

Extending XMSF with a Parameterized Action Representation for Agent Behaviors

Norman Badler

Jan Allbeck

Matt Beitler

Kathy Feng

Honghui Lu

Center for Human Modeling and Simulation
Computer and Information Science Department

200 S. 33rd St.

Philadelphia, PA 19104-6389

Tel: (215) 898-5862

Fax: (215) 573-7453

badler@seas.upenn.edu

When creating a framework for distributed or networked simulations, many design dimensions must be considered: bandwidth, synchronization, agent autonomy, agent control, latency, visualization, and interfaces¹. Often these considerations are diametrically opposed. We must balance, for example, the amount of control that we have over agents in the simulation with the amount of bandwidth that we have to control them. Early networked simulations broadcast position and orientation data over the network to every agent at every frame or clock tick. This enormous amount of information overwhelmed the available bandwidth, but gave the simulation designers great control. Later predictive methods such as dead reckoning were used to limit the packet frequency requirements and thus better utilize bandwidth, but at the expense of accuracy. Advancements in networking techniques and hardware have increased bandwidth, but our expectations of simulations have also increased. Emerging research techniques from computer graphics and artificial intelligence can be applied to building a smarter framework for distributed simulations.

Whether using a client-server or peer-to-peer architecture, packets describing agent actions must be formulated, sent, received, and interpreted. Until recently, animation interpreters had to be simplistic: consider, for example, the limited state control afforded to DI-Guy from Boston Dynamics. The autonomy of agents was limited – mostly to repetitive actions such as walking, running, or crawling – or motion captured units such as firing a weapon or falling. In either case motion control had to be explicit and fine-grained. Computational techniques, however, have advanced such that agents are not only acquiring the ability to perform individual actions on their own; they are also able to perform a series of contextually variable actions or behaviors autonomously. Such actions may include reaching for objects, moving head and eyes to attend to interesting nearby events, and adjusting locomotion to avoid obstacles and shift gaits as needed². Our goal is to develop action representations that can be explored by the underlying network to reduce communication, and at the same time guarantee consistent world state among distributed hosts.

Increasing the autonomy of agents can result in a decrease in necessary bandwidth. Consider, for example sending an agent's frame-by-frame joint angles for all of the actions necessary to animate an agent entering a building versus simply sending it a string: *enter the building*. Naturally with the detailed method the simulation has fine-grained control over the agent's performance, while with the simple instruction the simulation appears have little control. That is not the case, however. If the situation requires that the agent enter the building *carefully, through the blue door, or while watching the window above him*, there is no simple method to modify the detailed joint or motion capture data. If the actions are suitably parameterized, such modifiers may be carried immediately in the instruction itself and interpreted locally by the agent. Instructions between people carry information that both parties can use to drive implementing behaviors. Moreover, poor instructions may result in misunderstanding and incorrect actions. Simulations based at the instruction level may help expose potentially negative communication practices during a training session.

We are not arguing that *natural language* instructions should be used as the basis of a simulation packet structure; that would still require too much processing capability and interpretation in an agent, and the state of natural language processing is not quite ready for that role. But we can learn from this form of human-level communication some attributes that an efficient and effective distributed simulation packet structure might contain. Over the last few years, we have been developing a Parameterized Action Representation (PAR) based jointly the information requirements necessary to animate an embodied computer graphics agent as well as to represent the semantics of natural language action verbs, adverbs, and prepositions³. ***We propose codifying PAR in the XMSF⁴ framework and using it as the fundamental communication unit for large-scale distributed simulations.***

A PAR gives a complete description of an action. The PAR 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. PAR has been specifically designed to bridge the gap between natural language and animation. Natural language describes actions at a high level, leaving out many of the details that have to be specified for animation. Uninstantiated PARs (PARs lacking context specific parameter values) are stored in action dictionaries or databases called Actionaries in agents of the simulation. Table 1 shows a small portion of PAR as an XML DTD. Attached is a more complete description of PAR. Prior to a simulation run, all of the uninstantiated PARs are loaded into the agents' Actionaries. (For efficiency, an Actionary with default parameters may be common to all agents, who only store local parameters or exceptions.) This inheritance structure helps minimize the amount of information that must be sent over the network at run time. While the simulation is running, PARs can be exchanged over the network both as information packets and instructions for agents. Instruction PAR packets would be received by the intended agent, compared with the corresponding PAR stored in the agent's Actionary, and merged into an instantiated PAR for execution.

Consider the example above: "*enter the building carefully, through the back door, while watching the window above*" to illustrate the function of the PAR. The primary

instruction is *enter*, but there is little explicit specification of the agent’s path (though the target is given), there is no indication on how to open the door (though the specific “*back*” one desired is given)⁵, and there are other constraints on manner (*carefully*)⁶, reach (door knob), and attention (*watch the window*)⁷. An agent’s default action for entering a building may be *walking*, however, the adverb *carefully* might change *walking* into *sneaking*. Furthermore, “*through the back door*” would be used to influence path creation. These goals, attributes, and constraints must be converted to movements necessary to the action’s actual visible performance. PAR provides exactly the vehicle for carrying such information about an action. Unlike a PDU that describes minimal forward simulation parameters for an agent action, such as direction, velocity, and gait type, PAR offers a rich set of target, modifier, and constraint parameters for instructing an agent in a more goal-directed fashion.

```

<!ELEMENT PAR (parameterized_action)+>
<!ELEMENT parameterized_action (name, participants, applicability_conditions,
preparatory_specification, termination_conditions, post_assertion, during_conditions,
purpose, subactions, parent_action, previous_action, concurrent_action, next_action,
start, duration, priority, data, kinematics, dynamics, manner, adverbs)>
<! ELEMENT name (STRING)>
<! ELEMENT participants (agent-and-objects)>
<! ELEMENT kinematics (kinematics-specification)>
<! ELEMENT manner (manner-specification)>

