MGLAIR Agents in a Virtual Reality Drama
CSE Technical Report 2005-08

Stuart C. Shapiro* Josephine Anstey† David E. Pape† Trupti Devdas Nayak*
Michael Kandefer*

Orkan Telhan†
University at Buffalo, The State University of New York
Buffalo, NY 14260
{shapiro|jranstey|depape|td23|mwk3|otelhan}@buffalo.edu

March 30, 2005

Abstract

We provide an overview of the use of intelligent agents, implemented in the new MGLAIR architecture, in a virtual reality drama. For us, a virtual reality drama is a scripted play in which the computational agents are actors who have copies of the script, and one human audience member has been drafted to be a participant, but doesn't have a copy of the script. The computational actors must improvise reactions to the human partpicipant's actions, but keep the play moving along in as close agreement to the script as possible. The goal is to provide the human participant with a specific emotional experience. We explicate this philosophy; outline the previously described GLAIR architecture; explain the introduction of an organization into modalities that results in the new MGLAIR architecture; describe our current VR drama, *The Trial, The Trail*; and discuss the implementation of our actor-agents. Our discussion of the actor-agents focuses on their abilities to react to triggers (cues), their performance of contingent actions that are off the main-line arc of the script, their use of timers to pace the drama, and the organization of the cast of actor-agents into a multi-agent system.

1 Virtual Reality Drama

Our goal is to produce immersive virtual reality experiences that engage a participant as a central protagonist in compelling interactive dramas. We use a passive stereo system to project life-sized three dimensional images onto a large screen in front of the participant, who has tracking sensors on her head and both hands.

Virtual reality and mixed reality have been shown to be powerful presentation media for interactive fiction (Bobick, Intille, Davis, Baird, Pinhanez, Campbell, Ivanov, Schutte, and Wilson 1999; Laurel, Strickland, and Tow 1998; Shaw, Staff, Farr, Adams, vom Lehm, Heath, Rinman, Taylor, and Benford 2000), and using artificial intelligence for the construction of interactive narrative is considered by many to be a sine qua non. However, how one defines an interactive story determines how AI is used. Some define an interactive story as one in which the user's interventions must substantially effect the story-line (Szilas 2003). Following structural analyses of narrative such as (Barthes 1974; Propp 1968), they base their AI system around narrative rules that will allow them to generate storylines on the fly (Charles, Lozano, Mead, Bisquerra, and Cavazza 2003). We do not believe that a user-controlled structural organization or a semi-autonomous organization is required for an emotionally powerful interactive work. Instead, we view our virtual reality drama as a scripted play in which the computational agents are actors who have copies of the script, and one human audience member has been drafted to be a participant, but doesn't have a copy of the script. The computational actors must improvise reactions to the human participant's actions, but keep the play moving along in as close agreement to the script as possible. The goal is to provide the human participant with a specific emotional

*Department of Computer Science and Engineering
†Department of Media Study

experience. The participant's engagement with the story-line is increased if other entities, real or computer-controlled, are "co-present" in the virtual world (Ibanez, Aylett, and Ruiz-Rodarte 2003; Fencott 2003).

Our intelligent agents work co-operatively with our basic interactive dramatic structure, the snare, as discussed by Anstey et al. (2004). Each snare explicitly attempts to move the user from one emotional state to the next along a psychological arc. Like a novel, play or film, we use the narrative context to evoke an emotional response. The role of the actor-agents is to support this context with their actions and dialog, and more importantly to stimulate the user's emotions by simulating emotions of their own. In each snare the user is explicitly or implicitly encouraged to perform an activity or activities that we can detect with our tracking system. The agents and the virtual environment itself may respond immediately to the user's actions, but we also use the data we collect to interpret the user's state of mind, given the current narrative context.

