Mediating the Tension between Plot and Interaction

Brian Magerko and John E. Laird

University of Michigan
1101 Beal Ave.
Ann Arbor, MI 48109-2110
magerko, laird@umich.edu

Abstract

When building a story-intensive game, there is always the question of how much freedom to give the player. Give the player too little, and he may feel constrained and disconnected from the character he is controlling. Give him too much freedom, and the progression of the story may lag or stop altogether. This paper focuses on our attempt to find a balance between offering the player a high degree of interaction and providing a story-based experience where the player is a key character. Our approach is embedded in our Interactive Drama Architecture (IDA), which includes an omniscient story director agent who manages the player's narrative experience. The director agent uses a declarative description of the plot to track the player's progress, detect deviations from the plot, and make directions to supporting characters in the game. Our director is used within a game we have developed, called *Haunt 2*, which is an extension to the Unreal Tournament engine.

Introduction

The key problem with presenting a story-intensive game experience is that it is necessary to address the tension between telling a story and supporting a high degree of interaction for the player. The player is a variable character in the story; the actions he executes may help move the story forward, cause it to stall momentarily, or keep the story from progressing at all. A typical approach in games is to constrain the possible actions that the player has to choose from so that only actions consistent with the plot are available. The fewer constraints placed on the player's actions, the greater possible interactions the player can have with a rich environment. It follows that this increase in interaction leads us back to behaviors that can harm the progression of the plot.

Our approach to mediating this tension between plot and player behavior is the IDA (Interactive Drama Architecture) system, which is centered on the use of an omniscient story director agent that is responsible for maintaining the plot's progression (Magerko et al. 2004). Much like a human "dungeon master" does in some table top role-playing games, the director agent works with a pre-written story structure and attempts to guide the player through that story. The director follows along with the plot as it moves along, giving direction to characters when necessary to perform particular plot elements. The director agent also hypothesizes about the player's future behavior, trying to subtly steer the player away from those actions that may endanger the progression of the plot. Our game environment, called *Haunt 2*, consists of a fully structured story, synthetic characters that take part in the story, a 3-D world constructed with the Unreal Tournament engine (Magerko et al. 2004), and the story director agent, which will be the focus of this paper.

IDA uses player prediction to determine if the player's actions will endanger the plot. It is this capability that distinguishes it most from other interactive drama systems, such as the MIMESIS architecture and the Crosstalk framework (Young et al. 2004; Klesen et al. 2003). MIMESIS uses a fully structured plot, represented as a partial-order plan, and either incorporates unplanned player actions into the story or avoids them altogether if incorporating them is infeasible. The CrossTalk framework incorporates plan-based automatic dialogue generation with an author-defined narrative graph. Other approaches to interactive drama have taken a more modular approach to plot construction (Mateas and Stern, 2002; Weyhrauch 1997; Sgorous 1999). They rely on heuristically choosing plot elements as the player moves through the space of possible stories. Some systems have also included a player history as a model of user experience to help heuristically choose what plot elements should occur next (Szilas et al. 2003; Weyhrauch 1997). What these systems do not address is the preemptive alteration of the story state in subtle ways to avoid problematic player actions in the future (Beal et al. 2002). While some of the approaches above possibly provide a greater number of possible story orderings, IDA focuses on providing different possible plot content within the same specified plot structure from one gaming experience to the other. How we accomplish this is addressed in our discussion of the director agent, which is the focus of this paper.

Knowledge Maintenance

In order to make informed decisions about the state of the story, the director must maintain a comprehensive model of the world state. Figure 1 illustrates how this world model fits into the overall execution of the director. The director,

