Creating Agent Behaviors in Real-time

Norman I. Badler, Jan M. Allbeck, Rama Bindiganavale, Karin Kipper,
Michael B. Moore, William Schuler, Liwei Zhao,
Aravind K. Joshi, and Martha Palmer
Center for Human Modeling and Simulation
Computer and Information Science
University of Pennsylvania
Philadelphia, PA 19104-6389
215-898-5862
badler@central.cis.upenn.edu

1 Overview

A challenging research area is virtual reality systems suitable for training interpersonal interactions. In such a system, at least one person is the VR participant while one or preferably several more virtual human agents are engaged in activities in the same virtual space [7]. The participants, whether live or virtual, should interact as if all were real. This means that the virtual agents must have several characteristics afforded to real people:

1. They should exhibit sufficient situation awareness to precipitate relevant and appropriate decision-making. They should have human-like attention and, if possible, perceptual capabilities. At the minimum they should be able to "see" the actions of others [15] and make plausible assessments of behavior, action, and even intention.

2. They should understand verbal instructions for action [9, 11], the cessation of action, or the behaviors appropriate to future situations (standing orders). Training for leadership will require issuing orders, and the virtual agents must be knowledgeable about the content, meaning, and applicability of those orders. This is partly a natural language communication problem, but more importantly, a resolution of an interpreted instruction against the situational context of the virtual agent.

3. They should exhibit gestures, movements, and facial expressions appropriate to the primary actions they are engaged in, and these should appear credible and situationally meaningful to the live participant.

These three requirements demand significant improvements over embodied agent representations used in today's VR systems. While superficial appearance (shape, attire, equipment complement) is certainly important to the live participant's perception of the other

virtual beings, their appropriate behavior is arguably more crucial to training success. An approach to action selection (and execution) through situation awareness and run-time instructions is the research area we deem most critical to the next generation of VR training systems.

Needless to say, these capabilities must execute in real-time for effective training. Authoring training scenarios must therefore not tax the abilities of subject matter experts to change the course of the simulation as it is running. Capabilities (1) and (2) are the keys to run-time behavior modifications of virtual agents. Capability (3) lets us manifest internal states (observation, deliberation, and decision-making) of the virtual agent into human-like external actions.

In this paper we outline a broad and integrated approach to creating behaviors for real-time 3D embodied agents. We start with a brief summary of the sorts of instructions we wish to accommodate and the architecture we have designed and implemented to interpret and execute instructions in context. The architecture includes a parameterized action dictionary called an Actionary. Instantiated actions control a shallow software hierarchy of finite state machines and motion generators. Then we discuss various aspects of an autonomous agent model which support selection and control of a wide range of behaviors needed in a real-time training experience. We address the issues of movement naturalness by considering a parameterized system for expressing and animating the qualitative aspects of a gestural movement. We show how parametric action representations might be build from limited observations of performed movements. Finally, we close with some discussion of near future research directions.

2 Extensions to the PAR Architecture

The work presented in this paper is an extension of previous work found in [1, 3, 5], which uses natural language input to dynamically alter the behaviors of agents during real-time simulations. In that architecture we were able to give agents both immediate instructions such as "Approach the vehicle" and conditional instructions such as "If the driver has a weapon, draw your weapon and aim it at the driver". However, we were not able to give negative directives or constraints, such as "Do not stand in front of the car door" or "Do not walk on the grass." In order to successfully carry out these type of instructions, we need a planner which can dynamically alter an agent's behavior based on constraints imposed by new instructions.

We extend our current architecture to include a planner and the ability to keep a record or history of all actions successfully completed by the agent. This will also allow us to give conditional, iterative instructions to the agent, such as "Check every abandoned vehicle." The history feature ensures that the agent will correctly check every abandoned vehicle only once. The planner also allows us to give abstract instructions such as, "Take cover behind

the drum." This instruction does not include information about how to take cover behind the drum. The agent could walk, run, crawl, swim, or do any number of other translatory actions to position itself behind the drum. Also, the instruction has no information about the path the agent should take, nor how the spatial prepositional phrase "behind the drum" should be interpreted. The planner will provide the information that is needed for animation but not provided by the natural language input.

