Key Action Technique for Digital Storytelling

Hiroshi Mori, Jun'ichi Hoshino

University of Tsukuba, Graduate School of Systems and Information Engineering
{hmori, jhoshino}@esys.tsukuba.ac.jp

Abstract. Building story-based interactive systems are important for entertainment and education. In the storytelling system, a user can change the discourse of the story by talking with the character. The problem is that the scene goals in the realistic situations are complex and often multiple goals make effects to decide the current action. When a user asks something to the character, the character needs to arbitrate multiple goals based on the priorities, the current action plan, and the contents of the conversation. In this paper, we propose the method for controlling multiple temporal goals of the story character. The character controls its reaction to the user interactions by using temporal key action (TKA). TKA is a temporal sub-goal setting with time constraint and a priority value. When the TKA is newly added, the action sequences are interpolated using an action network. We created a story-based animation example in which the user can be a detective in the virtual London city.

1 Introduction

Controlling interactions of multiple characters are important for story-based interactive animation and computer games. Fig. 1 shows the example of typical story-based interactive animation. In the story environment, characters typically have multiple goals defined by the story settings. When a user interacts with the character, character's new goals are generated and spontaneous action sequences begin. For example, when the user asks the character to do something, the character needs to decide if the character can accept the request by considering its current goals and time constraints described in the story in advance. The character might reject the user's request due to the time limitations, or its mental status such as emotion and interpersonal relationships. When the character accepts user's request, the planned action sequences should be recalculated based on the user's requested temporal goals.

To construct such a complex story environment, interactive characters should have the following functions:

Generating time and location constraint action plan: When user and the character decides to meet at the specific location, the characters needs to plan action sequences by referencing time stamps. For example, when the user asks a character to "meet with person A at B street on C o'clock", the character should proper action sequences to satisfy that time and location conditions. By sharing the time and location specifications between multiple characters, synchronized action can be generated.

Arbitrating multiple temporal goals: A character has multiple temporal goals in story environment with different priorities. For example, when a user request an information about the accidents to a resident in a town, the resident might be busy due to his works. The user needs to change the priority by talking with the character.

In this paper, we propose the story-based interaction synthesis technique based on conversational control of temporal key actions. We control character's actions by using a key action technique. The key action specifies the important action and timing that characterize the discourse of the story. Key actions are interpolated using an action network.

The character controls its reaction to the user interactions by using temporal key action (TKA). TKA is a temporal sub-goal specification with time constraint and a priority value. When the TKA is newly added, the action sequences are interpolated using an action network. By integrating TKA and template-based language synthesis technique, the character can explain the current and future action plan. The user can dynamically change the priority of TKA through the conversation with the character. We created a story-based animation example in which the user can be a detective in the virtual London city.

Fig. 1. Snapshots from multiple character animation using key interaction control method. A user can change the discourse of the story by making a conversation with characters.

2 Previous Works

Behavior control of virtual humans has been one of the important topics in computer graphics. [Noser et al. 1995, 1996] presented a navigation model for animation characters using synthetic visions, whilst [Bandi and Thalmann 1998] discretized a synthetic environment into a 3D uniform grid to search paths for autonomous characters. IMPROV uses script language to control the actor's interactive motion[Perlin and Goldberg 1996]. [Faloutsos et al 2001] proposed a composer able controller for physically-based character animation. Goal directed action control using behavior network was proposed by [Maes 1989]. The proper behavior modules are selected by propagating activation from the goal behavior. [Swartout2001, Martinet2003] proposed story-based virtual reality systems using interactive characters.

[Funge 1999] proposed the cognitive modeling technique for virtual characters. Characters can control its actions by using goal specifications and planning technique.

[Mori 2004] has proposed the early concept of key action control method. The previous method provides static goal setting, and temporal goal arbitrations are not considered. In this paper, we focus on the problem of time constrained multiple goal arbitrations in the complex story situations.

