Dynamic, User-Centered Resolution in Interactive Stories

Nikitas M. Sgouros
Institute for Communication & Computer Systems
National Technical University of Athens
Zographou Campus, Greece, 157 73
sgouros@dsclab. ece. ntua.gr

Abstract

One of the most important parts of a story is its ending. This is the point in which all open questions and conflicts in the narrative must be resolved. This paper presents a dynamic resolution method for interactive stories. The type of stories we support allows the user to participate as one of the characters and influence dynamically the development of the plot. The rest of the cast consists of discrete computer characters. Our resolution method takes into account the motives of the user character to decide on: (i) the outcome of all character actions and (ii) the presentation sequence for these outcomes. The decision process is based on the current story context. In addition, it addresses the need to avoid ambiguities, to preserve consistency and to create suspense during the end. We have integrated this method in PEGASUS, an interactive story system set in ancient Greece.

1 Introduction

One of the most important parts of a story is its ending. This is the point in which all open questions and conflicts must be resolved in order for the audience to understand the main idea of the story and integrate all the plot developments around it. There are two types of possible endings. The first one resolves all open questions using the information provided in the story so far. The second one, known as *deus ex machina,* is based on the intervention of a character who has played no part in the story until that moment. Although quite popular in ancient dramatic practice, deus ex machina solutions have been dismissed very often in modern dramaturgy mainly for lack of believability.

This paper presents a resolution method for interactive stories that supports the first type of ending. Interactive stories are narratives in which the user takes part as one of the characters and interacts with the rest of the cast to influence dynamically the development of the plot. Cast interaction is based on the execution of appropriate actions by each character. Plot resolution, in this case, refers to the computation of the outcome of all character actions and the presentation of these results in the story. This work describes a resolution method that is *dynamic, user-centered* and *unambiguous.* Dynamic means that it does not support a priori specific types of ending, such as the happy end for the user character. Instead it determines the story outcome based on the contents of the current plot. As a result, this method can be used in conjunction with dynamic storyweaving methods to create stories in intelligent games or collaborative virtual worlds. User-centered means that the motives behind the actions of the user character in the story determine the final outcome. Consequently, the system presents the resolution as a consequence of user behavior in the story, thus enhancing the clarity and believability of the final outcome. Unambiguous means that the method decides on the results of *all* actions in the story, therefore leaving no questions unanswered to the user. This feature corresponds to the norm for closed, unambiguous endings found in classical theories of drama [1] and used in mainstream screen writing [3,4,5]. Furthermore, this method generates a presentation sequence for the final outcome that takes into account the need to create suspense during the conclusion. Finally, the decision on the results for all character actions preserves the causal dependencies between conflicting actions in the story.

Possible uses for this research include the development of dynamic storyweaving methods for interactive entertainment systems (e.g., stories, games, collaborative virtual worlds), or the creation of intelligent plot assistants for the design of these systems.

The rest of the paper is described as follows. Section 2 describes the input to the plot resolution method. Section 3 presents the steps used in this method. We give an example of the system at work in section 4. Finally, section 5 presents some related work, while section 6 is a conclusions and future work section.

2 System Input

Plot resolution accepts as input a symbolic description of the story content. This consists of a temporal sequence of the character actions in the story so far. This sequence is expressed in a special-purpose language consisting of predicates that describe these actions, their motivations and goals, along with the types of cast interaction they implement.

Resolution assumes that the behavior of each character is motivated by its *goals* in the story, the *roles* that it plays and its *relation* (e.g., friend, enemy) with the rest of the cast. In general, character behavior consists of either favorable or unfavorable interference between cast members, depending on whether the initiating character wants to harm or cooperate, respectively, with another cast member. More specifically, goals motivate each character to interfere with the rest of the cast in order to achieve them. Roles provide the norms regulating the behavior of each character in the story. Each cast member seeks to protect its role-relevant norms from violations by other characters by interfering unfavorably against the offenders. Analogously, each character is motivated to interfere favorably for cast members with similar normative beliefs. Finally, positive (e.g. friendship) or negative (e.g. enmity) social relations motivate the parties involved to either help or block, respectively, each other in the service of their goals.