The snare has some similarity to the concept of "beats" described by Mateas and Stern (2004). Both beat and snare have a dramatic goal for a short period of interaction, start with an assumption about how the user will respond, include checks on the user's reaction, and plans to handle deviations from that response. There is a difference of emphasis between beat and snare, the beat organises and communicates a short dramatic event, the snare maneuver's the user into the performance of some action that will be revealing of her psychological state. Whatever the apparent dramatic goal, the snare always has an underlying psychological goal. While snares may be as short as beats, they may also be built up into snare sequences designed to create psychological reversals. They may be much longer, with shorter snares nested inside longer snares. Longer snares are more akin to the acts and scenes of conventional drama.

2 The MGLAIR Architecture

The actor-agents are implemented according to the MGLAIR architecture, which is a modification of the GLAIR (Grounded Layered Architecture with Integrated Reasoning) architecture (Hexmoor, Lammens, and Shapiro 1993; Hexmoor and Shapiro 1997; Shapiro and Ismail 2003). Although there are other layered agent architectures, few if any have the same layers as (M)GLAIR:

The Knowledge Layer (KL) is the layer at which "conscious" reasoning takes place. The KL is implemented by the SNePS knowledge representation and reasoning system (Shapiro and Rapaport 1992; Shapiro and The SNePS Implementation Group 2004), and its SNeRE (SNePS Rational Engine) acting subsystem (Kumar and Shapiro 1994a; Kumar and Shapiro 1994b; Kumar 1996; Shapiro and The SNePS Implementation Group 2004). SNePS, in turn, is implemented in Common Lisp.

The Perceptuo-Motor Layer, Sublayer a (PMLa) Contains the Common Lisp implementation of the actions that are primitive at the KL. PMLa is implemented in a way that takes into account the top-level design of the agents, but is independent of the implementation of the agents' bodies.

The Perceptuo-Motor Layer, Sublayer b (PMLb) Implements the functions of PMLa taking into account the particular implementation of the agents' bodies. PMLb uses IP sockets to communicate with the PMLc (see the below section on Modalities).

The Perceptuo-Motor Layer, Sublayer c (PMLc) is on the other end of the IP sockets from the PMLb. Here, messages from the PMLb are passed to the appropriate SAL object, and sensory messages are passed back to the PMLb.

The Sensori-Actuator Layer (SAL) controls the sensors and effectors of the agent's virtual body.

The Environment is built using the Ygdrasil (Pape, Anstey, Dolinsky, and Dambik 2003) virtual reality authoring system. Ygdrasil is a plugin-based framework for VR applications that is based on SGI's OpenGL Performer toolkit (Rohlf and Helman 1994) and the CAVE library (VRCO, Inc. 2005). A virtual world in this system is a scene graph containing models, constructed in a 3D modelling package, sounds, and nodes that contain behaviors for objects in the world. New, application-specific behaviors are implemented as plugins written in C++, or dynamically loaded scripts written in Python. The combined scene graph and plugin/script approach allows a complete world to be constructed piece by piece, tying together many simpler elements. Real-time dynamic behaviors and interaction make use of an event-based structure; messages are passed between nodes in response to events. Messages include actions such as loading models, playing sounds, and moving objects.

3 The Trial The Trail

Our current project is *The Trial The Trail*, an interactive drama designed for an immersive VR system, and built on techniques used in *The Thing Growing* (Anstey, Pape, and Sandin 2000). Imagine Tarkovsky's Stalker, crossed with Alice Through the Looking Glass, crossed with Monty Python and the Holy Grail. Now imagine embarking on a guided journey through this warped yet familiar landscape. As you proceed your actions and interactions are logged, interpreted psychologically, and used to determine the outcome of your quest.

The two main characters in *The Trial The Trail* are Patofil and Filopat. As they explain the quest and introduce and take part in a series of absurdist challenges, they take up positions relevant to the psychological terrain of the drama. Patofil is reckless and insouciant, believes the journey is more important that the arrival, and is dubious about the quest's goal—to obtain one's heart's desire. Filopat follows rules, adheres to duty and fervently believes in the quest. Patofil stimulates the participant to disobey. Filopat represents authority and security. The participant is encouraged to side with one, then the other. These alliances implicitly include an adherence to the particular philosophical position of that agent.