Human motion synthesis techniques have also been extensively investigated. [Rose et al. 1998] introduced the framework of "verbs and adverbs" to interpolate example motions with a combination of radial basis functions. [Kovar et al. 2002] introduced a motion graph to represent transitions between poses of the captured motion data. A node of this graph represents a pose, and two nodes are connected by a directed edge if they can be followed from one to the other. [Lee et al. 2002] represented motion data with a graph structure, and provided a user interface for interactive applications. [Pullen and Bregler 2002] developed a method for enhancing roughly-keyframed animation with captured motion data. [Li et al. 2002] developed a two-level statistical model by combining low-level linear dynamic systems with a high-level Markov process. Generating gestures from natural languages was developed by [Cassell 1994, 2000].

In the narrative theory field, [Propp1958] proposed that classic folktales can be constructed by series of typical character actions. Propp named the character's typical action prototype functions. A function is an abstracted character action such as "The hero leaves home" and "The villain harms a member of the community". [Sgouros 1999] proposed a dynamic generation technique of interactive plots. The framework supports an Aristotelian plot conception, in which conflict between antagonistic forces develops out of an initial situation. Our multiple key action technique is useful for filling the gap between animation and the logic-based plot control technique because continuous animations can be produced from abstract story descriptions with motion-level consistency.

3 Key Action Technique

3.1 Overview of the key action method

Story can be considered as a collection of character's composite actions with a proper order and timing. Character's composite acting is modeled by a sequence of small actions. When the user interacts with the character, the character's actions locally changes, but we would like to maintain global consistency of actions.

To generate improvisational action with global consistency, we introduce *key actions* that corresponds to the important action in the story. Fig. 3 shows the Story's structure is described using series of key actions like Propp's functions [Propp1958]. Actor's action has degree of freedom between key actions. For example, if there is a key action describing "actorA find object B in room C", there are a lot of variations how actorA finds object B. Another actor or user may happen to talk to the actorA while it is searching in the room.

Key action interpolation is a function that generates action sequences between key actions. There are many possible interpolated actions. Proper action sequences are selected by referencing current situations and internal actor's status such as emotion parameters.

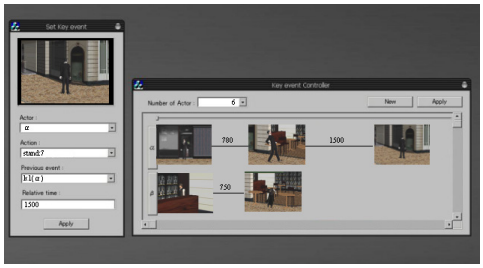

Fig. 2. Storyboard interface for key action setting.

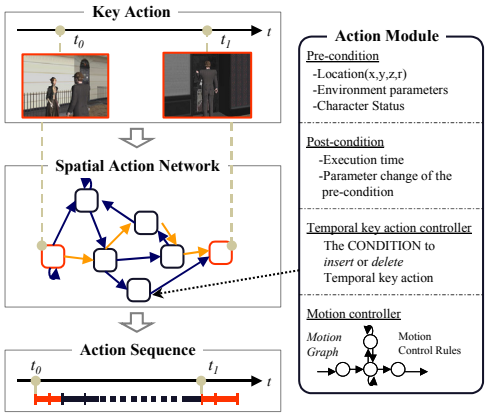

Fig. 3. Interpolating key action using spatial action network

3.2 Story descriptions using key action network

Fig. 2 is the example of storyboard interface to visually describe key actions[MORI04]. A key action ***k*** is defined by

k: ***Actor****[Actor ID]* ***Do*** *[Action ID]* ***at*** *[Previous Key Action ID,* Relative *time]*

Actor ID is actor identifier, *Action ID* is action module identifier to output actor motion(Details are described in 4.1.), *Previous Key Action ID* is identifier of the key action that become standard in providing execution time of this key action *k*. *Relative Time* is the specification of relative interval of time from *Previous key Action* to this key action *k*.

