Evaluating Accessible Graphical Interfaces for Building Story Worlds

Steven Poulakos[1], Mubbasir Kapadia[2], Guido M. Maiga[3], Fabio Zünd[3], Markus Gross[1,3], and Robert W. Sumner[1,3]

[1] Disney Research, Zurich, Switzerland
[2] Rutgers University, New Jersey, USA
[3] ETH Zurich, Switzerland

Abstract. In order to use computational intelligence to assist in narrative generation, domain knowledge of the story world must be defined, a task which is currently confined to experts. In an effort to democratize story world creation, we present an accessible graphical platform for content creators and end users to create a story world, populate it with smart characters and objects, and define narrative events that can be used to author digital stories. The system supports reuse to reduce the cost of content production and enables specification of semantics to enable computer assisted authoring. Additionally, we introduce an iterative, bi-directional workflow, which bridges the gap between story world building and story authoring. Users seamlessly transition between authoring stories and refining the story world definition to accommodate their current narrative. A user study demonstrates the efficacy of our system to support narrative generation.

Keywords: story world, domain specification, narrative generation

1 Introduction

A prerequisite to using computational intelligence for authoring digital stories [1] is to specify the underlying logical formalisms that describe the problem domain, also referred to as the *story world*. While a lot of work exists in making story authoring accessible to everyone [1], all these contributions assume a story world definition already exists. The story includes annotated semantics that characterize the attributes and relationships of objects and characters in a scene (state), different ways in which they interact (affordances), and how these affordances manipulate their state. The domain knowledge definition entails both the state and action space within a story world. Current languages and interfaces for specifying domain knowledge are confined to experts and the overhead of domain specification is high, often comparable to authoring the story from scratch [2]. This work aims to provide an accessible interface for building story worlds to democratize the use of computational narrative intelligence.

Event-centric authoring [3] encapsulates interactions that have narrative significance as logical constructs and provides an appropriate level of abstraction for authoring and reasoning about stories. This imposes an additional authoring overhead, as these events need to be specified, but mitigates the complexity of automation as it is now independent in number of actors and actor capabilities [2]. Recent research [4] has demonstrated the promise of event-centric representations for defining story worlds, in comparison to agent-centric approaches, and is the choice of domain representation used in this work.

We first describe an end-to-end graphical platform for systematically defining the building blocks of the story world using an intuitive, visual graphical interface. Story world building constitutes creating smart objects, defining their state and actions (or affordances), as well as defining a lexicon of events (multi-actor behaviors of narrative significance) that may take place in this story world. We then demonstrate the efficacy of our system through a user study, which involves the the creation of a Haunted Castle story world and authoring of stories within that world. The evaluation results reveal the potential of our graphical platform as well as the following benefits: bi-directional, iterative story world and story authoring process, reuse to reduce the cost of content creation and specification of event-level conditions to enable computational reasoning and assistance in authoring process, which reduces authoring complexity.

2 Related Work

The research community has addressed the problem of authoring digital narratives in two main ways. Manual approaches provide domain specialists with complete control over creating rich narrative content, while automated approaches rely on computational techniques to generate emergent interactive experiences. We refer the readers to comprehensive surveys on narrative authoring [1, 5].

Manual Authoring. Scripted approaches [6] describe behaviors as predefined sequences of actions. While providing fine-grained control, small changes often require far-reaching modifications of monolithic scripts. Improv [7] and LIVE [8] describe actor behaviors as rules based on certain conditions. These systems produce pre-defined behaviors appropriate for specific situations. However, they are not designed to generate complicated agent interactions with narrative significance. Facade [9] utilizes authored beats to manage the intensity of the story in addition to a generalized scripting language [10, 11] to manually authoring character interactions based on preconditions for successful execution.

Story Graphs can represent branching story lines [12] enabling user interaction as discrete choices at key points in the authored narrative. Behavior Trees (BT's) are applied in the computer gaming industry to design the artificial intelligence logic for non-player characters [13]. BT's enable the authoring of modular and extensible behaviors, which can be extended to control multiple interacting characters [14] and to provide a formalism for specifying narrative events.

Automated Narrative Generation. Domain-independent planners [15] provide a promising direction for automated narrative generation [16], however, at the cost of requiring the specification of domain knowledge. The complexity of authoring is transferred from story specification to domain specification.

