Logic Control for Story Graphs in 3D Game Narratives

Hui-Yin Wu, Tsai-Yen Li, and Marc Christie

IRISA/INRIA Rennes, National Chengchi University
hui-yin.wu@inria.fr, li@nccu.edu.tw, marc.christie@irisa.fr

Abstract. With the rising popularity of engaging storytelling experiences in gaming arises the challenge of designing logic control mechanisms that can adapt to increasingly interactive, immersive, and dynamic 3D gaming environments. Currently, branching story structures are a popular choice for game narratives, but can be rigid, and authoring mistakes may result in dead ends at runtime. This calls for automated tools and algorithms for logic control over flexible story graph structures that can check and maintain authoring logic at a reduced cost while managing user interactions at runtime.
In this work we introduce a graph traversal method for logic control over branching story structures which allow embedded plot lines. The mechanisms are designed to assist the author in specifying global authorial goals, evaluating the sequence of events, and automatically managing story logic during runtime. Furthermore, we show how our method can be easily linked to 3D interactive game environments through a simple example involving a detective story with a flashback.

Keywords: Interactive storytelling, Game narrative, Logic control, Story graph filtering

1 Introduction

As readers of stories, our brains are engaged in a mix of perceptual, cognitive, and logical activities in an attempt to comprehend the unfolding of events and immerse us in the story world. According to Branigan, comprehension of the plot can be interpreted in terms of two processes: the bottom-up perception of individual actions and events as they occur, and the top-down structural understanding of story goals, temporal order, and logical inferences [2]. In interactive digital storytelling, where the viewer can participate and alter the outcome of the story, the replay value of the story increases and the viewers are more immersed through participation in the story outcome.

Yet with the growing complexity of the story and managing outcomes of interaction, the task of authoring with regards to story logic and content becomes a big challenge for the story designer. Computational algorithms and the mature understanding of narrative structures provides story designers with better tools to create more personalized, engaging, and well-controlled narrative content. One structure is the branching structure, popularly adopted by existng games as game trees or story graph structures in existing research [15][9][12]. Yet maintaining logic over complex branching (and maybe even non-linear) stories at runtime is also a difficult task that requires laborious authoring to ensure that no illogicalities exist in the story graph. Also, user interactions may be greatly restricted or have low impact over the story due to the rigid structure of such graphs. Therefore, it is much desired to have a set of simple set of logic controls that integrates the story logic, user decisions, temporal arrangements, and authorial goals in one mechanism.

In this work, we propose a logic control algorithm specifically suited to maintain the logic control of the story graph structure. Our algorithm is suited to story graph structures that are interactive, require stronger control of logic, and possibly involve non-chronological story segments. The method takes as input a pre-authored (or generated) story graph and a set of user-defined authorial goals (e.g. ”a murder and a happy ending”). Our method then outputs a subgraph of the original story graph, and at each interactive plotpoint, subsequently prunes the story graph to maintain logicality at runtime. Moreover, when the temporality of the story is non-linear, involving embedded scenarios, the algorithm ensures user interactions within and outside of the embedded scenario extend to other parts of the story, ensuring that no matter in what sequence the story events play out, the logic remains consistent. The algorithm is linked to an authoring interface and scene in Unity to demonstrate the effect of logic control on the discourse in a real-time 3D environment.

The main contributions of this work is the design of a logic control mechanism for narratives with embedded plot lines that (1) enforces local story logic (such as preconditions, and post-effects of story events) and authorial goals to be upheld without running into unresolvable plot points, (2) manages story logic over user interactions of non-linear stories at runtime, and (3) performs dynamically in 3D environments.

In the following sections, we first outline the related work. We then briefly describe the specifications of the story graph representation before focusing on the logic control algorithm over the story graph. In Section 6 we present the output of our method in a real-time 3D environment with authorial goals, temporal variations, and user interactions. We finally conclude with possible applications of our method.

