Towards Integrating AI Story Controllers and Game Engines: Reconciling World State Representations

Mark O. Riedl
Institute for Creative Technologies
University of Southern California
13274 Fiji Way, Marina Del Rey, CA 90292 USA
riedl@ict.usc.edu

Abstract

Recently, many AI researchers working on interactive storytelling systems have turned to off-the-shelf game engines for simulation and visualization of virtual 3D graphical worlds. Integrating AI research into game engines can be difficult due to the fact that game engines typically do not use symbolic or declarative representations of characters, settings, or actions. This is particularly true for interactive storytelling applications that use an AI story controller to subtly manipulate a virtual world in order to bring about a structured narrative experience for the user. In this paper, I describe a general technique for translating between an arbitrary game engine's proprietary and procedural world state representation into a declarative form that can be used by an AI story controller. The work is placed in the context of building a narrative-based training simulation.

1 Introduction

Interactive storytelling systems are applications in which a story is presented to a user in such a way that the user has the ability to affect the direction and possibly even the outcome of story. The ability of the user to impact the story arc and outcome suggests a branching story structure [Riedl and Young, 2005]. Advanced 3D graphics rendering capabilities, such as those found in modern computer games, makes it possible and even desirable to implement interactive storytelling by situating the user in a 3D graphical story world. In this approach, the user, through her avatar, is a character in the story and is able to interact with the environment and other characters and possibly even play a role in the plot.

Computer games are perhaps the most pervasive example of an interactive storytelling system. However, in computer games, the user's interactivity with the world is typically bounded in such a way that the user's actions do not actually have an impact on the story arc. That is, computer games use story to motivate action but typically have little or no branching.

AI techniques have been applied to the problem of interactive storytelling for entertainment and training. A common technique among AI research in interactive storytelling is to separate the AI story control elements from the graphical, virtual world. An *automated story director* – often referred to as a *drama manager* [Kelso, Weyhrauch, and Bates, 1993] – is responsible for keeping the user and any non-player characters (NPCs) on track for achieving a particular narrative-like experience. An automated story director maintains a representation of the structure that the emergent user experience is expected to conform to and exerts influence over the user, the virtual world, and the NPCs in order to achieve this. Examples of interactive storytelling systems that use some notion of an automated story director are [Weyhrauch, 1997], [Mateas and Stern, 2003], [Szilas, 2003], [Young et al., 2004], and [Magerko et al., 2004]. Some interactive storytelling systems such as [Cavazza, Charles, and Mead, 2002] do not use an explicit story director. Instead, such systems rely on story to *emerge* from the behaviors of the NPCs and the user [Aylett, 2000].

Recently, many AI researchers working on interactive storytelling systems have turned to off-the-shelf game engines for simulation and visualization of virtual 3D graphical worlds (e.g. [Cavazza, Charles, and Mead, 2002], [Seif El-Nasr and Horswill, 2003], [Young et al., 2004], and [Magerko et al., 2004]). Game engines provide sophisticated graphical rendering capabilities with predictable frame rates, physics, and other advantages so that AI researchers do not need to devote resources to "reinventing the wheel."

Integrating AI research into game engines can however be difficult due to the fact that game engines typically do not use symbolic or declarative representations of characters, settings, or actions [Young and Riedl, 2003][1]. Action representations for many game engines such as first-person shooters are expressed as "micro-actions" – mouse clicks, key presses, etc. – and state representations are based on continuous vector positions, rotations, velocities and "flag" variables. AI character or story controllers such as [Cavazza, Charles, and Mead, 2002], [Young et al., 2004], and [Magerko et al., 2004] use declarative, symbolic representations of character, world, and story state. For example, $\texttt{Walk(agent}_1\texttt{, loc}_1\texttt{, loc}_2\texttt{)}$ is a discrete action and (at

[1] One exception is the commercial game described in [Orkin, 2004], which uses both proprietary and symbolic world state representations.

