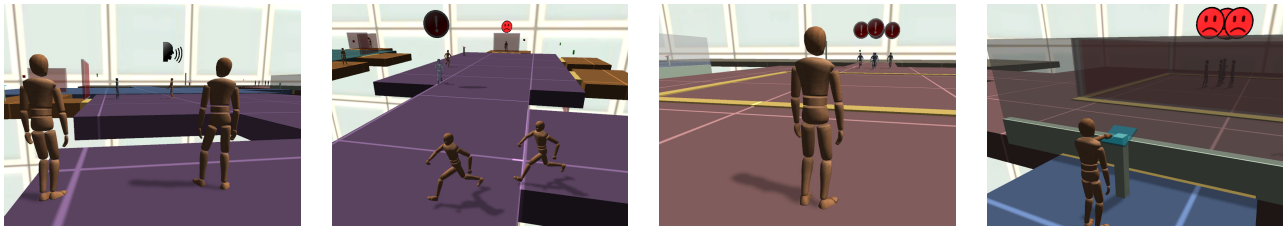
An Event-Centric Planning Approach for Dynamic Real-Time Narrative

Alexander Shoulson* Max L. Gilbert† Mubbasir Kapadia‡ Norman I. Badler§
University of Pennsylvania
Philadelphia, PA, USA



(a) (b) (c) (d)

Figure 1: *Characters exhibiting two cooperative behaviors. Two actors hide while a third lures a guard away (a), then sneak past once the guard is distracted (b). An actor lures the guards towards him (c), while a second presses a button to activate a trap (d).*

Abstract

In this paper, we propose an event-centric planning framework for directing interactive narratives in complex 3D environments populated by virtual humans. Events facilitate precise authorial control over complex interactions involving groups of actors and objects, while planning allows the simulation of causally consistent character actions that conform to an overarching global narrative. Events are defined by preconditions, postconditions, costs, and a centralized behavior structure that simultaneously manages multiple participating actors and objects. By planning in the space of events rather than in the space of individual character capabilities, we allow virtual actors to exhibit a rich repertoire of individual actions without causing combinatorial growth in the planning branching factor. Our system produces long, cohesive narratives at interactive rates, allowing a user to take part in a dynamic story that, despite intervention, conforms to an authored structure and accomplishes a predetermined goal.

CR Categories: I.3.7 [Computer Graphics]: Three-Dimensional Graphics and Realism—Animation;

Keywords: interactive narrative, behavior authoring, automated planning, events, behavior trees

*ashoulson@gmail.com
†gilbert.l.max@gmail.com
‡mubbasir.kapadia@gmail.com
§badler@seas.upenn.edu

MIG '13, November 06 - 08 2013, Dublin 2, Ireland

http://dx.doi.org/10.1145/2522628.2522629

1 Introduction

Complex virtual worlds with sophisticated, autonomous virtual populaces are an essential tool in developing simulations for security, disaster prevention, and interactive entertainment. Directed interactions between virtual characters help tell rich narratives within the virtual world and personify the characters in the mind of the user. However, creating and coordinating fully-fledged virtual actors to behave realistically and act according to believable goals is a significant undertaking. This difficulty is amplified when we design these characters to cooperate or compete according to their personal motivations and needs. In order to steer the trajectory of the narrative in a meaningful fashion, we need a virtual director capable of making narrative decisions based on an understanding of the world and its characters, including those characters' abilities and individual goals. This volume of information is too large to be managed in real-time, requiring a level of abstraction to temper the complexity of a rich world and the emergent nature of its characters' behavior.

Traditional stories are usually written in a form that emphasizes parsimony. Even the most prosaic description of characters engaging in a conversation is not likely to concern itself with details like the characters approaching and orienting towards one another, or listeners changing their posture to affect attention towards the speaker, unless these details deviate from the reader's expectation in a meaningful way. Similarly, a planning controller in a virtual world designed for narrative impact should not be concerned with details more significant to the execution of an animation than with the story being presented. Still, these mechanical elements cannot simply be excluded – as minor as the action may be, two actors in a conversation must still visibly approach and orient towards one another. To collapse a very large, mechanically-focused character action space into a series of tools available to a narrative planner, we define and use a domain of *events*: dynamic and reusable behaviors for groups of actors and objects that carry well-defined narrative ramifications.

Events temporarily preempt the autonomy of their participating actors with pre-authored, coordinated behaviors. For example, a conversation event would suspend the behavior of two participating actors, and dispatch commands for them to approach one another and play representational gesture animations. Using events as an encapsulation of the traditional atomic action planning domain affords an author several advantages:

Authoring Fidelity. Events are designed to carry out specific se-

quences of actions with some variability, and generally act with a single narrative intent. This allows an author to create emergence in the overarching narrative structure of a simulation between events while still ensuring that when an event occurs, it will proceed as intended. Since the results of a planner are difficult to verify in practice for large domains, being able to verify the behavior of each event prior to release reduces the tendency of the system to produce undesired effects.

Synchronization. Events can be used to coordinate very sophisticated multi-actor behaviors like conversation or dance that require precisely timed animations or procedural character controllers. Rather than depending on a planner to organize the subtasks of a compound character interaction, a pre-authored event can be written to properly time each action's execution between any number of actors involved.

Efficiency. An event takes full control of its participants and treats them as if they were limbs of the same entity. This allows multiple events to control numerous diverse sets of characters in parallel, and decouples the complexity of the planning domain from the number of agents in the world. Rather than searching through all possible actions a character can perform on all other objects and characters in the world, the search is limited to the set of possible events, which is a smaller and more tractable domain.

Events are more sophisticated than macro-operators [Botea et al. 2005] or task networks [Erol et al. 1994] that decompose into atomic actions when invoked. An event is a self-contained autonomous coroutine capable of making dynamic decisions during execution based on its own state, the state of its participants, and stochastic processes. Events have an arbitrary, variable, and finite runtime, and may ultimately fail during their execution. We author these events using Parameterized Behavior Trees (PBTs) [Shoulson et al. 2011] with additional meta-information that is used by the planner. Their graphical nature, expressiveness, synchronization constructs, and capability to make decisions make PBT-based events more suited for authoring behavior than macro-operators and task networks.

Contributions. This paper contributes a real-time planning framework that operates in the space of complex pre-authored character interactions. Narratives produced by our planner can react to the actions of a user and adapt to intervention that both helps and hinders the story objectives. We introduce a scheduler that converts a sequential narrative into a highly parallel simulation of simultaneous cooperative and competitive behaviors that all contribute to the overarching story. This produces dynamic virtual worlds with inhabitants that interact in intricate, meaningful ways beyond just the sequence produced by a plan. Our ideal application for this system is in building virtual worlds for games or simulations where a virtual populace is expected to interact with one another and a human user, carrying out narrative sequences with goals and causal relationships.

2 Related Work

Interactive narrative is quickly becoming a well-studied research topic with numerous, diverse methods. As an interdisciplinary field, interactive narrative builds upon techniques from crowd simulation, cognitive modeling, and automated planning.