There are two categories of key actions. *Static key actions* define the story actions described by creators in advance. *Temporal key actions* are determined by the actor's internal module. Key actions are more like human actor's script in movie scenes because characters need to add various actions between key actions.

4 Key Action Interpolation using Spatial Action Network

4.1 Action modules and action network

Key actions are interpolated with sequence of actions using action network. Action modules are abstract functions representing what and where the actions take place. Action modules consist of *pre-condition, post-condition, temporal key action controller, and motion controller*.

Temporal action controller(TKAC) controls the reaction of the character by temporarily control sub-goal settings. TKA has priority value in addition to the KA definition in Sec. 3.2. The priority value of the TKA is calculated in TKAC.

Action modules are connected by spatial action networks that represent possible connections of actions. An action network consists of a directed graph connecting action modules. Arcs represent direction of the module connections.

An action network is automatically generated from a collection of action modules. Continuity of the action is evaluated and the possible action modules are linked together. First the possible action modules are selected by the scene descriptions. For example, there is a suitable class of actions in breakfast scenes. Then, we evaluate the two types of continuity: 1) spatial continuity, and 2) pose continuity. Spatial continuity means that not all actions are executable in all the positions. Spatial continuity is evaluated by referencing a precondition of the action modules. Pose continuity means that the end of one action module and beginning of the other action module is connectable.

Action modules are abstracted by a controller, and we separate the actual implementation of the motion synthesis technique. In the experiments in this paper, we use a simple motion transition graph to connect roughly the continuous motion segments.

Action network is represented as $N^{\alpha}_{M} = (G=(M, A), length_M)$ where node $m \in$M is action module, graph G consists of directed arc $a \in A$, and $length_M(a)$ corresponds to execution time T_{tail} . Execution time of an action sequence can be represented by the total length of a partcial graph.

$$length_p(P) = \Sigma length_M(a) \tag{1}$$

4.2 Multiple path finding over action networks

Actors can improvise behaviors to satisfy key actions by searching optimal path in the action network. Fig. 4 shows the concept of key action interpolation. To interpolate key action, possible action sequences are evaluated based on time constraints. Key action interpolation can be done by extending Dijkstra method [Dijkstra1958].

In original Dijkstra method, only minimum cost path is selected. Therefore we apply Dijkstra method twice in forward and backward. We reverse directed graph when we apply Dijkstra method in backward. By adding cost of each nodes obtained

by backward and forward search, we can obtain the minimum path of three nodes {begin node, relay node, end node}.

Let us m_i is an action module specified by a key action k_i . When we thinks an action sequences to satisfy two key actions k_i and k_{i+1} , the time constraint condition can be represented by

$$length_k(k_i,\ \ k_{i+1}) = length_p(P_i) + \Delta T \quad (-T_e < T < T_e) \tag{2}$$

whereΔT is an margin time, and T_e is maximum time tolerance. Action sequences are calculated using the above time constraint conditions.

Using the time constraint conditions, the proper action sequence $P_i = (m_i, m_k, m_{k+1}, ..., m_{i+1})$ is generated using the following steps:

Apply Dijkstra method from k_i to k_{i+1}, and then from k_{i+1} to k_i. In the Dijkstra method, the minimum distances from the beginning node are stored in each node in the action network. Let $C^j_{forward}$, C^j_{back} be the minimum distance at node j.