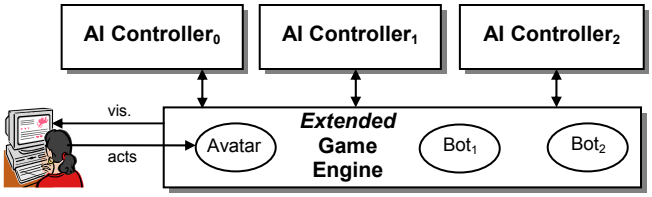

Figure 1: Generic architecture for an interactive storytelling system.

$\texttt{agent}_1\ \texttt{loc}_1)$ is a discrete term partially describing the world state.

AI technologies often use declarative and/or symbolic representations of the virtual environment, simplifying the world to only the aspects that are necessary for computation. Declarative representation facilitates robust reasoning about the simulation state such as regressive problem solving (e.g. planning and re-planning), predictive analysis (e.g. predicting plan failure), user goal recognition, agent belief-desire-intention modeling, and others. As far as automated story direction is concerned, [Young, 1999] describes the advantages of using a declarative, partial-order plan representation for narrative: (a) causal dependencies between actions ensure that all events are part of causal chains that lead to the outcome; (b) planning algorithms are general problem-solvers that "solve the problem" of piecing together the events of a narrative that achieves a particular outcome; and (c) story plans can be repaired by replanning to allow interactivity.

For an AI character or story controller to be closely integrated with a proprietary game engine, the AI system must transform the proprietary non-declarative world state in the game engine into a declarative form. For example, Mimesis [Young et al., 2004] overrides the game engine's user input routines in order to detect discrete user actions. The discretized user actions are correlated with plan operators that have declarative preconditions and effects with which to reason about changes to the world wrought by the user. Not all AI controllers use plan operator representations.

The remainder of the paper is laid out as follows. In Section 2, we describe a generic architecture for an interactive storytelling system. In Section 3, we describe a general technique for translating proprietary and procedural world representation from an arbitrary game engine into a declarative form that can be used by AI controllers such as automated story directors and autonomous agents. In Section 4, we briefly describe a narrative-based training simulation that motivates the need for the integration of an automated story director and autonomous agents with an arbitrary game engine.

2 A Generic Interactive Storytelling Architecture

A generic architecture for an interactive storytelling system is given in Figure 1. The architecture is based around a game engine and one or more AI controllers. AI controllers can be automated story directors or autonomous agents. Autonomous agents control the decision-making processes of non-player characters (NPCs). Even though a virtual world contains non-player characters, it is not necessarily the case that there must be an AI controller for each NPC. An automated story director, in addition to maintaining a branching narrative model, can be implemented such that it also directs the behaviors of NPCs, as in [Young et al., 2004]. If there is an automated story director, there is typically only one director. There does not necessarily have to be an automated story director for there to be interactive storytelling, as in [Cavazza, Charles, and Mead, 2002].

The game engine can be any game or simulation system that supports or can be extended to support interface to the automated story director and the virtual actors. Figure 1 refers to the game engine as an *extended* game engine because of its support for AI controllers. The game engine extensions are described in further detail in the next section. In general, the game engine is responsible for simulating a virtual world plus graphical presentation of the virtual world to the user who is embodied by an avatar. Each non-player character (NPC) that the trainee will be expected to interact with is represented graphically in the game engine as a *bot*. A bot is a physical manifestation of an NPC based on the proprietary graphical rendering of the character's body in the virtual world. Aside from processes for rendering, animating, and low-level path-planning, there is little or no intelligence in the bot. The higher-level intelligence of an NPC is relegated to one of the AI controllers that receive updates from the virtual world and issues control commands to bots.

It is possible – and sometimes even desirable – for the various AI controllers to communicate with each other to coordinate behaviors and world representations. For the remainder of this paper, we shall assume that all the AI controllers in an interactive storytelling system use the same world state representations and it is only the game engine that does not. Furthermore, we shall assume that there is at least one AI controller that is an automated story director.