Crowd Simulation. Crowd simulation is a mature field of research, concerned with the realistic simulation of large groups of lifelike characters in a virtual space. One primary focus is on navigation for virtual characters, which can be accomplished with social force models [Helbing and Molnár 1995], reactive behaviors [Reynolds 1999], synthetic vision [Ondřej et al. 2010], and predictive models [van den Berg et al. 2008]. Characters in virtual crowds often exhibit simple, scalable behavior with basic individual goals [Shao and Terzopoulos 2007] driven by heavy scripting or fuzzy logic [Massive Software Inc. 2010], where character interactions emerge as artifacts of collision avoidance [Pelechano et al. 2007]. Pelechano et al. [2008] give a more detailed survey of additional work in crowd simulation.

Cognitive Models. Numerous techniques have been studied for simulating the cognitive process of an autonomous agent. The decision-making process of a virtual character can be represented using scripts [Perlin and Goldberg 1996; Vilhjálmsson et al. 2007; Loyall 1997], neural networks [Blumberg 1997], hierarchical state machines [Menou 2001], partially-observable markov decision problems [Si et al. 2005], or decision networks [Yu and Terzopoulos 2007]. These represent methods for simulating an individual character, where cooperative behavior emerges as a requirement for accomplishing specific goals. In contrast, event-centric behavior using Smart Events [Stocker et al. 2010] or parameterized behavior trees [Shoulson et al. 2011] focuses on centralizing the logic for complicated multi-character interactions in order to simplify the authoring process.

Planning. One of the earliest planning systems, the STRIPS planner [Fikes and Nilsson 1971], lays down the framework of a described world state with operators, preconditions, and effects manipulating that state. Subsequent developments to the pervasive STRIPS archetype include the planning domain definition language [Mcdermott et al. 1998], cognitive-oriented planning [Funge et al. 1999; Porteous et al. 2010], hierarchical task networks [Erol et al. 1994], and planning with smart objects [Abaci and Thalmann 2005]. Our work can be seen as further exploration of the ideas presented by Kapadia et. al. [2011] in domain-independent planning for multi-actor behavior authoring. We share a similar focus on cooperative and competitive character interactions while simplifying the process to enable real-time performance and more direct author control.

Interactive Narrative. Interactive narrative systems draw on these techniques and others to create virtual worlds with a narrative focus. Façade [Mateas and Stern 2003], the first fully-realized interactive narrative system, uses natural language understanding and pre-authored narrative beats to create a story that adheres to an aristotelian narrative arc. Mimesis [Riedl et al. 2003] uses narrative planning [Li and Riedl 2011] with atomic agent actions, Thespian [Si et al. 2005] uses decision-theoretic agents, and PaSSAGE [Thue et al. 2007] guides a player through pre-scripted "encounters" based on the system's estimation of the player's ideal experience. Most of these systems employ some form of drama manager or director [Magerko et al. 2004], a virtual agent responsible for monitoring the state of the story and intervening according to narrative goals. Riedl and Bulitko [2013] also provide a more detailed survey of the current state of the art in interactive narrative.

Comparison to Prior Work. Our system creates a balance between two schools of interactive narrative techniques. Heavily scripted scenarios allow for computational efficiency and authorial control at the expense of flexibility and with a great deal of effort in content creation. In contrast, narrative planners that operate in the action space of each agent's individual capabilities quickly suffer from combinatorial explosion with large groups of agents. Additionally, like all highly emergent systems, they may produce unexpected and undesirable narrative plans. Our event-centric planner allows a virtual director to decide which events occur at which locations and with which participants, while still leaving the details of the underlying behavior in the hands of an author. This alleviates the back-and-forth planning complexity of multi-actor interactions, and al-

lows characters to exhibit a rich repertoire of actions suitable for a rich 3D environment (gestures, reaching, gazing, and so on) without affecting the branching factor of the simulation.

For example, the Automated Story Director (ASD) [Riedl et al. 2008] is an interactive narrative planner that uses traditional operators to execute the individual capabilities of actors in a world to create a story. An author designs an overarching narrative graph and the system attempts to adhere to the authored structure by producing planned strategies for the virtual actors and falling back on generated contingencies. Unlike the agent-centric ASD, which takes in an exemplar narrative and sets of character actions, our system event-centric uses a library of invokable events with narrative significance expressed by their pre- and postconditions. Our system uses events as global operators across the entire world population to achieve authored narrative goals, as opposed to each actor's local action set. Additionally, where the ASD would produce an interaction by producing the individual turn-taking actions for each character involved, our system invokes a single pre-authored event and lets the behavior of that event conduct the interaction, including carefully coordinated synchronous behavior. This trades some low-level emergence for more precise authorial control over sensitive operations like coordinated character animation.

3 Problem Domain and Formulation

For planning in the space of narrative, we use a generalized best-first planner operating on a problem domain defined as $\Sigma = \langle \mathbf{S}, \mathbf{A} \rangle$ where $\mathbf{S}$ is the combined state of all objects and actors in the environment, and $\mathbf{A}$ is the space of authored events. A particular problem instance is defined as $\mathbf{P} = \langle \Sigma, S_{\text{start}}, S_{\text{goal}} \rangle$ where $S_{\text{start}}, S_{\text{goal}} \in \mathbf{S}$ are the initial and desired world configurations. The planner generates a sequence of event transitions $\Pi(S_{\text{start}}, S_{\text{goal}})$ that leads the simulation from S_{start} to S_{goal}.

3.1 State Space

Each object in the world $\mathbf{W}$ is described as $o = \langle c, s \rangle \in \mathbf{W}$ where c is a controller and s is the object's state. We define s as $s = \langle r, p, d \rangle$ where r is a role index number, p is a 3-vector for world position, and d is a high-level description of an object as an unsigned bitmask. For example, a door will have a state description where "opened" and "locked" correspond to 0 or 1. The role index number defines an object's archetype, so that a character is different from a chair, and thus fills a different role in the narrative.

The overall state of the world is defined as the compound state of all of the objects in that world, $S = \{s_1, s_2, \ldots, s_n\}$ where s_i is the state of o_i. Objects can be actors, which have their own autonomy, or props, which cannot act unless involved in an event that directs their behavior. States are defined and hashed by name. For overarching world information describing the global state of the simulation, a special world control object with no physical representation can be read and written to as if it were a prop in the world. This is useful for maintaining narrative trajectories and story arcs.

The composite state presented to the planner is an encapsulation of each object's complete low-level state. Props and actors contain a great deal of information concerning their animation and procedural pose controllers, such as where in the world they should be gazing or reaching, or how near their current gesture animation is to completion, but this information is managed at the event level rather than at the planning level. A behavior author is responsible for placing in the proper connections so that when a door plays its closing animation, the corresponding state in its high-level description is also set to `Closed`. Additionally, entire objects may not be accounted for in the composite state space of the planner. Independent actors out of the planner's control may serve as confounding agents by interacting with objects tracked by the planner and altering their state in unforseen ways. The most obvious example of such an agent would be a human user, but virtual actors acting with their own autonomy may also fill this role for various purposes. Because these confounding agents may alter states critical to the progress of a plan, the planner is capable of monitoring required conditions and replanning when necessary.

