Parameterized Action Representation and Natural Language Instructions for Dynamic Behavior Modification of Embodied Agents

Norman I. Badler, Rama Bindiganavale, Jan Allbeck, William Schuler, Liwei Zhao, Seung-Joo Lee, Hogeun Shin, and Martha Palmer

Center for Human Modeling and Simulation
Department of Computer and Information Science
University of Pennsylvania
Philadelphia, PA 19104-6389
badler@central.cis.upenn.edu

Abstract

We introduce a prototype for building a strategy game. A player can control and modify the behavior of all the characters in a game, and introduce new strategies, through the powerful medium of natural language instructions. We describe a Parameterized Action Representation (PAR) designed to bridge the gap between natural language instructions and the virtual agents who are to carry them out. We will illustrate PAR through an interactive demonstration of a multi-agent strategy game.

Introduction

People communicate to share experiences, provide information, elicit responses, negotiate agreements, and modify behaviors. Instructions form an important communication subspace that addresses physical or behavioral actions performed by or with other people. Instructions provide a rich environment in which people must understand and carry out meaningful actions. Naturally, those actions must be done in a physical context and may vary across people or across situations. When we substitute humanlike characters as partners in instruction interpretation, a suitable computational framework must be used to transform language commands into visualized actions.

A consequence of studying and understanding person-to-person instructions is that we can use the same language-based instructions for embodied agent communications. Eventually, we can use this identical mechanism for agent-to-agent communication as well. A successful design and implementation of instruction understanding will open communication channels that blur the boundary between interactions with real and virtual beings.

Our research therefore addresses the problem of making embodied characters respond to verbalized commands in a context-sensitive fashion. Character movements are executed by motion generators which are themselves scheduled by a non-linear animation system of parallel finite state machines (PaT-Nets). To support both language understanding and animation, we have defined a Parameterized Action Representation (PAR) that instantiates and executes appropriate PaT-Nets. We conclude this very brief exposition with an example showing how character behavior can be dynamically modified by standing orders while the simulation is actually running.

Smart Avatars

In real-time applications, avatars (Wilcox 1998) are character representations driven directly by a real person. For movements not based on live performance, computer programs have to generate procedurally the right sequences and combinations of parameters to create the movements' desired actions. An embodied character that acts from its own motivations is often called an agent. We call an avatar controlled via instructions from a live participant a ''smart avatar." Parameters for its actions may come from the instruction itself, from the local object context, and from the avatar's available capabilities and resources. We have explored the contextual control of embodied agents, avatars, and smart avatars in a number of experiments including:

* Two-person animated conversation in Gesture Jack (Cassell et al. 1994).

* Medic interventions and patient physiological interactions in MediSim (Chi et al. 1995).

* Multi-parameter game behaviors in Hide & Seek (Badler et al. 1996).

* A real-time animated Jack Presenter (Noma and Badler 1997; Zhao 1998).

* Multi-user Jack-MOO virtual worlds (Shi et al. 1999).

* Virtual environment simulation using parameterized action representation (Badler, Palmer, and Bindiganavale 1999).

In the last two systems, we began to explore an architecture for interacting with characters that was solely language-based in order to explicitly approach a level of interaction comparable to that between real people. We focused on instructions for physical action to bound the problem, to enable interesting applications, and to refine a representation bridging natural language (NL) and embodied action.

Levels of Architectural Control

Building a character model that admits control from sources other than direct animator manipulations requires an architecture that supports higher-level expressions of movement. Although layered architectures for autonomous beings are not new (Brooks 1989; Johnson and Rickel 1997; Zeltzer 1990), we have found that a particular set of architectural levels seems to provide efficient localization of control for both graphics and language requirements. Our multi-level architecture is grounded in typical graphical models, articulation structures, and motion generators. The higher architectural levels organize these skills with parallel automata, use a conceptual representation to describe the actions a character can perform, and finally create links between natural language and action animation.

Parallel Transition Networks

Our parallel programming model is called Parallel Transition Networks, or PaT-Nets (Badler, Phillips, and Webber 1993). Other character animation systems, including Motion Factory's Motivate and New York University's Improv (Perlin and Goldberg 1996), have adopted similar paradigms with alternative syntactic structures. In general, network nodes represent processes. Arcs connect the nodes and contain predicates, conditions, rules, and other functions that trigger transitions to other process nodes. Synchronization across processes or networks is made possible through message passing or global variable blackboards to let one process know the state of another process.

PaT-Nets are effective programming tools but must be hand-coded in C++. No matter what artificial language we invent to describe human actions, it is not likely to represent exactly the way people conceptualize a particular situation. If we want human-like control, then we need a higher-level representation to capture additional information, parameters, and aspects of human action. We create such representations by incorporating natural language semantics into a Parameterized Action Representation.