Let action sequence P_i be the minimum path that satisfy the begin and end key actions, and relay key actions. The total length is $length_M(P_j) = C^j_{forward} + C^j_{back}$

$$length_k(k_i,\ \ k_{i+1}) \le length_p(P_j) + \Delta T \quad (\Delta T > 0) \tag{3}$$

Consider the action modules that are executable repeatedly. The original Dijkstra method does not count such nodes. When N repeating nodes are included in candidate path P_j,

$$length_p(P_j^{\prime}) = length_p(P_j) + \sum_{n=0}^{N-1} \sum_{k=0}^{K_{r+n}} k \cdot length_M(m_{r+n}) \tag{4}$$

where K is

$$K_{r+n} = length_k(k_i, k_{i+1}) - length_p(P_i) \,/\, length_M(m_{r+n}) \tag{5}$$

The minimum distance is selected as

$$|length_k(k_i,\ \ k_{i+1}) - length_p(P_i) - \Delta T)| \rightarrow min \tag{6}$$

4.3 Sharing key actions for cooperative actions

We need to synchronize the multiple characters' actions in most of the story. For example, when Actor_A meets Actor_B at location C on time D, two actors need to move to the location C and improvise actions until time D. The characters might spend time by shopping, having a coffee, or talking with friends. In this case, we use key action association. When two characters (actorA and actorB) meet and talk, the actorA and actorB have key actions of the same location and the same time.

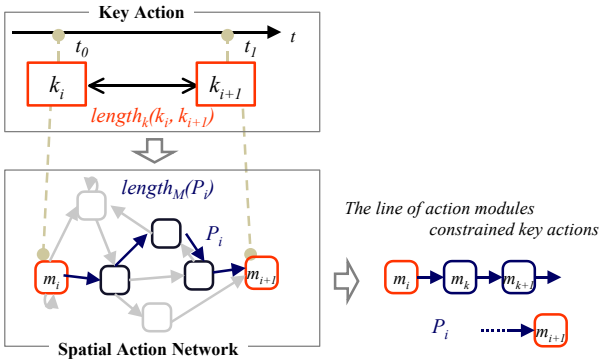

Fig. 4. Key action interpolation using the action network.

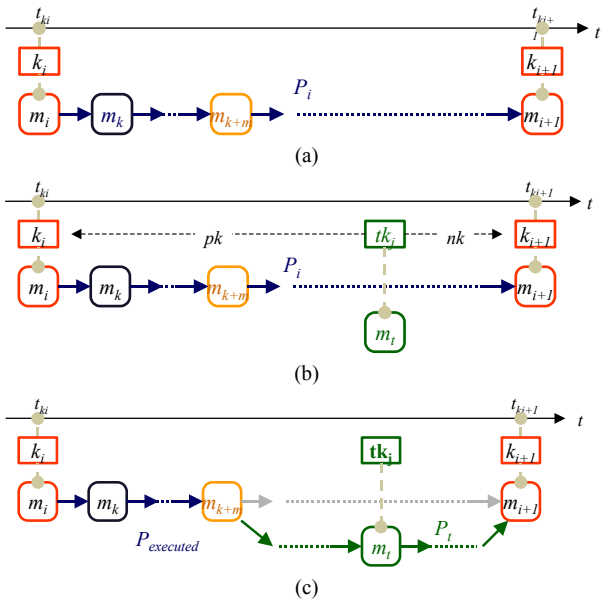

Fig. 5. Example of temporal key actions. Action modules can create temporal key actions to dynamically change the course of the actions, whilst maintaining the global story structure.

4.4 Controlling Temporal Key Actions

Key actions are used to control the character's action by the consequence of the user's interactions. The character is capable of controlling the temporal key actions. For example, when the user asks the character to do something, and the character has enough time, it will changes its temporal goal to answer. Temporal key action rules can be described as follows:

If (condition) then **insert** *or* **delete** *Temporal key action #n*

Fig. 5 shows the example of temporal key action control. Action modules can create temporal key actions to dynamically change course of actions, yet the global story structure is still maintained.

