Eurographics/ ACM SIGGRAPH Symposium on Computer Animation (2006)
M.-P. Cani, J. O'Brien (Editors)

"Walk to here": A Voice Driven Animation System

Z. Wang and M. van de Panne

University of British Columbia

Abstract

We present a novel interface for directing the actions of computer animated characters and camera movements. Our system takes spoken input in combination with mouse pointing to generate desired character animation based on motion capture data. The aim is to achieve a more natural animation interface by supporting the types of dialogue and pointing that might be used when one person is explaining a desired motion to another person. We compare our voice-driven system with a button-driven animation interface that has equivalent capabilities. An informal user study indicates that for the test scenarios, the voice-user interface (VUI) is faster than an equivalent graphical user interface (GUI). Potential applications include storyboarding for film or theatre, directing characters in video games, and scene reconstruction.

1. Introduction

A long standing goal of intelligent user interfaces is to develop tools and interfaces with which users can converse in the same way with the computer that they might converse with another person. In the case of character animation, this could take the form of a 'smart' system that embodies the understanding and skills of an animator, or, equivalently, a system that has 'smart characters' which embody some of the understanding and skills of actors. In this paper we present a system which implements the basic elements of the director-actor metaphor and we investigate its application to on-line and off-line character animation scenarios. The goals of such a system are to simplify content creation and to explore the expressive power of natural language for creating animated motion.

The capabilities of the system are best illustrated by example. The following example dialogue is used with our system to create the storyboarding example that is illustrated in Figure 8: *Front cam. Character one. Shake hands and say "Nice to meet you". Character two. Shake hands and say "Nice to meet you too." Action.* The current scene plays out, followed by additional user instructions: *Top cam. Character one. Walk to here and say "Come with me please; I will show you the destination." Then run to here and say "Hurry up, let's run." Then walk to here and say "Here we are." Character two. Follow him. Panning shot. Action.* The above dialogue demonstrates the use of spoken dialogue, interaction among multiple characters, the use of pointing to assign specific locations to diectics such as the word *'here'*, camera specifications, and basic path planning.

The remainder of this paper is structured as follows. Section 2 discusses previous related work. An overview of the various components of the animation system and their roles is given in Section 3. Section 4 describes describes the relevant system components in more detail. Section 5 presents results, and Section 6 concludes and presents ideas for future work.

2. Related Work

2.1. Voice-Driven and Multimodal Interaction

Developments in voice recognition technology over the past thirty years have facilitated the design of voice-based user interfaces augmented by other user input, such as drawing or pointing. An early and highly inspirational example of multimodal interfaces was that developed in 1980 by Bolt [Bol80]. The "Put-That-There" system used basic speech recognition and could also track a user's pointing actions. This system allowed users to move colored shapes around on a screen and allowed the free use of pronouns and other assigned names to refer to objects. The system was among the first to support a speak-and-point multimodal interface.

In recent years, more systematic studies have been carried out that examine the benefits, modalities, and pitfalls

that are commonly associated with multimodal interfaces that support voice commands combined with other modalities including pen and gestural input. An early study of voice-driven interfaces for spreadsheet entry points out issues that should be considered during the design of such an interface [Rud89]. The work of Oviatt and her colleagues [Ovi99, OCL04] has been particularly important in this area. Findings reveal that users increasingly favor multimodal communication as the task difficulty and communicative complexity increases [OCL04]. This points to the promise of multimodal techniques for non-trivial animation specification.

A number of common myths related to multimodal input are further described in [Ovi99]. For example, multimodal actions such as combining speech with pointing are most common when describing spatial information. An interesting caveat is that the speak-and-point pattern accounted for only 14% of all spontaneous multimodal gestures for a series of map-based tasks, with pen input being used more often to create graphics, symbols, and other lexical content. Also worthy of mention in our context is the finding that multimodal pen/voice language is briefer, syntactically simpler, and less disfluent than a user's unimodal speech [Ovi99]. Thus, while our system supports a limited, predefined vocabulary, there is in any case a natural tendency for users to revert to syntatically simpler language during multimodal actions such as the speak-and-point patterns that our system supports.