Language-Based Control

Even with powerful motion generators and PaT-Nets to invoke them, we still have to provide effective user interfaces to control, manipulate, and animate characters. We would also like to be able to instruct these characters while a simulation is running, so a user could dynamically refine an avatar's behavior or react to simulated stimuli without having to undertake a lengthy off-line programming session.

One promising and relatively unexplored option for giving run-time instructions to characters is a natural language interface. After all, instructions for real people are given in natural language, augmented with graphical diagrams, and, occasionally, live or previously generated animations. Recipes, instruction manuals, and interpersonal conversations all use natural language as a medium for conveying information about processes and actions (Badler et al. 1990; Badler, Phillips, and Webber 1993; Webber et al. 1995; Huffman and Laird, 1995). A natural language interface should be powerful enough to express conditional instructions and hypothetical situations, and it should be simple enough to use in a real-time application with little or no substantial formal user training.

Although the problem of parsing unrestricted text is by no means solved, broad-coverage grammars do exist that can account for many of the most common natural language constructions (Xtag, 1995). If we assume our interface will only be used for giving instructions grounded in an established virtual environment, we can ignore many of the problems of unknown words and syntactic constructions that make unrestricted text so hard to parse. This limited domain should also constrain syntactic and semantic ambiguity enough for existing statistical disambiguation techniques to produce reliable interpretations.

We are not advocating that animators throw away their graphical tools, only that natural language offers a communication medium we all know and can use to efficiently formulate run-time instructions for virtual characters. Some aspects of some actions are certainly difficult to express in natural language (such as precise locations and orientations of objects), but the availability of a language interpreter can make the user interface more closely simulate real interpersonal communication. Our goal is to build smart avatars that understand what we tell them to do in the same way humans follow instructions. These smart avatars have to be able to process natural language instructions into a conceptual representation that

can be used to control their actions. This representation is what we refer to as a Parameterized Action Representation.

Parameterized Action Representation (PAR)

A PAR (Badler et al. 1997) gives a description of an action. The PAR (Figure 1) has to specify the agent of the action as well as any relevant objects and information about path, location, manner, and purpose for a particular action. There are linguistic constraints on how this information can be conveyed by the language; agents and objects tend to be verb arguments, paths are often prepositional phrases, and manner and purpose might be in additional clauses (Palmer, Rosenzweig, and Schuler 1998). A parser and translator map the components of an instruction into the parameters or variables of the PAR, which is then linked directly to PaT-Nets executing the specified movement generators. Natural Language often describes actions at a high level, leaving out many of the details that have to be specified for animation (Narayanan 1997). The PAR bridges the gap between natural language and animations. More detail on the PAR and the architecture that executes them may be found in (Badler, Palmer, and Bindiganavale 1999).

PAR for Virtual Environment Training

The implemented "Virtual Environment for Training" is a prototype of a game of military strategies, in which a player controls a group of soldiers. Each soldier in the game has a different role to play, can make individual decisions, and react to changes in environment and the actions of others. Each civilian in the game can have a different personality and hence react differently to similar situations. The player monitors all the soldiers' actions and gives them explicit natural language instructions to modify their behaviors and actions, thus changing the strategy of the game.

The interactive demonstration scenario is a military checkpoint, with three soldiers whose job is to apprehend suspected spies. A separate agent process controls each of the soldiers. A process simulator (also an agent process) generates and controls vehicles, and operates traffic lights. As each vehicle approaches the checkpoint, one of the soldiers checks each civilian male driver's identification. If there is a match, the soldier is supposed to draw his weapon and take the driver into custody. All others are allowed to pass through the checkpoint.

PAR

participants:	**agent**: *AGENT* **objects**: *OBJECT list*
start:	*TIME*
applicability conditions:	*CONDITION boolean-expression*
preparatory specification:	**condition**: *CONDITION boolean-expression* **action:** *PAR*
subactions:	*PAR* ***constraint-graph***
execution steps:	primitive/complex
complex:	*subactions*
core semantics:	**motion**: *MOTION* **force**: *FORCE* **path**: *PATH* **purpose**: *PURPOSE* **termination**: *TERMINATION* **duration**: *DURATION* **manner**: *MANNER*
postassertions:	{assertions}
parent action:	*PAR*
previous action:	*PAR*
concurrent action:	*PAR*
next action:	*PAR*

MOTION

object:	*OBJECT*
caused:	*IPAR*
translational:	*BOOLEAN*
rotational:	*BOOLEAN*

FORCE

object:	*OBJECT*
point of contact:	*OBJECT LOCATION*

PATH

direction:	*DIRECTION*
start:	*LOCATION*
end:	*LOCATION*
distance:	*LENGTH*

PURPOSE

achieve:	*CONDITION* ***boolean-expression***
generate:	*PAR*
enable:	*PAR*

TERMINATION: *CONDITION* ***boolean-expression***

DURATION: *LENGTH*

MANNER

space:	*REAL*
weight:	*REAL*
time:	*REAL*
flow:	*REAL*

Figure 1. PAR Attributes

During this process, the soldiers may make strategic errors which cause them to get shot by enemy spies. In order to refine the checkpoint strategy, the player must give the checkpoint soldiers new standing orders so these errors won't happen again.