While we have a complete storyboard mapped out, we have started production in the middle of the drama, act 3, in which Patofil and the participant have been told by Filopat that they must stand all night in a vigil at a ruined chapel. In scene 1, Patofil and the participant are teleported to the mound where the chapel stands. After a short time one or the other tires of the vigil. At first they play with whisps that float through the air, and climb the ruins. Then they leave the mound in direct defiance of Filopat's injunctions. At this point they become separated. The scene ends with the participant hearing Patofil scream. In scene 2, the participant sees Patofil running, pursued by five bad guys. Three of these guys break off and surround the participant, taunting and pushing her. The sun rises (night only lasts a few minutes in this virtual world!). Filopat can be heard calling. The bad guys disappear.

4 Modalities

The difference between MGLAIR (Modal GLAIR) and GLAIR is that in MGLAIR, the PML is organized into modalities.

A modality is a hardware or software resource utilized by an intelligent agent for either sensing or acting. A single modality can support only a small number of behaviors at a time. For example an agent can typically use a speech modality to say only one thing at a time. However, behaviors that occupy different modalities can be simultaneous. For example, an agent may use a speech modality to say something while simultaneously using a navigation modality to change its location. A modality may be afferent (sensory), or efferent.

The actor-agents in *The Trial The Trail* have the following modalities.

Animation: Animation is a efferent modality that controls the movement of the actor-agent's body (its body language) independently of the location of the agent within the virtual world. Patofil and Filopat are particularly expressive, each having a head and a body with wing shaped arms and feet.

Hearing: Hearing is an afferent modality by which the agent "hears" what she, herself, says, and what the other actor-agents say. In the future we intend to include what the human participant says.

Mood: Instead of a realistic face, each character wears a mask. Controlled by this modality, the mask morphs between expressions conveying the agent's mood: happiness, sadness, anger, fear, etc.

Navigation: Movement of the agent from one point to another is controlled by the navigation modality. Obstacle avoidance and path planning are handled at the SAL.

Speech: Speech is an efferent modality by which the actor-agents deliver lines of speech. This is their main mode of interaction with the participant. We currently use prerecorded human speech, because we have not found computer-generated speech to be expressive enough of emotion.

Vision: Vision is an afferent modality by which the agents are made aware of the actions of the participant, themselves, and the other actor-agents, and of other changes and events in the world.

Several of the modalities provide feedback about what the agent, itself, says or does. This keeps the agent from starting to use an efferent modality while its body is still performing the previous action. For example, Patofil's hearing her own lines prevents her from starting to say something while she is still delivering her previous lines.

The Trial, The Trail runs on several computers. The KL, PMLa, and PMLb layers of each actor-agent run on computers in UB's Department of Computer Science and Engineering, (a separate program image for each agent), while the PMLc and SAL layers, as well as the environment, run on computers in UB's Department of Media Study. (one program image for all). Communication is via IP sockets organized by modality, one socket for each modality. The sockets essentially provide the "neural" mind-body connection of each actor-agent, with the "mind" running on one computer, and the "body" on another. Each socket-equipped modality runs in a separate process thread, providing for the parallel performance of actions, as long as they are in different modalities.

5 Triggers

Just as a stage actor uses various cues to know when to take some actions or deliver some lines, the actor-agents of our drama make use of triggers that are implemented in the world-model to identify the actions and locations of the virtual objects, agents, and the participant during their interaction in the world. We also make use of the concept of a stage direction agent, similar to a set manager, that is in charge of such procedural tasks as initiating the beginning and ending of each scene, starting and ending the action of other interactive objects (e.g. the flow of whisps), controlling the lighting of the scene (setting the sun, raising the moon).