2.2. Natural Language and Speech Interfaces in Support of Human Animation

The use of speech interfaces to author character motion is an under-explored area given the descriptive power of language and the relative ubiquity of speech recognition software. Speech-based systems designed for virtual training applications have been explored in [dOSG00], illustrated with examples of repairing an ATM switch and interacting with vendors in a virtual public mall. The users, represented by VRML-based avatars, can manipulate and interact in a VR system enabled with speech recognition for voice-based commands. Similarly, the system presented in [Apa02] implements a Voice User Interface (VUI) which maps predefined voice commands to a set of behavioral motions for humanoid avatars using Extensible 3D (X3D) graphics. The VUI maps between human voice commands and a set of animations of the avatar, thus providing voice access to the animation application. Their work demonstrates the possibility of integrating speech-recognition technology into networked virtual environments. These systems are principally designed for supporting interactive, first-person point-of-view scenarios. Our system targets storyboarding and more general character animation scenarios by supporting speak-and-point multimodal input, off-line animation, multiple character animation, spoken dialogue among characters, a set of cinematographic camera controls, and a richer motion vocabulary.

A text-based natural language interface was developed for human task simulation [JKBC91] and the graphical display of goal-directed tasks by an agent in a workspace. The inputs to the system are constrained natural language commands for specifying desired spatial relationships among objects in the workspace. To animate the behaviors, a semantics of action verbs and locative expressions is devised in terms of physically-based components, in particular the geometric or spatial relations among the relevant objects.

2.3. Computer Animation Interfaces

Many interfaces exist for computer animation systems, each making a particular tradeoff with regard to the degree of control over the motion and the ease of specification. A common interface for specifying animated locomotion involves the specification of goal locations or paths [LCR*02, AF02] possibly further annotated with a desired style. Performance animation allows an individual to directly act out a desired motion and have this mapped onto a character. By using a sufficiently strong prior model of the expected movements, this can be accomplished with video cameras and a limited number of markers [CH05] in lieu of a full motion capture system.

Labanotation is a written motion notation most commonly used for dance choreography, although it can also be used to author animated motions [CCP80]. Motion Doodles [TBvdP04] is a cursive motion notation useful for quickly specifying a limited repertoire of motions that involve both spatial, temporal, and timing parameters. More distantly-related work includes the use of scripts to specify autonomous behaviours [LRBW05] and personalized styles of motion [NF05]. The work of [HCS96] implements virtual cinematography techniques that we also exploit.

3. System Overview

Our voice-driven animation system is divided into a number of functional blocks: speech recognition, voice command analysis, motion controller, and camera controller. The role of each component is elaborated in the following subsection.

3.1. System Organization

The organization of our animation system and the path of user input and workflow through the system are illustrated in Figure 1.

- **Speech Recognition**: The Microsoft Speech API is integrated into our system as the speech recognition engine. It takes and processes the user speech from the microphone. If the speech matches one of the pre-defined grammar rules, the speech engine will trigger a recognition event

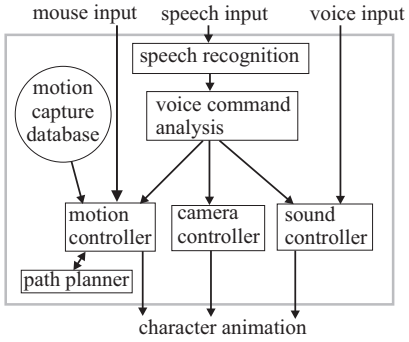

Figure 1: *System Architecture*

and pass the recognized rule ID to the analysis module.

- **Voice Command Analysis**: This module analyzes the recognized voice command received from the speech recognition engine and then extracts associated parameters, such as the walking speed of the character and the required field of view of the camera. These parameters are then passed to the relevant controllers.

- **Motion Controller**: This determines which motion the character will be performing and then loads the corresponding motion clips from the motion capture database. The controller is also responsible for the transition of consecutive motions to make the animation as smooth as possible. Additional mouse input may be provided by the user to indicate destination points or directions.

- **Path Planner**: After the motion and the destination have been determined by the motion controller, the path planner takes into account the location of obstacles, and generates a collision-free path to the destination along which the character will be moving. When a character is asked to follow another character, the path planner is also used to find out the shortest path to perform the following action.

- **Camera Controller**: Each camera movement supported by our system is implemented by the camera controller. The controller makes the necessary changes of the position and shooting directions of the camera whenever a camera-related voice command is issued by the user. A timer is used in the controller to produce the continuous zoom in/out effect as well as the pan shot over a specific point in the scene.

- **Sound Controller**: This consists of the sound recorder and sound player. The sound recorder records any voice input from the user other than the voice commands. The recorded sound is played back as the voice of the character by the sound player during the animation in order to create an audible dialogue between the virtual characters.