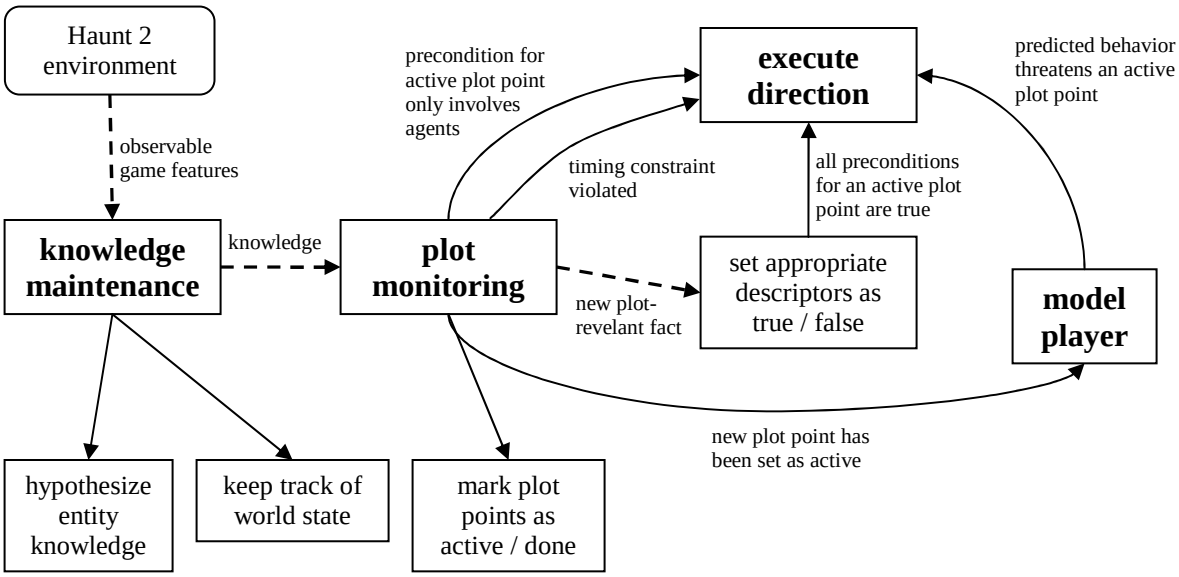

Figure 1. The execution cycle of the director agent.

using an omniscient view of the world, records any observable facts about the world's objects and entities (i.e. the synthetic characters plus the player). Along with gathering facts about the physical properties of the world, the director forms a hypothesis on each entity's knowledge base, trying to capture what knowledge each entity gathers as the story progresses. The model assumes that the player can learn new information from observing the world, being aware of his own character's state, or by learning information from spoken dialogue. Lines of dialogue are tagged with the information that they are intended to communicate. For instance, if the player moves into a room he has not been in before, the director will record that the player knows a) that the room exists, b) what objects are in the room, c) what entities are in the room at that time, d) any observable attributes about those objects and entities and e) any knowledge that is tagged to audible lines of dialogue. This knowledge helps the director decide what particular plot elements should be occurring at what time.

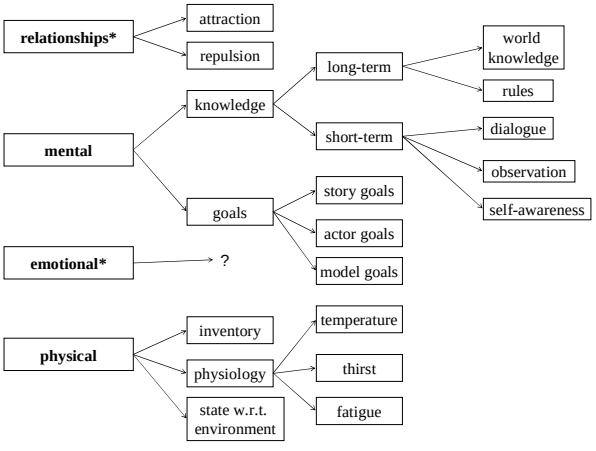

Figure 2. Taxonomy of knowledge used in Haunt 2.

The knowledge used in the plot representation, actor behaviors, and director's model of player behavior can be represented in an overall taxonomy, shown in Figure 2. The purpose of building such taxonomy is not to construct an exhaustive description of the kinds of information that could be represented in all interactive dramas, but rather to organize the knowledge that is used by our particular architecture for reasoning about and describing the world. By understanding and explicitly organizing the kinds of knowledge that is used in our system, we can have a better understanding of where our system's strengths and weaknesses are in expressivity.

The top level of the taxonomy organizes information by the different dimensions that can be used to describe game constructs (e.g. entities, items, how they relate together, and the directors' internal models of entities). The AI actors can be defined by their emotional, mental and physical states, while the items in the world can be describe by their physical state only. The director can include any and all of the knowledge in this taxonomy while maintaining a model of the world and the characters in it (e.g. keeping track of the relationships between two characters, the hypothesized knowledge that the player has of that relationship or the location of items). The mental constructs in the taxonomy are split into *knowledge*, which is the combination of an agent's or user model's long-term and short-term knowledge, and the *goals* that an agent or player model may pursue. The goals may come from the story representation, from the character's desires, or in the

case of the player model, from the hypothesized goals of the player. The parts of the taxonomy labeled with an "*" have not yet been implemented in Haunt 2. We're currently working on including a simple model of relationships between the characters, as well as incorporating an emotional into our agent architecture (Marinier and Laird 2004).

