A Two-Level Planning Framework for Mixed Reality Interactive Narratives with User Engagement

Manuel Braunschweiler
ETH Zurich
Disney Research

Steven Poulakos
Disney Research

Mubbasir Kapadia
Rutgers University

Robert W. Sumner
ETH Zurich
Disney Research

***Abstract*—We present an event-based interactive storytelling system for virtual 3D environments that aims to offer free-form user experiences while constraining the narrative to follow author intent. The characters of our stories are represented as smart objects, each having their own state and set of capabilities that they expose to the virtual world. Our narratives are represented as a collection of branching stories, where narrative flow is controlled by author-defined states. A user model is employed to evaluate the user's engagement with smart objects and events, based on proximity, interaction patterns and visibility to the user. A two-level online planning system is designed to find the best narrative trajectory along pre-authored stories, according to the user model, and to generate a story sequence to the best trajectory with Monte Carlo Tree Search. We present the capabilities of our interactive storytelling system on an example story and describe the adaptations required for modeling user engagement in AR and VR applications.**

***Index Terms*—interactive narratives, narrative generation, planning, artificial intelligence, Monte Carlo Tree Search, user perception**

I. Introduction

The goal of any interactive story is to give the user a high degree of agency during their experience. Unfortunately, user agency often conflicts with the plot intended by the story's author. Balancing user agency with engaging, author-defined narrative experiences is a challenge in Interactive Storytelling (IS). This balance is usually mediated using an experience management system, which monitors story progression and intervenes according to a model of narrative quality [1]. To enable reasoning about the progression and generation of interactive narratives using AI planning, a story domain representation is required [2]. To further adhere to narrative quality, a user model may be used to guide the planning algorithm towards a solution preferred by the player.

We propose an event-based storytelling platform to synthesize free-form interactive experiences within a 3D animated story world. It uses a novel story domain representation, consisting of smart objects and affordances, rather than Planning Domain Definition Language (PDDL) [3]. The possible storylines are created by an author in the form of branching stories, together with predefined user interactions that can be triggered at runtime. The stories can be loosely coupled, as an online planner is taking care of narrative generation to connect the storylines at runtime. In order to keep the user on a preferred trajectory, we employ a user model, based on the user's engagement with the virtual characters, to guide the planner.

This paper describes the following three contributions of our platform: (1) An interactive story domain representation based on smart objects, (2) an *engagement model* to infer user preference as well as user knowledge of important narrative information and (3) a two-level online planning framework to find optimal paths, and mediate user actions and inferred interests with author-intended story sequences.

We present the main concepts of our interactive storytelling system on an example story and describe the adaptations required for modeling engagement in AR and VR applications. The resulting applications are visualized in a video on the project page.

II. Related Work

Our approach takes inspiration from existing research in interactive narrative technology, as described below.

A. Experience Management

Interactive narratives require intelligence to guide the experience in a satisfying way. Riedl and Bulitko [1] advocate for the broader term "Experience Management" and provide a comparison of several interactive narrative systems in terms of author intent, character autonomy and player modeling. Plan-based experience management frameworks include Mimesis [4] and Merchant of Venice [5], which offer strong story-based control of virtual characters while simultaneously supporting automatically generated stories. The Mimesis system, for example, demonstrates the concept of narrative mediation, which exploits plan-based narrative representations to manage and respond to user interactions [6]. In their recent work [7] Robertson et al. leverage the user's perception of the world state to widen the space of possible accommodations of user interactions. We introduce a similar notion of user knowledge that can help the author to build the story around what the system assumes the user has learned about the narrative. Inspired by previous work, our system also aims to mediate between user interactions and author intent.

B. User Modeling

Another challenge for experience management systems is to provide a personalized experience for the user. Yu and Riedl [8] propose a data-driven approach to find the best fitting

path for a user to enjoy a set of possible stories. The system utilizes a user feedback mechanism as well as collaborative filtering to nudge the user towards the most appropriate story branch. The PaSSAGE system [9] proposes to find the best story path by selecting author-annotated encounters based on the user's play styles. In a follow-up work, Thue et al. [10] explore the selection of the best encounter based on a model of perceived user agency. Our system constructs a user model based on user knowledge of the story as well as engagement with virtual characters and objects, which is used to determine the optimal future narrative experience.

C. Narrative Generation

Since Young [11] first proposed AI planning for the task of interactive narrative generation, a variety of plan-based approaches have been adopted [2], [12]. As planning domains grow in the number of actions or participants, traditional planning techniques undergo a combinatorial explosion. Several planners like CPOCL [13], IPOCL [14] or Glaive [15] aim to reduce complexity by considering characteristics of character believability, such as intention or conflict. Kartal et al. [16] have demonstrated the use of Monte Carlo Tree Search (MCTS) to generate believable stories in domains having a high branching factor. They use a Bayesian story evaluation method to generate believable stories that achieve user defined goals. Soo et al. [17] extend this concept and designed a constrained MCTS algorithm that allows users to specify both hard and soft constraints. The soft constraints are expressed as a utility function to evaluate the quality of candidate stories, whereas hard constraints ensure plot coherence. Inspired by these approaches, we integrate an MCTS based algorithm within a novel domain representation to generate narratives that accommodate user interactions.