3.2. Microsoft Speech API

The Microsoft Speech API (SAPI) provides a high-level interface between an application and speech engines for recognition and synthesis. SAPI implements all the low-level details needed to control and manage the real-time operations of various speech engines, and thereby reduces the code overhead required for an application to use speech recognition and text-to-speech.

A Context-Free Grammar (CFG) is a structure that defines a specific set of words and the combinations of these words that can be used. In simplest terms, a CFG defines the sentences that are valid, and in SAPI 5, defines the sentences that are valid for recognition by a speech recognition engine. The CFG format in SAPI defines the grammar rules using extensible markup language (XML). The CFG Grammar compiler transforms the XML tags defining the grammar elements into a binary format used by SAPI-compliant speech recognition engines. The complete list of voice commands supported by our system is given in Figure 2.

Where possible, the voice commands are mapped in a direct fashion to the corresponding motion clips that define the character's capabilities. Fast and slow motions are achieved through a linear time warp applied to the original motion clips. Turning during walking is achieved through the repeated application of incremental rotations to the root coordinate frame.

4. Implementation

4.1. User Interface

Figure 3 shows the interface of our animation system. On the left, a window displays the animation being created. On the right is a panel with several groups of GUI buttons, one for each possible action. At the top of the button panel is a text box used to display the recognized voice commands, which is useful for verifying whether the voice recognition engine is working properly.

The GUI action buttons are used to provide a point of comparison for our voice-driven interface. Every GUI button has a corresponding voice command. The GUI only provides access to a subset of the voice commands because of the limited screen real estate that is available to house the GUI.

Many of the action buttons are used to directly trigger a specific motion of the character, while some of them have associated parameter to define a specific attribute of the action, such as the speed of the walking action, and the degree of the turning action. These parameters are selected using radio buttons, check box or drop-down list controls. The more

Character Commands	*Camera Action Commands*
walk [straight/fast/slowly]	front/side/top [cam]
[fast/slow] backwards	close/medium/long shot
run [fast/slowly]	zoom in/out [to
jump [fast/slowly]	close/medium/long shot]
[turn] left/right [by]	[over] shoulder cam/shot
[10-180] [degrees]	[on character one/two]
turn around	panning cam/shot
left/right curve	[on character one/two]
[sharp/shallow]	
wave [your hand]	*System Commands*
stop waving/put down	
[your hand]	draw box/obstacle/block
push	box/obstacle/block finish
pick up	[select] character one/two
put down	both [characters]
pass	action
throw	start/end recording
sit down	replay
stand up	reset
shake hands	save file
applaud	open file
wander around here	
follow him	

Figure 2: *Supported Commands. Square brackets denote optional arguments.*

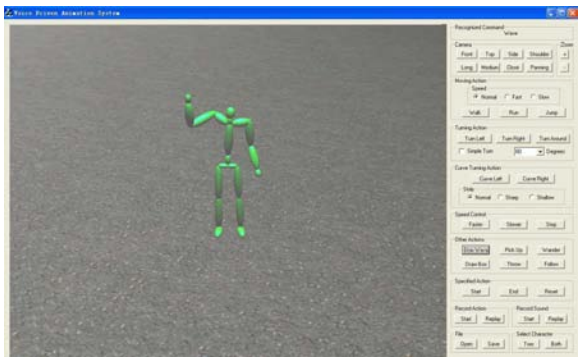

Figure 3: *GUI Interface of our system*

detailed an action is, the greater the number of parameters and visual controls that are required to be associated with the action button.

4.2. Motions and Motion Transition

The motions of character in our system are animated using motion capture data. The motions are captured using a Vicon motion capture system at 120 Hz for a character model with 44 degrees of freedom.

By default, the animated character is displayed in the rest pose. Transitions between different actions may pass through this rest pose, or may be a straight cut to the new action, depending on the types of the original and the target action. The allowable transitions are illustrated using the graph shown in Figure 4.

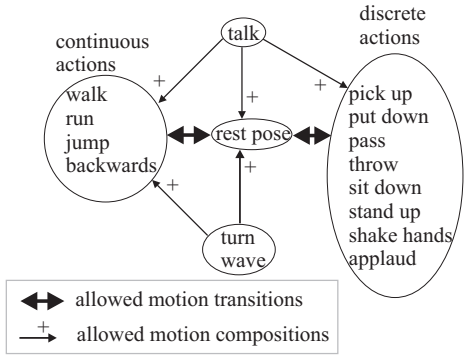

Figure 4: *Motion transition graph*

We divide the actions into several groups according to their attributes. Actions such as walk and run belong to continuous actions. Transition among these actions is done by a straight cut, which may result in a visible discontinuity in the motion. This choice was made to simplify our implementation and could be replaced by blending techniques.