2 Related Work

For more flexible narrative generative systems, Brooks [3] proposes a framework with structural, representational, and presentational environments. He also proposes that a computational narrative is comprised of a narrative structure, pieces of the story with representation information, and a reasoning strategy among the story pieces. Similar to Brooks’ framework, we realize our story structure and logic control with (i) the story graph structure with story units, (ii) the logic control algorithm, and (iii) the presentation in 3D environment. This section outlines the previous work on story structures and logic control mechanisms for 3D game and storytelling environments we observe in the design of our methods.

2.1 Story Structure and Units

To find a suitable story structure that provides enough flexibility for high-level plot arrangements, we survey previous approaches to branching and planning formalisms of interactive narrative.

Carmichael and Mould [4]adapts the story model of Chatman [5], using the concept of kernels and satellites to represent core events (kernels) and anticipated storylines (satellites) in open-world exploration games. This structure fits its original purpose to loosely-structured open-world exploration narratives by ensuring key events are played in order, but is too simplistic and general to construct multiple plotlines through variations of key events as story graphs could.

Our work is similar to the concept of plot-point graphs [15], and further explored by [9] where a number of important moments (i.e. plot points) are authored, and evaluation function is adopted to verify the sequencing of the plot points. However, this search-based approach is targeted towards story sequences that are linearly authored, requiring a full search algorithm over all permutations of plot points to construct an interactive story graph. Our work is targeted towards story graphs that were authored to be interactive.

Previous work on narrative formalisms come in the form of either planning or branching, leaning towards planning methods in generative systems. Though they are more restrictive in generative power than planning, branching structures are shown to be efficient when managing user decisions [13]. While planning algorithms provide the capacity for dynamicity in storytelling, their benefits are often insufficiently explored due to the limitations on the scalability of existing stories.

2.2 Logic Control in Interactive Storytelling

From the story point of view, logic control concerns the causal relations between events, as a key is to the lock it opens and a pen to the words it writes. When dealing with logic in stories, we are evaluating the causality of the author's creation in order to maintain believability in the audience. [12] Constructs graph-like structures for multiplayer storytelling, and introduces a logical control mechanism that can adapt to user interactions dynamically. [6] introduces interactive behaviour trees as a more author-friendly and flexible option. However, these approaches could still result in dead ends when users take certain choices unexpectedly. Also, they were not designed to accommodate embedded and temporally rearranged narratives, which is one of our strengths.

For some stories, telling the story chronologically may seem like a natural decision, whereas for others, such as detective stories, readers may enjoy deciphering what happened. Currently, there are a number of papers situated to discuss temporality in game narratives. We observe in text-based generative narratives, Montfort [8] focuses on syntactical issues, such as grammatical tense, using a tree representation of time. Lönneker [7] introduces an architecture for tense in "levels" of narrative, which is a basis for designing embedded story structures. Concerning logicality of temporal rearrangements, [10] and [14] tackle problems of timing in character or agent-based planning. Similarly, Bae and Young[1] use planning methods to create temporal rearrangements, mainly to invoke surprise arousal in viewers by hiding certain information and revealing it at emotionally intense moments.

However, the aim of these methods is to generate an arrangement of events that are consistent in their presentation (discourse) into a linear story, and not to ensure logic consistency of story content when user choices can have a strong impact on the plot. In

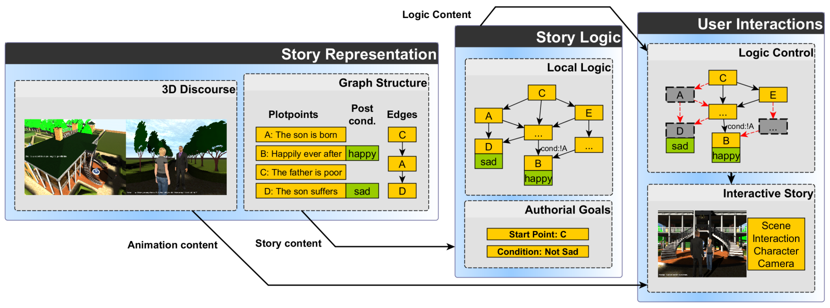

Fig. 1. In the system, the author can design story and animation content, and set authorial goals. The logic control uses a double traversal algorithm that checks and filters story nodes and edges so that no dead ends would result from user decisions.

contrast, our method maintains logic through an algorithm that enforces pre- and post-conditions while maintaining interactivity by producing a subgraph of all feasible paths (and not just one arrangement).