Standing orders are instructions from the player to correct the soldiers' errors. In the first situation, the soldier does not take cover while drawing his weapon at the driver and so gets shot by him. To correct this, the player gives the following standing order to that soldier: "When you draw your weapon at the driver, take cover from the driver behind your drum." This standing order is immediately parsed and stored as a Python script (Lutz 1996) in the rule-table. So, in the next trial of the simulation, as soon as the soldier draws his weapon, the standing order (now rule) forces him to take cover behind his drum correctly.

The player also notices that when one of the soldiers draws his weapon, the other two soldiers remain standing still, which leaves them vulnerable, so the standing order is given: "If Soldier1 draws his weapon at the driver, draw your weapon at the driver and take cover from the driver behind your drum." The command "take cover" takes two oblique arguments - the potential threat (in this case, the driver), and the desired cover (the steel drum). When this PAR is executed, the soldier moves to a place where the drum intersects the path between himself and the driver. But since the "take cover" action is parameterized at this high level, the soldier could be instructed to take cover from virtually any object (say, from one of his companions), behind any other object (say, behind the spy's car), and the simulation would accommodate it.

After the first spy is taken away, the scenario continues until a second suspect enters the checkpoint. This time, however, the suspect draws a gun as soon as the first soldier asks him for his identification, and shoots the soldier before he can react. Observing that the soldiers on the passenger side of the car could have seen the driver reach for the gun, the user gives two additional standing orders for soldiers 2 and 3:

* "When there is a driver, watch the driver."

* "If the driver reaches for a gun, warn Soldier1."

Once again the simulation is replayed and whenever there is a driver in the car at the checkpoint, the above standing orders force soldiers 2 and 3 to watch the driver. Moreover, when the driver reaches for his gun, the rules force those soldiers to warn the first soldier, before the driver can fire. So, the first soldier has time to draw his own weapon and take cover. Since all the previous orders are still in the system's memory, soldiers 2 and 3 also draw their weapons at the driver as soon as they see soldier 1 do so, and all three still take cover when they draw their weapons. The driver, outnumbered, quickly surrenders, and the soldiers successfully complete the exercise.

The "watch" action is classified as a preemptive action and so its PAR has a lower priority. This means that whenever the soldiers need to execute other actions, their agent processes will preempt the "watch" from their queues and execute the other actions. But after the other actions have been completed, if there is still a driver at the checkpoint, the rule resulting from the standing order "When there is a driver, watch the driver" will again force the soldiers to watch the driver. This results in a completely natural-looking correct scenario where the soldiers are always cautiously watching the driver. If they are interrupted and forced to do something else, they quickly finish that task and resume watching.

Discussion

The PAR architecture and its implementation is intended to provide a test bed for real-time smart avatars and agents who work, communicate, and manipulate objects in a synthetic 3-D world. Our goal is to make interaction with these embodied characters the same as with live individuals. We have focused on language as the medium for communicating instructions and finite state machines as the controllers for agent or object movements.

The structure described here is the basis for a new kind of dictionary we call an Actionary™. A dictionary uses words to define words. Sometimes it grounds concepts in pictures and perhaps even sounds and video clips. But these are canned and not parameterized - flexible and adaptable to new situations the way that words function in actual usage. In contrast, the Actionary uses PAR and its consequent animations to ground action terms. It may be viewed as a 3-D (spatialized) environment for animating situated actions expressed in linguistic terms. The actions are animated to show the meaning in context, that is, relative to a given 3-D environment and individual agents. Additionally, the Actionary enables low-bandwidth communication of instructions across a distributed multi-person simulation system since only the high-level NL commands need to be transmitted rather than low-level character movement data.

An instruction understanding system, based on natural language inputs, an Actionary translation, and an embodied agent could provide a non-programming interface between real and virtual people. We can describe tasks for others and see them carried out, whether they are real or virtual participants. Thus the door is opening to novel applications for embodied agents in games, job training, team coordination, manufacturing and maintenance, education, and emergency drills. As the Actionary grows, new applications should become ever easier to generate. And just as our human experience lets real people take on new

tasks, so too should embodied characters be adaptable to new environments, new behaviors, and new instructions.

