

Mediation in Mimesis Liquid Narratives

C. J. Saretto

cj@ndenial.com

R. Michael Young

Advisor

Liquid Narrative Research Group
Department of Computer Science
NC State University
Raleigh, NC 27695

Abstract

Films and novels effectively convey intriguing stories, powerful emotions, and meaningful messages to their audiences. Telling interactive stories in a virtual environment seems a natural progression of the narrative in light of modern technology. Unfortunately, current attempts often fail to engross the user by limiting his/her abilities to affect critical elements of the story. The Mimesis system attempts to tell liquid narratives that rewrite themselves as users take actions. In order to maintain the integrity of the story being told, Mimesis proposes a mediation system that arbitrates between the will of the user and the ends of the narrative. Speculative planning is utilized to determine actions the user could perform to threaten the world's current storyline and how the storyline could be rewritten to accommodate those actions. If the storyline cannot be effectively rewritten or if intervention is preferred for a given action, the system is responsible for determining a realistic way of preventing the action. This is done by substituting one of the action's failure modes for the action. An advanced controller determines whether to handle user actions threatening the current moment in the story via accommodation or intervention. A precaching scheme inside the virtual world environment keeps track of the controller's decisions and accommodates or modifies user actions in real-time.

Introduction

Films and novels are only successful in relaying their intended message when they fully captivate the attention of their audience. This attention is easily broken by outside forces beyond the narrative's control. However, the narrative must be very careful not to produce such a force itself. An unbelievable plot twist or badly written passage can just as easily pull the audience from the context of the narrative as unwelcome music from the next room. This problem is even more prominent in interactive narratives. When an audience member is unable to take a desired action, that audience member is pulled from the context of the narrative. The audience member becomes frustrated, and the brunt of that frustration rests not with a character in the narrative, but with the narrative itself.

The Liquid Narrative research group is a multidisciplinary group of faculty and students at North Carolina State University seeking to create successful interactive narrative environments. To do this, we are designing and building intelligent systems capable of creating structured interaction within virtual worlds. This structure works to eliminate narrative anomalies that may pull the audience member, or user in our case, out of a believable interactive narrative, and engross the user within a rewarding yet persistent virtual experience. The goal is to achieve the same kind of cognitive and affective responses to interactive stories as that seen in the participants of conventional narrative media such as the film or the novel. ²

The Mimesis project is the current focus of the Liquid Narrative group. The Mimesis system attempts to balance user control with narrative goals through a mediation process. A transparent mediator stands between the interactive narrative and each audience member. Audience members are allowed to take any action they please. Only when the audience member takes an action that jeopardizes the narrative's ends does the mediator step in and attempt to

resolve the issue in a realistic manner that does not break the audience member's context.

1. Problem

Many scholars have painstakingly detailed the composition and characterization of narrative. If narrative is the telling of a story, and the heart of a story is plot, then let us turn to Aristotle:

‘A whole is that which has a beginning, a middle, and an end. A beginning is that which does not itself follow anything by causal necessity, but after which something naturally is or comes to be. An end, on the contrary, is that which itself naturally follows some other thing, either by necessity, or as a rule, but has nothing following it. A middle is that which follows something as some other thing follows it. A well constructed plot, therefore, must neither begin nor end at haphazard, but conform to these principles.’¹

Let us extend Aristotle's definition to define a story as the combination of setting and plot. Thus a story has four basic parts: a setting, a beginning, an ending, and a series of actions with consequences. This may seem like an oversimplified definition. On the contrary, it proves to be just complicated enough to accomplish our goal.

The setting of a story is the virtual world in which the story takes place. The virtual world is a representation of a real or imaginary place with some defined physical layout. As necessitated by plot, there are some number of objects and actors in the virtual world. To facilitate action, there are a series of rules that dictate how objects behave, what actors can do, and what interactions among objects and actors are allowed.