Plot Monitoring

In an interactive drama, it is important that some part of the game architecture, either explicitly or implicitly, keeps track of where the player is in the progression of the story. As illustrated in Figure 1, the director agent follows the plot description and executes story direction when needed. The story in Haunt 2 takes place in a bed and breakfast inn. The player's character is murdered at the beginning of the game and re-awakens as a ghost. The rest of the game involves the player gathering information from the synthetic characters, trying to figure out who killed him, and manipulating the characters so that one of them (hopefully an innocent one) discovers the dead body and realizes a murder has taken place. It is the director's job to follow the player's journey through the plot and elicit the various dramatic events that take place as specified in the plot description.

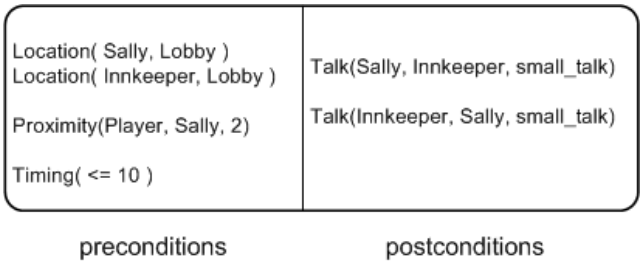

Figure 3. The first plot point in Haunt 2.

Scenes are defined by partially ordered atomic story events called *plot points*, as shown in Figure 3. Each plot point is comprised of a set of postconditions, which describes what happens in the world at this point in the story, and a set of preconditions, which describes what needs to be true in the world in order for these actions to be performed. In one of the earlier scenes of this very short story, the player is introduced to the characters "the Innkeeper" and "Sally" by overhearing a conversation of theirs in the main lobby. As shown in Figure 3, when the player is within earshot of the two characters, the Innkeeper and Sally should begin their conversation.

Both preconditions and postconditions have *logical descriptors* that describe something about the world. In our Figure 3, the statement *Location(Innkeeper, Lobby)* describes a relationship between an area in the world and a character. In order to allow more abstractly defined plot points, the language allows the author to use variables in descriptors. For instance, if it is not important exactly where the Innkeeper and Sally are, only that they are together with the player nearby, we can use a general set of preconditions, such as *Location(Sally, x)* and *Location(Innkeeper, x)*. When a descriptor that includes a variable is marked as true, the variable binding is also recorded (e.g. Sally and the Innkeeper are both in the lounge, with the player nearby, so *x* will be bound to *Lounge* in this plot point and any other point where it occurs). This simple change in the story representation language opens up the plot space; plot content is no longer fully specified by the author, but is instead partially determined by the actions of the characters and the player. Plot variables can be shared at a global level across plot points, allowing a variable that is instantiated early in the story to be referenced later. This is a representational detail that provides a very clear connection between player behavior and plot content; the choices that the player makes can have a very direct impact on the content of the plot he is taking part in.

It is the director's responsibility to compare its knowledge about the world and mark preconditions as true or false accordingly. As shown in Figure 1, a plot point cannot be considered for performance (i.e. the plot point is "active") until all of the point's predecessors in the plot structure have been performed. Once all of the preconditions for an active plot point are true, that point's postconditions are executed by the director (e.g. the director sends direction to the Innkeeper to begin the "small talk" conversation), the plot point is set to be inactive, and its children are set to be active. This is one of the cases where the director executes story direction.

Recognizing Errant Player Behavior

The director is responsible for both following the plot as it progresses and attempting to keep it moving if it stalls. The plot may not be able to continue if the player executes an action that threatens one or more preconditions from a plot point that has not yet been performed. The director is designed to not consider any action as a threat until the plot point's timing constraint has been violated. While a more complex approach would be to consider threats to *any* future precondition, to start with we are taking the simpler approach in our current design. When a plot point is set as active, it typically has a timing constraint associated with it. A *timing constraint* is a special precondition that signifies some pacing information for a particular plot point. Our architecture gives the author of an interactive drama the means to specify how quickly things should happen in the world. A timing constraint is initially stored as a relative value or range of values that represent "in what time range this plot point should be performed after the performance of its parent." In Figure 3, we can see that this plot point should fire by 10 time units after the beginning of the game. This allows the director to encode such timing concepts as urgency or pacing into the plot.