3 Actionary

The *Actionary* (Figure 1) is the core component of our system. It contains persistent, hierarchical databases of agents, objects, and actions. The objects are smart objects [12] that contain information necessary for object-agent interaction like grasp sites and constraint locations. The agents are treated as special objects and stored within the same hierarchical structure as the objects. The object hierarchy is constantly updated during the simulation, recording any changes in the environment or in the properties of the agents and objects. Actions are represented as PARs (Parameterized Action Representation). Each PAR can either be uninstantiated, containing only default properties for the action or be instantiated, containing specific information about the agent, objects, and other properties. Each PAR contains both low-level and high-level information. Low-level information includes motion, force, and path. High-level information includes applicability conditions, preparatory specifications, termination conditions, and manner. All the PARs are stored hierarchically within the *Actionary* and map to one or more motion generators.

PAR schemas are similar to PARs, except that their hierarchy is derived from natural language verbs and semantics, whereas the PAR hierarchy is derived from motion semantics within the animation domain. The hierarchy of PARs allows actions with physically similar animations to inherit properties from a common parent. The hierarchy of PAR schemas allows semantically similar verbs, such as motion verbs (like "go") or verbs of contact (like "hit"), to be closely associated with each other under a common parent that captures the properties these verbs all share [13]. Also, not all PAR schemas are directly associated with particular motions. For example, "Take cover" does not specify the translatory action (walking, running, swimming, etc.) that should be used to take cover. The highest nodes in the hierarchy are occupied by generalized PAR schemas which represent the basic predicate-argument structures for entire groups of subordinate actions. The lower nodes are occupied by progressively more specific schemas that inherit information from the more generalized ones. PAR schemas have been used to demonstrate that this representation can also be used as an interlingua in Machine Translation applications[14].

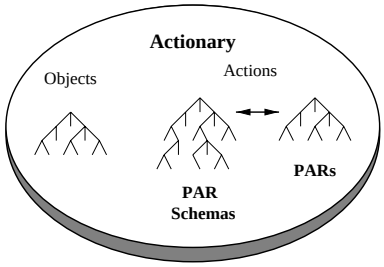

Figure 1: Actionary

4 Architecture

Controlling a virtual human model from sources other than direct animator manipulations requires an architecture that supports higher-level expressions of movement. We use a multi-level architecture system to provide efficient localization of control for both graphics and language requirements. The architecture is grounded in typical graphical models. Motion generators drive these graphical models to generate various motor skills, and endow the virtual humans with useful abilities like walking, reaching, grasping, climbing, etc.

At the next higher level, we use PaT-Nets to allow for simultaneous execution of the various motion generators thus allowing a human to "walk, talk, and chew gum" at the same time. PaT-Nets are effective programming tools, but do not represent exactly the way people conceptualize a particular situation, so at the next higher level, we use PARs and PAR-schemas to capture additional information, parameters, and aspects of human action.

At the highest level, we use natural language to instruct virtual humans while a simulation is running, so a user can dynamically refine an agent's behavior or react to simulated stimuli without having to undertake a lengthy programming session. This allows a non-technical user, such as a domain expert, to specify abstract, conditional behaviors, thereby scripting a simulation.