Current artificial intelligence planners, such as Longbow⁴, store the initial, current, and goal states of their problem space as a series of predicates. They then model all operations on the problem space as actions with preconditions and effects. The planner generates a plan by determining a series of actions that, when applied to the initial state of the problem space in a given order, will lead to the achievement of the goal state. The final plan includes a list of actions, a list of ordering constraints on those actions, and a list of causal links. Each causal link represents a single effect of one action that directly satisfies a single precondition of a following action.

Our definition of story is very applicable to the problem space model used by planners. The beginning of the story is the state of our virtual world when the first actor, controlled by an audience member (player), enters it. The end of the story is defined by a series of conditions that must all be satisfied. If we properly define predicates for our world, then the beginning and end of the story can easily be made into the initial and goal states of a plan.

In order for the story to play out, actors in the world must perform actions. Naturally, these actions will have consequences. Some actions will cause one or more of the conditions dictated by the story's ending to be satisfied. Proper definition of these actions will allow a planner to create a successful storyline.

The Mimesis system attempts to give a player the greatest freedom of action possible while maintaining the integrity of the narrative being told. An advanced artificial intelligence planning system represents the narrative as an aforementioned story³. At the beginning of a story, it creates a plan of action to guide the story to its end. An intelligent controller then begins manipulating the world and its actors according to the plan. It is highly unlikely that the player actors will behave as predicted by the plan, but a starting point is necessary. As players take actions, the controller is notified. The controller takes this information, creates a new initial plan state, requests a new plan from the planner, and begins executing with the new plan. Thus, a set past added with a planned future always leads to story completion.

What happens if the player does something that makes the ending impossible? To deal with the problem more directly, an example is in order: What if the player kills Grendel in the first act of the epic story Beowulf? Do we bring Grendel back to life for the final act? Do we not give the player the opportunity to attempt the early defeat by removing all of the player's control? Do we allow Grendel to survive a deadly blow? Almost all currently existing video game systems would take one of these three approaches. If the player desires to defeat Grendel in the first act, any of these solutions will seem unrealistic and break the player from the context of the narrative.

One of Mimesis's goals is to handle these types of situations with grace. The player must not be allowed to compromise the narrative by defeating Grendel so early. However, there are many acceptable solutions to the problem. Perhaps Grendel bats the player's sword from his hand in mid-swing to facilitate escape? This action is an obvious solution to the problem as presented. The means for implementing a decision process that could choose this course of action from infinite possibilities and execute it in a timely manner is not so obvious.

The traditional reactive planning approach would dictate that the world report all player actions back to the intelligent controller. The controller would then attempt to replan the story. If a new plan could not be generated, then the player action would need to be undone. This is unacceptable in an interactive narrative. To avoid undoing player actions, the world could request authorization from the controller for every action the player attempts. A slow network link or an action with complicated consequences could cause serious latency problems. Both traditional approaches break the player's context with the narrative, which is the exact problem they are attempting to solve.