Table 1 describes the primitives representing the possible character interactions and their outcome. Interference (predicates /+ and I- in Table 1) gives rise to the execution of one of possible actions for achieving it. These can be either actions for satisfying a goal or norm (predicate *Exec+* in Table 1), or counter-actions for impeding it (predicate *Exec-* in Table 1). For each action the motivating interference is noted (predicate *Motivates* in Table 1). Finally, action execution and interference can either succeed or fail (predicates *Succeeds* and *Fails* in Table 1).

For example, a character playing the priest role in the story, will seek to interfere with the behavior of the rest of the cast to impose all the norms associated with worshipping and obeying the gods. Furthermore, a character (e.g., *X)* with a goal of acquiring a valuable resource (e.g., gold) from some other character (e.g. *Y)* will seek to interfere favorably or unfavorably with Y to satisfy this goal. This interference will initiate a set of appropriate actions for materializing it. For example a stealing action for the resource in question would materialize an unfavorable interference, while an exchange of this resource would implement a favorable one for Y. These actions and, consequently, their associated interference can either succeed or fail.

3 Plot Resolution

Plot resolution refers to the computation of the outcome of all character actions and its presentation in the story. The resolution method works as follows. Initially, it determines all character motives that are affected positively or negatively by the actions of the user character in the story. Based on the importance of these motives, it decides whether all user actions will succeed or fail. The method then resolves the actions of the rest of the cast that oppose the user actions. Finally, the algorithm decides on the result of the remaining actions and composes a presentation sequence for the ending that increases suspense and preserves the causal dependencies between conflicting actions.

More specifically, plot resolution consists of the following sequence of steps:

1. Determine the motives of all user actions.
2. Determine the relative importance of these motives and resolve all user actions and the actions of the cast that oppose them.
3. Resolve the remaining actions.
4. Present the ending.

We present each one of these steps in more detail below.

Table 1: Behavior primitives involving characters *x* and y. Terms beginning with a small letter signify variables, while terms beginning with a capital letter denote constants.

Predicate	Interpretation
I+(x, y, +, g)	*y seeks to satisfy goal/norm g by helping x*
I-(x, y, -, g)	*y seeks to impede goal/norm g by obstructing x*
I+(x, y, -, g)	*y seeks to impede goal/norm g by helping x*
I-(x, y, +, g)	*y seeks to satisfy goal/norm g by obstructing x*
Exec+(x, a, g)	*x executes an action a for achieving goal/norm g*
Exec-(x, a, g)	*x executes an action a for impeding goal/norm g*
Motivates(i, a)	*interference i motivates the execution of action a*
Succeeds(a)	*action execution or interference a succeeds*
Fails(a)	*action execution or interference a fails*

3.1 Step 1

Step 1 examines the motives of all the actions of the user character and constructs two *motive sets*:

- *S+* lists the goals that will succeed and the norms that will be protected if all user actions in the story succeed.
- *S-* lists the goals that will fail and the norms that will be violated if all user actions succeed.

We refer to the motives in both sets as the *dominant motives* in the plot (*DM*). Assuming that *M* is the set of all character motives in the story, the symbolic definitions for these sets are:

$$S+ = \{m \mid m \in M \wedge Exec+(User, a, m)\}$$

$$S- = \{m \mid m \in M \wedge Exec-(User, a, m)\}$$

$$DM = S+ \cup S-$$

For example, if the personal goal of the user character in the current scene is to acquire a very precious jewel and, in this process, s/he tries to steal this piece from its current owner (e.g., *X*), thereby violating the norm protecting private property, then S+, S-, and DM will have the following form:

S+ = {Goal(User, Acquire(User, Jewel, X))}
S- = {Norm(Protect-Ownership)}
DM = {Goal(User, Acquire(User, Jewel, X)),
Norm(Protect-Ownership)}

We will say that two character actions *conflict* with each other, if they materialize reciprocal unfavorable interference between these characters and they seek to achieve opposite results with respect to a motive *m*. Symbolically:

Motivates(I-(x, y, +, m), a) ∧ Motivates(I-(y, x, -, m), b) →
→ Conflict(a, b)

/* IF x interferes against y with action a in order for motive m to succeed AND y interferes unfavorably against x with action b to block m THEN actions a and b conflict with each other. */

where the '-->" symbol separates the left from the right hand side of a rule, while the '∧' and '∨' symbols denote the conjunction and disjunction, respectively, of rule conditions or actions. The comment symbols (/*...*/) enclose an English interpretation of a rule.