Actions such as *pick-up* and *put-down* are classified as discrete actions. These actions are considered to be uninterruptable. Transition among these actions is straight forward, because every action needs to complete before the next one can begin. Transitions between continuous actions and discrete actions must pass through the rest pose, which serves as a connecting point in the transition.

Several additional actions such as talk, turn, and wave, can be composed with the current actions where they are compatible with each other. For example, talking can be added to the character while he is performing any action, and turning behaviors can be added to any of the continuous actions. Waving is another action that can be added to the continuous actions because it only changes the motion of the arm, while the motion of other parts of the body remains unchanged. A special motion clip is dedicated to blending the transition to and from the waving action of the arm.

Actions involving a pair of characters such as handshaking and pushing are accomplished by having the two characters walk towards each other and then stopping when within a fixed distance of each other. The characters then begin to perform the action at the same time.

4.3. Path Planning Algorithm

The character can perform two high level actions, namely walking to a destination while avoiding any obstacles, and

following another character as it moves throughout the space. These features are implemented using simple path planning algorithms as outlined below. We track a desired path by applying incremental rotations to a base walk clip as required. This could be replaced with a motion graph approach if desired. The choice is largely independent of interface design issues, however, and thus we opt for the simpler solution, which is also quite effective.

4.3.1. Obstacle Avoidance

Obstacles are modeled as rectangular blocks. The size and locations of the blocks are specified by the user and are given to the path planner. At each time step, the path planner examines the positions of the characters and detects if there is a potential future collision with the blocks. If this is the case, the path planner computes a revised path for the character(s) in order to avoid the obstacles. For simplicity, the obstacles are currently assumed to be rectangular blocks.

When a potential future collision is detected, the path planner adjusts the direction of the walking or running motion of the character according to the current position of the character and the destination. The character will follow the boundary of the block towards the destination until it reaches a corner. If there is no obstacle between the character and the destination, the character will move directly towards the destination. Otherwise, it will move onto the next corner, until it can reach the destination directly. The direction of motion of the character is updated only when the collision is detected or the character has reached the corner of a block.

4.3.2. Following Strategy

Another high level action that the character can perform is to follow another character. At each time step, the path planner computes the relative angle between the current heading of the pursuer and the direction to the character. If the angle is greater than a pre-defined threshold, the character will make a corrective turn to follow the target character.

If the other character has changed his action during the movement, e.g., from walking to running, the following character will also make the same change in order to keep pace. The character follows the path along which the other character is moving, and also enacts the same action during the movement.

4.4. Camera Control

Three kinds of static cameras are available, including the front camera, the side camera and the top camera. Each of these can be used in one of three settings, including the long shot, the medium shot, and the close shot. While each setting has its own camera position, the look-at point of the static camera is always on the center of the space.

There are three types of moving camera models in our system. The first is the *over-the-shoulder shot*. The camera is positioned behind a character so that the shoulder of the character appears in the foreground and the world he is facing is in the background. The look-at point is fixed on the front of the character, and the camera moves with the character to keep a constant camera-to-shoulder distance.

The second moving camera model provides a *panning shot*. With a fixed look-at point on the character, the camera slowly orbits around the character to show the different perspectives of the character. Regardless of whether or not the character is moving, the camera is always located on a circle centered on the character.

The last moving camera model is the *zoom shot*, including both zoom-in and zoom-out. The camera's field of view is narrowed or enlarged gradually over the time with a fixed look-at point on either the character or the center of the space. A zoom shot can be combined with over shoulder shot or panning shot simply by gradually changing the field of view of the camera during these moving shots.

4.5. Action Record and Replay

Our system can be used in online and off-line modes. In the online mode, the system immediately interprets spoken commands and places them in an action event queue. Using the off-line mode, the user gives a series of instructions followed by an "action" command to indicate the start of the animation. We use a structure called an *action event* to store information related to each of these instructions, such as the type and location of the action. Each time a new action event is created, it is placed onto a queue. When the animation is started, the system dequeues the action points one by one, and the character acts according to the information stored in each action event.