III. Overview

Our experience management system takes author-intended stories and controls the narrative progression based on user agency and engagement with the characters. To do so, the system: (1) continuously updates a user model, (2) determines an optimal path through the space of authored stories, (3) embeds user interactions into the running narrative considering the optimal path and (4) visualizes authored and emergent events as 3D animations.

The experience management system consists of the components visualized in Figure 1. The **Experience Manager** controls the playback of the story arc, observes user engagement, listens for user interactions and updates the other components as the story progresses. The **Story Domain Representation** (Section IV) contains the possible stories to tell, the story events, the story's state as well as the characters, objects and their states. The **User Model** (Section V) stores user engagement values for each smart object in the story, which are then used to find an optimal path in the space of authored stories. The **Two-Level Online Planning System** (Section VI) consists of two layers: the first calculates the previously mentioned optimal paths by maximizing a user engagement value, whereas the second layer generates story sequences in real-time in order to create a path to that optimal storyline.

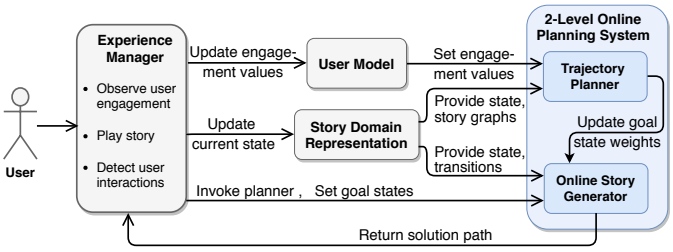

Fig. 1. The Experience Management System. The experience manager on the left updates the User Model and the current state of the Story Domain Representation. The two-level Online Planning System uses the two models to calculate an optimal path when confronted with the task of narrative generation due to a user interaction.

IV. Story Domain Representation

Our system builds upon `CANVAS` [18] for generating 3D animated stories and uses a similar domain representation, which is described in Section IV-A.

A. Domain Representation for Passive Narratives

Smart Objects $w = \langle q, \mathbf{F} \rangle$ represent the characters and objects in the story world. Each has a state q and a set $\mathbf{F}$ of affordances $f \in \mathbf{F}$, which it exposes to the story world. An **Affordance** $f(w_o, w_u) \in \mathbf{F}$ is a capability offered by a smart object w_o to be used by another smart object w_u. Invoking an affordance results in an animated interaction between the two smart objects that can change physical states. An affordance can also require that its participants fulfill certain physical state preconditions. **Physical States** are a binary representation of general smart object properties, such as attributes (e.g. `isStanding`), roles (e.g. `isHuman`) and relations $\mathcal{R}(w_i, w_j)$ between different smart objects w_i and w_j (e.g. `isInLoveWith`). These states are stored individually for each smart object.

An **Event** $e = \langle t, pre^e, \mathit{eff}^e \rangle$ defines a certain action performed by one or several smart object participants. Each event has a parameterized behavior tree [19] representation t, consisting of a sequence of affordances f, whose preconditions are collected to define pre^e, the preconditions for the whole event, and similarly the state effects of the whole event eff^e. An **Event Instance** $I = \langle e, w \rangle$ is an event e that has been assigned a list of valid smart object participants w. A **Beat** $B_i = \{I_1, \ldots, I_m\}$ contains one or more event instances I_i that are executed simultaneously. A **Story Arc** $\alpha = (B_1, B_2, \ldots, B_n)$ is an ordered sequence of beats that are replayed one after the other during execution of the story.

B. Domain Extensions for Interactive Narratives

In order to allow for interactive branching stories, our system introduces the following additional concepts:

Story States σ are defined during authoring and can be used to control the flow of the narrative through branching. A story state is never the effect of an event but has to be attached by

the author to a specific event instance. Accordingly, a **Story State Instance** σ^I is a story state that is conveyed by event instance I. **Experience States** ξ are very similar to story states. They are also attached to specific event instances but are only set if the user actually experiences them. Therefore, every **Experience State Instance** ξ^I is subject to different conditions, defining whether or not the user has consumed that story information. This decision is made at runtime depending on how engaged the user is with the event instance conveying this experience state. See Section V for more details. The **World State** s represents the collection of all physical states of every smart object, as well as the values for story states and experience states. To allow story states to control the branching, we extend story arcs to support world state preconditions, and story state and experience state invariants that have to remain true for the execution of that story arc. If the preconditions or invariant are not met, that story arc cannot be executed. This is conceptually similar to using state trajectory constraints as landmarks [20] used to decompose narrative generation tasks [5].

A **Story Graph** $\mathcal{A} = \{\alpha_1, \dots, \alpha_n\}$ is a collection of story arcs that are connected by follow-up arc relations as illustrated in Figure 2. When the narrative reaches the end of a story arc a follow-up arc is selected based on the currently set story states and experience states. Every story graph has one story arc, annotated with preconditions, that designates the entry point into this graph (illustrated in Figure 2 (a)). Our system supports multiple independent story graphs and can navigate between them by online planning, as visualized in Figure 3. This allows the story to be built in a modularized way. A **Storyline** is a valid sequence of story arcs that can be taken in order to traverse through a story graph.