6 The Scripts

Each actor-agent is provided with a script written in SNePSLOG (Shapiro and The SNePS Implementation Group 2004), a symbolic-logic-like interface language to SNePS, which is used for the examples in the rest of this paper. The agent internalizes its script as a set of beliefs in its KL. It decides how and when it should say its lines and perform its actions by reasoning over those beliefs, making use of the integrated reasoning/acting facilities of SNePS/SNeRE (Kumar 1990; Kumar and Shapiro 1994a). This is the major difference between SNeRE-based agents and those driven by acting languages like ABL (Mateas and Stern 2004) which are viewed as programming languages. Goals and behaviors are determined for ABL-agents by a kind of method call, and goal and behavior definitions are not represented in the same working memory as are facts about the current situation in the agents' environment. SNeRE, as an acting language, is closer to other logic-based acting languages such as Golog (Levesque, Reiter, Lespérance, Lin, and Scherl 1997) and the event calculus (Shanahan 1999), but there is not enough room in this paper to discuss the differences.

In the following discussion of SNeRE constructs, variables *p*, *p1*, ... range over propositions and policies, and *a*, *a1*, ... range over acts.

Act 3, scene 1 begins when the stage direction agent teleports the participant and Patofil to the mound where they are to spend the night in a silent vigil. Patofil's script says,

```
whendo(Location(I, On_Mound) and Location(User, On_Mound),
       do-all({believe(doing(Act3Scene1Plan1)),
               do(Act3Scene1Plan1)})).
```

This says that when Patofil believes that she and the participant are both on the mound, she should believe that she is performing the first plan of act 3, scene 1, and actually perform it. The policy `whendo(p,a)`, the control act `do-all({a1,...,ak})`, and the mental act `believe(p)` are part of SNeRE. The act `do(a)` is a standard technique we have adopted to specify the performance of an action that takes no arguments. The other symbols are specific to these agents. Once a `whendo(p,a)` policy is adopted, it acts like a demon. Afferent modalities perform a `believe` on a proposition that the agent has sensed something, and `believe` triggers forward inferencing. When this results in `p`'s being believed, `a` is performed.

The way Patofil is to perform the first plan of act 3, scene 1 is specified in the script as,

```
ActPlan(do(Act3Scene1Plan1),
        csequence(performAct(InitialSilence, Reverent, In_Front_Of_User, Neutral),
```

```
believe({whendo(said(I, InitialSilence), do(ConductVigil1)),
        whendo(said(I, NoTalking), do(ConductVigil2)),
        whendo(said(I, StandStraight), do(ConductVigil3)),
        whendo(said(I, ContemplateTheInf), do(ConductVigil4))})))).
```

`ActPlan(a1,a2)` is a SNeRE construct that denotes the proposition that a way to perform the act `a1` is by performing the act `a2`. If an act has multiple plans, one is chosen randomly. Since `ActPlan` is a proposition-valued function symbol (Shapiro 1993), it is easy to specify conditions under which certain plans are applicable (Kumar and Shapiro 1993). The control act `csequence` is one we recently developed that takes two acts and does them in reverse order. In this, a paradigm use of `csequence`, Patofil is first to believe (adopt) four policies, and then perform a four-part act, explained to her as:

```
all(u,a,n,m)(ActPlan(performAct(u,a,n,m),
                     do-all({say(u), animate(a), navigate(n), mood(m)})))
```

That is, simultaneously speak the lines *u*, perform the animation *a*, and perform the navigation *n*, while expressing the mood *m*. These four acts are primitive, and are implemented at the PMLa layer, each in its own modality.

The four policies adopted as part of `Act3Scene1Plan1` make use of Patofil's ability to hear her lines after she speaks them so that she performs the four-step vigil in order.