Figure 2 shows the new architecture of our system. The user inputs natural language instructions for a specific agent through a GUI. The *Natural Language Transducer* parses the instructions and translates them into situation calculus expressions containing the action information from the PAR schemas. These expressions are stored as *Desired Situations* in the *Agent Process*, and serve as temporally-extended goals of the agent, so they can function as constraints over long periods of time (like "Don't drive on the grass,") and persistent conditional rules (like "If there is a truck, open the gate,") as well as ordinary state-based goals (like "Close the gate,") without any special treatment. The situation calculus can also express information about spatial references (like 'in' and 'near') in a natural language input, by treating them as three-dimensional (spatial bounding box) constraints on situations, much as temporal references (like 'yesterday') can be treated as one-dimensional (time interval) constraints on situations. Since the input instructions may contain a certain amount

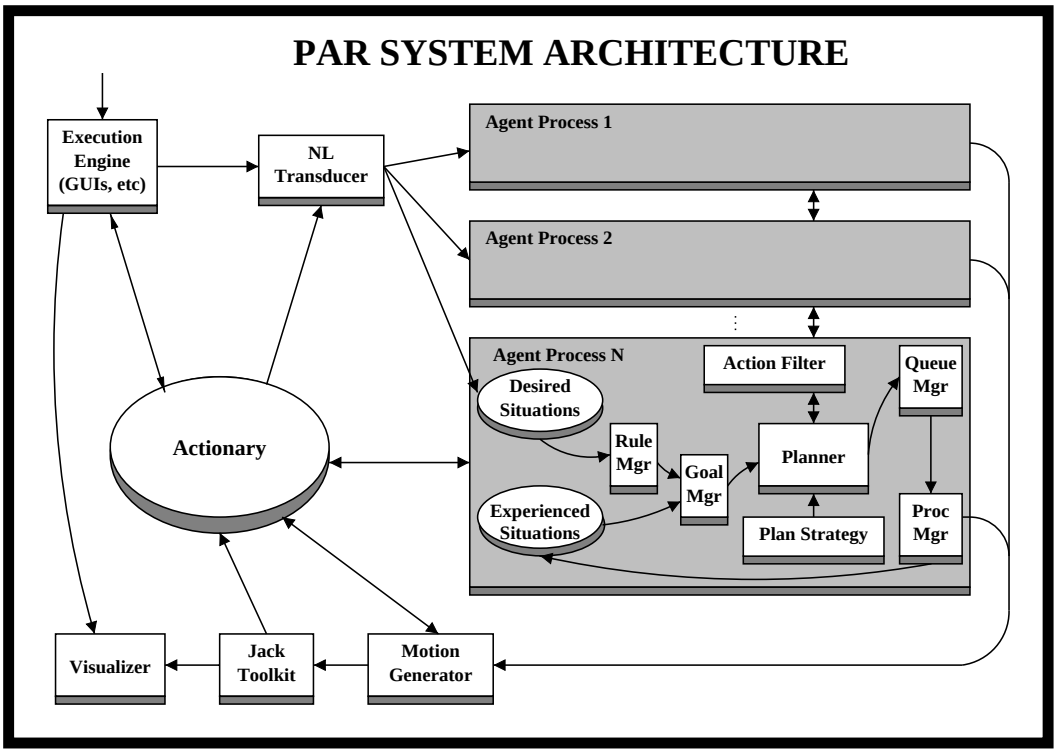

Figure 2: New Architecture Diagram

of syntactic ambiguity (such as prepositional phrase attachment ambiguity in the sentence "If Trainee 1 aims at someone with a bazooka, take cover," where either the *someone* has a bazooka or the *aiming* is with a bazooka), the parser can preserve this ambiguity in the situation calculus expression, so that the appropriate interpretation may be determined by later circumstances in the environment (depending on who Trainee 1 aims at, and what he aims with, for example). This environment-based disambiguation allows the interface to handle long sentences without having to maintain complex, domain-dependent disambiguation heuristics or a large statistical database.

Within the *Agent Process*, the situation calculus expressions are stored as a list of *Desired Situations* along with other situation calculus expressions generated from previous instructions. The *Desired Situations* set collectively represents the desired future behavior of the agent. The *Rule Manager* uses the *Actionary* to evaluate the expressions in the *Desired Situations* set and sends them to the *Goal Manager* when there are any changes. All actions successfully completed by the agent are stored as a list of *Experienced Situations* . The *Goal Manager* uses the *Desired Situations* and the *Experienced Situations* to determine new goals. It then sends the new goals to the *Planner*.