A **User Interaction** $I^U = \langle I_p, T, \delta, \lambda \rangle$ is an author-defined event that can be triggered by the user to influence the world state. It consists of a story time window δ during which the user can cause the (partially) populated event instance I_p to be executed by activating the set of triggers T. Triggers can include platform specific inputs, world state conditions, as well as proximity to and visibility of specific smart objects. If I_p is only partially populated, the remaining participants are dynamically selected based on the rules defined in the policies λ. Policies select smart objects w_i at runtime based on a combination of proximity to a specific smart object, the smart object that the user clicked on, the current state of a smart object or their relations to other smart objects. These policies give the author the tools to create interaction events that affect different smart objects depending on the context in which they are triggered. The event instance I_p can also carry story state instances, allowing for more narrative control. For example, executing a user interaction that conveys story state `secretRevealed` during a story arc with invariant `¬secretRevealed` would force the experience manager to immediately leave that story arc and search for a valid one to execute, as illustrated in Figure 3 (d).

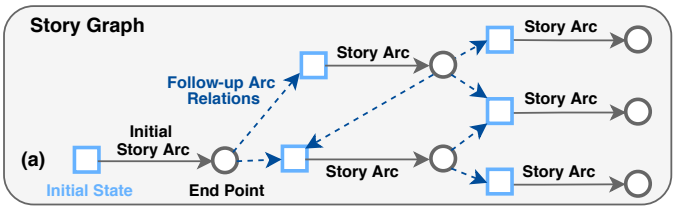

Fig. 2. Story Graph. A story graph has exactly one single point of entry (a). Each contained story arc has an initial state with preconditions and an end point, which can lead to several follow-up arcs.

V. A Model of User Engagement

Since our system relies on many storylines that could be executed, we require a model of the user's preferences to help the experience manager deciding on the best one. Following this storyline results in a personalized experience.

A. User Engagement

While the story is being played, the experience manager observes the user according to how involved they are with a smart object w_o or event e. We capture this involvement for each smart object and event individually in the form of engagement values. Engagement values are continuously calculated as a combination of the user's proximity to w_o, how often the user interacts with w_o and how well w_o is visible to the user:

$$\begin{aligned} v_{w_o}^{proximity} &= \min(1, \sqrt{1/d(w_o, u)}), \\ v_{w_o}^{interaction} &= \begin{cases} 1, & \text{if a click on } w_o \text{ was registered} \\ 0, & \text{otherwise} \end{cases} \\ v_{w_o}^{visibility} &= \sqrt{A(w_o) \cdot C(w_o)}, \end{aligned} \tag{1}$$

where $d(w_o, u)$ describes the distance between the user u and the smart object w_o. $A(w_o)$ represents the relative size of the smart object's enclosing box on screen and $C(w_o)$ describes how central the smart object is relative to the user's field of view. To calculate $C(w_o)$ we first need:

$$\begin{aligned} \Delta x &= (\vec{z}_{w_o} - \vec{z}_{cam}) \cdot \vec{x} \quad \text{and} \\ \Delta y &= (\vec{z}_{w_o} - \vec{z}_{cam}) \cdot \vec{y}, \end{aligned}$$

where $\vec{z}_{cam}$ is the normalized forward vector of the camera, $\vec{z}_{w_o}$ is the normalized vector from the camera to where smart object w_o is visible on screen and $\vec{x}$ and $\vec{y}$ represent the x- and y-axis unit vectors, respectively. With these we calculate the field of view in direction of where the smart object resides:

$$\gamma_{FOV} = \frac{\Delta x \cdot \phi_{FOV} + \Delta y \cdot \theta_{FOV}}{\Delta x + \Delta y},$$

where ϕ_{FOV} is the camera's horizontal and θ_{FOV} its vertical field of view in degrees. γ_{FOV} is then used to normalize the angle between $\vec{z}_{cam}$ and $\vec{z}_{w_o}$, which we define to be the centrality of smart object w_o:

$$C(w_o) = \frac{\arccos(\vec{z}_{cam} \cdot \vec{z}_{w_o})}{0.5 \cdot \gamma_{FOV}}.$$

The engagement value v_{w_o} for smart object w_o at any given moment is then calculated as:

$$v_{w_o} = \frac{1}{3}(v_{w_o}^{\textit{proximity}} + v_{w_o}^{\textit{interaction}} + v_{w_o}^{\textit{visibility}}) .$$

The **User Model** can be formalized as $\mathcal{M} = \langle V_W, V_E \rangle$, where $v_{w_o} \in V_W$ depict the user's engagements for each smart object w_o and $v_e \in V_E$ are the user's engagement values for each event e. Engagement values v_{w_o} are calculated at a regular interval during execution of the story. The values stored in the user model are a running average of these measurements.

In order to infer engagement values v_e for an event e, we require event e to be featured in an event instance $I = \langle e, w \rangle$ of the story. During I's execution, the experience manager calculates the engagement values v_{w_o} with the participants of I and averages these values over the course of I's execution to get an estimate v_I for how engaged the user was with that event instance. The average engagement of an event e is then calculated as the running average of all engagement values v_I of executed event instances featuring e.