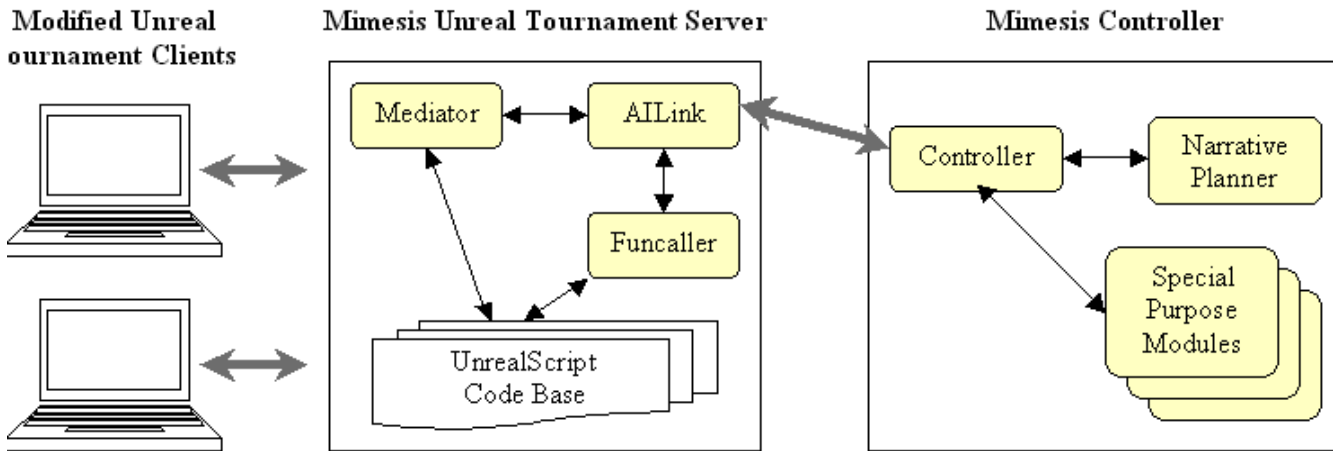


Figure 1: Mimesis System Architecture

2. Solution

Mimesis proposes a player Vs narrative mediation scheme using advanced caching and speculative planning techniques to handle these situations in a manner that is fair, believable, and imperceptible to the player.

2.1. Mimesis Background and Architecture

The Mimesis system is comprised of a Longbow planner variant, an intelligent controller, and a modified version of Unreal Tournament (an off-the-shelf commercial game). Unreal Tournament (UT) provides the Mimesis team with a stable, network-enabled, client-server 3-D gaming system to implement virtual worlds. Modifications to UT are written in a language called UnrealScript. UnrealScript is a Java-like, object-oriented language with built-in support for concepts such as time slicing, state, and network replication. By extending the existing UnrealScript classes, Mimesis programmers can easily modify the existing behavior of UT and create new behaviors.

The Mimesis system, as shown in Figure 1, is split into three primary components: a modified UT client, the Mimesis Unreal Tournament Server, and the Mimesis Controller. The Mimesis Unreal Tournament Server (MUTS) is an Unreal Tournament Server with additional worlds, characters, sounds, music, and Unreal Script code. The modified UT client contains a minimally necessary subset of the MUTS additions. The Mimesis Controller (MC) is made up of the planner and intelligent controller. The modified UT client, MUTS, and MC communicate with each other via network socket connections. This allows each component to reside on a different machine or for all

components to reside on the same machine without reconfiguration.

When a MUTS is started, it establishes a connection to an MC via the AILink. The AILink is a network socket module that relays communication between the MC and various MUTS components. The MUTS tells the MC specifics such as which story is to be told and what the world looks like. The MC then creates a plan of action to drive the story to completion. Lastly, the MUTS waits for the MC to issue commands or a player to connect.

When a UT client connects to a MUTS, the client is given a world state to replicate. Once the client has successfully set up its world, it begins accepting and requesting updates to the world state. As actors and objects move about and perform actions in the world, all clients are notified so that they can accurately represent the world to their users. Whenever the user of a client moves or performs an action, that information is passed to the server. The server updates the world state, informs the MC of the change, and sends notification to all clients.

2.2. Failure Modes

In order to facilitate the explanation of the mediation system, we must first introduce the concept of failure modes. In traditional AI planning systems, actions are autonomous. They have set preconditions and their effects are always assumed to be true. Some real-time controllers will check to see if an executed action had the desired effect, but they do not consider the different ways the action might fail before executing that action.

A failure mode is a link between an action and a similar action that has the same preconditions, but at least one different effect. Failure modes allow the MC to consider ways to deal with user actions that have undesired results.