In the case of the previous example let us assume that character X interferes unfavorably against the user to defend the norm protecting private property. Let us further assume that X materializes this interference by chasing the user after s/he tries to steal the jewel. This action of X conflicts with the user action with respect to the norm that protects private ownership.

3.2 Step 2

Step 2 decides whether *S+* or *S-* will prevail based on a hierarchy indicating the relative importance of motives in the story. This hierarchy can be provided either by the original author of the story (i.e., the one who created the cast and assigned roles in the story), or by the user before the story begins. Furthermore, the comparison of the motive sets with respect to their importance can be done in various ways.

For example, the motive hierarchy in PEGASUS is provided by the story author. Furthermore, this hierarchy indicates that all personal goals occupy the lowest point in it, therefore PEGASUS considers norm-related motives to be more important than the personal goals of the cast. In comparing motive sets, PEGASUS resolves the plot in favor of the set with the motive that is higher up in its hierarchy. Therefore, in the case of the example of Step 1 PEGASUS decides that *S~* will prevail, since norm-related motives are more important than personal goals in its hierarchy.

If there is no hierarchy of importance for the character motives, or if there is no conflict between them (i.e., S- = ∅), then the method chooses to provide an upbeat resolution for the user, therefore *$+* prevails. The justification for this decision is that audiences generally prefer happy endings [4,5].

If S+ is deemed to be more important that *S-* then all user actions in the story succeed, otherwise all the actions of the cast that conflict with the user succeed. This prevalence rule is symbolically described as:

Prevails(S+) ∧ Exec+(User, a , m) → Succeeds(Exec+(User, a, m))
Prevails(S+) ∧ Exec-(User, a , m) → Succeeds(Exec-(User, a, m))

/* IF motive set S+ prevails AND the user performs an action or counter-action a related to some motive m THEN a succeeds. */

Prevails(S-) ∧ m ∈ S- ∧ Exec-(User, a, m) ∧ Exec+(x, b, m) ∧
∧ Conflict(a, b) → Succeeds(Exec+(x, b, m))
Prevails(S-) ∧ m ∈ S+ ∧ Exec+(User, a, m) ∧ Exec-(x, b, m) ∧
∧ Conflict(a, b) → Succeeds(Exec-(x, a, m))

/* IF motive set S- prevails AND character x executes an action b that conflicts with an action a of the user THEN b succeeds. */

In general, if an action or counter-action succeeds during the resolution, then its associated goal or norm succeeds or fails, respectively. Symbolically:

Succeeds(Exec+(x, a, g)) → Succeeds(g)
Succeeds(Exec-(x, a, g)) → Fails(g)

Furthermore, the outcome of an action related to a goal or norm *g* has an impact on the outcome of all its conflicting actions in the story. In particular, the success of an action that seeks to achieve *g* causes all its conflicting actions to fail. The method notes the resolution order for the character actions in this case with the *Resolves* predicate. This order is used later for scheduling the presentation of action outcomes. Symbolically, we have the following pair of *resolution order* rules:

Succeeds(Exec+(x, a, g)) ∧ Exec-(y, b, g) ∧ Conflict(a, b) →
→ Fails(Exec-(y, b, g)) ∧ Resolves(a, b)

/* IF the execution of action a in support of goal g succeeds AND b is a counter-action against g THEN b fails AND the system notes that the outcome of action a resolves the outcome of action b. */

Succeeds(Exec-(x, a, g)) ∧ Exec+(y, b, g) ∧ Conflict(a, b) →
→ Fails(Exec+(y, b, g)) ∧ Resolves(a, b)

/* IF the execution of counter-action a against goal g succeeds AND b is an action supporting g THEN b fails AND the system notes that the outcome of counter-action a resolves the outcome of action b. */

In the case of the jewel acquisition example in Step 1, because S- prevails, then X succeeds in its chasing action. Consequently the user fails to steal this jewel according to the first of the two resolution order rules above.