The animation created in online or off-line mode can be recorded and replayed at a later time. During recording, a *record event* stores the type and starting frame of each action being recorded. The difference between a record event and an action event is that a record event can be used to store the information of online actions, and it contains the starting frame but not the location of each action. If some part of the animation is created off-line, the corresponding action events will be linked to the current record event. All the record events are placed in a queue. Once the recording is done, the animation can be replayed by revisiting the record events from the queue. Since the record events are designed to be serializable objects, they can also be stored in a file and opened to replay the animation in the future.

4.6. Sound Controller

The Wave Player and Recorder Library [Bel05] is integrated into our system as the sound controller. This allows spoken dialogue to be added to an animation, as may be useful for storyboarding applications. The user starts recording

spoken dialogue using a spoken "talk" or "say" command. Each recorded sound clip is attached to the current action of the character. During the recording of the sound, the voice recognition process is temporarily disabled in order to avoid any potential false recognition by the speech engine. When the recording is done, the user clicks a button on the interface panel to stop the recording, which enables voice commands to be accepted once again.

5. Results

5.1. Voice-Driven Animation Examples

The animation examples described below are created using voice as input, with the exception of the user using mouse input to draw the blocks and to point to the destinations of the movement of the character. We also refer the reader to the associated videos, which more directly convey the capabilities of the system.

5.1.1. Online Animation Example

Figure 6 shows a set of frames summarizing an animation created from a series of voice-based commands, as given below each image. When a voice command is recognized, the character responds interactively. With no voice input, the character retains its current behavior rather than stopping. Thus, actions such as walking, running, idling, and wandering all continue until they are interrupted.

5.1.2. Off-line Animation Example

Figure 7 shows an example of creating off-line animation combined with online camera controls. During the off-line mode, the camera is switched to the 'top cam' (plan view camera) and mouse input is used to specify the locations of the block obstacles and the destination points used to disambiguate the deictics used in the spoken input. Each action event is associated with at least one off-line motion specified by the voice command. When the *Action* command is recognized, the animation is generated by processing the action events in sequence. During playback, the user has the option of further interactively commanding the choice of camera model.

5.1.3. Storyboarding Example

Figure 8 (see color plates) shows an example of using our system as an animation storyboarding tool. In this example, the user wants to act out a short skit, where two people meet and greet each other, and then one of them asks the other person to follow him to a destination. At first they are walking. After some time the leading person suggests they should hurry and thus they both start running together. Finally, they arrive at the destination after circumnavigating any obstacles along the way. This animation takes less than a minute to create.

5.2. Comparison between GUI and VUI

We have conducted an informal user study on a group of ten people to compare our voice user interface (VUI) with the traditional graphical user interface (GUI). The users have no prior experience in using the voice interface and all of them have a minimum of five years experience with graphical user interfaces.

Before using our system for the first time, the users need to provide the speech engine with a 10-minute training session that involves reading paragraphs of text. Once the training is complete, the users are given 10 minutes to familiarize themselves with the VUI by creating short animations using voice commands. They then have another 10 minutes to familiarize themselves with the GUI. After that, each of them is asked to create one online and one off-line animation using both interfaces according to the scripts that are given to them. The scripts consist of a series of actions, an example of which is listed along the *x*-axis of Figure 5. The cumulative time is recorded as the user creates each action in the script using the VUI and the GUI, in turn. The mean cumulative time for all users is plotted on Figure 5 for the online animation scenario. A similar result was obtained for the offline animation scenario, which included commands that exercise the speak-and-point patterns.

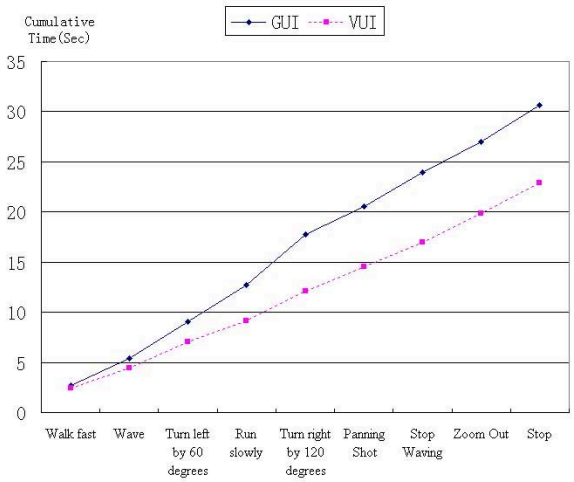

Figure 5: *Mean cumulative time for creating online animation using both interfaces*

For the given scenarios, the voice user interface (VUI) is more efficient than using the graphical user interface (GUI) to create the same animation, whether online or off-line. We note that when using GUI to create complex actions such as *Turn right by 120 degrees* and *Walk fast to here and pick it up*, it takes considerably more time than creating simple actions such as *Wave* and *Draw Box*, due to the fact that complex actions involve searching and clicking more buttons in

the graphical user interface. When using voice commands to create the same animation, there is a less notable difference between the time required to specify simple and complex actions.