Some example high-level states for objects in the world are as follows:

- Door: [`Open`, `Locked`, `Guarded`]
- Character: [`HasKey`]
- Guard (extends from Character): [`Trapped`, `Dazed`]

3.2 Action Space

Our planner does not operate in the space of each actor's individual capabilities. Rather, our system's set of actions is taken from a dictionary of *events*, authored parameterizable interactions between groups of characters and props. Because events encapsulate mechanical details related to animation, actors can have a rich repertoire of possible actions without making an impact on the planner's computational demands. Two characters involved in an interaction could have dozens of possible actions for displaying diverse, nuanced sets of gesticulations. Where a traditional planner would need to expand all of these possibilities at each step in the search, our planner only needs to invoke the conversation event between the two characters and let the authored intelligence of the event itself dictate which gestures the characters should display and when.

3.2.1 Formulating Narrative Events

Events are pre-authored dynamic behaviors that take as parameters a number of actors or props as participants. When launched, an event suspends the autonomy of any involved object and guides those objects through a series of arbitrarily complex actions and interactions. Events are well-suited for interpersonal activities such as conversations, or larger-scale behaviors such as a crowd gathering in protest. Each event is temporary, possessing a clear beginning and at least one end condition. When an event terminates, autonomy is restored to its participants until they are needed for any other event. Each event is defined as

$$e = \langle t, c, \mathbf{R} = (r_1, \ldots, r_m), \phi : \mathbf{R} \times \mathbf{W}^m \rightarrow \{0, 1\}, \delta : S \rightarrow S' \rangle$$

where the PBT t contains event behavior, c is the event's cost, the role list $\mathbf{R}$ defines the number of objects that can participate in the event and each participant's required role index number, the precondition function ϕ transforms the list of m roles and a selection of m objects from the world into a true or false value, and the postcondition function δ transforms the world state as a result of the event. The precondition function will return true if and only if the given selection of objects satisfies both the roles and authored preconditions of the event. Mechanical details that are encapsulated within an event (such as walking animations, or body pose configuration), also cannot be used as pre- or postconditions.

Roles for an event are based on an object's narrative value. A conversation may take two human actors, whereas an event for an orator giving a speech with an audience might take a speaker, a pedestal object, and numerous audience members. Preconditions and postconditions define the rules of the world. For example, a character can pull a lever only if the lever can be reached from the character's position. Figure 2 illustrates the pre- and postconditions

Event *UnlockDoor*(Prisoner : a, Door: d):
Precondition ϕ**:**
`Closed`(d) ∧ ¬`Guarded`(d)
∧ `Locked`(d) ∧ `CanReach`(a,d);
Postcondition δ**:**
¬`Locked` (d)

Figure 2: *Event definition for UnlockDoor.*

for an *UnlockDoor* event. This event metadata would be accompanied with a PBT that instructs the character to approach the door and play a series of animations illustrating the door being unlocked.

An event's cost is partially authored and partially generated. An author designing an event where a prisoner obtains a key from a guard may specify that sneaking up to the guard and stealing the key may be less costly than assaulting the guard outright to obtain the key. Other costs, such as the distance to a navigation goal where an event takes place, are more mechanical and can be automatically determined by the system at runtime. Since we are planning in a narrative space, it is important to author costs so that the planner picks events that are not necessarily the most efficient, but rather are the most consistent to the story. A more appropriate way to think of an event's cost would be to consider the risk to a viewer's suspension of disbelief, rather than to consider the theoretical energy expended by the participants to accomplish the goal. Currently, there is no single algorithm to determine cost, and so this is left as an authoring task.

3.3 Goal Specification

Goals can be specified as: (1) the desired state of a specific object, (2) the desired state of *some* object, or (3) a certain event that must be executed during the simulation. These individual primitives can be combined to form narrative structures suited for the unfolding dynamic story. When searching through $\mathbf{A}$ for an event sequence that satisfies the narrative, the planner uses preconditions to determine possible events, and postconditions to generate future world states after the execution of one or more events. Requirements on the desired state of *some* object are translated into the composite world space by adding multiple satisfiable conditions to detect an end state. The goal stipulating that an event must occur places a hard constraint on any generated plan, rejecting branches even if they have achieved every other goal with minimal cost.

Goals are a manner of specifying the narrative structure of the virtual world. Much of the emergent story using this system will occur when the planner attempts to achieve a narrative objective despite the efforts of confounding agents in the world (including but not limited to the player). It is important to view goals not as means to solve a task, but as the desired climax or conclusion of a story arc. Sequences of goals could be achieved in order to enforce structure on a story that can deviate from expectation due to user interaction. The selection of goals is currently author-specified, though intelligent dynamic selection of goals would allow more user freedom and is a topic we intend to explore in future work.

4 Planning in Event Space

Currently, it is challenging to automatically compute a cost estimate between arbitrary states in our state space, and a manual definition would significantly add to the authorial burden. As a result, our system uses a generic best-first planner to produce sequences of events that satisfy a given narrative. The core of our planner serves as a baseline for future work in which we can explore different methods of automatically computing heuristics [Kapadia et al. 2011] and subsequently exploit the benefits of state-of-the-art real-time planners [Likhachev et al. 2005].

The set of permissible events at each search node $I \in \mathbf{I}$ determines the transitions in the search graph. A valid instance of an event e is defined as $I_e = \langle e, O \in \mathbf{W}^{|\mathbf{R}_e|} \rangle$ with $\phi_e(\mathbf{R}_e, O) = 1$. This implies that the event's preconditions have been satisfied for a valid set of participants O, mapped to their designated roles $\mathbf{R}_e$. Algorithm 1 describes the procedure for computing the transitions $\mathbf{I}$ from a given state S. Planning terminates when either a goal has been reached, or the process exceeds a timeout period (a few seconds).

Data: world object list $\mathbf{W}$
Data: authored event library $\mathbf{E}$
Result: $\mathbf{I}$
foreach $e = \langle t_e, c_e, \mathbf{R}_e, \phi_e, \delta_e \rangle \in \mathbf{E}$ **do**
  foreach $r \in \mathbf{R}_e$ **do**
    create list ℓ_r ;
    foreach $o = \langle c_o, \langle r_o, p_o, d_o \rangle \rangle \in \mathbf{W}$ **do**
      if $r_o == r$ **then**
        $\ell_r = \ell_r \cup \{o\}$;
      end
    end
  end
  let $n = |\mathbf{R}_e|$;
  let $P = \ell_{\mathbf{R}_e^1} \times \ell_{\mathbf{R}_e^2} \times \ldots \times \ell_{\mathbf{R}_e^n}$;
  foreach $p = (o_1, o_2, \ldots, o_n) \in P$ **do**
    if $\phi_e(\mathbf{R}_e, p) == 1$ **then**
      $\mathbf{I} = \mathbf{I} \cup \{(e, p)\}$;
    end
  end
end

Algorithm 1: *Populating the set of possible event instances* $\mathbf{I}$ *to build the action space for each world state. Note that the role lists* ℓ_r *are reusable and do not need to be rebuilt for each event iteration once constructed.*

The transitions $\mathbf{I}$ are used to produce a set of new states: $\{S' | S_e = \delta(S, I_e) \forall I_e \in \mathbf{I}\}$ by applying the effects of each candidate event to the current state in the search. Cost is calculated based on the event's authored and generated cost, and stored alongside the transition. The planner generates a sequence of event transitions $\Pi(S_{\text{start}}, S_{\text{goal}})$ from the S_{start} to S_{goal} that minimizes the aggregate cost:

$$\Pi(S_{\text{start}}, S_{\text{goal}}) = \underset{\{I_i | 0 \leq i < n\}}{\operatorname{argmin}} \sum_{i=0}^{i<n} c(S_i, I_i)$$