When new temporal key action tk_j is added to an action sequences $P_i = (m_i, m_k, m_{k+1}, \ldots, m_{i+1})$ between key actions k_i, k_{i+1}, new action sequences $P_i = (m_i, m_k, m_{k+1}, \ldots, m_{k+m}, \ldots, m_{i+1})$ is generated. Let the action sequence between m_i and m_{k+m} be $P_{executed}$, the application condition of tk_j is as follows:

$$Length_p(P_t) \leq length_p(P) - length_p(P_{executed}) \quad (7)$$

When a new TKA satisfied the above conditions, the TKA is inserted in the action sequence, and the new action sequence is dynamically generated.

5 Conversational Control of Multiple Temporal Key Actions

Characters typically have multiple scene goals described in the story. These predefined multiple goals contain daily sequences and story events. When a user talks with the character, the character needs to arbitrate the priority of multiple goals based on the current action plans and the contents of the conversation. The character controls its reaction to the user interactions by using temporal key action (TKA). TKA is a temporal sub-goal setting with time constraint and a priority value. By integrating TKA and template-based language synthesis technique, the character can explain the current and future action plan.

5.1 Arbitrating multiple TKA using priority values

When there are many possible TKAs at the current situation, the character can not adopt all TKAs within limited time. Therefore the character needs to select proper TKA to generate a proper action sequence.

TKA has a priority value as described in Sec. 4.1. We examine the possibility of adding a new TKA by dynamically calculate a temporal action sequences. If the temporal action sequences are executable, the character add a new TKA in the current action sequences. Let us the current action sequences be calculated from two KA and one TKA : $\{KA_a, TKA_a, KA_b\}$. When new TKA_b is generated, the character temporarily generate an action sequences $\{KA_a, TKA_a, TKA_b, KA_b\}$. If the new action sequence is executable, the current action sequence is replaced to the new one.

Arbitrating process of multiple TKAs can be done by the following steps:

1) Sorting TKAs by using priority values within the planning scope $T_{planing}$. Select candidates by thresholding sorted TKA with threshold value C.

2) Calculate a temporal action sequence by applying the TKA with the highest priority.

3) Check the time constraint of the new temporal action sequence. If the action sequence is executable, remove the TKA from the candidate list. Then, go to 2) and apply new TKA with the second priority value.

4) Apply 3), 4) to the all candidates TKAs in the list.

In step 2), the generation of a temporal action sequence can be done as follows. Let m_{execut} be the action module that are currently executing. Let the sorted and thresholded temporal key actions are $\boldsymbol{tka}=(tk^{(1)}, tk^{(2)}, \ldots, tk^{(n)})$. First, we calculate the action sequence from m_{execut} to a closest story key action KA with $tk^{(1)}$. If it is possible to calculate a temporal action sequence considering time constraint, $tk^{(1)}$ is adopted. When the priority values of TKAs are changed by the situation change or user's utterance, the sorted order of TKAs are changed. Then the new action sequences are re-computed.

5.2 Conversational Control of TKA

The user can dynamically change the priorities of TKAs by the conversation with the character. We created a story-based animation example in which a user can be a detective in the virtual London city. 5.2.1 describes an interactive action planning technique using an utterance from the user. 5.2.2 describes a template-based conversation synthesis technique using an action plan.

5.2.1 Changing the priority of TKA by user's utterance

User's utterance gives an influence to the priority of TKA. When a user talks to a character, TKA controllers analyze the utterance by using a standard lexical analyzer. Priority control rules in TKA controller decide the influence value on effect value. When the priority value changes, the action sequence is updated using the arbitration method in Sec. 4.3.

5.2.2 Generating character's conversational response

Characters respond to user's requests by generating conversation. When a character received a user's utterance, the character matches the input text and condition of the conversation templates. Conversation template consists of user's utterance, a history of executed action sequence, story key action, TKA, and non-executed TKA. For example, in the detective RPG game, when the user asks "Could you let me know about A?", and the character's priority is low, the character may select an answer "Could you come back later because I'm busy doing X".

6 Examples

We applied the proposed method to a multi-user RPG game. In this game, the user plays the role of Sherlock Holmes in a virtual London city. User can control the locomotion of the character using a keyboard or a game controller pad. The system displays the candidate utterances to the user. In this example, 6 characters are controlled.