For example, domain specification has been demonstrated to enable multi-actor interactions that conform to narrative constraints [17, 18]. However, they cannot be dynamically changed to accommodate user input. Narrative mediation systems [19] automatically generate sub stories that consider the ramifications of possible user interaction. However, these systems produce story graphs with significant branching that are difficult to edit by humans. Virtual directors or drama managers [20] may also accommodate user input while steering agents towards pre-determined narrative goals [21]. Thespian [22] uses social awareness to guide decision-theoretic agents. PaSSAGE [23] estimates a player's ideal experience to guide the player through predefined encounters.

Agent-Centric Domain Specification. Agent-centric approaches [24] build up each character as an individual and explore the space of all possible combinatorial character actions to generate stories. Authoring the characteristics and capabilities of individual characters is decoupled from specifying the story itself, and the complexity of automated narrative generation is combinatorial in the number of characters and the different capabilities of each character.

Event-Centric Domain Specification. Events are a layer of abstraction on top of agent-centric authoring which encapsulate multi-actor interactions that have narrative significance. Event-centric approaches [3, 25, 2] plan in the space of pre-authored narratively significant interactions, thus mitigating the combinatorial explosion of planning in the action space of individual character actions.

Story World Building. Regardless of the mode of story authoring, an underlying representation of the domain in which the story is authored, referred to as the story world, needs to be first specified. This task is traditionally tedious and confined to experts. Wide Ruled [26] offers intuitive graphical interfaces for specifying the problem domain which can be used by narrative generation engines. However, these systems are not compatible with animated stories. Recent work [4] demonstrates the potential of event-centric representations for building story worlds that can be used to author animated stories. However, the task of story world building and story authoring are isolated from each other.

Comparison to Prior Work. Our work complements ongoing research in computational narrative intelligence and advocates for providing accessible metaphors for building story worlds. We build on top of prior work [4] to enhance the features of the system to include affordance and behavior creation. We propose a bi-directional workflow, leveraging reuse and supporting automation, that allows authors to seamlessly transition between story world building and story authoring.

3 Domain Specification for Automated Narrative Generation

In order to use computational intelligence for narrative generation, content creators and story writers need to specify the domain knowledge of the story world, which can be used by an intelligent system for reasoning, inference, and ultimately story generation. This includes annotating semantics that characterize the attributes and relationships of objects and characters in the scene (state), different ways in which they interact (affordances), and how these affordances manipulate their state. Many intelligent systems for automated generation are similar in this regard [1]. However there exists a tradeoff between the complexity of authoring and the computational complexity of generating stories depending on type of domain representation used. Our previous work [4] introduces preliminary concepts for story world building, and we include these concepts here for completeness. Using these building blocks, we describe an event-centric representation of domain knowledge for narrative generation [2].

Preliminaries. We introduce smart objects and affordances as the building blocks for creating story worlds.

Smart Objects. The virtual world $\mathbf{W}$ consists of smart objects [27] with embedded information about how an actor can use the object. We define a smart object $w = \langle \mathbf{F}, s \rangle$ with a set of advertised affordances $f \in \mathbf{F}$ and a state $s = \langle \theta, R \rangle$, which comprises a set of attribute mappings θ, and a collection of pairwise relationships R with all other smart objects in $\mathbf{W}$. An attribute $\theta(i, j)$ is a bit that denotes the value of the j^{th} attribute for w_i. Attributes are used to identify immutable properties of a smart object such as its role (e.g., a button or a person) which never changes, or dynamic properties (e.g., `IsPressed` or `IsStanding`) which may change during the story. A specific relationship R_a is a sparse matrix of $|\mathbf{W}| \times |\mathbf{W}|$, where $R_a(i, j)$ is a bit that denotes the current value of the a^{th} relationship between w_i and w_j. For example, an `IsFriendOf` relationship indicates that w_i is a friend of w_j. Note that relationships may not be symmetric, $R_a(i, j) \neq R_a(j, i) \; \forall \; (i, j) \in |\mathbf{W}| \times |\mathbf{W}|$. The state of each smart object is stored as a bit vector encoding both attributes and relationships.