3 A Middleware Substrate for Integrating a AI Controllers into a Game Engine

In the generic architecture for an interactive storytelling system described in the previous section, the automated story director and any autonomous agents are assumed to use a shared declarative representation of world state. The game engine, however, is not assumed to use a world state representation that is deterministic or shared with the other components. In fact, it is assumed that the game engine does *not* use a declarative representation! However, it is vital that the AI controllers are aware of the state of the simulation in the game engine. An automated story director, in particular, must be aware of the changes to the world state that are caused by the actions of the user. Agents must also be aware of changes in the world state to be able to react appropriately and believably. The solution to reconciling world state representations between an arbitrary game engine and AI controllers described here is motivated by the generic architecture. However, it is our belief that the solu-

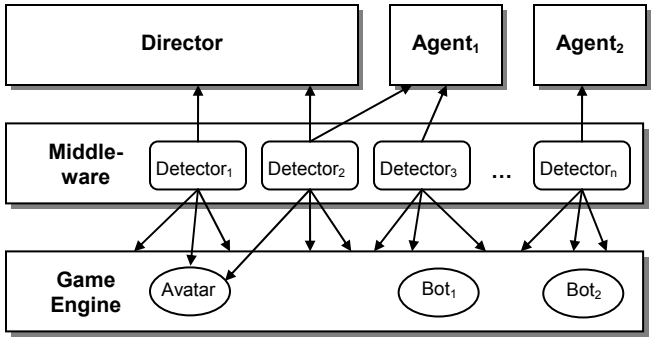

Figure 2: Middleware for detecting and translating game engine world state.

tion is general enough to apply to other interactive storytelling systems.

The procedural, non-declarative state representation maintained by the game engine must be translated into a declarative representation shared by the automated story director and the actors. One way to transform the game engine's representation into a declarative form is through a middleware substrate that interfaces directly with the game engine through scripting or through an API such as that proposed in [van Lent, 2004]. A middleware approach may be inappropriate for computer game production where AI is only guaranteed a small portion of a computer's processing time and efficiency is therefore essential. However, AI game research is not necessarily beholden to production constraints. Researchers in automated story direction often resort to a distributed architecture where graphical and simulation processing occurs on one computer while AI processing occurs on one or more other computers. In this case, a middleware solution is favorable because it abstracts away the procedural nature of the game engine and allows AI researchers to focus on theories, algorithms, and cognitively plausible representations of narrative.

The proposed middleware substrate implements *state detectors* and *proprioceptive detectors* that efficiently access the game engine's proprietary state variables (such as object locations, rotations, velocities, flags, etc.) to derive discretized information about the game engine and push that information to any system modules that request updates. Figure 2 shows a conceptualization of the middleware substrate.

3.1 State Detectors

State detectors determine if discrete state declarations are true or false. For each atomic, ground sentence used by the automated story director or an autonomous agent to represent some aspect of world state, there must be a detector that can recognize whether it is true or not in the simulation. Note that for efficiency purposes a single state detector can be responsible for more than one fact.

An example of a state detector is one that determines whether `(in-speaking-orientation user ?npc)` is true for some non-player character in the world, meaning the NPC and player are close by, facing each other, etc. When a sentence of this form is true, the player and agents can engage in conversation (either can take the initiative). This world state can be important to agents who need to know if they can engage the user in dialogue and to an automated director if the story requires some conversational exchange between user and another character before the story can continue. Whether a sentence of this form is true or not can be computed from the distance between the user's avatar and the bot and the directional orientation of avatar and bot towards each other. A single detector can be responsible for determining whether the relationship holds or does not hold for all NPCs in the world, as opposed to state detectors for each NPC.