B. Experience State Instances

When an event instance I conveys an experience state instance ξ^I, we calculate the additional engagement value v_{ξ^I} to assess whether the user experienced ξ^I or not. This value is calculated in a similar fashion as v_I. The difference is that, depending on the conditions the author set to be necessary for experiencing ξ^I, only a subset of the terms in Equation 1 is considered for calculating v_{ξ^I}. For example, if look-at is not required, then $v_{w_o}^{\textit{visibility}}$ from Equation 1 is not considered. This might be the case when the story instance conveying the information is a dialogue for which the user only has to be in hearing range but is not required to actually see the dialogue being played out. Also, $v_{w_o}^{\textit{proximity}}$ is slightly modified, such that the value turns 0 if the user is further away than a maximum distance set by the author. If the average v_{ξ^I} remains above a certain threshold after I has finished, the experience manager assumes that the information was consumed and the experience state is set.

VI. Two-Level Planning for Online Narrative Generation

In order to embed user interactions into the running story and follow an optimal path, we developed a planning system that consists of two stages: (1) finding the best path along the authored stories and (2) synthesizing event sequences to reach the previously determined optimal storyline. The first task is handled by the **Trajectory Planner**, whereas the latter is solved with Monte Carlo Tree Search (MCTS) for **Online Narrative Generation**.

A. Trajectory Planner

In narratives, the shortest story path is rarely the best one. Our trajectory planner aims to score possible paths through a story graph according to how well they fit a trajectory defined by the author. Such a trajectory could represent suspense, user awareness, emotions of characters or other measures that can be inferred from the interactive narrative. For now, we aim to maintain high engagement, which simplifies the trajectory to a straight line over time with value 1. The trajectory planner evaluates each story arc of a story graph relative to the engagement values stored in the user model and represents each storyline as a sequence of values, which it then compares to the given trajectory. In our case, the algorithm returns for each story graph the storyline with highest average engagement value.

1) Implementation Details: The trajectory planner performs an exhaustive search and represents each possible path through a story graph as a series of values (one for each story arc) that can then be fitted to the author defined trajectory. Therefore, we evaluate the engagement estimate v_{α_i} of every story arc α_i as follows:

Engagement Value for Contained Event Instances First we evaluate the story arc for how engaging its event instances are.

$$v_{I_i} = v_e \cdot \frac{1}{|w|} \sum_{w_o \in w} v_{w_o} ,$$

where w are the participants of the event instance $I_i = \langle e, w \rangle$.

We then average the v_{I_i} of all event instances $I_i \in \mathbf{I}$, the set of event instances in α_i, to get our final estimate $v_{\mathbf{I}}$.

Engagement Value for Participants We calculate the frequency r_{w_o} at which some smart object w_o appears in α_i as:

$$r_{w_o} = \frac{N(w_o)}{\sum_{w_i \in \mathbf{W}} N(w_i)} ,$$

where $N(w_o)$ depicts how often w_o participates in some event instance and $\mathbf{W}$ is the set of all smart objects participating in the story arc α_i.

Finally, the cumulative engagement value for the participants in α_i is calculated as

$$v_{\mathbf{W}} = \sum_{w_i \in W} r_{w_i} v_{w_i} .$$

Engagement Value for Story Arc The two measures above are averaged to estimate the engagement value of the story arc α_i as

$$v_{\alpha_i} = \frac{1}{2}\big(v_{\mathbf{I}} + v_{\mathbf{W}}\big) . \tag{2}$$

With these values, we can score each acyclic storyline through a story graph according to its engagement over time and select the one that is closest to the author-defined trajectory. Since we perform this calculation for each story graph separately, we can assign each story graph $\mathcal{A}_i$ a score equal to the score of its best storyline. This allows us to compare the different story graphs and find the best suiting one. In case that the user no longer seems interested in the current storyline we can attempt to guide the story towards a story graph with a higher score.

B. Online Narrative Generation

An optimal path is of no use if we are unable to find a way to get there or stay on it. Therefore, we require an online planner that can synthesize event sequences leading to the optimal storyline. However, planning in narrative state space is known to be P-SPACE complete [21]. This means that for large narratives, we have to constrain the state space or accept the fact that we cannot search the whole state space for a solution path. Monte Carlo Tree Search (MCTS) has been successfully applied to several high dimensional problem domains like the game Go [22] or the domain of General Game Playing [23]. Additionally, its anytime characteristic is very useful for real-time applications, as it allows a best next step to be extracted at any time, even though a full solution plan might not yet be available. It is for these reasons that we explore the approach of using an MCTS algorithm to synthesize event sequences in real-time.

1) Monte Carlo Tree Search: MCTS is an iterative search algorithm that samples the search space and builds an asymmetric search tree. Each node of the tree represents a state s and stores statistics about how often a transition a was sampled in this state ($N(s,a)$), how many times this node was reached ($N(s)$), and the average reward $Q(s,a)$ obtained after applying transition a in state s. The child nodes represent the states directly reachable over some transition a. MCTS performs several iterations to sample the search tree before converging on a best next step. Each iteration starts at the root node s_0 of the tree and can be split up into four steps: *Selection*, *Expansion*, *Simulation* and *Backpropagation*.