Affordances. An affordance $f = \langle w_o, \mathbf{w}_u, \boldsymbol{\Phi}, \Omega \rangle$ is an advertised capability offered by a smart object that takes the owner of that affordance w_o and one or more smart object users $\mathbf{w}_u$, and manipulates their states. For example, a smart object such as a ball can advertise a *Throw* affordance, allowing another smart object to throw it. A precondition $\boldsymbol{\Phi} : \mathbf{s}_{\mathbf{w}} \leftarrow \{\texttt{TRUE}, \texttt{FALSE}\}$ is an expression in conjunctive normal form on the compound state $\mathbf{s}_{\mathbf{w}}$ of $\mathbf{w} : \{w_o, \mathbf{w}_u\}$ that checks if f can be executed based on their current states. A precondition is fulfilled by $\mathbf{w}$ if $\boldsymbol{\Phi}_f(\mathbf{w}) = \texttt{TRUE}$. The postcondition $\Omega : \mathbf{s} \rightarrow \mathbf{s}'$ transforms the current state of all participants, $\mathbf{s}$ to $\mathbf{s}'$ by executing the effects of the affordance. When an affordance fails, $\mathbf{s}' = \mathbf{s}$.

Narrative Generation. The aim is to generate a narrative $\Pi(\mathbf{s}_s, \mathbf{s}_g)$, which satisfies an initial state $\mathbf{s}_s$ and through a series of state transitions results in the desired goal state $\mathbf{s}_g$.

Event-centric Domain Knowledge. Event-centric domains introduce events as an additional layer of abstraction. Events are pre-defined context-specific interactions between any number of participating smart objects whose outcome is dictated by the current state of its participants. Events serve as the building blocks for authoring complex narratives. An event is formally defined as $e = \langle t, \mathbf{r}, \boldsymbol{\Phi}, \Omega \rangle$ where t is a logical representation of a coordinated interaction between multiple actors. t takes any number of participating smart objects as parameters where $\mathbf{r} = \{r_i\}$ define the desired roles for each participant. r_i is a logical formula specifying the desired value of the immutable attributes $\theta(\cdot, j)$ for w_j to be considered as a valid candidate for that particular role in the event. A precondition $\mathbf{\Phi} : \mathbf{s_w} \leftarrow \{\texttt{TRUE}, \texttt{FALSE}\}$ is a logical expression on the compound state $\mathbf{s_w}$ of a particular set of smart objects $\mathbf{w} : \{w_1, w_2, \ldots w_{|\mathbf{r}|}\}$ that checks the validity of the states of each smart object. $\mathbf{\Phi}$ is represented as a conjunction of clauses $\phi \in \mathbf{\Phi}$ where each clause ϕ is a literal that specifies the desired attributes of smart objects, and relationships between pairs of participants. A precondition is fulfilled by $\mathbf{w} \subseteq \mathbf{W}$ if $\mathbf{\Phi}_e(\mathbf{w}) = \texttt{TRUE}$. The event postcondition $\Omega : \mathbf{s} \rightarrow \mathbf{s}'$ transforms the current state of all event participants $\mathbf{s}$ to $\mathbf{s}'$ by executing the effects of the event. When an event fails, $\mathbf{s}' = \mathbf{s}$. An event instance $I = \langle e, \mathbf{w} \rangle$ is an event e populated with an ordered list of smart object participants $\mathbf{w}$.

State Space. The overall state of the world $\mathbf{W}$ is defined as the compound state $\mathbf{s} = \{s_1, s_2 \cdots s_{|\mathbf{W}|}\}$ of all smart objects $w \in \mathbf{W}$, which is encoded as a matrix of bit vectors. $\mathbf{s_w}$ denotes the compound state of a set of of smart objects $\mathbf{w} \subseteq \mathbf{W}$. The state space $\mathbb{S}_e$ represents the set of all possible world states $\mathbf{s}$.

Action Space. The event-centric action space $\mathbb{A}_e = \{e_1, e_2 \cdots e_m\}$ is defined as the set of all m events that may occur between any permutation of smart objects in the world $\mathbf{W}$.

Narrative Generation. A narrative $\Pi(\mathbf{s}_s, \mathbf{s}_g) = \langle e_1, e_2 \ldots e_n \rangle$ is defined as a sequence of events that transform the state of the world from its initial state $\mathbf{s}_s$ to the desired goal state $\mathbf{s}_g$ that represents the desired outcome of the narrative. An event-centric representation of domain knowledge helps mitigate the combinatorial complexity of authoring individual characters in complex multi-character interactions and its variants have recently gained prominence in the games industry [2]. While this does impose the additional overhead of authoring events, it offers greater control of narrative progression.