3.2 Proprioceptive Detectors

Proprioceptive detectors apply only to the user's avatar and are used to determine if the user has performed certain discrete actions. The purpose of a proprioceptive detector is for the user's avatar to declare to listening AI controllers, "I, the user's avatar, have just performed an action that you might have observed." Bots do not need proprioceptive detectors because their behavior is dictated by an AI controller; success or failure of bot behaviors can be confirmed by comparing the expected world state changes with actual world state changes. Agents can be made aware of each others' observable actions through direct back-channel communication.

An example of a proprioceptive detector is one that determines when the user has moved from one discrete location in the world to another. That is, it determines whether the declarative action `Walk(user, ?loc1, ?loc2)` has been performed. This declaration can be important to agents who observe the user leaving or arriving. This declaration can also be important for an AI controller such as an automated director that needs to know about the effects of the action: `(at user ?loc2)` and $\neg$`(at user ?loc1)`. However, this information can be derived through state detectors as well without concern for how those effects were achieved (e.g. `Walk` versus `Run`).

3.3 Detector Integration with the Game Engine

While state changes can be determined from discrete action representations such as those used in planning systems, the purpose of detecting user actions is primarily for sensor input to the autonomous agent AI controllers. When NPCs and the user interact, the agents will need to know the observable actions that the user performs, whether they are physical or discourse acts, instead of inferring them from local world state changes. State detectors, however, are still necessary above and beyond proprioceptive detectors because the user's input into the game engine is through "micro-actions" – mouse clicks, key presses, etc. Many micro-actions string together to produce discrete actions. However, it may be the case that the user performs micro-actions that change the world state but are not aggregated into a recognizable discrete action. Thus, it is possible for the simulation state in the game engine to become out of sync with that of the story director and the actors. One solution is to define discrete action representations at a finer level of

detail. The approach advocated here is to detect high-level actions that are necessary for user-agent interactions and allow state detectors to fill in the rest.

It is possible to integrate symbolic and procedural representations. Orkin [2004] describes a technique for individual AI characters to perform real-time, goal-oriented planning in a game engine using action representations that combine both symbolic preconditions and effects with procedural preconditions and effects. However, it is not clear whether such a technique could be applied to an AI story director since a story director does not directly act in the world as an AI character does. We believe the middleware substrate approach advocated in this paper to be more general and flexible.

4 Towards a Narrative-based Training Simulation

In this section, we describe an interactive storytelling system built on top of a game engine that uses a multitude of AI controllers, including an automated story director and several autonomous agents. The various types of AI controllers use different AI technologies and consequently have different declarative world representations. The middleware substrate approach is capable of meeting all of the information requirements of the heterogeneous collection of AI controllers without requiring any to be tightly integrated with the game engine. The following discussion describes the purpose of the system and motivates the necessity of having different types of AI controllers operating simultaneously.

The interactive storytelling system we describe here is a *narrative-based training simulation*. Simulations have been used for training skills and situation awareness. For training tacit knowledge such as the operational and procedural skills required for adaptive military leadership, it is advantageous to situate the trainee in a realistic environment. A virtual reality simulator is a good start. However, it is advantageous that trainees are situated in an environment whose situational evolution is directed. The advantages are that the trainee can be exposed to a larger context, multiple learning objectives can be strung together in a particular order, and the trainee can gain valuable experience in dealing with successions of problems that are interrelated in a lifelike manner (instead of running separate, and thus disjoint, training exercises). Since pure simulations are open-ended, there is no guarantee that the world will evolve in a sustainable manner. That is, the structure of the trainee's experience is not guaranteed to contain certain events or situations after the first few actions. The actions of the trainee and any autonomous agents can cause the world to evolve in a way that is undesirable from the perspective of the trainee being exposed to situations of pedagogical value.

4.1 Story Control for Training

Our narrative-based training simulation uses a high-level AI control structure to try to manipulate a simulation such that the world state, at least at a high level of abstraction, evolves in way that corresponds to a given model of narrative. The way in which this is achieved is necessarily different from more entertainment-oriented interactive storytelling systems. One difference between training and entertainment applications is that the trainee must learn about second- and third-order effects of their actions, meaning that it is important for realistic emergence of situation. An entertainment application can ignore the effects on the world that do not contribute to the story. This emergence [Aylett, 2000] must be carefully balanced against the overarching, high-level narrative model of the story director.