The director can also execute direction because of hypothetical future player behavior. Our architecture is designed to try to avoid conflicting player behavior before it happens. The system models short-term player behavior and treats the results of that model as a hypothesis of future player behavior. In order to model the player's behavior, whenever a plot point is finished, the director creates an internal copy of the world state. The director also has a simple internal, rule-based model of the player's behavior. The director runs that model on the simulated world, executing rules to simulate how the world would respond to the player's actions, and observing what plot elements are affected by the model's actions. The model may return a "success," meaning that an active plot point's preconditions are fulfilled by player behavior, or a "failure," indicating that no active plot points are fulfilled. For example, after the game has begun, the director may observe some of the player's actions, which consist of staying in the room where he was created. The director then creates a copy of the world, runs the player model on that copy, and returns the result that the player will remain in that room (probably examining objects) until the next plot point's timing constraint is violated in the simulation. Therefore, the modeling result is a "failure" and the director should execute some director action to get the player closer to Sally and the Innkeeper's forthcoming conversation.

Reconciling Errant Behavior with the Plot

When there is a problem, real or hypothesized, with the flow of the story, the director dynamically alters the world to get the story back on (or to stay on) track. In principle, the director should be able to change any accessible parameter in the game state to guide the player's experience. What we describe here are the current implemented capabilities of the director in Haunt 2.

The director can affect the state of three major components of the world: the synthetic characters, the objects in the world, and the environment. The synthetic characters in Haunt are rule-based, goal-oriented agents implemented in Soar (Laird et al. 1987), with long-term knowledge stored as productions and all other knowledge stored as working memory elements. Their behavior is determined by their long-term knowledge, the information present in working memory, and an internal physiological model, which includes physical attributes such as *thirst* and *temperature* (Magerko et al. 2004). An agent may decide to go order a drink at the bar if its thirst level is too high and there are no important story related actions to carry out at the moment. The director can give the characters new goals (e.g. "get a drink"), information about the state of the world (e.g. "the player is in the lobby now"), or specific atomic actions for them to perform (e.g. "perform dialogue line #2 now, speaking to Sally). These directions change the working memory of the agents, and therefore alter their behavior. It is also possible for the director to change a character's physiology to indirectly affect behavior (e.g. making a character thirsty or cold). The director has a library of directions to choose from, each labeled to help match it to the appropriate situation.

The director can create or remove objects from the world, as well as change several physical parameters associated with that object (e.g. location). This may be especially useful if the user is predicted to alter, or actually has altered, an important object in an irreversible manner. For example, the player may have unwittingly destroyed an old book that contained a piece of information key to the story. As opposed to the story coming to a halt, the director can create a new book with the same information in a part of the house the user has not been to yet, or place it on the person of one of the characters. There is an important interplay here between the hypothesized knowledge base of the player and what the director can do. Having such a knowledge base as an input into the decision-making process of "what can I do to move the story along?" adds a check on the believability of the action. Creating a book out of thin air in a room that the player has just left is not as subtle or as believable as creating another copy in the hands of a character that is elsewhere in the building or creating it in a room that the director knows the player has not visited.

In terms of environmental story direction, we have given the director control over lighting and sound triggers in the world. If the director wishes to attract the player to a particular nearby room, there are sound triggers that are accessible to the director that may be triggered in that room (e.g. the clock chiming in the lounge). Such actions are useful as subtle attractors to different areas, objects, or even characters in the world. If the player is in the lobby, but we need him to be in the lounge, then chime the clock's bells loudly as a new event in the world that may draw the player towards the sound. The director can also attempt to attract or repel the player from a particular room by manipulating light levels in the building. If the player is hanging out in the lobby, but really should be moving on to the lounge, the director can raise the light level in the lounge, giving some dialogue to the Innkeeper like "Ah, that's better! Now I can see who I'm talking to," or even turn out the lights in the lobby, directing the Innkeeper to say loudly "Sounds like we've blown a fuse downstairs. I'll look into it after we're done with this chat of ours."

Discussion

The current language used by the director for giving directions is comprised of goals (and lists of goals) that the director can send to an agent to fulfill a particular plot element. In our taxonomy shown in Figure 2, this knowledge is the intersection of story goals and actor goals (i.e. the goals shared between the story and the agents). It is up to the designer to be sure that the directions can actually be achieved by the actors; if an actor gets a command that it does not recognize, it will be ignored.