6.1 Key action-based behavior generation

We show the example of key action control of a café stuff and a male character. Table 1 and 2 show story key actions in the café scene. We would like to have a scene that the café staff and the make character bumps each other before the café. The characters should improvise actions between the specified actions based on their roles.

To create such a scene, we specify the key actions of going out an office ($k_0^{(Male)}$), stagger before the cafe ($k_1^{(Male)}$), standing at the crossings ($k_2^{(Male)}$) for a male actor. For staff's key actions, standing before a counter ($k_0^{(staff)}$), picks up a garbage ($k_1^{(staff)}$). By specifying "stagger before the cafe ($k_1^{(Male)}$)" and "picks up a garbage ($k_1^{(staff)}$)" at the same time stamps, we can generate the action sequences that satisfy the time constraint.

We show snapshots of the produced animation in Fig. 6. The animation sequence with action networks is in the video. Red nodes in the action network are key actions, green nodes are TKA, and orange lines show the action sequences that satisfy key action settings.

6.2 Changing character's priority via conversation

The user controls Holmes, and asks information about the incident from the café staff. First, the café staff answers "I'm busy" because of TKAs created by the customer characters inside the cafe. However, after Holmes explains about the importance of the hearing, the priority of café staff's TKA changes. The action sequences are recomputed, and action modules to answers the questions are selected. The result of the generated animation sequences are in the video.

Table 1. Key action example of a make actor

Key action	Script
$k_0^{(Male)}$	**Actor**(*Male*) **Do**(*open_door:18*) **at**(*none, 0*)
$k_1^{(Male)}$	**Actor**(*Male*) **Do**(*collide:32*) **at**($k_0^{(Male)}$, *780*)
$k_2^{(Male)}$	**Actor**(*Male*) **Do**(*stand:3*) **at**($k_1^{(Male)}$, *1500*)

Table 2. Key action example of a café stuff

Key action	Script
$k_0^{(Staff)}$	**Actor**(*Staff*) **Do**(*counter:1*) **at**(*none, 0*)
$k_1^{(Staff)}$	**Actor**(*Staff*) **Do**(*pick_up:31*) **at**($k_0^{(Staff)}$, *750*)

Fig. 6. Correspondence of key actions and snapshots of the generated animation sequence.

7 Conclusion

In this paper, we propose the story-based interaction synthesis technique based on conversational control of temporal key actions. Behaviors of the characters are controlled by changing the priorities of the characters. Discourse of the story is changed by accumulating such changes.

The limitations of the current technique may be the conversation synthesis technique is still limited. Integration with the more sophisticated conversation modules will be useful. In future, we extend our approach to the more complex story situations and interactions with many characters.

References

ICKMORE, T., CAMPBELL, L., VILHJ´ALMSSON, H., AND YAN H.,. Conversation as a system framework: Designing embodied conversational agents. 2000. In J. Cassell, J. Sullivan, S. Prevost, and E. Churchill, editors, Embodied Conversational Agents. MIT Press, Cambridge, MA

COHEN, M. F. 1992. Interactive spacetime control for animation. ComputerGraphics(Proc.SIGGRAPH'92)26, 293–302.

DIJKSTRA, E. W. 1959. A note on two problems in connection with graphs. Numerische-Mathematik1, 269–271.

FALOUTSOS, P., PANNE, M. TERZOPOULOS, D..2001. Composable Controllers for Physics-based Character Animation. Proceedings of ACM SIGGRAPH 2001, pp 251–260

FUNGE, J, TU, X. , TERZOPOULOS, D. , "Cognitive Modeling: : Knowledge, reasoning and planning for intelligent characters," Proc. ACM SIGGRAPH 99 Conference, Los Angeles,

CA, August, 1999, in Computer Graphics Proceedings, Annual Conference Series, 1999, 29-38.