A second difference between training and entertainment applications of interactive storytelling is that in systems for training the AI story controller should be relatively resilient to branching. That is, the given high-level narrative model achieves a certain sequence of learning objectives that has pedagogical value. Branching to alternative narrative arcs should be possible, but only when absolutely necessary. Furthermore, any alternative narrative branch should be as similar as possible to the original narrative model and contain the same pedagogical value. Branching story in entertainment applications only require the consequent alternative narrative branches to have entertainment value and can consequently deviate more in order to comply with the apparent desires of the user.

A third difference between training and entertainment applications of interactive storytelling is that in systems for training, the automated story director should not intervene with the actions of the trainee. This is important because one does not want the trainee to learn that certain incorrect or inappropriate actions are okay because they will be caused to fail. It is also important for the trainee to learn from her mistakes, even if it means "game over." This is in contrast to [Young et al., 2004] which describes an entertainment-oriented interactive storytelling system that is capable of subtly intervening with user actions to preserve the content of the narrative. For training, user actions that are not in accordance with the narrative model should either cause an alternative branch to be taken or result in failure with feedback about what was wrong.

4.2 Architecture for a Narrative-Based Training Simulation

We believe that we can achieve the nuances of interactive storytelling for training purposes with a combination of automated story direction and semi-autonomous agents. The heterogeneity of AI controllers makes a middleware approach to integration with a game engine desirable. The architecture for the narrative-based training simulation is given in Figure 3.

The three main components to the architecture are: the game engine, the automated story director, and the semi-autonomous virtual actors. The game engine is any game engine or simulation that includes the middleware substrate for interfacing with an automated story director and AI characters. The automated story director is an AI controller that has a branching narrative model and is capable of determining whether the simulation state in the game engine matches – or at least is not contradictory to – the narrative

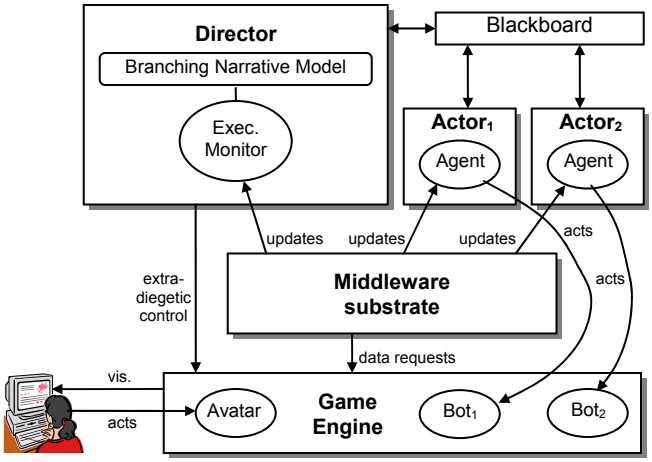

Figure 3: Architecture for a narrative-based training simulator.

model. Additionally, the automated story director is capable of manipulating the *extra-diegetic* effects of the game engine as well as the semi-autonomous virtual actors. Extra-diegetic aspects of the game engine are those involving the visualization of the world such as music, cinematography (e.g. [Jhala, 2004]), and lighting (e.g. [Seif El- Nasr and Horswill, 2003]), and not the actual simulation state.