If all of the characters and objects could participate equally in an event, then the branching factor in planning would be $|\mathbf{E}|\binom{|\mathbf{W}|}{|\mathbf{R}|}$, and grow prohibitively large. In order to curb the combinatorial cost of this search process, we divide actors and objects into role groups. Though there may be many objects in the world, very few of them will fill any particular role, which greatly reduces the size of the event instance list. Checking an object's role is a fast operation which we use for filtering before the more costly operation to validate the event's preconditions on its candidates. The maximum branching factor of this technique will be $|\mathbf{E}|(max_{e \in \mathbf{E}} |\mathbf{R}_e|)^{max_{r \in \mathbf{R}} |\ell_r|}$, and the effective branching factor can be calculated by taking average instead of maximal values. In other words, the branching factor is bounded by the number of events, the maximum number of participants in any event, and the size of the role group with the most members.

Role groups mitigate the effect of the total number of objects in the world from on the growth of the branching factor as long as new objects are evenly divided into small groups. This increases efficiency and allows the system to scale to large groups of actors and objects at the cost of flexibility (all of the objects in a role group may be busy when needed) and authoring burden (the author must ensure that objects are divided properly). Note that the branching factor also grows independently of an actor or object's individual capabilities, allowing characters to exhibit rich and varied behaviors without affecting the planner's overhead. Reducing the branching factor is the subject of future work, including a metric called *Salience* that filters participants based on their history of involvement in past events [Shoulson et al. 2013].

5 Runtime and Simulation

While the planner is responsible for generating sequences of events to produce a narrative, a second component is needed to instantiate those events in a proper order and manage their execution. To do so, we use an event scheduler that monitors a queue of pending events and a list of currently running event behavior trees. The scheduler is responsible for selecting which events to instantiate in order, and for monitoring the world state to detect when preconditions of an event in the plan have been invalidated, necessitating a replan.

5.1 Event Loading and Dispatch

Unlike a contained sequence of actions in a single problem domain, our system presents a fully-fledged simulation with many independent actors and moving components. While the story plan itself is a strict sequence, we use a specialized event scheduler that enables us to dispatch multiple events simultaneously if they are not in competition for resources. This makes the simulation appear more lifelike by maximizing actor activity and preventing long periods of time where characters idly wait for the next step in the plan. Multiple discrete "threads" of a story can progress simultaneously in different areas of the world to present a compelling narrative that is driven by the démarches of its characters.

The product of a generated plan is a sequence of events and lists of objects from the world that fill the parameters of those events. Despite the planner's ordering, events using disjoint sets of resources can be safely executed in parallel. Upon receiving a plan, our scheduler queues the plan's event sequence and prepares to dispatch those events to the planner's selected participants. At each scheduler update, the scheduler updates all of the currently running events, and then iterates through the list of queued events to launch those events for which their participants are all available and their preconditions are satisfied. Multiple events can be run in parallel if their lists of participating objects do not overlap. Scheduler updates occur at a fixed frequency and also handle tasks such as cleaning up completed events and restoring autonomy to their participating actors.

In order to maximize the amount of synchronization that can occur, authors of events can wrap complex sequences of actions into *compound events*. Compound events appear to the planner as one large event that takes a group of participating objects. However, when placed on the scheduler's event queue, the compound structure is broken down into a number of component sub-events that each take fewer parameters than the whole. As an example, in a simplified version of our scenario in which three characters escape from a prison, we illustrate a room with a simple puzzle in Figure 3. The three main characters enter the room from the left and must exit on the right. Buttons B1 and B2 open the door at the bottom of the diagram, and must be pushed simultaneously, while button B3 opens the door at the top of the diagram. The pathway to the room with buttons B1 and B2 are blocked by a guard G. The event(s) to solve this could be authored in a number of ways, including one large event that takes all seven parameters and walks the characters through the problem.

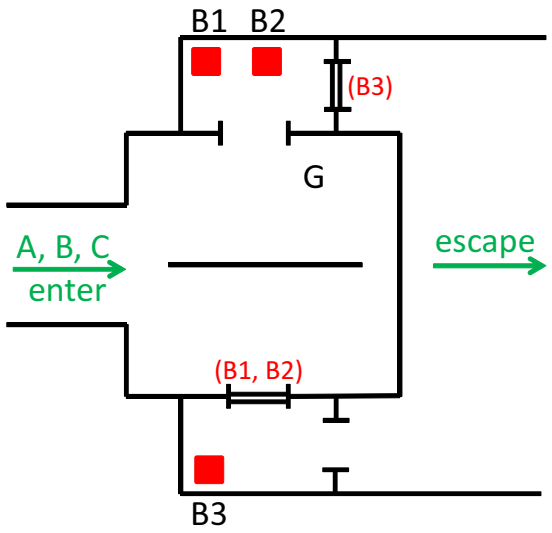


Figure 3: *Map of the environment for the distract-and-escape scenario.*

Regardless of how the solution appears to the planner, however, the events can be broken down for parallel execution. Figure 4 illustrates the resources used for the various sub-events to solve the puzzle. While the events in the sequence may still be predicated on state transitions, they may still partially overlap. As character A distracts the guard, characters B and C can press the buttons to open the bottom door. Then A can escape from the guard, press B3, and all three characters can escape simultaneously. Compound events are another tool available to authors to control the branching factor of the planning process, allowing the planner to only see a very coarse-grained event such as "escape from room", without sacrificing qualities like synchronized cooperative behaviors.

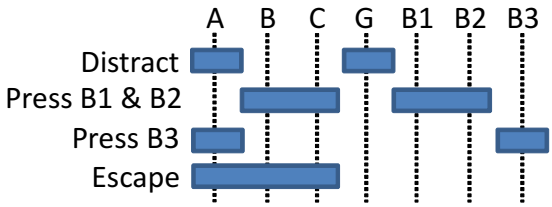


Figure 4: *Synchronization breakdown of the distract-and-escape scenario.*

5.2 Handling Dynamic World Changes

The scheduler receives a notification when any external object in the world, including a user's avatar, has unexpectedly changed the state of the environment. If the change affects an entry in the world state used by any currently-running events' pre- or postconditions, then those events are immediately terminated, the event queue is cleared, and the planner is instructed to replan from the current affected world state. If the change does not alter any of the current events' preconditions or postconditions, then the scheduler forward-simulates the events in the queue and ensures that given

each sequential world state, the next event in the queue can be reached. Note that we check postconditions to see whether user intervention has achieved the effect of an event before the event itself could terminate (for example, if the user moves an item that an actor in an event was supposed to move). If any event in the queue is detected as unreachable due to the unexpected change, then the queue is emptied and the planner generates a new sequence of events. This allows the planner to respond to changes from either a user or adversarial virtual characters acting in opposition to the goals of the story. Further work on plan and story repair [Likhachev et al. 2005; Paul et al. 2011] may be suitable for our planning environment, but would be the subject of future work.