The *Planner* solves an abstract planning problem where the initial state comes from the object hierarchy in the *Actionary*. The *Planner* begins by evaluating the situation calculus expressions and querying the Actionary for PARs which will meet the goals. Before the query results are sent to the *Planner* , the *Action Filter* first eliminates some of the PARs from the set of results based on the agent's capabilities, and sorts the remaining PARs based on the agent's characteristics. For example, the *Planner* may ask for all of the translatory PAR actions. If one of the actions returned is "walk" and the agent is an infant not capable of walking, "walk" will be removed from the set. Also, if the agent is a happy child, the *Action Filter* may prefer "skip" to "walk," but if the agent is a prominent, business woman "walk" would be preferable. The ordering of actions is based on the agent's role, perception of the situation, culture, personality, and emotions. During the planning process the *Planner* determines which actions to include in the plan based on the list of actions sent from the *Action Filter*, it determines the structure of the plan based on parameters of the *Plan Strategy*, and it eliminates some of the possible plans, based on the constraints found in the situation calculus expressions. The *Planner* instantiates the PARs with information about the specific agents and objects involved in the actions as well as information about how the actions should be performed. This includes information not available in the natural language directives. For example, a directive such as "Walk across the street," does not include information about the path, but this information must be provided before the action can be executed [16].

These instantiated PARs are processed by the *Queue Manager* and *Process Manager* and then sent to the pre-registered *Motion Generators* for the actual execution of the action. It is within these motion generators that all the parameters of the IPAR are finally resolved. These generators access the Actionary for information on agents, objects, and the current

state of the environment. During the execution of the action, the motion generators update the Actionary with the status of the ongoing action. At the end of the action, they use post assertions to update the Actionary with the various changes in the environment. After successful completion of the action, the *Process Manager* adds the relevant information about the action to the list of *Experienced Situations*.

5 Agent Model

The PAR architecture contains *Agent Processes*, which contain high-level mechanisms for the agents to perform actions. In order for an agent to perform an action, it needs only to place the appropriate IPAR on its action queue. It also allows multiple agents in the environment to communicate with each other through message passing. By providing these mechanisms, the architecture allows the designer of autonomous agents for virtual environments to concentrate on other aspects of the design like creating believable agents with individuality.

The agent, whether acting autonomously or being controlled by a user, acts under the influence of *many* variables. These are a few of the parameters of an agent that might influence its behavior:

- Roles are learned, generalized guidelines for behavior. Roles are more about expectations and perceptions than what is actually said or done, and an agent may have more than one role.
- Cultural associations condition the way an agent perceives, acts, and reacts. Culture helps in determining the importance and immediacy of the activities of life.
- Situation is a collection of variables that characterize the agent's model of the world. A key idea is that the situation representation can vary over the agents.
- Personality is a pattern of behavioral, temperamental, emotional, and mental traits for an individual. The way an agent perceives, acts, and reacts is influenced by its personality.
- Emotions are generated through the agent's construal of and reaction to the consequences of events, actions of agents, and aspects of objects. Emotions effect not only which actions the agent chooses to perform, but also the manner in which they are performed.

6 Manner

One essential component of agent behavior is adequate body and gesture movements. In a real-time procedural animation system, it is even more important to have the performance

of appropriate movements parametrically controlled by the agent's internal states. One mechanism for doing this is through the *manner* field of the PAR. Manner describes how a particular movement is performed within a space of reasonable variations in *Effort* and *Shape* dimensions from Laban Movement Analysis [4].