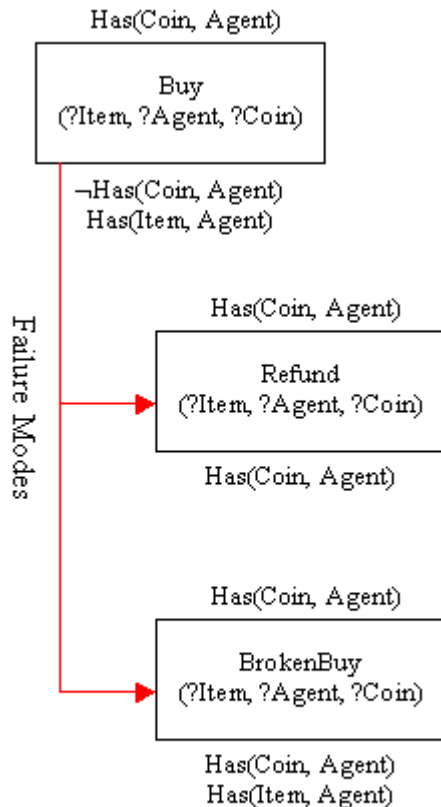


Figure 2: Vending Actions

Let us consider the example shown in Figure 2. The action of purchasing an item from a vending machine is named Buy. The precondition for Buy is that the agent purchasing the item has a coin. The effects of Buy will be that the agent no longer has the coin, but that the agent now has the desired item.

An attempt to purchase an item from a vending machine could fail. The agent's coin might fall through the machine and not be accepted. On the other hand, the machine could be broken and give the agent the desired item as well as returning the coin. These actions are represented as Refund and BrokenBuy respectively in Figure 2. They are both failure modes of the action Buy. If an attempt is made to insert the action Buy into a plan and one or more of the action's effects is a threat, it could be replaced with one of its failure modes that does not have the threatening effect(s).

2.3. Mediation

In order to mediate between the will of the player and the ends of the narrative, the mediation system must have some knowledge of the problem space. It must know what player actions are possible in the world, which of those actions threaten the current narrative plan but can be worked around, and which of those actions will cause irrevocable

damage to the current narrative plan. The system must also be able to monitor the actions of a user as they are performed allowing it to keep its narrative plan updated and stop or substitute user actions as necessary.

The Mediator is an UnrealScript object that exists as part of the persistent world. The key to its operation is that a UT client never takes action on its own. Rather, it informs the UT server that an action should occur. Unfortunately, these client-server exchanges are not publicly interceptable. In order to intercept this information, the MUTS only allows specialized players of type MPlayer to exist in the world. MPlayers are specialized player actors that always request permission to perform an action from the Mediator. MPlayers also inform the Mediator of any events that occur to them such as bumping into a wall, entering a new zone, or another character coming into view. Using these self-mediating characters, the Mediator doubles as a discrete event generator for player actions. All discrete events are reported via the AILink to the MC. In this way the MC is constantly aware of player actions and can replan the storyline as needed.

When the MC would otherwise be idle, it spends its time performing speculative planning. The MC looks through the causal links of the currently executing plan for links that player actions could threaten. The MC sends these actions to the Mediator for monitoring.

Next the MC creates "what if" scenarios to determine the effects of the threatening actions on the narrative plan. The MC must make a decision to either accommodate the user and allow the action or to intervene and prevent the action by substituting it with one of the action's failure modes. The details of this decision are not trivial, but they are beyond the scope of this paper. In the event that a user action prevents the narrative from reaching its end, then the action must be prevented.

Once the MC decides whether to accommodate or to intervene for an action, it must update the Mediator's record of that action. Actions selected for intervention are updated by sending the Mediator alternate actions to perform in place of the offensive actions. Actions selected for accommodation are updated by sending the Mediator meta-data to flag the action for accommodation.

As the Mediator receives player actions to monitor and updates to those actions from the MC, it stores them in a data structure called the "bad list". As players attempt to perform actions in the world, the bad list is consulted to ensure the action is safe. If the action is not in the bad list, it is considered to be safe and is allowed. If the action is in the bad list, the action is treated as unsafe and handled in the following manner:

1. If the action is flagged for intervention, the replacement action is performed and the original action is canceled. A message indicating that the action was performed and replaced is sent to the MC.