5.3 Intelligent Ambient Character Behavior

In virtual environments with many actors, we do not expect every actor to constantly be involved in events as the story progresses. At times, characters will be waiting for others to perform tasks and advance the narrative sequence. This can produce periods of downtime where actors are not explicitly given a task to perform by the story planner. Fortunately, the event-centric behavior paradigm allows for characters to maintain their own autonomy when not involved in an event. We extend this functionality so that characters not only maintain a level of activity when waiting for their next story action, but act with an awareness of their current role in the narrative.[1]

Objects in our world are annotated with special *observation points*, which serve as placeholder geometry where a character can safely stand and watch that particular object while it is in use. When a character is idle, we determine which event from the scheduler that character will be involved in next. Given the event, we pick one of the non-actor objects scheduled to be involved in that event, and direct the character to stand in one of that object's observation points as long as the character can safely do so. This produces realistic ancitipatory behavior such as a character standing by a door while waiting for another character to unlock and open it. The characters can then be co-opted at any time once their event starts to continue advancing the narrative.

6 Results

We author a narrative-driven "prison break" scenario, where prisoner characters need to overcome a number of hurdles including locked doors, traps, alarms, and adversarial guards in order to exit their cells. This simulation runs at or above 30 frames per second, with near-instaneous replanning to react to user interaction. Section 6.1 discusses the virtual environment we designed for the scenario, Section 6.2 describes the objects within the virtual world and their state descriptions, Section 6.3 describes the process of authoring multi-actor events, Section 6.4 describes the generated narratives, and Section 6.5 discusses our system's ability to react in real-time to both helpful and confounding user intervention.

6.1 Environment Design

Figure 5 displays a map of the test environment we designed for demonstrating our system. The goal is for all of the prisoners to safely escape their cells and flee the prison. In order to accomplish this goal, the prisoners must evade guards and open doors either using keys (for locked doors) or buttons (for their controlled doors, indicated by matching color).

[1]This can also strongly influence camera control, as described by Markowitz et al. [2011].

The map is annotated with the following points of interest (red circles): (1) the north cellblock contains three prisoners in cells, one guard with a key, and a button that opens two doors in the south cellblock (orange doors), (2) the south cellblock contains a sleeping guard with a key and four locked prisoners, (3) the exit to the cellblock is controlled by two buttons on the interior that must be pressed simultaneously to open (dark blue) or one button on the exterior, (4) the exit from the north cellblock is blocked by another guard, and requires three doors to be opened in sequence using a key and then two buttons (light blue, purple), (5) a hiding spot for prisoners to evade the guards in the north cellblock, (6) a cage that rises from the floor and can trap prisoners or guards caught inside, (7) a control room with buttons controlling the cage (red) or the alarm at the exit (green) that will attract guards to it when activated, and (8) the goal zone, representing the exit from the prison.

6.2 Object State Description

Our world state comprises instances of eight object types, each with their own individual state.

Object	State Fields
Door	`Closed`, `Locked`, `Guarded`, `Room1`, `Room2`
Prisoner	`HasKey`, `CurrentRoom`, `Position`
Room	`Guarded`, `AdjacentRooms`
CageTrap	`Active`
Button	`Active`, `ControlledObject`
Guard	`HasKey`, `Trapped`, `Dazed`, `Position`
Alarm	`Active`
Key	(no state information)

The state of an object determines the conditions for its use. For example, a door with a guard standing near it is considered guarded, and cannot be safely opened or unlocked without first removing or disabling the guard. The same applies for guarded rooms, which cannot be safely entered. Buttons are linked to the objects they activate, controlling doors, the cage trap, or the alarm. Guards are unable to move or react when dazed or trapped, and locked doors can only be opened by characters that have a key. For special cases like a door that can only be opened by two simultaneous button presses, we use proxy objects that comprise the state of two or more objects in the world.

6.3 Authored Events

Table 1 illustrates the major events used in our planning scenario, with their parameters, preconditions, postconditions, and a summary of their behavior trees. Pre- and postconditions involving character positions are omitted for clarity, but are handled, in general, with a system of room adjacency and world annotations in the form of waypoints. These events were authored as generic reusable structures and placed in a library available to the planner. The process of introducing new behavior is straightforward, and can simply be accomplished by adding a new event that manipulates the objects in the world. By using events, we were able to add a rich variety of behavior to the world with a minimal degree of authorial burden. These events, such as the six-actor event to trap four guards in a cage, allow for rich multi-character interactions without causing a combinatorial growth in the planning branching factor.

6.4 Generated Narrative

In a supplemental video, we demonstrate four different generated narrative plans. For each of these plans, the goal of the simulation was for each prisoner to be able to reach the exit location of the prison complex map. The initial configuration of the world matches

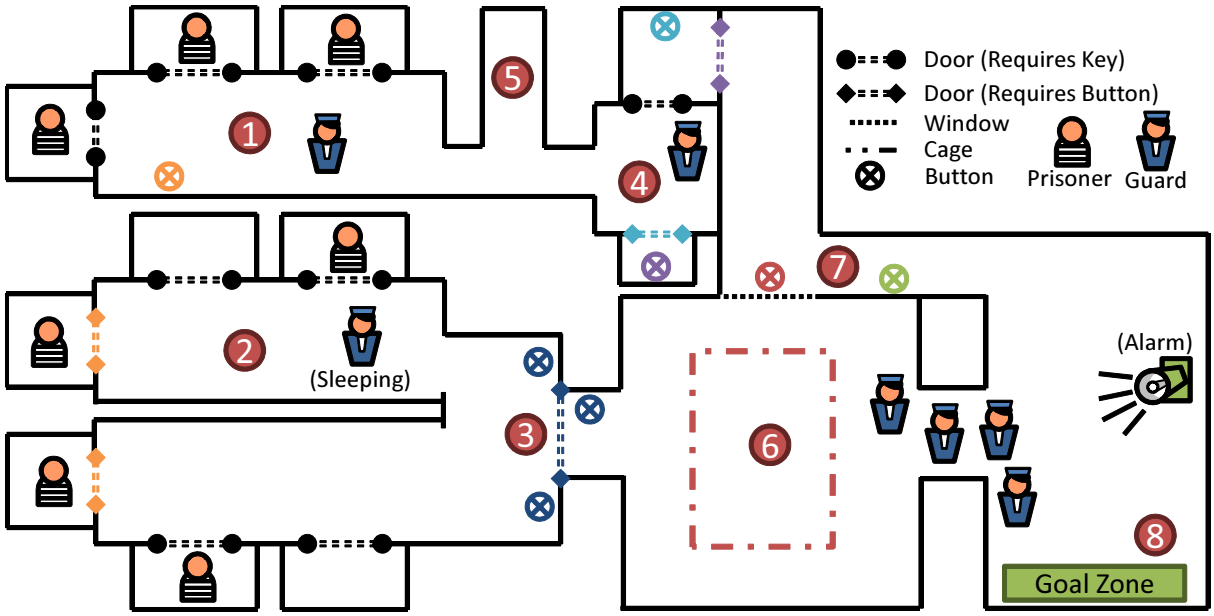


Figure 5: *Problem definition map for the "prison break" scenario. Prisoners must escape from their cells and escape from the complex by unlocking doors and evading guards.*