7 Timers

Patofil performs the fourth step of the vigil as specified in the script as,

```
ActPlan(do(ConductVigil4),
   performAct(DontLeaveTheHill, Reverent,
              Stand_Still, Eyes_Closed))
```

That is, say "Don't leave the hill", stand in a reverent manner without moving, and wear the "eyes closed" mask. But if the participant is obedient, for how long should Patofil just stand there doing nothing? The following policy is that it should be for a maximum of 14 seconds:

```
whendo(said(I, DontLeaveTheHill),
       start_timer(14, Giggle, Laugh, Stand_Still, Happy, Act3Scene1Plan1))
```

An arbitrary number of timers may be started with the primitive act `start_timer(`*t,u,a,n,m,p*`)`, which associates a new timer with the plan *p*, sets it to expire in *t* seconds, and adopts the policy that when the timer expires, Patofil should `performAct(u,a,n,m)`. So, after 14 seconds of neither Patofil nor the participant doing anything, Patofil should giggle, act like she's laughing, and wear the happy mask, but still not move. This is just one example of a timer being used to limit an activity that is not being done in order to achieve some detectable goal.

Timers may also be paused, restarted, and cancelled.

8 Contingencies

If the participant moves during the first part of act 3, scene 1, when Patofil is encouraging her to maintain a silent vigil, this will be detected by the VR equipment, and the environment modeler will send a "user fidgeting" message to Patofil via the socket in her vision modality. Following this policy,

```
wheneverdo(Location(I, On_Mound)
           and Location(User, On_Mound) and AgentAct(User, Fidgeting),
           snif({if(doing(Act3Scene1Plan1),
                    snsequence(pause_timer(Act3Scene1Plan1),
                               do(ReprimandFidgeting)))}))
```

Patofil will pause the `Act3Scene1Plan1` timer and reprimand the participant. Once the timer is restarted, Patofil will resume her 14 seconds silence.

The SNeRE act `snif({if(`*p1*`,`*a1*`), ..., if(`*pk*`,`*ak*`)})` will nondeterministically perform one of the *ai* acts whose proposition *pi* Patofil believes. In this case $k = 1$. The SNeRE control act `snsequence(`*a1*`,`*a2*`)` will perform the two acts *a1* and *a2* in order. The SNeRE policy `wheneverdo(`*p*`,`*a*`)` is like the `whendo(`*p*`,`*a*`)` policy, but once a `whendo` is adopted it will automatically be cancelled (disbelieved) after its first triggering, whereas a `wheneverdo` can be triggered repeatedly. So Patofil will repeatedly reprimand the participant if the participant repeatedly fidgets.

Since Patofil might reprimand the participant multiple times, she has multiple ways to do so:

```
ActPlan(do(ReprimandFidgeting),
        do-all({do-one({say(Concentrate), say(StopFidgeting),
                        say(Meditate), say(YoureMeantToStayS)}),
                animate(Reverent),
                navigate(Stand_Still),
                mood(Eyes_Closed)}))
```

The SNeRE act `do-one({`*a1, ..., ak*`})` performs one of the argument *ai* acts randomly. So each time Patofil reprimands the participant, she might do so by saying a different line.

In general, contingincies are handled by the demons set up by `whendo` or `wheneverdo` policies, that, in turn, pause timers or otherwise interrupt main-line plans.

9 The Cast as a Multi-Agent System

The cast of computational actor-agents constitutes a multi-agent system. This is especially the case in act 3, scene 2, in which three bad guy agents are required to cooperatively harass the human participant while another two chase Patofil off stage. Thus, this scene contains six actor-agents in three groups: Patofil; two agents who chase Patofil; and three agents who cooperatively taunt and push the human participant. The most interesting group from a multi-agent perspective is the last one. These agents must react to each other as well as to the human participant.