3.3 Step 3

Step 3 proceeds in two stages. The first stage determines the outcome of character actions not resolved during step 2. More specifically, if *S+* prevails according to step 2 then the method allows all unresolved character actions driven by motives in *S-* to succeed and those driven by motives in S+ to fail. On the other hand, if *S-* prevails then the algorithm allows all unresolved character actions driven by motives in *S+* to succeed and those driven by motives in *S-* to fail.

The justification for these decisions is that user interest in the story will increase, when the algorithm introduces an unexpected plot twist, just before it presents the outcome of the user character actions. In particular, assuming that S+ prevails in Step 2, then Step 3 ensures that the user will initially watch its allies (i.e., characters supporting motives in S+ and reacting against motives in *S-)* fail and its opponents (i.e., characters reacting against motives in S+ and supporting motives in *S-)* succeed in their endeavors. Consequently, the user will expect to be defeated as well, only to be pleasantly surprised, when s/he will be presented with the favorable resolution for his/her actions computed in step 2. On the other hand, if we assume that *S-* prevails in step 2, then step 3 ensures that the user will initially watch its allies succeed and its opponents fail. Consequently, the user will expect to win, only to become adversely surprised when s/he starts failing. In both cases, the dramatic effect of the plot resolution will be emphasized.

The second stage of step 3 determines randomly the outcome of the character actions that are not driven by any of the dominant motives. Consequently, plot resolution exhibits a degree of variability, even when the user has followed the same sequence of actions in a scene. This feature is applicable in computer game applications, where the user has to go through a sequence of levels repeatedly, before s/he masters the skills that will allow him/her to move to more advanced levels.

3.4 Step 4

Step 4 schedules the presentation of all action outcomes. This presentation follows the resolution order established by the application of the resolution order rules during the previous steps. In particular, if action *a* resolves some other action *a',* then the system presents the outcome of action *a* before that of a'. If we assume that the binary relation ' *a « b* ' denotes that the outcome of its first argument (i.e., *a)* is presented earlier than the outcome of its second argument (i.e., *b),* then this rule can be described as:

Resohles(a, a') -> a « a' (1)

/ IF action a resolves actions a' THEN present the outcome of a before the one for a'. */*

This rule favors a sequence, in which the success of an action (e.g. *a)* is presented first, to explain the failure of all other actions that oppose the motives of *a.* In this case, the user realizes why the actions that fail, do so, thereby increasing the believability of plot resolution.

Furthermore, if two actions are either resolved by, or are scheduled to follow a common action, then their outcomes are presented in a reverse chronological order from the one with which these actions were originally introduced in the story. The justification for this rule is that because the user participates as one of the characters of the story, s/he perceives more recent situations in the narrative as more immediate. Consequently, s/he expects that these situations will be resolved first. If we assume that *Follows(x, y)* denotes that y is a later plot development than x, then this rule is described as:

$\text{Resolves}(a, b) \wedge \text{Resolves}(a, c) \wedge \text{Follows}(b, c) \rightarrow c \ll b$ (2)

/ IF action a resolves both actions b and c AND c follows b in the plot THEN present the resolution for c before the one for b. */*

$a \ll b \wedge a \ll c \wedge \text{Follows}(b, c) \rightarrow c \ll b$ (3)

/ IF the resolution for action a precedes that for actions b and c AND c follows b in the plot THEN present the outcome for c before the one for b. */*

Finally, if the action of a character different from the user has been resolved directly by the system, then the presentation of this outcome precedes the presentation of the outcome for the user actions. This rule supports the presentation of the plot twist described in step 3, that seeks to surprise the user during plot resolution. Furthermore, by pushing the resolution of the user actions at the end of the presentation, the system increases user suspense. This is the case, because the user is not aware of the fate of its endeavors until the last possible moment. Symbolically, this presentation rule is described as:

$\text{Exec+}(x, a, g) \vee \text{Exec-}(x, a, g) \rightarrow \text{ActionSubject}(x, a)$ (4)

/ IF x executes an action or counter action a THEN x is the subject of this action. */*

$\text{ActionSubject}(\text{User}, a) \wedge \text{ActionSubject}(x, a') \wedge x \neq \text{User} \wedge \wedge$
$\text{Resolves}(\text{System}, a') \rightarrow a' \ll a$ (5)

/ IF the user executes some action a AND character x executes an action a' AND x is not the user AND action a' is resolved directly by the system THEN the presentation of the outcome of a' will precede the presentation of the outcome of a. */*

4 An Example

We have tested this method in PEGASUS, an interactive story set in Ancient Greece. During one of the scenes in PEGASUS the user visits Eretria, a town with a big port. In Eretria the user has the goal of obtaining a ship to sail to Troy. The local cast for this scene consists of Apateonis, the local trader, Elanthos the king of Eretria, and the user. The initial motives for the characters indicate that the king is an enemy of Apateonis. In fact, Elanthos has issued a decree forbidding anyone to make business with Apateonis. Furthermore, Apateonis is a dishonest trader. In the following, we assume that:

A = Apateonis, E = Elanthos, U = User, G = Goal(User, Obtain(Ship)), RK = Norm(Respect-King), PO = NormfProtect-Ownership)

where RK is the norm demanding respect for the king, while PO is the norm protecting resource ownership.

The resolution algorithm accepts as input the temporal sequence of actions shown in Table 2. This sequence can be summarized by hand as follows:

«The user wants to obtain a ship. Consequently, he tries to hire one from Apateonis in exchange for a precious diamond that he owns. This action violates the royal decree that forbids to make business with Apateonis. Apateonis is dishonest and tries to steal the diamond from the user. The user reacts and attempts to confront the thief. Elanthos, the king seeks to punish the user for violating his decree. The user attempts to hide from the king. Elanthos tries to punish Apateonis as well for his attempt to steal the diamond. Apateonis seeks to hide from the king.>>

At this point, the system applies Step 1 of the resolution method and detects the motives of user actions in time steps #1, #4 and #7 in Table 2. Based on these actions the algorithm creates the following motive sets:

S+={G.PO} S- = {RK} DM = {G,PO,RK}

meaning that the user tries to satisfy his goal of acquiring a ship and s/he also complies with the norm that protects private ownership. On the other hand the user opposes the norm that demands respect for the king and his orders.

During Step 2, PEGASUS uses its hierarchy of motives to determine which set will prevail. This hierarchy indicates that protecting ownership is more important than obedience to the king, while the personal goal of the user is less important than either of the remaining user motives. Consequently, the system decides that S+ will prevail. This decision allows all user actions to succeed.

The algorithm now resolves the character actions that conflict with the user actions based on the resolution order rules. As a result we get:

Succeeds(Exec-(U, Hide(U), RK)) ∧ Exec+(E, Punish(U), RK) ∧
∧ Conflict(Hide(U), Punish(U)) →
→ Fails(Exec+(E, Punish(U), RK)) ∧ Resolves(Hide(U), Punish(U))

/ Elanthos fails to punish the user because the user successfully hides from the king. */*

Similarly we get that Apateonis fails to steal the diamond from the user (i.e., action #2 fails) because the user successfully confronts him (i.e., action #4 succeeds) and/or also because the system has decided that the exchange action of the user will succeed (i.e., action #1 succeeds).

At this point, the only unresolved actions describe the conflict between Elanthos and Apateonis (Table 2, rows 9-11). Step 3 notices that these actions are driven by a norm-related motive in S+. Because S+ prevails, the algorithm decides that the hiding action of Apateonis which opposes the norm for protecting ownership in *S*+ will succeed to increase user suspense. This causes the punishment action pursued by Elanthos to fail according to the resolution order rules. With this decision the resolution method determines the outcome of all the actions in this scene and step 4 of the algorithm is activated to present this plot resolution.