We have considered two different approaches for determining which direction is appropriate for any particular descriptor. Our first approach to this was similar to that in Weyhrauch's MOE (Weyhrauch 1997) director. Descriptors were annotated with specific directions to help fulfill them. When a particular story element needed to be encouraged, it was annotated with the exact direction needed. We have since opted for a more modular approach to representing directions in the agent. Descriptors are annotated with a classification, such as *proximity* or *knowledge*. This classification denotes what strategies are most appropriate for a particular descriptor. When a descriptor is marked as needing direction, the director examines the entire set of directions, matches on those that are of the same classification, and then chooses between whichever are applicable for this particular situation. For example, there may be two direction rules written in the agent for the *proximity* class, one that involves only synthetic characters and one that involves a synthetic character and the player. If *Proximity(Player, Sally, 1)* requires direction, then the director would match that descriptor to the *Proximity* action that deals with the player and another synthetic character. This approach allows for the reuse of director actions across multiple descriptors, the authoring of actions that can apply to many descriptors or just a single one, and encourages future work in researching the different kinds of strategies humans use in story mediation (e.g. the strategies used by dungeon masters in table-top games or online games like *Neverwinter Nights*).

At the current stage of our work, our player model is a barebones representation of player behavior. Our focus is on incorporating a player model and showing that it is effective in preemptive story guidance. There are still open questions with how to more accurately and efficiently include the synthetic characters' behaviors into the modeling process. Future work will involve enriching the model, verifying its effectiveness, and examining the incorporation of more complex methods of modeling, such as learning the likelihoods of the different possible goals in the model. The verification of the effectiveness of using a predictive model of player behavior in an interactive drama would be a significant contribution to the field and is the goal of our future work.

While we are attempting to show the usefulness of player prediction in an interactive drama, we would like to explore additional approaches in the future, such as the incorporation of a depth-limited search into the director's decision-making process. When querying a predictive model, we are in effect asking the question "Is the player *likely* to reach a future plot point?" If the answer is "no," to this question, it seems relevant to also ask "Is it even *possible* for the player to get to the next point?" If the player is unlikely to get further in the plot, but it is indeed possible to, the appropriate actions for the director to take may be different then if the world has been changing so it is actually impossible to move forward, as in our earlier example of destroying an important plot device. Therefore, our future work will focus on incorporating a search procedure into the director's decision-making process if the predictive model returns a "false" result.

References

Beal, C. R., Beck, J., Westbrook, D., Atkin, M., Cohen, P. 2002. Intelligent Modeling of the User in an Interactive Environment. AAAI 2002 Spring Symposium Series: Artificial Intelligence and Interactive Entertainment.

Klesen, M., Kipp, M., Gebhard, P., Rist, T. 2003. Staging exhibitions: methods and tools for modeling narrative structure to produce interactive performances with virtual actors. *Virtual Reality* 7(1): 17-29.

Laird, J. E., A. Newell, Rosenbloom, P.S. 1987. Soar: An architecture for general intelligence. *Artificial Intelligence* 33(3): 1-64.

Magerko, B., Laird, J. E., Assanie, M., Kerfoot, A., Stokes, D. 2004. AI Characters and Directors for Interactive Computer Games. 16th Innovative Applications of Artificial Intelligence Conference. Forthcoming.

Marinier, R. & Laird, J. (2004). Toward a Comprehensive Computational Model of Emotions and Feelings. International Conference on Cognitive Modeling 2004. Forthcoming.

Mateas, M. and A. Stern. 2002. A Behavior Language for Story-Based Believable Agents. AAAI 2002 Spring Symposium Series: Artificial Intelligence and Interactive Entertainment.

Sgorous, N. M. 1999. Dynamic Generation, Management, and Resolution of Interactive Plots. *Artificial Intelligence* 107: 29-62.

Szilas, N., Marty, O., Réty, J. 2003. Authoring Highly Generative Interactive Drama. 2nd International Conference on Virtual Storytelling.

Weyhrauch, P. 1997. Guiding Interactive Drama. PhD Thesis, Carnegie Mellon.

Young, R. M., Riedl, M. O., Branly, M. Jhala, A., Martin, R.J., Saretto, C.J. 2004. An Architecture for Integrating Plan-based Behavior Generation with Interactive Game Environments. *Journal of Game Development* 1(1): 51-70.