Selection In the selection step, the algorithm decides on the next child node of the search tree to visit. This decision is usually done by equally weighting exploitation (select child node with highest average value) and exploration (select child node with lowest visit count). This is usually done by using the Upper Confidence Bound applied to Trees (UCT) [24]:

$$UCT(s) = Q(s,a) + k\sqrt{\frac{\log N(s)}{N(s,a)}},$$

where k is usually chosen to be $\sqrt{2}$ and attempts to balance the left term (exploitation) with the right term (exploration).

The child node with highest UCT value is selected and the iteration continues at this child with another selection step unless we reached a leaf node. At a leaf node we either perform *Expansion*, in case that the maximum depth of the tree is not yet reached, or *Simulation* respectively.

Expansion In the expansion step, all states that are reachable with some transition from the current node are added as child nodes. This way the search tree grows. However, before any of these children is allowed to undergo Expansion as well, they all have to be visited at least once.

Simulation When the iteration reaches a leaf node or a node that has not yet been visited, a random simulation of transitions is performed until a goal state or the maximum search depth is reached. The final state s_k of this simulation is evaluated according to a heuristic function. The heuristic value $h(s_k)$ is then used in the *backpropagation* step.

Backpropagation In this step, the heuristic value $h(s_k)$ achieved at the end of the simulation step is added to the average $Q(s,a)$ of each node that was visited when traversing from the root to the sampled leaf node. The visit count $N(s)$ and $N(s,a)$ of each visited node is also increased. With this, the iteration has finished and a new one can start at the root.

2) Implementation Details: The MCTS algorithm implemented in our system uses the standard Upper Confidence Bound applied to Trees (UCT) [24], which has previously been used for narrative generation [16]. At the leaf nodes we perform simulations based on an evolutionary algorithm similar to the one proposed in [25] until a goal state or a predefined maximum depth is reached.

Evolutionary Algorithm Since completely random simulations in high dimensional spaces are not very effective, we direct the simulation step towards solutions with higher heuristic value by using an evolutionary algorithm. Therefore, each leaf node contains a population $\mathcal{P} = \{p_0, p_1, \dots p_n\}$ of simulations $p = \{a_0, a_1, \dots a_k\}$ that have been evaluated during past iterations. Each p is associated with the heuristic value $h(s_k)$ it achieved at its final state. The population of each leaf node is initialized with one random simulation p_0, which from there on undergoes local mutations or is recombined with paths from the same population. In comparison to the approach in [25], where transitions do not have any preconditions, we have to assure however, that the mutated path is still a valid one. If preconditions are no longer met at some point during a mutated simulation, a random valid transition is introduced instead. Every mutated path is eventually evaluated and added to the population. Whenever the population's maximum size is reached, the lowest scoring specimen p is removed, training the population towards higher rewards. In order to assure enough diversity among the specimen, completely random simulations are performed approximately every third simulation and added to the population according to the same rules.

Heuristic Function The final world state s_k reached by a simulation is evaluated according to the following state distance heuristic:

$$h(s_k) = w(g) \cdot \big(1 - d(s_k, g)\big)\,,$$

where g is the goal state, $w(g)$ the weight for that goal state as set by the experience manager and $d(s_k, g)$ the normalized state distance between s_k and g. Since we have binary states, $d(s_k, g)$ is equal to the normalized number of state bits that have to be switched in order to arrive at the goal state g, which can be calculated efficiently. Additionally, goal states can be simplified by using **state masks**. These masks define which bits of the goal state really need to be met and which ones are not relevant and can therefore

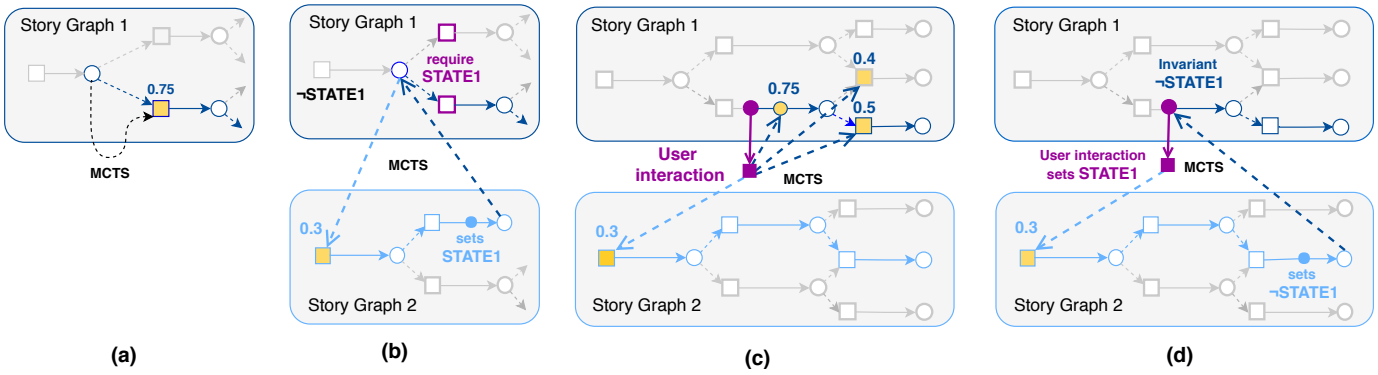

Fig. 3. Controlling the Story. The optimal path for each story graph is outlined in color. Goal states are highlighted in orange and are annotated with their respective weights. (a) shows the case where the MCTS needs to generate a path to reach the follow-up arc. (b) shows a scenario, where the follow-up arcs are blocked, because they require story state STATE1. (c) shows the case, where a user interaction causes a diversion from the story and the possible goal states for the MCTS. (d) shows the case, where the executed user interaction conveys a story state that makes it impossible to return to the current story graph, due to its invariant. Here, some event on story graph 2 sets STATE1 back to false, allowing the execution to return to the interrupted story graph.