Step 4 schedules the presentation of the outcomes to the user. In particular, rule (1) produces the following results:

Resolves(Hide(U), Punish(U)) → Hide(U) << Punish(U)

/ Present the outcome of the hiding of user from Elanthos before the outcome of the punishment attempt of the user by Elanthos, because the outcome of hiding resolves this punishment attempt. */*

and similarly:

Confront(A) << Steal(Diamond)

/ Present the outcome of confrontation between the user and Apateonis before the outcome of the stealing action of Apateonis. */*

Exchange(Diamond, A) << Steal(Diamond)

/ Present the outcome of the stealing attempt of the diamond after the outcome of the exchange of this diamond. */*

Hide(A) << Punish(A)

/ Present the outcome of the hiding of Apateonis from Elanthos before the outcome of the punishment attempt of Apateonis by Elanthos. */*

Table 2: Unresolved character actions and conflicts in chronological succession.

t	Actions & Conflicts
1	Exec+(U, Exchange(Diamond, A), G) Exec-(U, Exchange(Diamond, A), RK) /* The user tries to exchange the diamond for a ship. */
2	Exec-(A, Steal(Diamond), G) Exec-(A, Steal(Diamond), PO) /* Apateonis seeks to steal the diamond from the user. */
3	Conflict(Exchange(Diamond, A), Steal(Diamond)) /* The exchange and stealing actions conflict. */
4	Exec+(U, Confront(A), G) Exec+(U, Confront(A), PO) /* The user seeks to confront Apateonis. */
5	Conflict(Steal(Diamond), Confront(A)) /* The stealing action and the confrontation conflict. */
6	Exec+(E, Punish(U), RK) /* Elanthos seeks to punish the user. */
7	Exec-(U, Hide(U), RK) /* The user attempts to hide from Elanthos. */
8	Conflict(Punish(U), Hide(U)) /* The punishment and hiding actions conflict. */
9	Exec+(E, Punish(A), PO) /* Elanthos tries to punish Apateonis. */
10	Exec-(A, Hide(A), PO) /* Apateonis tries to hide from Elanthos. */
11	Conflict(Punish(A), Hide(A)) /* The punishment and hiding actions conflict. */

Rule (2) gives:

Resolves(System, Confront(A)) ∧ Resolves(System, Hide(U)) ∧
∧ Follows(Confront(A), Hide(U)) → Hide(U) << Confront(A)

/ Present the outcome of the hiding of the user from Elanthos before the outcome of the confrontation attempt of the user. This is the case because both actions are resolved by the system and the user seeks to hide after s/he has attempted to confront Apateonis. */*

Similarly:

Confront(A) << Exchange(Diamond, A)

/ Present the outcome of the confrontation between the user and Apateonis before the outcome of the diamond exchange. */*

Hide(A) << Hide(U)

/ Present the outcome of the hiding of the Apateonis from Elanthos before the outcome of the hiding of the user from the king. */*

Rule (3) suggests the following sequence:

Hide(A) << Hide(U) ∧ Hide(A) << Punish(A) ∧
∧ Follows(Hide(U), Punish(A)) → Punish(A) << Hide(U)

/ Present the resolution of the punishment of Apateonis by Elanthos before the resolution of the hiding attempt of the user from Elanthos. This is the case, because both actions are scheduled after the resolution of the hiding of Apateonis from Elanthos and the hiding of the user precedes the punishment attempt of Apateonis by Elanthos. */*

and similarly:

Punish(U) << Confront(A)

/* Present the outcome of the punishment attempt of the user by Elanthos before his confrontation with Apateonis. */

Table 3 describes the presentation sequence for all the events in the story that is determined by Step 4 of plot resolution. This sequence can be summarized by hand as follows:

«Apateonis succeeds in hiding from Elanthos, therefore the king is unable to punish him. The user successfully hides from the king, therefore Elanthos cannot punish him as well. The user successfully confronts Apateonis. As a result the user succeeds in exchanging the diamond and Apateonis fails in his stealing actions

PEGASUS runs on a Windows PC. Plot resolution is built on top of an ATMS-based rule engine written in C++, similar to the one described in [6].

5 Related Work

The resolution method accepts as input an action sequence produced by a plot control method similar to the one described in [11].