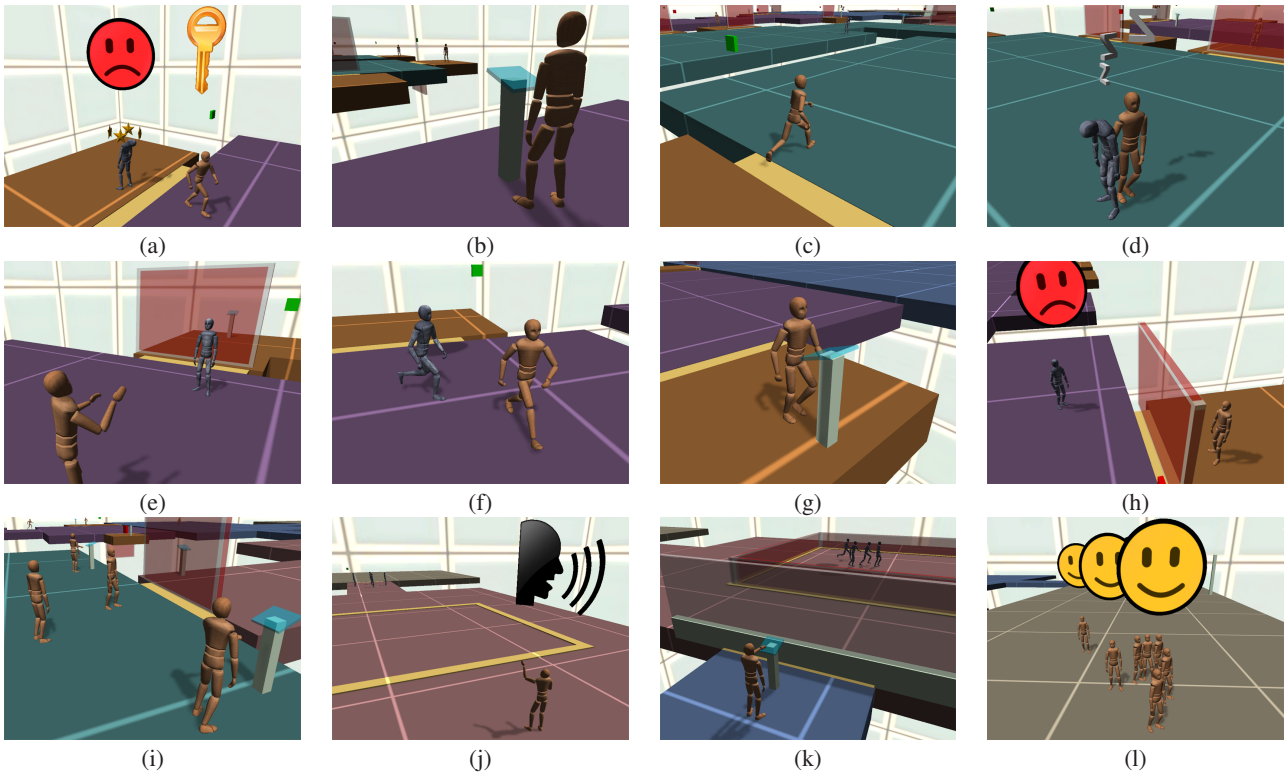


Figure 6: *Storyboard for the first generated narrative plan with no intervention.*

that of Figure 5, with prisoners in their cells, and guards with keys guarding the prison exits.

Figure 6 provides a storyboard of the first generated narrative plan from the default environment. The plan begins with a prisoner in the north cellblock (area (1) in Figure 5) luring the guard into his cell, assaulting him, and stealing his key (a). Afterwards, that prisoner frees the others in the north cellblock, and presses the button (b) to release the south cellblock (orange button, Figure 5). Two prisoners are released (c) in the south cellblock, and one steals a key from the sleeping guard (d). Meanwhile, in the north cellblock, a prisoner lures a guard away from the door (e, f) so that two other prisoners

Event *DistractGuard*(Guard : g; Door : d;
Prisoner : a, b, c; Waypoint : u, v):
Precondition ϕ:
Guarded(d) ∧ CanReach(a,g)
∧ CanReach(b,u) ∧ CanReach(c,v);
Postcondition δ:
¬Guarded (d);
Behavior Summmary:
b hides at u
c hides at v
a draws away g

Event *EscapeCell*(Guard : g; Door : d;
Prisoner : a):
Precondition ϕ:
CanReach(g,d) ∧ HasKey(g)
∧ Closed(d) ∧ Locked(d);
Postcondition δ:
HasKey(a) ∧ ¬HasKey(g)
∧ Trapped(g) ∧ Locked(d);
Behavior Summmary:
a calls g
OpenDoor(g,d)
a dazes g
StealKey(g,a)
a closes and locks d

Event *SoundAlarm*(Guard : g, h, i, j;
Alarm : a; Button : b; Prisoner : p):
Precondition ϕ:
CanReach(p,b) ∧ Controls(b,a)
∧ ¬Active(a);
Postcondition δ:
Active(a);
Behavior Summmary:
p presses b
g, h, i, j approach a

Event *StealKey*(Guard : g; Prisoner : a):
Precondition ϕ:
CanReach(a,g) ∧ HasKey(g)
∧ IsDazed(g) ∧ ¬HasKey(a);
Postcondition δ:
HasKey(a) ∧ ¬HasKey(g);
Behavior Summmary:
a approaches g
a reaches g to take key

Event *OpenDoor*(Agent : a; Door : d):
Precondition ϕ:
CanReach(a,d) ∧ Closed(d)
∧ ¬Locked(d);
Postcondition δ:
¬Closed(d);
Behavior Summmary:
a approaches d
a opens d

Event *ExchangeKey*(Prisoner : a, b):
Precondition ϕ:
CanReach(a,b) ∧ HasKey(a)
∧ ¬HasKey(b);
Postcondition δ:
HasKey(b) ∧ ¬HasKey(a);
Behavior Summmary:
a approaches b
a gives b the key

Event *TrapGuards*(Guard : g, h, i, j;
Prisoner : a, b; Button : u; CageTrap : t;
Waypoint : w):
Precondition ϕ:
CanReach(a,w) ∧ CanReach(b,u)
∧ OnPathTo((g..j),w,t) ∧ ¬Active(t)
∧ Controls(u,t);
Postcondition δ:
Trapped(g) ∧ Trapped(h)
∧ Trapped(i) ∧ Trapped(j)
∧ Active(t);
Behavior Summmary:
a approaches w
a calls out to g, h, i, j
g, h, i, j approach a
b presses u
t traps g, h, i, j

Event *TrapGuardsAlarm*(Guard : g, h, i, j;
Prisoner : a, b; Button : u, v;
CageTrap : t; Alarm : x):
Precondition ϕ:
CanReach(a,u) ∧ OnPathTo((g..j),x,t)
∧ ¬Active(t) ∧ Controls(u,t)
∧ CanReach(b,v) ∧ Controls(v,x)
∧ ¬Active(x);
Postcondition δ:
Trapped(g) ∧ Trapped(h)
∧ Trapped(i) ∧ Trapped(j)
∧ Active(x) ∧ Active(t)
Behavior Summmary:
SoundAlarm(g,h,i,j,x,v,a)
TrapGuards(g,h,i,j,a,b,t)

Table 1: *Summary of major events used in our narrative scenarios. Note that events may invoke other events during their execution.*

can escape by pressing the correct buttons (g) (area (4) in Figure 5). The luring prisoners enter the guarded area (above (4) in Figure 5) and shut the door, locking the guard behind them (h). In the south cellblock, two prisoners simultaneously press the door buttons (i) to open the door (dark blue, area (3) in Figure 5). A prisoner runs out into the cage room (area (6) in Figure 5), and attracts the guards to him (j). Another prisoner presses the button to trap the guards (k). Afterwards, all seven prisoners escape (l). Figure 1 highlights some of the cooperative behaviors performed during the scenario.