Acknowledgments

This research is partially supported by U.S. Air Force F41624-97-D-5002, Office of Naval Research K-5-55043/3916-1552793, DURIP N0001497-1-0396, and AASERTs N00014-97-1-0603 and N0014-97-1-0605, DARPA SB-MDA-97- 2951001, NSF IRI95-04372, SBR-8900230, and IIS-9900297, Army Research Office ASSERT DAA 655-981-0147, NASA NRA NAG 5-3990, and Engineering Animation Inc.

References

Badler, N., M. Palmer, and R. Bindiganavale. 1999. Animation control for real-time virtual humans. *Communications of the ACM* 42(7):65--73.

Badler, N., C. Phillips, and B. Webber. 1993. *Simulating humans: Computer graphics animation and control*. New York: Oxford University Press.

Badler, N., B. Webber, W. Becket, C. Geib, M. Moore, C. Pelachaud, B. Reich, and M. Stone. 1996. Planning for animation. In N. Magnenat-Thalmann and D. Thalmann, eds., *Interactive Computer Animation*, 235-262. New York: Prentice-Hall.

Badler, N., B. Webber, J. Kalita, and J. Esakov. 1990. Animation from instructions. In N. Badler, B. Barsky, and D. Zeltzer, eds., *Making them move: Mechanics, control, and animation of articulated figures*, 51--93. San Francisco: Morgan-Kaufmann.

Badler, N., B. Webber, M. Palmer, T. Noma, M. Stone, J. Rosenzweig, S. Chopra, K. Stanley, J. Bourne, and B. Di Eugenio. 1997. Final report to Air Force HRGA regarding feasibility of natural language text generation from task networks for use in automatic generation of Technical Orders from DEPTH simulations. Technical report, CIS, University of Pennsylvania.

Brooks, R., A robot that walks: Emergent behaviors from a carefully evolved network. *Neural Computation* 1(2):253--262.

Cassell, J., C. Pelachaud, N. Badler, M. Steedman, B. Achorn, W. Becket, B. Douville, S. Prevost, and M. Stone. 1994. Animated conversation: Rule-based generation of facial expression, gesture and spoken intonation for multiple conversational agents. In *Computer Graphics*, 413--420. New York: ACM SIGGRAPH.

Chi, D., B. Webber, J. Clarke, and N. Badler. Casualty modeling for real-time medical training. *Presence* 5(4):359--366.

Huffman, S. and J.E. Laird, 1995. Flexibly Instructable Agents. *Journal of Artificial Intelligence Research* 3: 271—324.

Johnson, W.L., and J. Rickel. 1997. Steve: An animated pedagogical agent for procedural training in virtual environments. *SIGART Bulletin* 8(1--4):16--21.

Lutz, M.1996. *Programming Python*. Sebastapol: O'Reilly.

Narayanan, S. 1997. Talking the talk is like walking the walk. In *Proceedings of the 19th Annual Conference of the Cognitive Science Society*, 000--000. Palo Alto, Calif.

Noma, T., and N. Badler. 1997. A virtual human presenter. In *IJCAI '97 Workshop on Animated Interface Agents*, Nagoya, Japan.

Palmer, M., J. Rosenzweig, and W. Schuler. 1998. Capturing motion verb generalizations with synchronous tag. In P. St. Dizier, ed., Predicative forms in NLP. *Text, speech, and language technology series*, (250--277). Dordrecht, The Netherlands: Muwer Press.

Perlin, K., and A. Goldberg. 1996. Improv: A system for scripting interactive actors in virtual worlds. In *Computer Graphics*, 205-216. New York: ACM, SIGGRAPH.

Shi, J.,T. J. Smith, J. Granieri, and N. Badler. 1999. Smart avatars in JackMOO. In *Proceedings of IEEE Virtual Reality*, 156--163. Los Alamitos: IEEE Computer Society.

Webber, B.,N. Badler,B. Di Eugenio, C. Geib, L. Levison, and M. Moore. 1995. Instructions, intentions and expectations. *Artificial Intelligence Journal* 73:253--269.

Wilcox, S. K. 1998. *Web developer's guide to 3D avatars*. New York: Wiley.

XTAG Research Group, 1995. A lexicalized tree adjoining grammar for English. Technical Report, University of Pennsylvania.

Zeltzer, D. 1990. Task-level graphical simulation: Abstraction, representation, and control. In N. Badler, B. Barsky, and D. Zeltzer, eds., *Making them move: Mechanics, control, and animation of articulated figures*, 3--33. San Francisco: Morgan-Kaufmann.

Zhao, L., and N. Badler. 1998. Gesticulation behaviors for virtual humans. In *Proceedings of Pacific Graphics*, 161--168. Los Alamitos: IEEE Computer Society.