3 Overview

For the implementation of story logic control, we design a workflow for game narratives comprised of three components: (1) authoring, (2) logic control over temporality and user interactions, and (3) 3D presentation. The relation between the components of the framework and the presentation platform are shown in Figure 1.

The authoring component involves authoring of the basic event of the story, and linking them up into a story graph. The logic control then takes over to ensure pre- and post- conditions on each event is upheld, and removes any illogicalities in the story graph. Finally, the method is linked to a 3D presentation and allows users to experience and interact with the story in real time. The next section begins to outline our method by explaining the story representation: the basic story units, the story graph, and how the story graph represents complex structures such as embedded plotlines.

4 Story Representation

Here we introduce the authoring of the graph-based story representation for the purpose of demonstrating the logic control. Note that, in this section though we coin the term "authoring" to describe the process of building the story graph, the story graph does not necessarily have to be manually authored. We envision the capability of planning or search algorithms that could generate such a story graph. The story graph is therefore intended to be seen as a formalisation or a generalisation of graph-based representations of branching narratives that are widely used in current games.

4.1 The Building Blocks of Story

Interactive narratives for games are often comprised of basic units of action, dialogue, and visuals or audio effects. In this paper, we refer to these units as plotpoints.

To help the logic control identify the plotpoints, postconditions can be attached to the plotpoint, and serve as markers that the logic control can evaluate. Postconditions have either boolean values (e.g. the plotpoint concerns "Sara", "murder", and "mansion") or integer values (e.g. the plotpoint has the effects of "happy+=1"). Using the above postconditions as an example, if the story goes through a plotpoint that has the postconditions "murder;happy+=1" then the global parameter "murder" is assigned the value of "true" while the value of the global parameter "happy" is incremented by 1.

The plotpoints do not need to be authored in any specific order, and any postconditions can be attached to the plotpoints, which act like postconditions that can be evaluated later on in the story. We do not restrict the size, content, or scope of a plotpoint. A story, or even multiple varying stories can be composed out of the plotpoints simply by linking them in a specified order. The linking and maintaining of logic between plotpoints is described below.

4.2 Establishing Local Story Logic

On their own, each story plotpoint simply represents a unit within the story. When a number of plotpoints related to a same scene are grouped together, we call the grouped plotpoints a "move" adopting the terminology from [11] to describe a complete sequence.

But a number of questions remain: How is each plotpoint within a move related to other plotpoints? Arrangements of the plotpoints within the move are achieved by linking plotpoints to each other with directed edges that specify a total order between plotpoints. Plotpoints can exist in moves without any edges, but then signifies that the plotpoint has no logical relation with any other plotpoint in the move. Every move has an empty entering and exiting plotpoint (Figure 2). The first plotpoint in the move is the entering plotpoint, which directs to all the plotpoints that can serve as the first action in the move. All plotpoints that are the last plotpoint in the move point to an exiting plotpoint, which is either a concluding point in the story (thus a "The End" of the story), or the entrance point of another move.

The story representation can also allow embedding, which refers to the process of jumping back and forth to an external move B from a plot point in move A: Optionally, a plotpoint within move A can also embed to another move C. Upon finishing the plotpoint, the story plays the embedded move B (suspending the current point in move A) and when move B is finished, the story returns to the embedding plotpoint in move A.

Apart from the order of the events, edges are also a way to control logic. preconditions can be placed on the edges, such as the boolean precondition "$murder$ = `false`" (meaning that a plotpoint with the postcondition tag "murder" must not precede this plotpoint) or "$happy \geq 2$" (meaning that the integer postcondition "happy" must have a value of 2 or more at this point). These preconditions can be grouped with boolean operators (**AND**, **OR**, and **NOT**) to form evolved preconditions to control the story. As edges can be placed between any two plotpoints that can logically follow each other, we can envision many circumstances when one node can have two or more exiting edges pointing to different story nodes respectively. Note that preconditions are placed on edges linking plotpoints, and not on plotpoints themselves, keeping the plotpoint as minimal as possible so that it may be reused in multiple stories flexibly. This is an important property in the story graph, where the plot line actually branches: the separation of the plot leading from a plotpoint is where user intervention can change the outcome of the story (by leading to different end nodes within the story graph).