```

Table 1: Sample PAR DTD

The PAR has to include information about applicability, preparatory, and termination conditions so that a run-time PAR interpreter can fill in the movements from the action specifications. PAR is, by definition, parameterized, because details of the action depend on the action’s participants, including agents, objects, and other attributes. Object descriptions must include the actions (PARs) that the object can perform (*affordances*) and what state changes they cause. The path, the number of steps it will take to get to the door, the hand used to open the door, and so on, depend on the context. With such a representation, agents can become increasingly autonomous. Moreover, if several agents are given the same PAR, they will naturally interpret the actions in their own context, thus yielding variations and additional behavioral richness.

When controlling agents over a network, action synchronization issues must be addressed. Generally, agents’ *movements* need not be tightly synchronized across the network, but *interaction* events need to be distributed to keep all instantiations or “ghosts” of an agent up-to-date. Creating agents that are more autonomous is likely to result in greater divergences. Penn’s MELD framework⁸ explores a solution to this problem in a peer-to-peer environment, but further solutions need to be investigated. We think that PAR provides an opportunity to synchronize agents by action rather than by low-level position and state.

The level of detail of objects has long been a subject of graphics research. Large scale, distributed simulations give us the opportunity to expand the level of detail concept to actions as well. Nearby actions involving objects may need to be enacted using inverse kinematics. At further distances, similar actions could be enacted by replaying motion

capture data. It is not necessary to *display* actions that are outside an agent's circle of influence⁹¹⁰. Nonetheless, other agents may still need to be aware of the action that an agent is performing, the consequences of the action, and whether or not the action was performed successfully. ***Thus PAR can be used to communicate agent activities even if those actions are not directly seen or even executed; PAR can be a cognitive representation for conveying action information between agents.*** PAR might even be used to define "need-to-know" multi-cast groups. PAR uses fields that may be loaded, modified, interpreted, and transferred like data packets, and are generally used as dynamic information objects. An additional bonus with PAR is that its language origins allow PAR contents to be output as a sentence, making PAR a convenient resource for After Action Summaries and Reviews.

Penn proposes a one-year project to:

- Develop an XML binding for PAR suitable for the XMSF environment: XPAR.
- Study and construct a rich object representation with object affordances.
- Rebuild Actionary for XPAR (creating a structure agnostic with respect to a specific database implementation)
- Integrate XPAR into the XMSF environment.
- Benchmark performance of XPAR in a typical simulation environment.
- Investigate multi-cast communication and synchronization with PAR 'packets'.
- Work with other XMSF partners to develop a working demonstration of XPAR/XMSF technology.

We are prepared to begin this project immediately; budget on request.

References

- ¹ S. Singhal and M. Zyda. Networked Virtual Environments: Design and Implementation. Addison-Wesley, 1999.
- ² N. Badler, R. Bindiganavale, J. Allbeck, W. Schuler, L. Zhao, and M. Palmer. "Parameterized Action Representation for virtual human agents," In J. Cassell, J. Sullivan, S. Prevost, and E. Churchill (eds.), Embodied Conversational Agents, MIT Press, 2000, pp. 256-284.
- ³ N. Badler, M. Palmer, and R. Bindiganavale. "Animation control for real-time virtual humans," *Communications of the ACM* 42(8), August 1999, pp. 64-73.
- ⁴ <http://www.movesinstitute.org/xmsf/xmsf.html>
- ⁵ B. Douville, L. Levison, and N. Badler. "Task level object grasping for simulated agents," *Presence* 5(4), pp. 416-430, 1996.
- ⁶ L. Zhao, M. Costa, and N. Badler. "Interpreting movement manner." Proc. Computer Animation 2000 Conference, IEEE Computer Society, Philadelphia, May 3-5, 2000, pp. 112-120.
- ⁷ S. Chopra-Khullar and N. Badler. "Where to look? Automating attending behaviors of virtual human characters," *Autonomous Agents and Multi-Agent Systems* 4(1/2), 2001, pp. 9-23.
- ⁸ J. Shi, N. Badler, and M. Greenwald. "Joining a real-time simulation: parallel finite-state machines and hierarchical action level methods for mitigating lag time," Proc. 9th Conference on Computer Generated Forces, Orlando, FL, May, 2000.
- ⁹ K. Morse, L. Bic, & M. Dillencourt. Interest Management in Large Scale Virtual Environments. *MIT PRESENCE-Teleoperators and Virtual Environments*, February 2000.
- ¹⁰ M. Stytz. "Distributed virtual environments," *IEEE Computer Graphics & Applications* 16(3), pp. 19-31, 1996.