Our design for the expression of manner is based on the EMOTE Model [2, 10]. EMOTE allows the specification of Effort and Shape parameters used to modify independently defined limbs and torso movements. The underlying movements are specified through key time and pose information. Key pose information may be generated by an external process; for example, a procedurally generated motion, or motion captured from live performance. Given an underlying movement, manner information can be used to compute specific Effort and Shape parameters, which in turn are used to vary the performance of the motion. Effort and Shape parameters are expressed numerically in the range of -1.0 and +1.0. The extreme values correspond to extreme attitudes. For example, a +1.0 value in Effort's Weight factor corresponds to a Strong movement; a -1.0 value in Shape's Vertical dimension corresponds to a Rising movement. Effort parameters are translated further into low-level parameters, while Shape parameters are used to modify key pose information. In addition, different Effort and Shape parameters may be specified for different parts of the body involved in the same movement. Moreover, Effort and Shape parameters may be phrased across a set of movements, similarly to communicative phrasing with an expressive content consonant with the principal utterance. Working from movements stored as key time and pose information, the manner parameters can execute the movements to speed, space, force, or fluidity and these in turn can be related back to the internal states, personalities and emotions of the agent.

7 Building Parameterized Action Representations from Observation

A PAR can be created either from a GUI or from observing a real human performing the task. For virtual humans, motion capture and procedural animation are two of the popular approaches for action generation. We have built the CaPAR (Captured PARs) interactive system [6] which combines both these approaches to generate agent-size neutral representations of meaningful actions in the form of a PAR. Just as a person learns new complex physical tasks by observing another person do it, the CaPAR system observes a single trial of a human performing some complex task involving interaction with self or other objects in the environment and automatically generates semantically rich information about the action. This information can be used to generate similar constrained motions for agents of different sizes.

We use motion capture techniques to acquire human movements of a complex action. By computing motion zero-crossings and geometric spatial proximities, we isolate significant

events, abstract both spatial and visual constraints from an agent's action, and segment continuous motion sequences. By breaking up a given complex action into several simpler subactions and analyzing them independently, we build individual PARs for each of them. We can then combine several PARs into one complex PAR representing the original activity. Within each motion segment, we extract semantic and style information about the action. The style information allows us to generate the same constrained motion in other differently sized virtual agents by copying the end-effector velocity profile, by following a similar end-effector trajectory, or by scaling and mapping force interactions between the agent and an object. The extracted style information is stored in the corresponding agent and object models. The semantic information is stored in a PAR as a high level description of an action. The CaPAR system generates PAR attributes at both the low-level (constraints, style of execution, etc) and the high-level (default preparatory specifications and termination conditions of the observed action).

8 Directions

We believe that Natural Language instructions, parameterized actions, EMOTE, and observed movement inputs will provide effective methods for the on-line creation of behaviors for embodied animated agents. Preliminary implementations are all these components exist and are mostly integrated. The EMOTE system will be brought into the PAR architecture and the planner will be integrated before the end of 2000. The virtual checkpoint simulation is already running and we plan to add a live participant to interact with the other virtual autonomous agents within a year. In addition, another scenario is being developed based on a virtual maintenance technician validating task instructions on an actual vehicle. This application will demonstrate the re-usability and extensibility of the Actionary. The planner will be exercised, too, as tasks may fail for any number of reasons such as presence of hazards, lack of access, ordering constraints, and so on.

Overarching desires for this effort include the better appearance in the human forms, more detailed environments, and faster execution of the whole system. We would like to have other human models controlled by the motion generators in the system, and are examining other possibilities that might be compatible with the EAI Jack Toolkit interface API. For more detailed environments we are looking into full vehicle CAD models and more elaborate urban models to expand our two primary applications in virtual training and equipment maintenance. The main issues are availability of suitable data and its polygon volume. Game engines are an interesting alternative for the OpenGL graphics interface we presently use. Finally, although the interpreted Python software gives us a performance hit, it has proven invaluable in software development and interactivity in the system as a whole. Faster computing platforms are inevitable.

A more specific research direction is connecting spatial referents in the textual instruc-

tions with the objects or parts of objects in the world [8]. We are investigating a combination of propositional and geometric reasoning to ground spatial terms (prepositional phrases) in the current or future contexts. We have already studied directional prepositions as predictors of general access or locomotion paths [16]. Such instructions are also likely to interact with the visual and attentional aspects of the agent [2].