When making the above comparison between GUI and VUI, we have not taken into account the extra time spent in the voice recognition training session. We note that the voice-recognition training session is required only once for a new user, and this time is amortized across future use of the system. Oviatt et al. [Ovi99] also note that other advantages of multimodal systems may often be more important than any gains in speed. Particularly, errors can drop significantly for multimodal interactions. The current system has only limited multimodal interaction (speak-and-point), but given the spatial nature of animation, additional multimodal patterns are likely to be beneficial.

6. Conclusions

We have presented a voice-driven interface for creating character animation scenarios based on motion capture data. It provides basic support for multiple characters, spoken dialogue, speak-and-point interaction, basic path planning, integrated autonomous character behaviors, and interactive camera control. We demonstrate the system using online and offline creation of short storyboard-like scenarios. To our knowledge, it provides a first attempt at directly implementing a director-actor metaphor. Our system provides a starting point for further exploration of voice-driven animation systems and future 'smart' animation systems. Future systems of this type have the potential to open up content creation to a significantly wider audience.

6.1. Challenges and Limitations

Achieving the best results required enabling speaker-dependent recognition with a high quality microphone and a predefined grammar. We believe that the system is perhaps best suited for online animation because it is here that the voice-driven interface affords a user the capability to issue interactive commands while remaining focussed on the motion itself. It is also well suited for commands that readily mix spatial descriptors with other motion descriptors.

The space of possible human motions and behaviors is very large and full of subtleties that takes actors a lifetime to master. Natural language processing and computational linguistics is highly challenging and has made halting progress over the past four decades. Speech recognition remains fallible and works best in the context of a pre-structured grammar and user-specific training. It thus comes as no surpise that our system is limited in many ways. It currently supports a limited motion repertoire based on the available motion clips and their connectivity. While capturing and adding new motions is a straightforward process, it will necessitate a more sophisticated motion controller, especially for motions that require interactions between multiple characters.

Certain types of interactions may not be easy to express in spoken language, such as the simultaneous actions of a number of characters. Support for voice-driven layered animation would help with this, as would a richer repertoire of autonomous character behaviors. Additional support for motion styles, e.g., various styles of walks, and emotional overtones lend themselves well to linguistic description. In some cases, it may be impossible for characters to follow a given set of directives because of inherent ambiguities or conflicts. A means is then needed to communicate the nature of the problem back to the user in these situations.

Our speaker dependent voice recognition system achieves a high recognition rate, at the expense of requiring a training session for each user in order to train the speech recognition engine. A further constraint of any voice recognition system is that it requires a quiet environment.

6.2. Future Work

Adding more motions and a corresponding richer motion vocabulary is an obvious next step for testing scalability. It is here that the predefined grammar is most likely to become a bottleneck. It would be interesting to study the spoken and gestural communication used to communicate a particular scenario between two people. The person with the role of 'director' could be shown a paricular scene from an existing film, which they then are required to communicate to an 'actor'. This wizard-of-oz style scenario would provide useful information about how spatial gestures, directing-by-demonstration, and other modes are intermixed. Another direction for future work is allowing for absolute and relative constraints to be placed on the time of events. This would simplify the required user directives at the expense of additional complexity for the system in synthesizing a motion plan that is consistent with the directives.

References

[AF02] ARIKAN O., FORSYTH D. A.: Interactive motion generation from examples. *ACM Trans. Graph. 21*, 3 (2002), 483–490.

[Apa02] APAYDIN O.: Networked humanoid animation driven by human voice using extensible 3d (x3d), h-anim and java speech open standards. *Master's Thesis at Naval Postgraduate School* (2002).

[Bel05] BELETSKY A.: Wave player and recorder library. *http://www.codeguru.com/Cpp/G-M/multimedia/audio/article.php/c9305/* (2005).