We employ what Mateas and Stern (2004) call the "many-minds" approach to multi-agent systems, as opposed to the "one-mind" approach, or their "joint goals" approach. The act an agent performs is determined by reasoning at its Knowledge Layer (its "mind"), running on one computer system. The act is sent to the agent's "body", a collection of Ygdrasil nodes running on another computer system, as a message across the socket contained in the act's modality. A socket node receives the message and uses the Ygdrasil message handling system to pass it on to the appropriate node or nodes that can parse the message and react. The message handling system also prepends the agent's identifier to the message, and broadcasts it to the appropriate socket of all the agents in perceptual range, including the one that first sent the message. The PMLa of each perceiving agent then parses the message into the acting agent and the act, and performs a `believe` on the proposition that that agent has performed that act.

For example, if bad guy agent one performs the act of jostling the participant, he sends a "Jostle" message across the socket to the Ygdrasil navigation socket node, which sends the message on to a navigation/travel manager node that handles the navigation of the agent's body model in the world. The Ygdrasil message handling system then prepends `BG1_` to the message, and broadcasts to all the agents in perceptual range the message `BG1_Jostle` across their visual sockets. All those agents then believe that bad guy one is performing the act of jostling.

This message passing format, unlike others (Finin, Labrou, and Mayfield 1997), does not include a receiver for the message. The primary reason is that these are not peer-to-peer messages, but messages broadcast to the environment, and received by all agents in sensory proximity.

The bad guys engage in two types of cooperative behavior: sequential actions and blocking actions. Sequential actions are used during the dialogue portions of the drama and follow a strict script. The approach used for this is the same used by Patofil in the previous scene to order her utterances. However, instead of one individual agent perceiving its own utterances to establish a scripted order, these agents perceive their own and other agent utterances to accomplish the same task. For example, the script given to bad guy one says,

```
whendo(AgentSaid(Patofil, HelpMe),
       performAct(LookAnotherOne, Fierce, Surround_User, Happy))
```

while the script given to bad guy two says,

```
whendo(AgentSaid(BadGuy1, LookAnotherOne),
       performAct(HalloLittleOne, Menacing, Surround_User, Angry))
```

According to these policies, bad guy one will say, "Look another one", and move to surround the participant when Patofil says "Help Me". Bad guy two will then continue the dialogue upon hearing bad guy one's utterance by saying "Hallo little one" and moving to help surround the participant.

Blocking actions are actions that must not be interfered with by the other agents during the action's performance. For this scene, such actions include blocking the user's movement, jostling the user, and pushing the user. Whenever an agent perceives that another agent is performing one of these actions, it believes that that agent is performing an action block. The script for every agent says,

```
all(x)({AgentAct(x, Jostle_User), AgentAct(x, Push_User), AgentAct(x, Block_User)}
       v=> {ActionBlock(x)})
```

The SNePSLOG formula {*p1, ..., pn*} `v=>` {*q1, ..., qm*} says that if any of the *pi* is believed, then belief is justified in any of the *qj*.

Whenever an agent believes that another agent is blocking, it will only perform those actions that will not interfere with the blocking action. For example, this script segment provides several ways to push the user, but says that if some agent is blocking, then instead of pushing, just keep surrounding the participant, not saying anything, but looking angry and menacing:

```
ActPlan(do(pushUser),
        withsome(?x,
                 ActionBlock(?x),
                 performAct(Silent, Menacing, Surround_User, Angry),
                 do-one({performAct(WeJustWantToHave, Push, Push_User, Happy),
                         ...,
                         performAct(YeeHaa, Push, Push_User, Happy)})))
```

The SNeRE control act `withsome(?x, p(?x), a(?x), da)` attempts to find some individual *i* that satisfies the proposition *p(?x)*, and if it finds one, performs the act *a(?x)*{*i/?x*}, otherwise, it performs the act *da*.