KOVAR, L., GLEICHER, M., AND PIGHIN, F. 2002. Motion graphs. ACMTransactionson-Graphics(Proc.SIGGRAPH2002)21,3, 473–482.

LEE, J. AND SHIN, S. Y. 1999. A hierarchical approach to interactive motion editing for human-like fi gures. *Computer Graphics (Proc.SIGGRAPH'99)33*, 395–408.

LI, Y., WANG, T., AND SHUM, H.-Y. 2002. Motion texture: A two-level statistical model for character motion synthesis. *ACM Transactions on Graphics(Proc.SIGGRAPH2002)21,*3, 465–472.

MAES, P.1989. How to do the right things. Connection Science Journal, Special Issue on Hybrid Systems

MATEAS, M.. STERN, A. 2000. Towards Integrat-ing Plot and Character for Interactive Drama, Socially Intelligent Agents: The Human in the Loop, AAAI symposium.

MARTIN K, MICHAEL K, PATRICK G, THOMAS R. 2003. Staging Exhibi-tions: Methods and tools for modelling narrative structure to produce interactive performances with virtual actors, In: Special Issue of Virtual Reality on Story-telling in Virtual Environments, Springer-Verlag .

MORI, H, HOSHINO, 2005, Key Action Control of Virtual Actors, Proceedings of IEEE VR 2005, (accepted as a poster)

NOSER, H., PANDZIC, I. S., CAPIN,T.K., THALMANN, N. M., AND THALMANN, D. 1996. Playing games through the virtual life network. In Proc.Alife'96.

NOSER, H., RENAULT, O., THALMANN, D., AND THALMANN, N. M. 1995. Navigation for digital actors based on synthetic vision, memory, and learning. ComputersandGraph-ics19,1, 7–19.

PERLIN,K. GOLDBERG,J A. 1996. Improv:a system for scripting interactive actors in vir-tual worlds, SIGGRAPH'96, pp. 205 – 216

PROPP, V. 1958 . Morphology of the Folktale", University of Texas Press

PULLEN, K. AND BREGLER, C. 2002. Motion capture assisted animation: Texturing and synthesis. *ACM Transactions on Graphics (Proc.SIGGRAPH2002)21,*3, 501–508.

RICKEL, J., JOHNSON, W., 1999. Animated agents for procedural training in virtual reality: Perception, cognition, and motor control. Applied Artificial Intelligence, 13:343–382

RICKEL, J., JOHNSON,J L.. 1999. Virtual humans for team training in virtual reality. In Proceedings of the Ninth International Conference on Artificial Intelligence in Education, pages 578–585. IOS Press

ROSE, C., COHEN, M. F., AND BODENHEIMER, B. 1998. Verbs and adverbs: Multidimen-sional motion interpolation. *IEEE Computer Graphics and Applications 18,*5, 32–40.

TOWNS, S. VOERMAN, J. ,CALLAWAY,C. LESTER, J. 1997. Coherent gestures, locomo-tion, and speech in life-like pedagogical agents, Proceedings of the 3rd international confer-ence on Intelligent user interfaces. pp. 13-20

SGOUROS, N. M. 1999. Dynamic Generation, Management and Resolution of Interac-tive Plots, Artificial Intelligence, vol. 107, no. 1, pp.29-62.

SWARTOUT, W., HILL, R., GRATCH, J., JOHNSON, W.L., KYRIAKAKIS, C., LABORE, K., LIND-HEIM, R., MARSELLA, S., MIRAGLIA, D., MOORE, B., MORIE, J., RICKEL, J., THIEBAUX, M., TUCH, L., WHITNEY, R., and DOUGLAS, J. 2001. Toward the Holodeck: Integrating Graph-ics, Sound, Character and Story. ,Proceedings of the Fifth Interna-tional Conference on Autonomous Agents 2001, ACM Press, pp.409-416.