4 A Graphical Platform for Building Story Worlds

Our Story World Builder (SWB) is designed to build up components of a full story world with required semantics to enable computer-assisted narrative generation. The graphical platform is built within the Unity3D game engine. Our demonstration system conforms to the event-centric representations of Shoulson et al. [3] and integrates with the CANVAS story authoring system of Kapadia et al. [2]. CANVAS provides a storyboard-based metaphor for visual story authoring of event and event participants, and it utilizes partial-order planning to enable computer-assisted generation of narratives. The underlying representation of the story world is general and can be easily used within other computer assisted narrative generation systems, such as PDDL.

Story world building begins with the creation of a new story world project and the specification of a scene. The scene specification can involve inclusion of static scene elements, environment lighting and navigation paths in the environment. Building a story world continues with three system components, which are described below: (1) Smart Object Editor: defining a set of smart objects

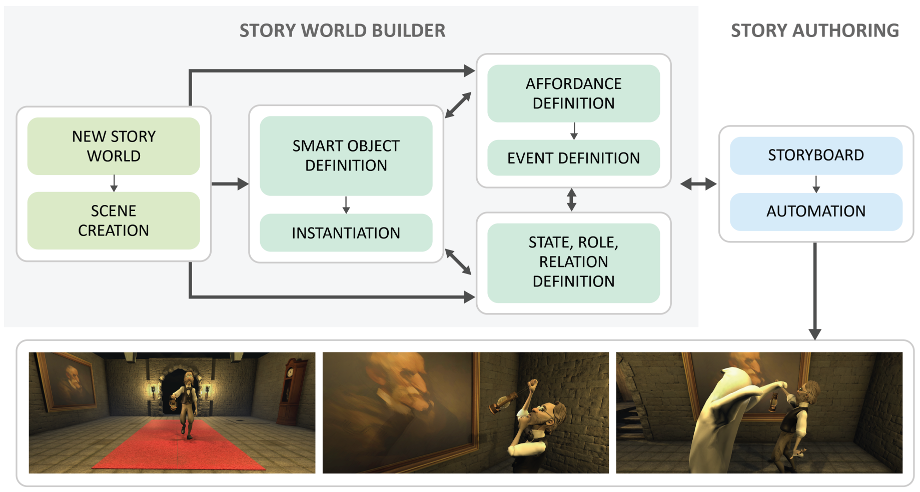


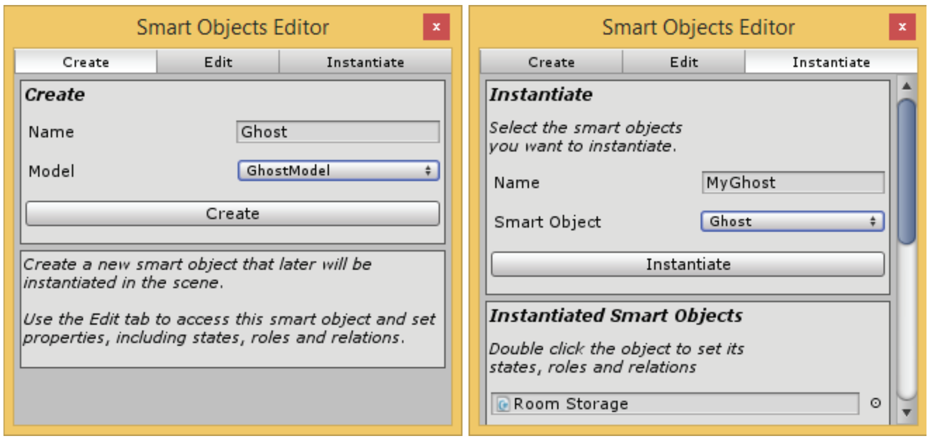
Fig. 1: Overview for building a story world and authoring stories. A scientist enters the haunted castle, investigates a painting and is then spooked by a ghost.

and characters, and instantiating them in the scene. (2) State Editor: defining states, roles and relationships. (3) Affordance and Event Editors: defining capabilities for smart objects and the events which use them. Figure 1 illustrates bi-directional relationship between these components and illustrates an example story that was generated within the Haunted Castle scenario.