[Bol80] BOLT R. A.: "put-that-there": Voice and gesture at the graphics interface. In *SIGGRAPH '80: Proceedings of the 7th annual conference on Computer graphics and interactive techniques* (New York, NY, USA, 1980), ACM Press, pp. 262–270.

[CCP80] CALVERT T. W., CHAPMAN J., PATLA A.: The integration of subjective and objective data in the animation of human movement. In *SIGGRAPH '80* (1980), pp. 198–203.

[CH05] CHAI J., HODGINS J. K.: Performance animation from low-dimensional control signals. *ACM Trans. Graph. 24*, 3 (2005), 686–696.

[dOSG00] DE OLIVEIRA J. C., SHIRMOHAMMADI S., GEORGANAS N. D.: A collaborative virtual environment for industrial training. In *VR '00: Proceedings of the IEEE Virtual Reality 2000 Conference* (Washington, DC, USA, 2000), IEEE Computer Society, p. 288.

[HCS96] HE L., COHEN M. F., SALESIN D. H.: The virtual cinematographer: a paradigm for automatic real-time camera control and directing. In *SIGGRAPH '96* (1996), pp. 217–224.

[JKBC91] JUNG M. R., KALITA J. K., BADLER N. I., CHING W.: Simulating human tasks using simple natural language instructions. In *WSC '91: Proceedings of the 23rd conference on Winter simulation* (Washington, DC, USA, 1991), IEEE Computer Society, pp. 1049–1057.

[LCR*02] LEE J., CHAI J., REITSMA P., HODGINS J., POLLARD N.: Interactive control of avatars animated with human motion data. *ACM Trans. Graph. 21*, 3 (2002), 491.

[LRBW05] LOYALL A. B., REILLY W. S. N., BATES J., WEYHRAUCH P.: System for authoring highly interactive, personality-rich interactive characters. In *Proceedings of Symposium on Computer Animation* (2005).

[NF05] NEFF M., FIUME E.: Aer: Aesthetic exploration and refinement for expressive character animation. In *Proceedings of Symposium on Computer Animation* (2005).

[OCL04] OVIATT S., COULSTON R., LUNSFORD R.: When do we interact multimodally? cognitive load and multimodal communication patterns. In *Proceedings of the Sixth International Conference on Multimodal Interfaces (ICMI 2004)* (2004).

[Ovi99] OVIATT S.: Ten myths of multimodal interaction. *Communications of the ACM 42*, 11 (Nov 1999).

[Rud89] RUDNICKY A. I.: The design of voice-driven interfaces. In *HLT '89: Proceedings of the workshop on Speech and Natural Language* (1989), pp. 120–124.

[TBvdP04] THORNE M., BURKE D., VAN DE PANNE M.: Motion doodles: an interface for sketching character motion. *ACM Trans. Graph. 23*, 3 (2004), 424–431.