Finally, we are leaving aside the input requirements for an autonomous agent who interacts with a live player. The issues of gesture recognition, facial expression recognition, and speech input are all important components for a successful virtual reality training experience with interacting artificial agents. We expect to adapt some of the existing techniques in this area to provide inputs for situation assessment in our agents.

9 Acknowledgments

This research is partially supported by U.S. Air Force F41624-97-D-5002, Office of Naval Research K-5-55043/3916-1552793, AASERTs N00014-97-1-0603 and N0014-97-1-0605, DARPA SB-MDA-97-2951001, NSF SBR-8900230 and IIS-9900297, Army Research Office ASSERT DAA 655-981-0147, NASA NRA NAG 5-3990, and Engineering Animation Inc.

References

[1] N. Badler, R. Bindiganavale, J. Allbeck, W. Schuler, L. Zhao, and M. Palmer. A parameterized action representation for virtual human agents. In *Embodied Conversational Agents*, pages 256–284. MIT Press, 2000.

[2] N. Badler, D. Chi, and S. Chopra. Virtual human animation based on movement observation and cognitive behavior models. In *Proceedings of Computer Animation*, pages 128–137, 1999.

[3] N. Badler, M. Palmer, and R. Bindiganavale. Animation control for real-time virtual humans. *Comm. of the ACM*, 42(8):65–73, Aug. 1999.

[4] I. Bartenieff and D. Lewis. *Body Movement: Coping with the Environment.* Gordon and Breach Science Publishers, New York, 1980.

[5] R. Bindiganavale, William Schuler, Jan M. Allbeck, Norman I. Badler, Aravind K. Joshi, and Martha Palmer. Dynamically altering agent behaviors using natural language instructions. In *Proceedings of Autonomous Agents*, pages 293–300, 2000.

[6] Rama Bindiganavale. *Building Parameterized Action Representations from Observation.* PhD thesis, CIS, University of Pennsylvania, 2000. In preparation.

[7] T. Capin, I. Pandzic, N. Magnenat-Thalmann, and D. Thalmann. *Avatars in Networked Virtual Environments*. Wiley, Chichester, England, 1999.

[8] Marc Cavazza and Ian Palmer. A prototype for natural language control of video games. In *Proceedings of VSMM'99*, pages 36–45, 1999.

[9] D. Chapman. *Vision, instruction, and action*. PhD thesis, Massachusetts Institute of Technology, 1990.

[10] D. Chi, M. Costa, L. Zhao, and N. Badler. The EMOTE model for effort and shape. In *ACM SIGGRAPH Computer Graphics*, 2000. To appear.

[11] S.B. Huffman and J.E. Laird. Flexibly instructable agents. *Journal of Artificial Intelligence Research*, 3:271–324, 1990.

[12] M. Kallmann and D. Thalmann. Direct 3d interaction with smart objects. In *Proceedings of ACM VRST'99*, 1999.

[13] Karin Kipper, Hoa Trang Dang, and Martha Palmer. Class-based construction of a verb lexicon. In *Proceedings of the Seventh National Conference on Artificial Intelligence (AAAI-2000)*, Austin, TX, July-August 2000.

[14] Karin Kipper and Martha Palmer. Representation of actions as an interlingua. In *Proceedings of the Third Workshop on Applied Interlinguas, held in conjunction with ANLP-NAACL 2000*, Seattle, WA, April 2000.

[15] H. Noser and D. Thalmann. A rule-based interactive behavioral animation system for humanoids. *IEEE Transactions on Visualization and Computer Graphics*, 5(4):218–307, 1999.

[16] Yilun Dianna Xu and Norman I. Badler. Algorithms for generating motion trajectories described by prepositions. In *Proceedings of Computer Animation*, pages 33–39, 2000.