5 Logic Control

Once the story graph has been either manually designed or generated by algorithms, we then demonstrate our method to perform logic control on this representation. Story logic comes into light as the author wants to achieve storytelling goals while managing user interaction in possibly non-chronological story lines. Like matching keys to their locks, the process of establishing story logic requires a series of pre- and postconditions; the more elaborate the plot design, the more complicated this web of preconditions, and the higher likelihood of dead ends, unresolvable user interactions, or unachievable story goals.

In this section, we introduce our logic control method over the generic story graph. The method is designed to assist the author in specifying global authorial goals, evaluate the sequence of events, and automatically manage story logic during runtime.

5.1 Authorial Goals

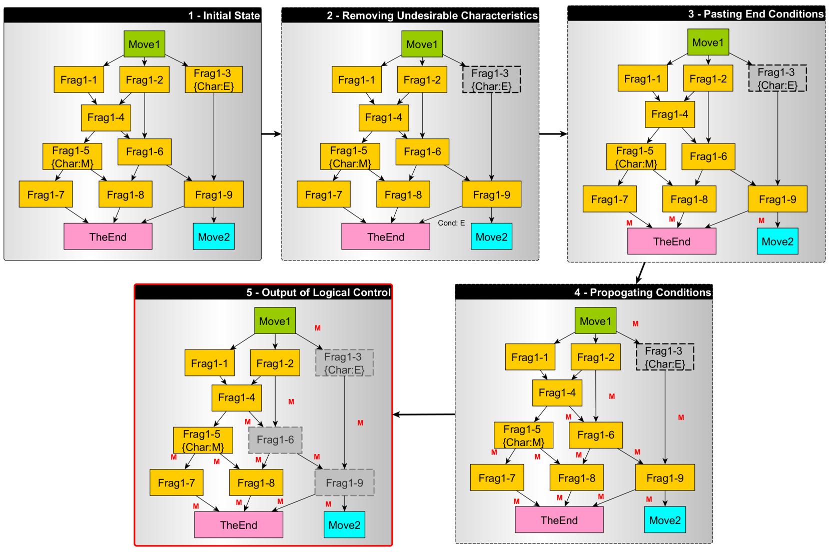

Fig. 2. Given an authored move and authorial goal ¬ E ∧ M ("(**NOT** Evil) **AND** Magic')', the algorithm (1) removes plotpoints contradicting the goal (E), (2) pastes the goal (M) as an end precondition, (3) reverses and (4) propagates the preconditions on all edges, and (5) removes dead ends. As a result, remaining paths ensure the story logic as well as authorial goals.

Authorial goals are defined as the storytelling goals that the author wants to achieve. The design of authorial goals corresponds with Chatman's theory of modes of plot, where certain combinations of postconditions will result in specific emotions in the audience. For example, Chatman defines that a good hero that fails will arouse sympathy, while an evil hero that succeeds invokes a feeling of disgust. Such statements can be easily expressed as preconditions on the story graph as an evaluation of the postconditions of the plotpoints that occur in the story. Taking tan example, "**AND**($HeroFail$ = `true`; $HeroKindness \geq 2$)" and "**AND**($HeroFail$ = `false`; $HeroKindness \leq -2$)" respectively, where in the story graph, there are ending nodes with boolean postconditions of $HeroFail$ and nodes that accumulate or decrement the value of the integer parameter $HeroKindness$. Authorial goals are to be placed on the whole story graph and seen as preconditions that must be fulfilled before the end of the story. If we have an authorial goal of "$Magic$ = `true`" it would mean that somewhere in the story, there must exist a plotpoint tagged with the postcondition $Magic$.