There has been significant research in AI for the creation of interfaces that feature believable interactive characters (see [7] for an overview) or for the creation of interactive story systems [12]. This work has concentrated mainly on portraying the emotional state of these characters, on supporting full-body interactive video environments, or on developing directed improvisation paradigms in which computer characters improvise a joint course of behavior following users' directions [7,9,10]. In addition, AI researchers in this field have conducted live experiments designed to understand how to create interactive drama [9]. In these experiments human actors simulated a computer system for interactive drama. Our work complements all this research by addressing computational issues not raised by any of the previous approaches, such as dynamic resolution techniques for interactive drama.

Storyweaving has been investigated in TALE-SPIN [8]. This system provides a simple model for story generation that focuses on describing the stereotypical problem-solving behavior of story characters in pursuit of various goals. TALE-SPIN supports deus ex machina types of endings because the user does not participate in the story as one of the characters. Instead s/he has a directing role resolving all plot developments. In contrast, the computer decides dynamically on the resolution of all plot developments in our approach, based on the higher-level motives of the cast and the dramatic power of the story ending. Consequently, this feature increases the believability of the resulting plots.

6 Conclusions

We have described a dynamic, user-centered resolution method for interactive stories. This method can be used for the development of intelligent plot assistants for screenwriters or directors of interactive entertainment systems. In addition it can be incorporated in dynamic plot generation systems for interactive stories, games and/or collaborative virtual worlds.

Future work in this area includes the development of dynamic multimedia presentation techniques that will be integrated with this resolution method, along with resolution algorithms for non-linear plot generation systems that cycle, contrast or repeat multiple story lines [3].

Table 3: Final resolution sequence.

t	Outcome of Actions
1	Succeeds(Exec-(A, Hide(A), PO))
2	Fails(Exec+(E, Punish(A), PO))
3	Succeeds(Exec-(U, Hide(U), RK))
4	Fails(Exec+(E, Punish(U), RK))
5	Succeeds(Exec+(U, Confront(A)), PO))
6	Succeeds(Exec+(U, Exchange(Diamond, A), G))
7	Fails(Exec-(A, Steal(Diamond), PO))

References

1. Aristotle, *Poetics,* trans. I. Bywater, in *Introduction to Aristotle,* ed. Richard McKeon, 2nd ed., Chicago, Univ. of Chicago Press, 1973.
2. Sgouros, N. M, Papakonstantinou, G., Tsanakas, P., Dynamic Dramatization of Multimedia Story Presentations, Accepted in *Intelligent User Interfaces (1UI-97),* (Orlando FL, January 1997).
3. Pfister, M., *The Theory and Analysis of Drama,* Cambridge University Press, 1988.
4. Lucey, P., *Story Sense: Writing Story and Script for Feature Films and Television,* McGraw-Hill, 1996.
5. Blacker, I. R., *The Elements of Screenwriting,* Macmillan, 1986.
6. Forbus, K. D., de Kleer, J., *Building Problem Solvers,* MIT Press, 1993.
7. Maes, P., *Artificial Life Meets Entertainment: Lifelike Autonomous Agents,* Communications of the ACM, vol. 38, no. 11, November 1995.
8. Meehan, J., TALE-SPIN, in *Inside Computer Understanding,* Schank, R. C. and Riesbeck, C. K. (eds), Lawrence Erlbaum Associates, 1981.
9. Kelso, M. T., Weyhrauch, P., Bates, J., *Dramatic Presence.* Technical Report CMU-CS-92-195, School of Computer Science, Carnegie Mellon University, December, 1992.
10. Hayes-Roth, B., Brownston, L., Sincoff, E., Directed Improvisation by Computer Characters, Technical Report, KSL-95-04, Department of Computer Science, Stanford University, 1995.
11. Sgouros N. M., Tsanakas P., Papakonstantinou G., A Framework for Plot Control in Interactive Story Systems, *Proceedings of AAAI-96,* Portland, OR, USA.
12. Waters, R. C, The 1995 AAAI Spring Symposia Reports; Interactive Story Systems: Plot and Character, *AI Magazine,* vol. 16, no. 3, Fall 1995.