be excluded from the distance calculation. The experience manager masks out all those bits of a goal state that are not relevant for the successful execution of the story arc that would be reached when arriving at this goal state. This is similar to planning in PDDL, where goal states are usually only a set of specific literals that have to be satisfied. These state masks are calculated offline for every authored beat of a story arc, therefore not impacting real-time performance.

3) Planning in the Story Domain: The nodes of the search tree represent world states, whereas the edges represent the event instances applicable in the respective world state. It is important to note that the MCTS is not allowed to generate event instances that change story states or experience states. These can only be changed by events that are part of an authored story arc or an author intended interaction event.

In the most simple case, the planner has to be invoked when the end of a story arc is reached and a plan to satisfy the preconditions of the follow-up arc has to be found, as shown in Figure 3 (a). If all of the follow-up arcs are blocked due to non-matching story states or experience states, then the experience manager has to plan towards another story graph as illustrated in Figure 3 (b). Whenever the user triggers a user interaction that changes the world state s, the MCTS is invoked to determine a path, after which the story can seamlessly continue on the optimal path, as illustrated in Figure 3 (c) and (d).

The possible **goal states** for the search are (1) the initial world state of the beat that got interrupted by a user interaction, (2) the initial world state of the beat that would have followed, (3) the initial world states of the follow-up arcs of the current story arc and (4) the initial world states of all other reachable story graphs. Since the MCTS only operates in physical state space, the experience manager further reduces the set of goal states to those, whose story state preconditions and invariants are met at the initial state of the search. Each goal state g_k is then weighted by the score of the best path through the story graph $\mathcal{A}_i$ on which g_k resides. With these weights, the trajectory planner guides the online story generation, as the experience manager only selects the goal state with highest weight for planning. Additional weights apply depending on which of the four goal state categories above g_k belongs to.

For each plan, we grant the MCTS planner 1500ms of time, which is an upper limit for maintaining a real-time experience [5]. If the MCTS cannot find a full solution path in the given time, the experience manager selects the best next step from the root node for execution and updates the root accordingly. During the event instance's execution, the MCTS has time to calculate the next step or find a full solution path. The MCTS only stops once its root node represents the goal state of the search, which means that the experience is back on track of an authored story arc.

VII. Results

We present the key concepts of our system on an interactive story and elaborate on the changes required to embed such stories in an AR or VR environment. Also, we evaluate our MCTS implementation within a more challenging story domain.

A. Example Interactive Story

The example interactive story takes place in our Haunted Castle scenario. Its domain consists of 10 smart objects and 75 possible event instances, which resulted in an average branching factor of approximately 19 for online narrative generation.

The interactive story follows the concepts illustrated in Figure 3. It consists of three story graphs: the main story graph and two additional ones, telling background information. Moving this information into separate story graphs allows to access them from any other place in the story through story states. After the initial story arc, the author wants the user to learn more about the background of the characters before continuing. Therefore, the follow-up arcs are blocked by story

state preconditions. These can only be satisfied by visiting one of the other two story graphs, similarly to Figure 3 (b). The trajectory planner evaluated one of the two as being more engaging, due to the user's previous engagements with characters and events. Therefore, the MCTS planned towards the story graph containing the best scoring storyline.

Once the small story graph ended, the experience manager attempted to continue on the story graph it left off from. There, the previously blocked follow-up arcs were now unlocked due to a story state that was set at the end of the small story graph. If the user still wanted to learn about the other character, then they could trigger an interaction event that sets a story state that is incompatible with the invariant of the current story arc, as illustrated in Figure 3 (d). If executed, this forced the experience manager to plan towards a compatible story arc, which it found in the other small story graph.

Further user interactions allowed to make characters pick up or drop objects, which in theory could cause any of the diversions indicated in Figure 3 (c). The additional weights indicated there ensured however that the story stayed on the current story arc, unless a much more engaging storyline had been identified by the trajectory planner.

For this story, the MCTS planner was given 1500ms before being polled for the best next step. The trajectory planner returned results almost instantaneously, which was also due to the small search space.

B. AR & VR Application

We deployed our storytelling system in preliminary AR and VR applications. The AR application allows the user to discover a story in our Haunted Castle scenario in a top down view, as can be seen in Figure 4. Interactions with the smart objects are enabled through touch input or proximity. The latter required some minor adaptations, since in AR the camera is much further away than in a first person experience. In order to allow proximity calculations for interactions and engagement values, the player is embodied as a glowing orb in the scene. The orb follows the center of the AR device's camera and is restricted to stay within the navigable boundaries of the scene. Proximity calculations are performed relative to this orb.