We aim to ensure (1) that the local logic in the previous section did not resolve in any dead ends or unresolvable plot lines, and (2) that the authorial goals are ensured to be achieved (if a solution exists). We design the logic control algorithm for the purpose of upholding both the local logic and authorial goals (see Algorithm 1). The algorithm uses a double depth-first traversal to prune the story graph and output a subset of validated paths with reinforced preconditions on the edges to ensure that authorial goals are met. There are four stages in the algorithm to carry out authorial control: removal of contradicting plotpoints, paste end preconditions, reversal and propagation of preconditions, and the final validation and dead end removal.

Algorithm 1 LogicControl (node N, goals G)

```
1:  if $G$ violates descriptors of node $N$ then
2:      remove node $N$, and node's incoming and outgoing edges
3:  end if
4:  for all sons $s$ of node $N$ do
5:      if $s$ has not been visited then
6:          tag $s$ as visited
7:          LogicControl($s$, $G$)
8:      end if
9:  end for
10: if all outgoing edges of $N$ have preconditions then
11:     for all outgoing edges $e$ in node $N$ do
12:         $cond \leftarrow cond \vee$ preconditions in edge $e$
13:     end for
14:     for all descriptors $d$ of node $N$ do
15:         $cond$ = negateDescription ($d$, $cond$)
16:     end for
17:     for all incoming edges $e$ of node $N$ do
18:         add preconditions $cond$ to edge $e$
19:     end for
20: end if
```

Removal of contradictory plotpoints From the story representation, plotpoints can contain a number of boolean or integer postconditions. The removal of contradictory plotpoints excludes plotpoints containing undesirable boolean postconditions. The algorithm takes as input the boolean authorial goals, performs a DFS traversal on the story graph to remove any plotpoints with postconditions that contradict with the goal, where an authorial goal requires a boolean postcondition to be false. For example, for the authorial goal "**AND**($HeroFail$ = `false`; $HeroKindness \leq -2$)" all plotpoints with the postcondition $HeroFail$ are automatically removed, since by definition, they automatically contradict the goal.

This ensures that no plotline will go through undesirable plotpoints. Boolean goals that are evaluated as true (i.e. "$SomeChar$ = `true`") as well as integer goals are not evaluated for this step.

Pasting end preconditions The algorithm then pastes the authorial goals as end preconditions on all the incoming edges of the end nodes. The reason for doing this is to ensure that, before the story concludes, these goals will be upheld. However, this step would result in possible unresolvable plot lines if not all of the goals are fulfilled before the story comes up to this point. The next step, the reversal and propagation addresses this problem.

Reversal and propagation of preconditions As mentioned previously, pasting goals alone cannot ensure that a story will conclude. It is possible to choose a path in the story graph that reach an end node, but cannot find any possible path that achieves the goal, thus causing the story to fail at runtime. To prevent this, our algorithm does a second traversal through the story graph bottom-up. For every plotpoint, it concatenates the preconditions from the outgoing edges, and pastes them to the incoming edges. We refer to this step as the propagation. However, this task is not just a copy-paste of preconditions from one edge to another.

Before preconditions are propagated, they are first reversed; the plotpoint checks its own postconditions against the preconditions. If a boolean precondition is fulfilled (for example, a plotpoint with the postcondition $HeroFail$ is propagating the precondition "$HeroFail$ = `true`") it will see the precondition as fulfilled, and will remove it. Integer preconditions such as "$HeroKindness \leq -2$" are reversed by reversing the calculation done by the plotpoint (for example, the propagated precondition "$HeroKindness \leq -2$" will be translated as a "$- = 1$" precondition when encountering the integer postcondition "$HeroKindness \leq -1$"). This allows the integer value to increase and decrease freely throughout the story, thus creating highs and lows in the plotline. Finally, repeated boolean preconditions and redundant integer preconditions can be eliminated. Though not required, this step ensures that the preconditions are concise and do not expand much throughout the propagation.

