such as a touch screen. Other typical inputs include a camera-based system as described herein, simulated instruments, microphones, or game controllers. Alternatively input can be provided by a keyboard and a pointing device, e.g., a mouse or a trackball, by which the user can provide input to the computer or game console. Other kinds of devices can be used to provide for interaction with a user as well; for example, feedback provided to the user can be any form of sensory feedback, e.g., visual feedback, or auditory feedback; and input from the user can be received in any form, including acoustic, speech, or tactile input.

[0151] The above described techniques can be implemented in a distributed computing system that includes a back-end component, e.g., as a data server, and/or a middleware component, e.g., an application server, and/or a front-end component, e.g., a client computer or game console having a graphical user interface through which a user can interact with an example implementation, or any combination of such back-end, middleware, or front-end components. The components of the system can be interconnected by any form or medium of digital data communication, e.g., a communication network. Examples of communication networks include a local area network ("LAN") and a wide area network ("WAN"), e.g., the Internet, and include both wired and wireless networks.

[0152] The computing/gaming system can include clients and servers or hosts. A client and server (or host) are generally remote from each other and typically interact through a communication network. The relationship of client and server arises by virtue of computer programs running on the respective computers and having a client-server relationship to each other.

[0153] The invention has been described in terms of particular embodiments. The alternatives described herein are examples for illustration only and not to limit the alternatives in any way. The steps of the invention can be performed in a different order and still achieve desirable results.

What is claimed:

1. A method, executed on a game platform, for providing a smooth animation transition in a game comprising:

(a) providing, during play of the game, an event timeline comprising event markers denoting points in time on the event timeline, each event marker associated with an animation segment from a plurality of animation segments;

(b) providing a first marker on the event timeline indicating a first animation segment to be displayed on the display at a first time with respect to event timeline and a second

marker on the event timeline indicating a second animation segment to be displayed on the display at a second time with respect to event timeline;

(c) determining, during play of the game, that the second time is approaching;

(d) determining a set of animation segments to be blended together to provide a smooth transition from the first animation segment to the second animation segment; and

(e) blending among the set of animation segments based on the determination made in step (d).

2. The method of claim **1** wherein the set of animation segments to be blended comprises the first animation segment and the second animation segment and blending comprises blending at least some of the first animation segment with at least some of the second animation segment.

3. The method of claim **1** wherein the set of animation segments to be blended comprises the first animation segment, a bridge animation segment, and the second animation segment, and blending comprises blending at least a some of the first animation segment with at least some of the bridge animation segment and blending at least some of the bridge animation segment with at least some of the second animation segment.

4. The method of claim **1** further comprising:

before the first time, judging a first player's performance of the game; and

at and after the second time, judging a second player's performance of the game.

5. The method of claim **1** wherein the second animation segment is determined based on a difficulty of the game.

6. The method of claim **1** wherein determining the set of animation segments to be blended comprises determining if there is a bridge animation segment in a table, where the bridge animation segment is looked up based on the first animation segment and the second animation segment.

7. The method of claim **1** wherein the first marker and the second marker are associated with the same animation.

8. A method, executed on a game platform, for providing smooth audio transitions in a song comprising:

(a) providing, during play of a game, an event timeline comprising event markers denoting points in time on the event timeline, the event timeline associated with an audio track played during play of the game, the audio track comprising a first audio segment and a second audio segment;

(b) providing a first marker on the event timeline indicating a first time associated with the first audio segment and a second marker on the event timeline indicating a second time with respect to the event timeline associated with the second audio segment;

(c) determining, during play of the game, that the second time is approaching;

(d) determining a set of audio segments to be blended to transition from the first audio segment to the second audio segment; and

(e) blending among the set of audio segments based on the determination made in step (d).

9. The method of claim **8** wherein the set of audio segments to be blended comprises the first audio segment and the second audio segment.

10. The method of claim **9**, wherein blending among the first audio segment and the second audio segment comprises crossfading from at least some of the first audio segment to at least some of the second audio segment.

11. The method of claim **8** wherein the set of audio segments to be blended comprises the first audio segment, a bridge audio segment, and the second audio segment.

12. The method of claim **11**, wherein blending among the first audio segment, the bridge audio segment, and the second audio segment comprises:

crossfading from at least some of the first audio segment to at least some of the bridge audio segment; and

crossfading from at least some of the bridge audio segment to at least some of the second audio segment.

13. The method of claim **11**, wherein blending among the first audio segment, the bridge audio segment, and the second audio segment comprises muting at least some of the first audio segment, playing at least some of the bridge audio segment, and muting at least some of the second audio segment.

14. The method of claim **13** wherein muting at least some of the first audio segment, playing at least some of the bridge audio segment, and muting at least some of the second audio segment comprises muting the first audio segment for its final beat, muting the second audio segment for its first two beats, and playing the bridge audio segment for three beats during the muted last beat of the first audio segment and muted first two beats of the muted second audio segment.