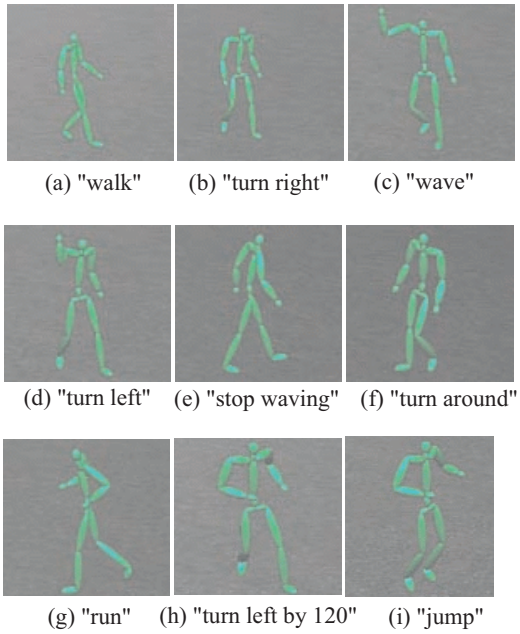

Figure 6: *Online animation example*

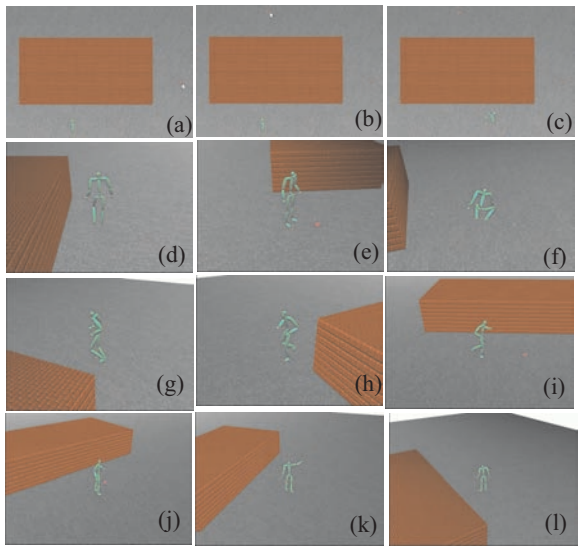

Figure 7: *Off-line animation: (a) "Walk to here and pick it up"; (b) "Then run to here and throw it"; (c) "Action"; (d) "Shoulder cam"; (e) "Panning shot"; (i) "Zoom out";*

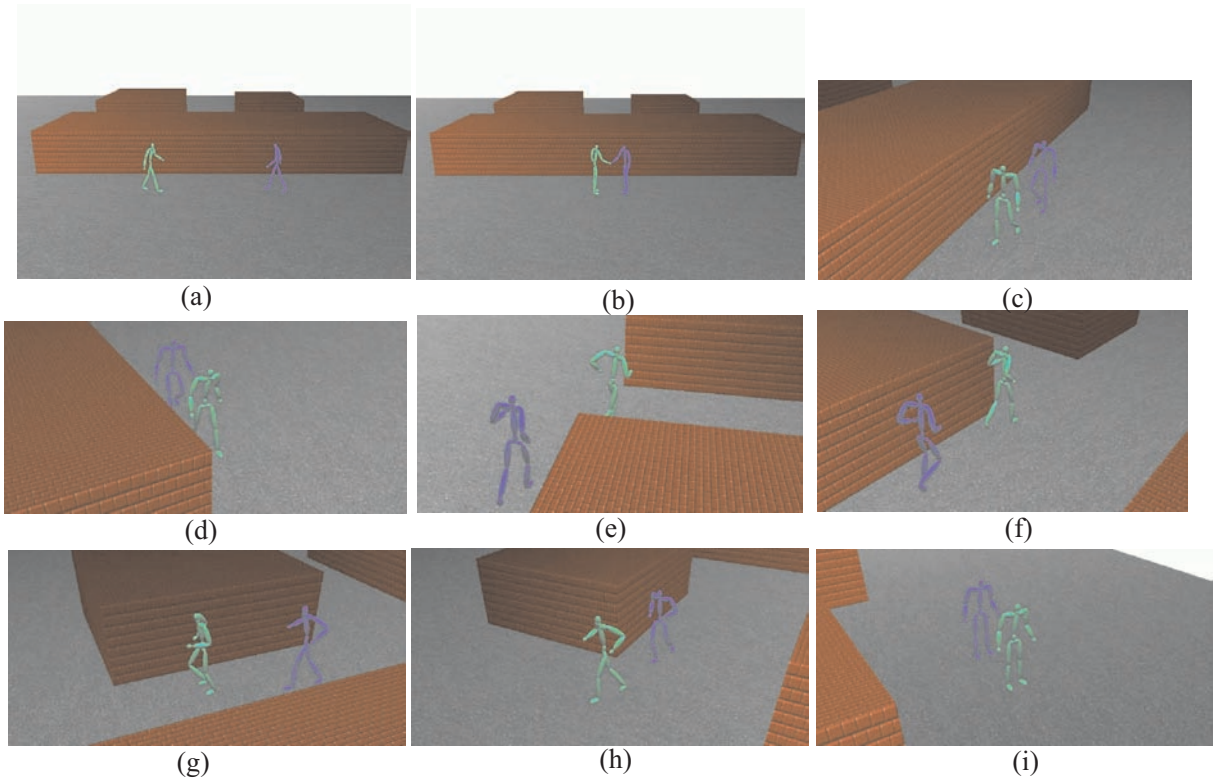

Figure 8: *Storyboarding Demo: "Front cam. Character one. Shake hands and say 'Nice to meet you'. Character two. Shake hands and say 'Nice to meet you too.' Action."* The current scene plays out (a)–(b), followed by additional user instructions: *"Top cam. Character one. Walk to here and say 'Come with me please; I will show you the destination.' Then run to here and say 'Hurry up, let's run.' Then walk to here and say 'Here we are.' Character two. Follow him. Panning shot. Action."* The remainder of the scene plays out (c)–(i).