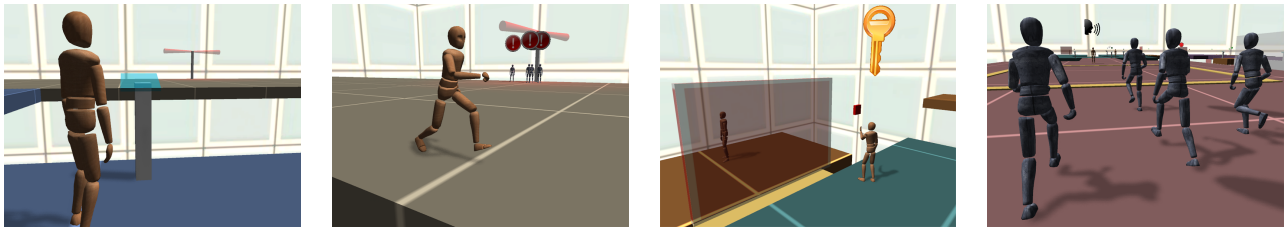
We altered the environment by disabling the button in the north cellblock (near (1) in Figure 5, orange), which prevented the two doors in the south cellblock (orange) from being opened by the north wing prisoners. This one change to the world state produced a drastically different narrative experience.

The second narrative began in the same way as the first, with the north cellblock prisoner stealing a key from a guard and releasing the other two prisoners (Figure 6(a)). However, with the button disabled, the south cellblock prisoners could not be released, so the events featured in Figure 6(c, d, i) do not occur. Instead, an alternative sequence of actions occurs, displayed in Figure 7. To lure the guards away from the gate room exit, a prisoner from the north cellblock activates the alarm (m). This attracts the guards to it so that a prisoner can sneak past them (n) and enter the south cellblock. After stealing a key from the sleeping guard (as in 6(d)), that prisoner releases the other south cellblock prisoners (o), and lures the guards back into the trap after the alarm is deactivated (p). Once again, all seven prisoners safely escape. This demonstrates our system's ability to react and adapt to static changes in the environment by generating alternative narrative plans.

6.5 Reacting to User Intervention

Our system can perform rapid replanning when faced with an invalided plan. We demonstrate two scenarios in which a user interacts with the system while a plan is currently active. In the first scenario, the user manually opens two doors in the north wing of the complex (near (4) in Figure 5, light blue and purple). This prevents the prisoners in the north wing from having to open those doors by pressing their associated buttons. As soon as either door is opened, the planner detects the change, invalidates the active plan, and immediately produces a new plan from the current state of the world. Where the original plan instructed the prisoners to open those doors after distracting the guard blocking them, the new adapted plan allows the north wing prisoners to bypass that step entirely.

In a second scenario, we demonstrate a situation where the user acts in direct opposition to the goal, illustrated in Figure 8. While the narrative is taking place, the user moves the four guards (q) from the cage room (near (4) in Figure 5) closer to the door (r) at the exit of the south cellblock (near (3), blue). This prevents the prisoners from using the diversion tactic they employed in the original plans to trap the guards in the cage. To adapt to this change, the planner produces a new narrative where the prisoners use the alarm to lure the guards back out of the cage room (s), and intercept them with the



(m) (n) (o) (p)

Figure 7: *Adaptations to the narrative plan resulting from an environment change.*

(q) (r) (s) (t)

Figure 8: *The user selects and moves four guards (q, r), resulting in a real-time narrative plan adaptation (s, t).*

cage as they pass over it (t). This showcases our system's ability to continue to progress the narrative towards an authored goal despite confounding intervention, intentional or otherwise, by a user.

7 Conclusion and Future Work

This paper describes a framework for creating real-time interactive narratives using the event-centric authoring paradigm. Our method allows us to design complex and intricate multi-character interactions with a high degree of control fidelity in order to synchronize cooperative and competitive character behaviors. Planning in the space of events rather than in individual character action spaces allows characters to exhibit a large repertoire of individual capabilities without causing combinatorial growth in the planner's branching factor. This allows us to produce and simulate long, cohesive narratives at interactive rates.

Our system has a number of limitations:

Required Roles. To contain the branching factor of populating events with characters and props, we are required to divide the environment's populace into many different narrative roles. This limits the number of objects that can be selected for an event, but comes at the cost of reducing the flexibility of the system to adapt to confounding user input. To alleviate these restrictions, we are examining other ways to filter the populace according to suitability for a particular event. This can include criteria such as agent priming, its proximity to the event's location, and previous events in which an agent has participated [Shoulson et al. 2013].

Authoring Burden. Currently, events must be manually authored with their behavior, preconditions, postconditions, and cost. While feasible for a small number of events, this can quickly become prohibitive for larger, more interesting virtual worlds. Complex events can also grow unpredictable in nature, making it difficult to manually extrapolate their pre- and postconditions. Despite this burden, however, the freedom and flexibility of an event-driven system allows us to automatically generate extremely diverse narratives that are free-form and responsive to user intervention. While authoring events can be manually intensive, it alleviates the need for an author to manually design the entire narrative, which severely limits user agency. To counteract the authoring burden for event design, we are actively exploring ways to partially automate the event authoring process. This revolves around a system that understands the nature of certain objects in the world, and the ramifications of interacting with them [Shoulson et al. 2013].

Non-determinism. Our current event specification depends on events acting deterministically within the world. This poses a significant limitation for authors both in designing more interesting events, and in handling errors that result from occasionally unreliable systems like procedural reaching and path-finding. Modeling events as nondeterministic functions with multiple probable outcomes would allow our planner to more robustly account for chaotic environments, system error, and unexpected user input.

Acknowledgements

The research reported in this document/presentation was performed in connection with Contract Number W911NF-10-2-0016 with the U.S. Army Research Laboratory. The views and conclusions contained in this document/presentation are those of the authors and should not be interpreted as presenting the official policies or position, either expressed or implied, of the U.S. Army Research Laboratory, or the U.S. Government unless so designated by other authorized documents. Citation of manufacturers or trade names does not constitute an official endorsement or approval of the use thereof. The U.S. Government is authorized to reproduce and distribute reprints for Government purposes notwithstanding any copyright notation heron.

References

ABACI, T., AND THALMANN, D. 2005. Planning with smart objects. In *Computer Graphics, Visualization and Computer Vision: WSCG*, University of West Bohemia, 25–28.