Each non-player character (NPC) that the trainee will be expected to interact with is represented by a pairing of two components: a bot and an AI controller called an *actor*. Bots are described in Section 2. An actor[2] contains within it an autonomous agent decision-making process that has beliefs, desires, and intentions and uses sensors to react to the environment as it attempts to achieve its intentions. Examples of AI character technologies that have been applied to animated, virtual agents are Soar [Rickel et al., 2002], HAP [Loyall, 1997], ABL [Mateas and Stern, 2003], and hierarchical task networks [Cavazza, Charles, and Mead, 2002]. We do not make any commitment to the type of agent technology used in the narrative-based training simulation except that the agent decision-making process is wrapped in additional logic that is aware of the narrative goals of the automated director and is *directable*. A directable agent is one whose behavior and reasoning can be manipulated by an external process [Blumberg and Galyean, 1995; Assanie, 2002]. The actor itself is aware of the narrative goals of the automated director and takes direction from the automated director. Direction from the automated director takes one of two forms:

- Direction to achieve some world state that is desirable to the automated director and moves the plot forward.
- Direction that constrains the internal, reactive decision-making process – which is only aware of its own beliefs, desires, intentions and sensory input from the environment – from choosing actions, behaviors, or dialogue that contradicts or invalidates the automated director's narrative model.

Both types of direction are essential. The first type of direction is the primary mechanism through which the automated director pushes a story forward and is necessary because the actors cannot be relied on to autonomously make decisions that are always favorable to the automated director. The second type of direction is important in any situation where actors do have some autonomy to form and reactively pursue their own goals. Autonomy means that actors can potentially choose actions, behaviors, or dialogue that contradicts the narrative model of the automated director and even make it impossible for the narrative and all of its branches to continue coherently.

The final component in Figure 3 is a blackboard. Rist, André, and Baldes [2003] demonstrate a blackboard to be an effective channel of communication between autonomous agents and story directors. Here, the blackboard serves two purposes. First it contains a specific world state that is shared between the director and the actors. Note that this world state may be different than the world state held by the actor's internal agent processes because the internal agent processes are responsible for reacting to local environmental conditions and should not necessarily be aware of things outside the scope of its senses. Actors only receive state updates and knowledge about user avatar actions that are within range of the bots' senses and necessary for reactivity within the environment. The blackboard, however, contains a global representation of the entire virtual world, including the internal state of all the NPCs. This privileged information is only accessible to the directable processes that wrap the autonomous agent decision-making processes.

The second purpose of the blackboard is a communication channel between the automated story director and the actors. In particular, the director sends directives to the actors so that they will achieve certain world states that are advantageous to the narrative development as well as constraints so that the actors do not perform actions that make it impossible for the plot to advance. Conceivably, actors can also communicate amongst themselves to coordinate their performances.

5 Conclusions

In an interactive storytelling system such as the narrative-based training simulator described here, the graphical rendering of the virtual world and story world characters is separate from the AI control processes for story direction and agent decision-making. Game engines notoriously use proprietary and procedural representations for world state whereas AI controllers such as an automated story director often use declarative and/or symbolic world state representations. The approach presented here is a middleware substrate that uses actor and state detectors to produce declarations about the simulation world state and push state changes onto the story director and autonomous actors. While this approach is taken in the context of the architecture for a narrative-based training simulator, the middleware

[2] Gordon and van Lent [2002] lay out the pros and cons of agents that are realistic models of humans versus agents that are actors.

substrate approach is expected to be general enough to be applicable to many interactive storytelling systems.

Acknowledgements

The project or effort described here has been sponsored by the U.S. Army Research, Development, and Engineering Command (RDECOM). Statements and opinions expressed do not necessarily reflect the position or the policy of the United States Government, and no official endorsement should be inferred.

References

[Assanie, 2002] Mazin Assanie. Directable synthetic characters. In *Proceedings of the AAAI Spring Symposium on Artificial Intelligence and Interactive Entertainment*, 2002.

[Aylett, 2000] Ruth Aylett. Emergent narrative, social immersion and "storification." In *Proceedings of the 1st International Workshop on Narrative and Interactive Learning Environments*, 2000.

[Blumberg and Galyean, 1995] Bruce Blumberg and Tinsley Galyean. Multi-level direction of autonomous agents for real-time virtual environments. In *Proceedings of SIGGRAPH,* 1995.