Validation and removal of dead ends Since preconditions are propagated up to the top of the story graph, it is fast to identify which edges have preconditions that cannot be fulfilled. The remaining step traverses the graph once more removing any dead ends (i.e. a plotpoint with no feasible outgoing edges), then outputting the sub-graph with all the feasible plot lines.

Figure 2 illustrates this process.

5.2 Embedding

Our method can also control logic in embedded moves that represent temporal rearrangements, such as flashbacks. As described previously, an embedded move is an internal representation within the plotpoint allowing it to embed another existing move like a sub-plot.

Given a start point in the story, the story progresses sequentially and chronologically down a feasible story path. When deciding whether an embedding should occur, the algorithm checks whether the embedded move has been played before. If it hasn't, embedding occurs; otherwise, the story just continues to the next plotpoints. The algorithm automatically determines what sequence to show events, whether an embedding should take place, records the progression of the story, and returns the story to the original point after the embedding occurs.

However, when embedding user interaction happen simultaneously, we need to ensure that user decisions extend to the embedded move at runtime. And vice versa, we need to ensure that decisions made within the embedded move extend to the the rest of the story. For example, if at the story entry point, an event such as the murder of Actor C is assumed to have occurred, then in a flashback of the crime taking place, only story lines that lead up to the murder of Actor C should be feasible.

5.3 Managing of User Interactions for Embedded Storylines

The algorithm we have described above not only enforces the achievement of authorial goals over a story graph, but also reinforces local logic on the edges by propagating them until they are fulfilled. By definition of this logic control algorithm, it is guaranteed that all paths that the user may choose in the graph must have at least one feasible path (chronologically) that (1) can terminate the story, (2) achieves all the authorial goals, and (3) does not contain any illogicalities.

We designed a second algorithm for user interaction, Algorithm 2, to solve the problem of logic control for user decisions with embedding. It extends the previous algorithm by propagating, for each decision the user takes, the preconditions on the outgoing edges of the plotpoint throughout the story graph, as if they were new end preconditions. In this way, the plotpoint is ensured to be reachable and to be reached for all the remaining paths. The graph is then re-shaped for each decision. This algorithm also maintains a set of flags to record the current level of embedding, and the embedding point, ensuring that the story returns to the correct plotpoint after finishing embedding.

Our mechanism for logic control in the stories is consistent with Branigan's mode of comprehension of narrative [2]. Each story plotpoint is understood as a bottom-up independent unit of action with certain postconditions in relation to its content, and the story graph is a top-down logic control unit, with a filtering algorithm that (i) preserves preconditions between narrative plotpoints, and (ii) assists the author in meeting authorial goals in the form of end preconditions.

Algorithm 2 UserInteraction (Decision D)

```
1: if current node $N$ has $embedding$ then
2:     $embedLevel$ += 1
3:     push node $N$ into levels stack $L$
4:     next node $N'$ = embedding entrance point $N_e$
5:     $embedFlag$ = $true$
6: end if
7: if $embedFlag$ is $false$ and $embedLevel > 0$ then
8:     current node $N$ = pop $L$
9: end if
10: get next adjacent node $N'$ from current node $N$ and decision $D$
11: for all incoming edges $e$ of node $N'$ do
12:     $cond \leftarrow cond \vee$ preconditions in edge $e$
13: end for
14: LogicControl(story entrance point $s$,$cond$)
15: return next node $N'$
```

6 Results

6.1 Demonstration

We integrated our logic control method and authoring interface into an interactive storytelling environment built on Unity called the "Theater". The Theater takes as input a pre-authored story graph as well as predesigned animation content for each plotpoint. It actively communicates with the logic control, presents the animation content for the plotpoints in the order designated by the logic control, requests suitable user choices and presenting them to users, and returns choices made by users.

We demonstrate the logic control on an example scenario involving a dissatisfied woman, her millionaire husband, her secret lover, and a local investigator. The woman is unhappy with her current life and wants to run away with her lover. She makes decisions on whether to confront her husband, which results in some conflicts, and the husband is killed. An investigator arrives and questions the woman. Based on the previous decision, the woman will have options to lie, to confess, to escape, or to protect her lover, each leading to different consequences.