15. The method of claim **9** wherein the first marker and the second marker are associated with the same audio segment.

16. The method of claim **11** wherein the first marker and the second marker are associated with the same audio segment.

17. The method of claim **11** wherein determining the set of audio segments to be blended comprises determining if there is a bridge audio segment in a table, where the bridge audio segment is looked up based on the first audio segment and the second audio segment.

18. A computer program product, tangibly embodied in a non-transitory computer readable storage medium, for providing a smooth animation transition in a game, the computer program product including instructions being operable to cause a data processing apparatus to:

(a) provide, during play of the game, an event timeline comprising event markers denoting points in time on the event timeline, each event marker associated with an animation segment from a plurality of animation segments;

(b) provide a first marker on the event timeline indicating a first animation segment to be displayed on the display at a first time with respect to event timeline and a second marker on the event timeline indicating a second animation segment to be displayed on the display at a second time with respect to event timeline;

(c) determine, during play of the game, that the second time is approaching;

(d) determine a set of animation segments to be blended together to provide a smooth transition from the first animation segment to the second animation segment; and

(e) blend among the set of animation segments based on the determination made in step (d).

19. A computer program product, tangibly embodied in a non-transitory computer readable storage medium, for providing smooth audio transitions in a song, the computer program product including instructions being operable to cause a data processing apparatus to:

(a) provide, during play of a game, an event timeline comprising event markers denoting points in time on the event timeline, the event timeline associated with an audio track played during play of the game, the audio track comprising a first audio segment and a second audio segment;

(b) provide a first marker on the event timeline indicating a first time associated with the first audio segment and a second marker on the event timeline indicating a second time with respect to the event timeline associated with the second audio segment;

(c) determine, during play of the game, that the second time is approaching;

(d) determine a set of audio segments to be blended to transition from the first audio segment to the second audio segment; and

(e) blend among the set of audio segments based on the determination made in step (d).

20. An apparatus for providing a smooth animation transition in a game, the apparatus comprising:

(a) means for providing, during play of the game, an event timeline comprising event markers denoting points in time on the event timeline, each event marker associated with an animation segment from a plurality of animation segments;

(b) means for providing a first marker on the event timeline indicating a first animation segment to be displayed on the display at a first time with respect to event timeline and a second marker on the event timeline indicating a second animation segment to be displayed on the display at a second time with respect to event timeline;

(c) means for determining, during play of the game, that the second time is approaching;

(d) means for determining a set of animation segments to be blended together to provide a smooth transition from the first animation segment to the second animation segment; and

(e) means for blending among the set of animation segments based on the determination made by element (d).

21. An apparatus for providing smooth audio transitions in a song, the apparatus comprising:

(a) means for providing, during play of a game, an event timeline comprising event markers denoting points in time on the event timeline, the event timeline associated with an audio track played during play of the game, the audio track comprising a first audio segment and a second audio segment;

(b) means for providing a first marker on the event timeline indicating a first time associated with the first audio segment and a second marker on the event timeline indicating a second time with respect to the event timeline associated with the second audio segment;

(c) means for determining, during play of the game, that the second time is approaching;

(d) means for determining a set of audio segments to be blended to transition from the first audio segment to the second audio segment; and

(e) means for blending among the set of audio segments based on the determination made by element (d).

22. A system for providing a smooth animation transition in a game, the system comprising:

a display; and

a game platform configured to:

(a) provide, during play of the game, an event timeline comprising event markers denoting points in time on the event timeline, each event marker associated with an animation segment from a plurality of animation segments;

(b) provide a first marker on the event timeline indicating a first animation segment to be displayed on the display at a first time with respect to event timeline and a second marker on the event timeline indicating a second animation segment to be displayed on the display at a second time with respect to event timeline;

(c) determine, during play of the game, that the second time is approaching;

(d) determine a set of animation segments to be blended together to provide a smooth transition from the first animation segment to the second animation segment; and

(e) blend, for display on the display, among the set of animation segments based on the determination made in step (d).

23. A system for providing smooth audio transitions in a song, the system comprising:

an audio output; and

a game platform configured to:

(a) provide, during play of a game, an event timeline comprising event markers denoting points in time on the event timeline, the event timeline associated with an audio track played during play of the game, the audio track comprising a first audio segment and a second audio segment;

(b) provide a first marker on the event timeline indicating a first time associated with the first audio segment and a second marker on the event timeline indicating a second time with respect to the event timeline associated with the second audio segment;

(c) determine, during play of the game, that the second time is approaching;

(d) determine a set of audio segments to be blended to transition from the first audio segment to the second audio segment; and

(e) blend, for output through the audio output, among the set of audio segments based on the determination made in step (d).

* * * * *