This restriction is only lifted when the blocking action is finished, as indicated by an *action*_Finished message from the virtual world via the agent's vision modality. The following policy, which all the bad guys have, lifts the blocking action when jostling is finished:

```
all(x)(wheneverdo(AgentAct(x,Jostling_Finished),
                  disbelieve(AgentAct(x,Jostle_User))))
```

The SNeRE mental act `disbelieve(p)` results in the agent no longer believing the proposition *p*. SNeBR (Martins and Shapiro 1988), the SNePS Belief Revision (Assumption-Based Truth Maintenance) System, causes the agent to no longer believe `ActionBlock(x)` once `AgentAct(x,Jostle_User)` is disbelieved.

10 Implementation Status

Everything described in this paper has been implemented. Act 3 of *The Trial, The Trail* has been presented privately to a number of participants from our institution, but not yet publicly to a general audience.

11 Conclusions

The MGLAIR architecture, along with the SNePS/SNeRE integrated knowledge representation, reasoning, and acting system are proving to be valuable approaches to the building of intelligent agent-actors in virtual reality dramas. Modalities simplify the organization of acts, especially those that can be performed simultaneously. Incorporating

IP sockets with modalities facilitates the organization when the "mind" and the "body" of the agent are running on different machines. Self-perception via the afferent modalities allows the "mind" to pace its acting properly. Timers are used for activities, that don't have goals. Contingincies are handled by demons that pause timers and interrupt main-line plans. Each agent-actor is an independent intelligent agent that cooperates with others via sensing and communication in accordance with the script it has been given.

References

Anstey, J., D. Pape, and D. Sandin (2000). The Thing Growing: Autonomous characters in virtual reality interactive fiction. In *IEEE Virtual Reality 2000*. IEEE.

Anstey, J., D. Pape, S. C. Shapiro, O. Telhan, and T. D. Nayak (2004). Psycho-drama in VR. In *Proceedings of The Fourth Conference on Computation Semiotics (COSIGN 2004)*, Croatia, pp. 5–13. University of Split.

Barthes, R. (1974). *S/Z*. New York: Hill and Wang.

Bobick, A., S. Intille, J. Davis, F. Baird, C. Pinhanez, L. Campbell, Y. Ivanov, A. Schutte, and A. Wilson (1999, August). The Kidsroom: A perceptually-based interactive and immersive story. *Presence 8*(4), 367–391.

Charles, F., M. Lozano, S. J. Mead, A. F. Bisquerra, and M. Cavazza (2003). Planning formalisms and authoring in interactive storytelling. In *Technologies for Interactive Digital Storytelling and Entertainment Conference*, pp. 216–225. Fraunhofer IRB Verlag.

Fencott, C. (2003). Agencies of interactive digital storytelling. In *Technologies for Interactive Digital Storytelling and Entertainment Conference*, pp. 152–163. Fraunhofer IRB Verlag.

Finin, T., Y. Labrou, and J. Mayfield (1997). KQML as an agent communication language. In J. M. Bradshaw (Ed.), *Software Agents*, pp. 291–316. Menlo Park, CA: AAAI Press/The MIT Press.

Hexmoor, H., J. Lammens, and S. C. Shapiro (1993, April). Embodiment in GLAIR: a grounded layered architecture with integrated reasoning for autonomous agents. In D. D. Dankel II and J. Stewman (Eds.), *Proceedings of The Sixth Florida AI Research Symposium (FLAIRS 93)*, pp. 325–329. The Florida AI Research Society.

Hexmoor, H. and S. C. Shapiro (1997). Integrating skill and knowledge in expert agents. In P. J. Feltovich, K. M. Ford, and R. R. Hoffman (Eds.), *Expertise in Context*, pp. 383–404. Cambridge, MA: AAAI Press/MIT Press.

Ibanez, J., R. Aylett, and R. Ruiz-Rodarte (2003). Storytelling in virtual environments from a virtual guide perspective. *Virtual Reality, Special Edition on Storytelling in Virtual Environments 7*(1), 30–42.

Kumar, D. (1990). An integrated model of acting and inference. In D. Kumar (Ed.), *Current Trends in SNePS*, Number 1600 in Lecture Notes in Artificial Intelligence, pp. 55–65. Berlin: Springer-Verlag.