- If the action is flagged for accommodation, the action is allowed. A message indicating the accommodation was performed is sent to the MC.
- If the action is not flagged, the Mediator sends a high priority message to the MC requesting a decision to accommodate or intervene. If the Mediator does not receive a response from the MC in a predetermined timeout period, the MC accommodates the action and informs the MC via a message.

The current Mediator prototype handles three player actions and one player event. These are Fire, SwitchWeapon, Trigger, and ChangeZone. The Fire action occurs when the player attempts to fire a Weapon. Fire could have failure modes such as FireUnloadedWeapon or FireButMiss. The SwitchWeapon action occurs when a player tries to draw a weapon or switch to a different weapon from the currently drawn weapon. It is important to note that a weapon in UT is simply an object held in the hand that can be utilized (fired). The term ‘weapon’ is unfortunate as possible weapons range from rocket launchers to scientific equipment.

The Trigger action occurs when a player comes in contact with an MTrigger. MTriggers are invisible objects in the world that report to the Mediator. When a player comes in contact with them, they inform the mediator what action they wish to perform. MTriggers are used for actions such as opening doors.

ChangeZone occurs when a player enters a new zone in the world. World designers define zones when the world is built. ChangeZone is unique because it is an event. Events can only be observed. Thus the Mediator only serves to inform the MC when events occur, as it cannot prevent them.

3. Example

As an example of the mediation process, let us examine a short, simple story in detail.

Story Setting

Our story is set on a small island. There is a vending machine on the island as well as a drawbridge leading off the island. The vending machine and drawbridge are both coin operated. The vending machine uses the actions described earlier in Figure 2 and Section 2.2. There is a player character named Bob on the island.

Story Beginning

Our story begins with Bob on the island. Bob has a coin in his possession. We define the initial state as:

OnIsland(Bob)

At(Start, Bob)
Has(Coin1, Bob)

Story Ending

Our story ends when Bob successfully travels off the island. We define the goal state as:

¬OnIsland(Bob)

Action Definitions

Action: Go(?there, ?here, ?agent)

Preconditions:

At(here, agent), ¬At(there, agent)

Effects:

¬At(here, agent), At(there, agent)

Action: OpenBridge(?coin, ?agent, ?bridge)

Preconditions:

At(bridge, agent), Has(coin, agent), ¬Open(bridge)

Effects:

¬Has(coin, agent), Open(bridge)

Action: LeaveIsland(?agent, ?bridge)

Preconditions:

At(bridge, agent), Open(bridge), OnIsland(agent)

Effects:

¬OnIsland(agent)

Action: Buy(?item, ?agent, ?coin, ?machine)

Preconditions:

At(machine, agent), Has(coin, agent)

Effects:

Has(item, agent), ¬Has(coin, agent)

Failure Modes:

Refund(item, agent, coin, machine)

BrokenBuy(item, agent, coin, machine)

Action: Refund(?item, ?agent, ?coin, ?machine)

Preconditions:

At(machine, agent), Has(coin, agent)

Effects:

Has(coin, agent)

Action: BrokenBuy(?item, ?agent, ?coin, ?machine)

Preconditions:

At(machine, agent), Has(coin, agent)

Effects:

Has(item, agent), Has(coin, agent)

Initial Plan

When Bob enters the world, a simple initial plan is generated as shown in Figure 3. The plan shows that Bob need only go to the drawbridge, use his coin to open the