BLUMBERG, B. M. 1997. *Old tricks, new dogs : ethology and interactive creatures*. PhD thesis, Massachusetts Institute of Technology.

BOTEA, A., ENZENBERGER, M., MLLER, M., AND SCHAEFFER, J. 2005. Macro-ff: Improving ai planning with automatically learned macro-operators. *Journal of Artificial Intelligence Research 24*, 581–621.

EROL, K., HENDLER, J., AND NAU, D. S. 1994. Htn planning: Complexity and expressivity. In *AAAI*, AAAI Press, 1123–1128.

FIKES, R. E., AND NILSSON, N. J. 1971. Strips: A new approach to the application of theorem proving to problem solving. Morgan Kaufmann, IJCAI, 608–620.

FUNGE, J., TU, X., AND TERZOPOULOS, D. 1999. Cognitive modeling: knowledge, reasoning and planning for intelligent characters. In *SIGGRAPH*, ACM Press/Addison-Wesley, 29–38.

HELBING, D., AND MOLNÁR, P. 1995. Social force model for pedestrian dynamics. *Phys. Rev. E 51* (May), 4282–4286.

KAPADIA, M., SINGH, S., REINMAN, G., AND FALOUTSOS, P. 2011. A behavior-authoring framework for multiactor simulations. *Computer Graphics and Applications 31*, 6, 45–55.

LI, B., AND RIEDL, M. 2011. Creating customized game experiences by leveraging human creative effort: A planning approach. In *Agents for Games and Simulations II*, F. Dignum, Ed., vol. 6525 of *LNCS*. Springer, 99–116.

LIKHACHEV, M., FERGUSON, D. I., GORDON, G. J., STENTZ, A., AND THRUN, S. 2005. Anytime Dynamic A*: An Anytime, Replanning Algorithm. In *ICAPS*, 262–271.

LOYALL, A. B. 1997. *Believable Agents: Building Interactive Personalities*. Ph.d. thesis, Carnegie Mellon University.

MAGERKO, B., LAIRD, J. E., ASSANIE, M., KERFOOT, A., AND STOKES, D. 2004. Ai characters and directors for interactive computer games. In *Innovative Applications of Artificial Intelligence*, AAAI Press, 877–883.

MARKOWITZ, D., KIDER, J., SHOULSON, A., AND BADLER, N. 2011. Intelligent camera control using behavior trees. In *Motion in Games*. Springer, 156–167.

MASSIVE SOFTWARE INC., 2010. Massive: Simulating life. www.massivesofware.com.

MATEAS, M., AND STERN, A. 2003. Integrating Plot, Character and Natural Language Processing in the Interactive Drama Facade. In *Technologies for Interactive Digital Storytelling and Entertainment*, Fraunhofer IRB Verlag, vol. 9, 139–151.

MCDERMOTT, D., GHALLAB, M., HOWE, A., KNOBLOCK, C., RAM, A., VELOSO, M., WELD, D., AND WILKINS, D. 1998. Pddl - the planning domain definition language. Tech. Rep. TR-98-003, Yale Center for Computational Vision and Control.

MENOU, E. 2001. Real-time character animation using multi-layered scripts and spacetime optimization. In *ICVS*, Springer-Verlag, 135–144.

ONDŘEJ, J., PETTRÉ, J., OLIVIER, A.-H., AND DONIKIAN, S. 2010. A synthetic-vision based steering approach for crowd simulation. *ACM Trans. Graph. 29*, 4 (July), 123:1–123:9.

PAUL, R., CHARLES, D., MCNEILL, M., AND MCSHERRY, D. 2011. Adaptive storytelling and story repair in a dynamic environment. In *ICIDS*, Springer-Verlag, 128–139.

PELECHANO, N., ALLBECK, J. M., AND BADLER, N. I. 2007. Controlling individual agents in high-density crowd simulation. In *Symposium on Computer Animation (SCA)*, Eurographics, 99–108.

PELECHANO, N., ALLBECK, J. M., AND BADLER, N. I. 2008. *Virtual Crowds: Methods, Simulation, and Control*. Synthesis Lectures on Computer Graphics and Animation. Morgan & Claypool Publishers.

PERLIN, K., AND GOLDBERG, A. 1996. Improv: a system for scripting interactive actors in virtual worlds. In *SIGGRAPH*, ACM Press, 205–216.

PORTEOUS, J., CAVAZZA, M., AND CHARLES, F. 2010. Applying planning to interactive storytelling: Narrative control using state constraints. *ACM Trans. Intell. Syst. Technol. 1*, 2 (Dec.), 10:1–10:21.

REYNOLDS, C. 1999. Steering Behaviors for Autonomous Characters. In *Game Developers Conference*.

RIEDL, M. O., AND BULITKO, V. 2013. Interactive narrative: An intelligent systems approach. *AI Magazine 34*, 1, 67–77.

RIEDL, M., SARETTO, C. J., AND YOUNG, R. M. 2003. Managing interaction between users and agents in a multi-agent storytelling environment. In *AAMAS*, ACM Press, 741–748.

RIEDL, M. O., STERN, A., DINI, D. M., ALDERMAN, J. M., AND REY, M. D. 2008. Dynamic Experience Management in Virtual Worlds for Entertainment, Education, and Training. *International Transactions on Systems Science and Applications 3*, 1 (Mar.), 23–42.

SHAO, W., AND TERZOPOULOS, D. 2007. Autonomous pedestrians. *Graph. Models 69* (September), 246–274.

SHOULSON, A., GARCIA, F., JONES, M., MEAD, R., AND BADLER, N. I. 2011. Parameterizing Behavior Trees. In *Motion In Games*, Springer, 144–155.

SHOULSON, A., KAPADIA, M., AND BADLER, N. I. 2013. Paste: A platform for adaptive storytelling with events. In *Intelligent Narrative Technologies 6*, To Appear.

SI, M., MARSELLA, S. C., AND PYNADATH, D. V. 2005. Thespian: An architecture for interactive pedagogical drama. In *Artificial Intelligence in Education*, IOS Press, 595–602.

STOCKER, C., SUN, L., HUANG, P., QIN, W., ALLBECK, J. M., AND BADLER, N. I. 2010. Smart events and primed agents. *Intelligent Virtual Agents 6356*, 15–27.

THUE, D., BULITKO, V., SPETCH, M., AND WASYLISHEN, E. 2007. Interactive storytelling: A player modelling approach. In *AIIDE*, AAAI Press, 43–48.

VAN DEN BERG, J., LIN, M. C., AND MANOCHA, D. 2008. Reciprocal velocity obstacles for real-time multi-agent navigation. In *ICRA*, IEEE, 1928–1935.

VILHJÁLMSSON, H., CANTELMO, N., CASSELL, J., E. CHAFAI, N., KIPP, M., KOPP, S., MANCINI, M., MARSELLA, S., MARSHALL, A. N., PELACHAUD, C., RUTTKAY, Z., THÓRISSON, K. R., WELBERGEN, H., AND WERF, R. J. 2007. The behavior markup language: Recent developments and challenges. In *Intelligent Virtual Agents*, 99–111.

YU, Q., AND TERZOPOULOS, D. 2007. A decision network framework for the behavioral animation of virtual humans. In *Symposium on Computer Animation (SCA)*, Eurographics, 119–128.