Kumar, D. (1996, January). The SNePS BDI architecture. *Decision Support Systems 16*(1), 3–19.

Kumar, D. and S. C. Shapiro (1993, April–September). Deductive efficiency, belief revision and acting. *Journal of Experimental and Theoretical Artificial Intelligence (JETAI) 5*(2&3), 167–177.

Kumar, D. and S. C. Shapiro (1994a, May). Acting in service of inference (and *vice versa*). In D. D. Dankel II (Ed.), *Proc. FLAIRS 94*, pp. 207–211. The Florida AI Research Society.

Kumar, D. and S. C. Shapiro (1994b, March). The OK BDI architecture. *Int. J. on AI Tools 3*(3), 349–366.

Laurel, B., R. Strickland, and R. Tow (1998). Placeholder: Landscape and narrative in virtual environments. In C. Dodsworth (Ed.), *Digital Illusion*, pp. 181– 208. New York, New York: ACM Press.

Levesque, H. J., R. Reiter, Y. Lespérance, F. Lin, and R. B. Scherl (1997, April–June). GOLOG: A logic programming language for dynamic domains. *The Journal of Logic Programming 31*(1–3), 59–83.

Martins, J. P. and S. C. Shapiro (1988). A model for belief revision. *Artificial Intelligence 35*, 25–79.

Mateas, M. and A. Stern (2004). A behavior language: Joint action and behavioral idioms. In H. Prendinger and M. Ishizuka (Eds.), *Life-Like Characters: Tools, Affective Functions, and Applications*, pp. 135–161. Springer.

Pape, D., J. Anstey, M. Dolinsky, and E. J. Dambik (2003, August). Ygdrasil—a framework for composing shared virtual worlds. *Future Generation Computer Systems 19*(6), 1041–1049.

Propp, V. A. (1968). *Morphology of the Folktale*. Austin and London: University of Texas Press.

Rohlf, J. and J. Helman (1994). IRIS Performer: A high performance multiprocessing toolkit for real-time 3D graphics. In *SIGGRAPH 94*, pp. 381–394. SIGGRAPH: ACM Press.

Shanahan, M. P. (1999). The event calculus explained. In M. J. Wooldridge and M. Veloso (Eds.), *Artificial Intelligence Today*, Number 1600 in Lecture Notes in Artificial Intelligence, pp. 409–430. Berlin: Springer-Verlag.

Shapiro, S. C. (1993, April–September). Belief spaces as sets of propositions. *Journal of Experimental and Theoretical Artificial Intelligence (JETAI) 5*(2&3), 225–235.

Shapiro, S. C. and H. O. Ismail (2003, May). Anchoring in a grounded layered architecture with integrated reasoning. *Robotics and Autonomous Systems 43*(2–3), 97–108.

Shapiro, S. C. and W. J. Rapaport (1992, January–March). The SNePS family. *Computers & Mathematics with Applications 23*(2–5), 243–275.

Shapiro, S. C. and The SNePS Implementation Group (2004). *SNePS 2.6.1 User's Manual*. Buffalo, NY: Department of Computer Science and Engineering, University at Buffalo, The State University of New York.

Shaw, J., H. Staff, J. R. Farr, M. Adams, D. vom Lehm, C. Heath, M.-L. Rinman, I. Taylor, and S. Benford (2000). Staged mixed reality performance "Desert Rain" by Blast Theory. http://www.nada.kth.se/erena/doc/aD7b3.html.

Szilas, N. (2003). Idtension: a narrative engine for interactive drama. In *Technologies for Interactive Digital Storytelling and Entertainment Conference*, pp. 187–203. Fraunhofer IRB Verlag.

VRCO, Inc. (2005). CAVELib: Overview. http://www.vrco.com/CAVELib/OverviewCAVELib.html.