The story has two possible initiating points: one from the chronological start, when the women decides to take action; and one is when the millionaire is already dead and the investigator arrives. The second initiating point will invoke a flashback on the woman when she recalls what had happened prior to the investigator's arrival. In both cases, the user is offered the same amount of decision, and the logic control ensures that all stories generated are plausible. The two plausible plot lines are shown in Figure 3.

The accompanying video shows the flashback version of this story, demonstrating how the logic control achieves story goals while ensuring all preconditions are met over user decisions and the non-linear storyline at runtime.

Link to video: https://vimeo.com/129289640

An addressing of all three aspects–temporality, interactivity, and authorial goals–for complex storyliens has not been displayed before in existing contributions.

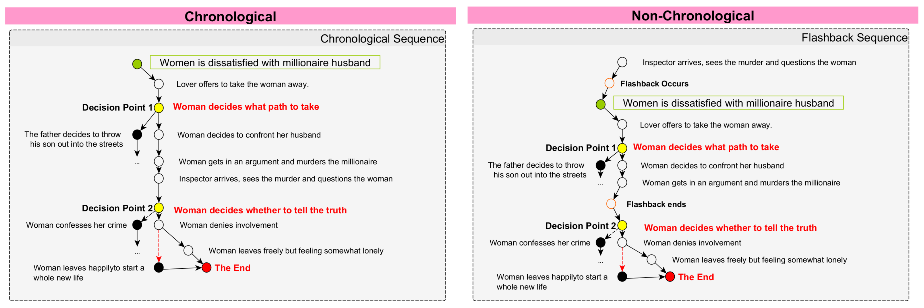
Chronological
Chronological Sequence
Women is dissatisfied with millionaire husband
Lover offers to take the woman away.
Decision Point 1
Woman decides what path to take
The father decides to throw his son out into the streets
Woman decides to confront her husband
...
Woman gets in an argument and murders the millionaire
Inspector arrives, sees the murder and questions the woman
Decision Point 2
Woman decides whether to tell the truth
Woman confesses her crime
Woman denies involvement
...
Woman leaves freely but feeling somewhat lonely
Woman leaves happilyto start a whole new life
The End
Non-Chronological
Flashback Sequence
Inspector arrives, sees the murder and questions the woman
Flashback Occurs
Women is dissatisfied with millionaire husband
Lover offers to take the woman away.
Decision Point 1
Woman decides what path to take
The father decides to throw his son out into the streets
Woman decides to confront her husband
...
Woman gets in an argument and murders the millionaire
Flashback ends
Decision Point 2
Woman decides whether to tell the truth
Woman confesses her crime
Woman denies involvement
...
Woman leaves freely but feeling somewhat lonely
Woman leaves happilyto start a whole new life
The End

Fig. 3. The example story with two plot lines that involve non-chronological events.

6.2 Evaluation

Though an authoring interface is currently under development to evaluate the authoring potential of the story graph, we conducted a qualitative pilot study on 5 users to gain feedback on the logic control of the algorithm.

All five users were university students from different backgrounds. Each were asked to experience the story in four modes: (1) text, (2) non-interactive animated, (3) interactive animated, and (4) interactive animated with story goal (they were given the choice to select a story goal of "happy" or "sad").

In the post survey, users mentioned that while Mode 3 was more enjoyable, but Mode 4 had a higher capacity for creativity as an author. The selection of story goals offered them control over how they, as an author, would like the story to unfold. When asked what they would like to use the system for, all five users noted the control they had on the story in Mode 4, and its potential for creativity as an author. One user particularly noted from the perspective of narrative creation, he would be interested in using the system of Mode 4 by adding his own fragment to create a new storyline with a reinforced goal. Another user stated that Mode 4 would be helpful in creative writing with its branching narrative as compared to traditional linear narratives.