[Cavazza, Charles, and Mead, 2002] Marc Cavazza, Fred Charles, and Steven Mead. Planning characters' behaviour in interactive storytelling. *Journal of Visualization and Computer Animation*, 13: 121-131, 2002.

[Gordon and van Lent, 2002] Andrew Gordon and Michael van Lent. Virtual humans as participants vs. virtual humans as actors. In *Proceedings of the AAAI Spring Symposium on Artificial Intelligence and Interactive Entertainment*, 2002.

[Kelso, Weyhrauch, and Bates, 1993] Margaret Kelso, Peter Weyhrauch, and Joseph Bates. Dramatic presence. *Presence: The Journal of Teleoperators and Virtual Environments*, 2(1), 1993.

[Jhala, 2004] Arnav Jhala. *An Intelligent Cinematic Camera Planning System for Dynamic Narratives*. Masters Thesis, North Carolina State University.

[Loyall, 1997] Brian Loyall. *Believable Agents: Building Interactive Personalities*. Ph.D. Dissertation, Carnegie Mellon University, 1997.

[Magerko et al., 2004] Brian Magerko, John Laird, Mazin Assanie, Alex Kerfoot, and Devvan Stokes. AI characters and directors for interactive computer games. In *Proceedings of the 16th Innovative Applications of Artificial Intelligence Conference*, 2004.

[Mateas and Stern, 2003] Michael Mateas and Andrew Stern. Integrating plot, character, and natural language processing in the interactive drama Façade. In *Proceedings of the 1st International Conference on Technologies for Interactive Digital Storytelling and Entertainment*, 2003.

[Orkin, 2004] Jeff Orkin. Symbolic representation of game world state: Towards real-time planning in games. In *Proceedings of the AAAI Workshop on Challenges in Game Artificial Intelligence*, 2004.

[Rickel et al., 2002] Jeff Rickel, Jon Gratch, Randall Hill, Stacy Marsella, David Traum, and Bill Swartout. Toward a new generation of virtual humans for interactive experiences. *IEEE Intelligent Systems*, July/August 2002.

[Riedl and Young, 2005] Mark Riedl and R. Michael Young. From linear story generation to branching story graphs. In *Proceedings of the 1st Conference on Artificial Intelligence and Interactive Digital Entertainment*, 2005.

[Rist, André, and Baldes, 2003] Thomas Rist, Elisabeth André, and Stephen Baldes. A flexible platform for building applications with life-like characters. In *Proceedings of the 2003 International Conference on Intelligent User Interfaces*, 2003.

[Seif El-Nasr and Horswill, 2003] Magy Seif El-Nasr and Ian Horswill. Real-time lighting design for interactive narrative. In *Proceedings of the 2nd International Conference on Virtual Storytelling*, 2003.

[Szilas, 2003] Nicolas Szilas. IDtension: A narrative engine for interactive drama. In *Proceedings of the 1st International Conference on Technologies for Interactive Digital Storytelling and Entertainment*, 2003.

[van Lent, 2004] Michael van Lent. Combining gaming and game visualization with traditional simulation systems. Invited talk at the *Serious Games Summit*, 2004.

[Weyhrauch, 1997] Peter Weyhrauch. *Guiding Interactive Fiction*. Ph.D. Dissertation, Carnegie Mellon University.

[Young, 1999] R. Michael Young. Notes on the use of planning structures in the creation of interactive plot. In *Proceedings of the AAAI Fall Symposium on Narrative Intelligence*, 1999.

[Young and Riedl, 2003] R. Michael Young and Mark Riedl. Towards an architecture for intelligent control of narrative in interactive virtual worlds. In *Proceedings of the 2003 International Conference on Intelligent User Interfaces*, 2003.

[Young et al., 2004] R. Michael Young, Mark Riedl, Mark Branly, Arnav Jhala, R.J Martin, and C.J. Saretto. An architecture for integrating plan-based behavior generation with interactive game environments. *Journal of Game Development*, 1: 51-70, 2004.