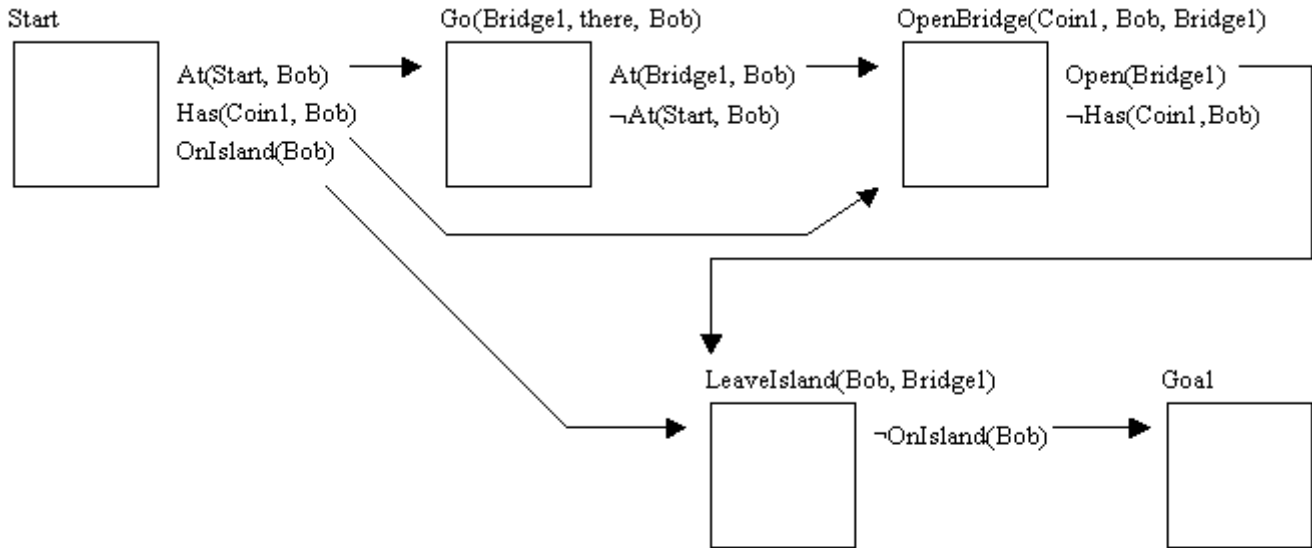


Figure 3: Initial Plan for Example

bridge, and leave the island to complete the story. The current time is somewhere between the initial state and Bob reaching the drawbridge. There are two causal links spanning the current time. These are the provision of $At(Start, Bob)$ to $Go(there, Bridge1)$ and $Has(Coin1, Bob)$ to $OpenBridge(Coin1, Bob, Bridge1)$ from the initial state.

There are two possible user actions that threaten these causal links, they are Go and Buy respectively. The MC engages in speculative planning to determine how to handle the event that Bob attempts to execute one of these actions. The MC sees, by creating a speculative plan, that if Bob goes somewhere other than the bridge, he can go to the bridge later. The MC chooses to accommodate this action. The MC also creates a speculative plan where Bob attempts to buy something. In this plan, Bob is now short the coin he needs to open the drawbridge. The MC must choose one of the following options:

1. Intervene by substituting the failure mode $Refund$ for Buy .
2. Intervene by substituting the failure mode $BrokenBuy$ for Buy .
3. Accommodate the action Buy , but create a new subplot that results in the drawbridge being open. Perhaps place a new coin in the world for Bob to find, or create a new character to open the bridge for Bob.

Assuming that the MC chooses option 1, the initial bad list for the Mediator is:

Action	Mediation
$Go(there, here, Bob)$	$Accommodate$
$Buy(Coin1, Bob, machine)$	$Refund(Coin1, Bob, machine)$

Fortunately for our example, Bob goes to a vending machine before he goes to the drawbridge. The Mediator accommodates this action, and the Mediator informs the MC of the accommodation. Bob is hungry, so he attempts to purchase a bag of corn chips from the vending machine. When Bob attempts this Buy action, the Mediator steps in and replaces it with the $Refund$ action. Bob's coin falls through the machine and is returned to him, and the MC is notified of the substitution.