7 Discussion and Conclusion

One main drawback of our approach is, being a graph-based traversal, it cannot rank or evaluate the quality of a story line as planning algorithms do with well-designed heuristics. Every feasible path in the story graph is considered equally probable. Therefore, the algorithm cannot solve a problem such as searching for a best match. This maintains the simplicity of the algorithm, but limits its tolerance to what it would consider a good plot line. In this paper, we have proposed a method for logic control of interactive narratives with regards to authorial goals, user interaction, and temporal rearrangements. The algorithm we designed performs a traversal on the story graph, thereby re-shaping interactive story graphs to fulfil authorial goals and maintain story logic. We demonstrated the output of the logic control in a 3D virtual environment.

As an extension of evaluating story postconditions and understanding of story structure and content, we believe our approach can be further developed to enhance real-time context-aware storytelling techniques. The logic control could provide story level information such as emotion, timing, perspective, and genre to the discourse level such that the virtual camera can make decisions on viewpoint and compute suitable sequences of shots. The mechanism for story filtering and authorial goal design also provides an exciting move towards even more tailored and personalised storytelling experiences for interactive digital storytelling.

References

1. Bae, B.c., Young, R.M.: A Use of Flashback and Foreshadowing for Surprise Arousal in Narrative Using a Plan-Based Approach. In: First Joint International Conference on Interactive Digital Storytelling, ICIDS 2008. pp. 156–167. Springer Berlin Heidelberg, Erfurt, Germany (2008)
2. Branigan, E.: Narrative Comprehension and Film. Routledge (1992)
3. Brooks, K.M.: Do Story Agents Use Rocking Chairs? The Theory and Implementation of One Model for Computational Narrative. In: MULTIMEDIA '96: Proceedings of the fourth ACM international conference on Multimedia. pp. 317–328. ACM Press (1996)
4. Carmichael, G., Mould, D.: A Framework for Coherent Emergent Stories. In: Proceedings of Foundations of Digital Games 2014. Florida, USA (2014)
5. Chatman, S.: Story and Discourse: Narrative Structure in Fiction and Film. Conell University Press (1980)
6. Kapadia, M., Falk, J., Fabio, Z., Marti, M., Sumner, R.W.: Computer-Assisted Authoring of Interactive Narratives. In: ACM SIGGRAPH Symposium on Interactive 3D Graphics and Games (i3D). vol. 2 (2015)
7. Lönneker, B.: Narratological Knowledge for Natural Language Generation. In: Wilcock, G., Jokinen, K., Mellish, C., Reiter, E. (eds.) Proceedings of the 10th European Workshop on Natural Language Generation (ENLG 2005). pp. 91–100. Aberdeen, Scotland (Aug 2005)
8. Montfort, N.: Ordering events in interactive fiction narratives. In: Proceedings of the AAAI Fall Symposium on Intelligent Narrative Technologies. pp. 87–94 (2007)
9. Nelson, M.J., Mateas, M.: Search-Based Drama Management in the Interactive Fiction Anchorhead. Proceedings of the First Annual Conference on Artificial Intelligence and Interactive Digital Entertainment pp. 99–104 (2005)
10. Porteous, J., Teutenberg, J., Charles, F., Cavazza, M.: Controlling Narrative Time in Interactive Storytelling. In: The 10th International Conference on Autonomous Agents and Multiagent Systems - Volume 2. pp. 449–456. AAMAS '11, International Foundation for Autonomous Agents and Multiagent Systems, Richland, SC (2011)
11. Propp, V.: Morphology of the Folktale. University of Texas Press, 2 edn. (1968)
12. Riedl, M., Li, B., Ai, H., Ram, A.: Robust and Authorable Multiplayer Storytelling Experiences. Proceedings of the Seventh International Conference on Artificial Intelligence and Interactive Digital Entertainment pp. 189–194 (2011)
13. Ryan, M.: Avatars of story. University of Minnesota Press (2006)
14. Shoulson, A., Gilbert, M.L., Kapadia, M., Badler, N.I.: An Event-Centric Planning Approach for Dynamic Real-Time Narrative. In: Proceedings of the Motion on Games - MIG '13. pp. 99–108. ACM Press, Dubin (2013)
15. Weyhrauch, P.W.: Guiding Interactive Drama. Ph.D. thesis, Pittsburgh, PA, USA (1997)