In our VR application, the user is immersed inside the Haunted Castle and experiences an interactive story from a first person perspective, as illustrated in Figure 5. There were no issues concerning proximity calculations. The only adaption required was to use the controller of the VR device for interaction and navigation input beyond the tracking area.

C. MCTS Planner Performance

The MCTS had no issues for planning in our Haunted Castle scenario and was able to generate plans within the 1500ms timelimit per step, with no noticeable latency in our interactive application. In order to evaluate the MCTS planner on a more challenging problem domain, we created one consisting of 10 smart objects, 27 events and 314 possible event instances, which resulted in an average branching factor of approximately 43. We generated several random goal states and let the MCTS find a path to them in an iterative fashion. The MCTS was granted 1500ms for each step. The MCTS planner was able to produce a valid plan for a majority of these randomly generated goal state instances within a reasonable amount of steps. However, there were a few problem instances where the planner was unable to return a solution. This can be attributed to the strict time bounds we placed on the planner and local minima encountered during the search. The local minima is caused by one or more events reducing the state distance by a large amount. The MCTS would usually execute these solutions first as they provide a good heuristic score. However, if they have to be temporarily undone later, in order to execute another required event, the MCTS may stay in this local minima.

Tests were performed on an Intel i7 6700K running at 4.0 GHz and 32 GB of RAM.

Fig. 4. The AR application recognizes the map of the Haunted Castle as AR marker and projects the interactive story world onto it.

Fig. 5. The VR application allows to get closer into the story while preserving the known engagement calculations and user interactions.

VIII. Limitations & Future Work

The user engagement model requires a user study for further evaluation and could be extended with more advanced concepts and heuristics. For example, eye-tracking could give a more reliable estimate of what the user is engaged with. Also, only because a story arc is not featuring a character w_o in its event instances does not mean the other participants are not talking about w_o, which would also be very interesting for a user, who is interested in w_o. Currently, the trajectory planner cannot detect this and would rank such a story arc rather low. Having author annotations for story arcs would help the trajectory planner to score them higher.

Additionally, the trajectory we chose for evaluation does not consider any temporal aspect. If the trajectory were to be a Freytag's pyramid, then the author might want to define how long the rising action lasts in comparison to the exposition or denouement. However, for this to work we would require an estimate for how long a story arc takes to execute.

Despite being intended for planning in high dimensional spaces, the reliability of the MCTS, when it comes to proposing a best next step, is decreasing for high branching factors. Therefore, improved heuristics for search acceleration, and analysis of domains to identify and resolve dead ends and local minima, are potential avenues of future exploration. Possible techniques for reducing branching factor include: (1) only considering smart objects used in the story arc of the goal state, (2) novelty pruning [26] or (3) authoring and leveraging character intentions [27].

Due to the iterative planning approach, we cannot perform any intervention of user interactions. This means that our story domain needs to be carefully crafted such that states are always reversible or the author creates a story graph to handle the effects of exceptional user interactions.

IX. Conclusion

We proposed an experience management system for interactive storytelling that leverages user engagement to drive narrative progression. Our domain representation includes one or more story graphs annotated with story states and experience states to allow the author to effectively constrain the narrative paths. We describe the construction of a user model based on engagement with characters and objects in the story. Our two-level online planning system first computes optimal trajectories within story graphs, and second finds a solution path to the best trajectory using Monte Carlo Tree Search. We described the adaptations required for modeling user engagement in AR and VR applications, and presented the main concepts of our interactive storytelling system on an example story.

Acknowledgment

We would like to thank Maurizio Nitti and Alessia Marra for creating the 3D assets of the Haunted Castle, and Mattia Ryffel and Shaula Möller-Telicsak for making the AR and VR apps a reality and improving the authoring GUI. This work was (partly) funded through the EC H2020 Framework Programme project FLAME, Grant Agreement No. 731677.

References

[1] M. O. Riedl and V. Bulitko, "Interactive Narrative: An Intelligent Systems Approach," *AI Magazine*, vol. 34, no. 1, pp. 67–77, 2013.

[2] R. M. Young, S. Ware, B. Cassell, and J. Robertson, "Plans and planning in narrative generation: a review of plan-based approaches to the generation of story, discourse and interactivity in narratives," *Sprache und Datenverarbeitung, Special Issue on Formal and Computational Models of Narrative*, vol. 37, no. 1-2, pp. 41–64, 2013.

[3] M. Fox and D. Long, "PDDL2.1: An extension to PDDL for expressing temporal planning domains," *J. Artif. Int. Res.*, vol. 20, no. 1, pp. 61–124, Dec. 2003.

[4] R. M. Young, M. O. Riedl, M. Branly, A. Jhala, R. Martin, and C. Saretto, "An architecture for integrating plan-based behavior generation with interactive game environments," *Journal of Game Development*, vol. 1, no. 1, pp. 51–70, 2004.

[5] J. Porteous, M. Cavazza, and F. Charles, "Applying planning to interactive storytelling: Narrative control using state constraints," *ACM Trans. on Intelligent Systems and Technology (TIST)*, vol. 1, no. 2, p. 10, 2010.