The MC now knows that it has represented the vending machine to Bob as not accepting his coin. It may choose to continue this representation, or it may choose to placate Bob by updating the Buy action to be accommodated.

In this example, the MC chooses for the vending machine to never accept Bob's coin. Bob does not get his corn chips. This may displease Bob, but it does so in a believable manner that does not break Bob's context with the narrative. Bob may choose to seek out another vending machine. It would not be prudent for all vending machines on the island to return Bob's coin. As shown here, the Mediator keeps the MC updated on Bob's actions. It is the MC's responsibility to keep the Mediator's bad list updated in a manner that produces believable interventions and accommodations.

4. Conclusions

The Mimesis team assumes that our stories will give the player enough freedom such that the majority of his/her actions are safe or accommodated. If this holds true, mediation provides the Mimesis system with significant advantages. Network traffic between clients and the MUTS is no higher than between standard UT clients and servers. The bad list cache significantly reduces the latency of user actions by eliminating the need to consult the MC on every user action. Since the MC is not required to approve actions, it is free to spend its time on speculative planning.

Mediation is a new idea in virtual worlds research and narrative story telling. Though the Mediator and MC are only in its preliminary stages, the Longbow planner used by the MC is already capable of efficiently handling complex situations and generating hierarchical plans. As work on the Mediator progresses, it is our goal to develop the MC's functionality in direct parallel.

Most importantly, this work is very far reaching and can be applied to any type of interactive narrative story telling system. In fact, one of the challenges for the Mimesis system is to properly represent actions taken in a complex 3D environment as discrete events. 2D or text based systems could possibly make more productive use of these concepts presented here in the short-term.

5. Future Work

There are many actions and events already defined by the UT event structure. Work is now being performed to determine which of these will be relevant to our purposes, where new actions can be built into the UT event structure, and where information needed for mediation will not be satisfied by the UT event structure.

The Mediator currently checks every action or event against all entries in the bad list before approving it. Future work will center on creating more efficient data structures for determining acceptance. As the list of detectable actions and events grows, and larger stories dictate larger bad lists, the Mediator must become more efficient.

One possibility not addressed by the current mediation system is the participation of other actors in the world during intervention. Indiana Jones might be distracted from picking up an ancient key when the ruin around him starts shaking violently. Alternately, his companion might grab his hand and warn him that the key is cursed. Both of these scenarios cannot be handled entirely by failure modes. The Mediator would need to deny the player request and instantaneously set the events of the more complicated intervention into motion. Current intelligent camera and scene control work for the Mimesis system may help enable scenarios such as this in the future.

Acknowledgements

This author could not have developed, implemented, or published this research without the constant support of his faculty advisor, Dr. Michael Young, and the North Carolina State Department of Computer Science. This paper was produced in fulfillment of the NC State Computer Science Undergraduate Honors Program.

The work of the Liquid Narrative group has been supported by a number of sources, including a Faculty Research and Development grant from the NC State Office of the Provost, and equipment grants and gifts from the NC State Department of Computer Science and Microsoft Research.

References

1. Aristotle, 350 BCE. Poetics, <http://classics.mit.edu/Aristotle/poetics.html>.
2. Young, R. Michael, 2001. An Overview of the Mimesis Architecture: Integrating Intelligent Narrative Control into an Existing Gaming Environment in The Working Notes of the AAAI Spring Symposium on Artificial Intelligence and Interactive Entertainment, Stanford, CA, March 2001.
3. Young, R. Michael, 1999. Notes on the Use of Plan Structures in the Creation of Interactive Plot in The Working Notes of the AAAI Fall Symposium on Narrative Intelligence, Cape Cod, MA, 1999.
4. Young, R. Michael, 1994. A Developer's Guide to the Longbow Discourse Planning System, University of Pittsburgh Intelligent Systems Program Technical Report 94-4.