[6] M. Riedl, C. J. Saretto, and R. M. Young, "Managing interaction between users and agents in a multi-agent storytelling environment." ACM Press, 2003, p. 741.

[7] J. Robertson and R. M. Young, "Perceptual experience management," *IEEE Transactions on Games*, 2018.

[8] H. Yu and M. O. Riedl, "Optimizing players expected enjoyment in interactive stories," in *Proc. of the 11th Annual AAAI Conf. on Artificial Intelligence and Interactive Digital Entertainment*, 2015, pp. 100–106.

[9] D. Thue, V. Bulitko, M. Spetch, and E. Wasylishen, "Interactive storytelling: A player modelling approach." in *Proc. of the 3rd AAAI Conf. on Artificial Intelligence and Interactive Digital Entertainment*, 2007.

[10] D. Thue, V. Bulitko, M. Spetch, and T. Romanuik, "A computational model of perceived agency in video games." in *Proc. of the 7th AAAI Conf. on Artif. Intelligence and Interactive Digital Entertainment*, 2011.

[11] R. M. Young, "Notes on the use of plan structures in the creation of interactive plot," in *AAAI Fall Symp. on Narrative Intelligence*, 1999.

[12] J. Porteous, "Planning Technologies for Interactive Storytelling," in *Handbook of Digital Games and Entertainment Technologies*. Springer Singapore, 2017, pp. 393–413.

[13] S. G. Ware and R. M. Young, "CPOCL: A narrative planner supporting conflict," in *Proc. of the 7th AAAI Conf. on Artificial Intelligence and Interactive Digital Entertainment*. AAAI Press, 2011, pp. 97–102.

[14] M. O. Riedl and R. M. Young, "Narrative planning: Balancing plot and character," *J. Artif. Int. Res.*, vol. 39, no. 1, pp. 217–268, Sep. 2010.

[15] S. G. Ware and R. M. Young, "Glaive: a state-space narrative planner supporting intentionality and conflict," in *AAAI Conf. on Artificial Intelligence and Interactive Digital Entertainment*, 2014, pp. 80–86.

[16] B. Kartal, J. Koenig, and S. J. Guy, "User-driven narrative variation in large story domains using monte carlo tree search," in *Proc. of the 2014 Int. Conf. on Autonomous Agents and Multi-agent Systems*, 2014.

[17] V.-W. Soo, C.-M. Lee, and T.-H. Chen, "Generate Believable Causal Plots with User Preferences Using Constrained Monte Carlo Tree Search," in *AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment*, 2016.

[18] M. Kapadia, S. Frey, A. Shoulson, R. W. Sumner, and M. Gross, "CANVAS: Computer-assisted narrative animation synthesis," in *Proc. of the ACM SIGGRAPH/Eurographics Symp. on Computer Animation*. Goslar, Germany: Eurographics Assoc., 2016, pp. 199–209.

[19] A. Shoulson, F. M. Garcia, M. Jones, R. Mead, and N. I. Badler, "Parameterizing behavior trees," in *Proc. of the 4th Int. Conf. on Motion in Games*. Berlin, Heidelberg: Springer-Verlag, 2011, pp. 144–155.

[20] J. Hoffmann, J. Porteous, and L. Sebastia, "Ordered landmarks in planning," *J. Artif. Int. Res.*, vol. 22, no. 1, pp. 215–278, Nov. 2004.

[21] T. Bylander, "Complexity results for planning," in *Proc. of the 12th Int. Joint Conf. on Artificial Intelligence - Volume 1*, 1991, pp. 274–279.

[22] D. Silver, A. Huang, C. J. Maddison, A. Guez, L. Sifre, G. Van Den Driessche, J. Schrittwieser, I. Antonoglou, V. Panneershelvam, M. Lanctot *et al.*, "Mastering the game of go with deep neural networks and tree search," *Nature*, vol. 529, no. 7587, p. 484, 2016.

[23] H. Finnsson and Y. Björnsson, "Simulation-based approach to general game playing," in *Proc. of the 23rd National Conf. on Artificial Intelligence - Volume 1*. AAAI Press, 2008, pp. 259–264.

[24] L. Kocsis and C. Szepesvári, "Bandit based monte-carlo planning," in *Proc. of the 17th European Conf. on Machine Learning*. Berlin, Heidelberg: Springer-Verlag, 2006, pp. 282–293.

[25] D. Perez, S. Samothrakis, and S. Lucas, "Knowledge-based fast evolutionary MCTS for general video game playing," in *2014 IEEE Conf. on Computational Intelligence and Games*, Aug 2014, pp. 1–8.

[26] R. Farrell and S. G. Ware, "Fast and diverse narrative planning through novelty pruning," in *Proc. of the 12th AAAI Int. Conf. on Artificial Intelligence and Interactive Digital Entertainment*, 2016, pp. 37–43.

[27] J. Teutenberg and J. Porteous, "Efficient intent-based narrative generation using multiple planning agents," in *Proc. of the 2013 Int. Conf. on Autonomous Agents and Multi-agent Systems*, Richland, SC, 2013.