Smart Object Editor. Smart objects represent all of the characters and props that influence the story world. In the context of our overall formalism, smart objects have state and offer capabilities to interact with other smart objects and influence the state of the story world. Smart object characters are identified as “smart characters”. We differentiate smart characters because they require additional components to, for example, support inverse kinematics to do complex physical actions, such as pressing a button and grasping objects. All humanoid smart characters can have the same base set of capabilities.

Figure 2 (a) shows the interface for creating a smart character called “Ghost”, which is embodied by a ghost model. Once the smart character is created, the user may then edit properties of the smart object, accessed via the Edit tab. The Instantiate tab, visualized in Figure 2 (b), is then used to create one or more instances of the Ghost smart object within the scene. Each instance has a unique name, for example “MyGhost”. Instantiated smart objects are listed at the bottom of the tab. Additional components are provided to specify other properties, including a representative icon for use in the authoring system.

State, Role and Relation Editor. The State Editor enables the creation of state attributes, roles and relationships available for use with smart objects in the world. States include high level descriptions of objects. For example, the


Smart Objects Editor

Create | Edit | Instantiate

Create

Name: Ghost

Model: GhostModel

Create

Create a new smart object that later will be instantiated in the scene.

Use the Edit tab to access this smart object and set properties, including states, roles and relations.

(a)

Smart Objects Editor

Create | Edit | Instantiate

Instantiate

Select the smart objects you want to instantiate.

Name: MyGhost

Smart Object: Ghost

Instantiate

Instantiated Smart Objects

Double click the object to set its states, roles and relations

Room Storage

(b)

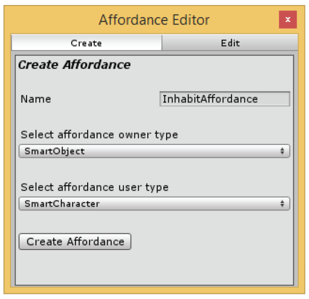
Fig. 2: **Smart Objects Editor.** (a) Create and (b) Instantiate smart objects.

state `IsInhabiting` is true when a ghost is inhabiting another object. Roles may be created to specify that all ghost smart objects have the role `IsGhost`. Relationships existing in the world may also be defined. For example, ghosts may have an `IsAlliedWith` or `IsInhabitedBy` relation. Each smart object provides a component for editing its states, roles and relations.

Affordance and Event Editors. Following the event-centric representation [3], events are defined as Parameterized Behavior Trees (PBT) [28], which provide a graphical, hierarchical representation for specifying complex multi-actor interactions in a modular, extensible fashion. Event creation involves specification of Affordance PBT, Event PBT, and event-based planning.

Affordance PBT. Affordances specify the capabilities of smart objects. An affordance owner offers a capability to the affordance user. Figure 3 (a) demonstrates an example "InhabitAffordance", which is owned by a smart object and used by a smart character. Figure 3 (b) represents an example behavior tree to achieve the affordance. A sequence control node is used to specify that a smart character user (a ghost) will jump into a smart object (the painting) and become invisible. The material property of the smart object will change to reflect that it is now inhabited. Affordances can be much more complex. An advantage of specifying affordances is that they may be reused in many events.

Event PBT. Events utilize the affordances and conditional logic to compose more complex forms of interaction. Figure 3 (c) and (d) present the creation of the InhabitEvent. This simple example uses a sequential control node to specify that a smart character (a ghost) will first approach a smart object (the painting) before inhabiting it. Events may have multiple smart object and character participants, however they must be consistently specified throughout the event behavior tree nodes. We use names of characters from an example scenarios to maintain consistency. Additional components are provided to specify a represen-



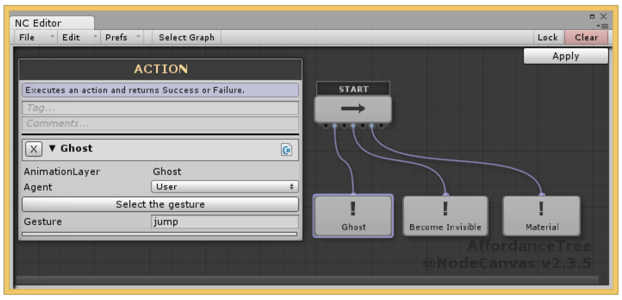
Create Affordance

Name InhabitAffordance

Select affordance owner type
SmartObject

Select affordance user type
SmartCharacter

(a)



Executes an action and returns Success or Failure.

Tag...

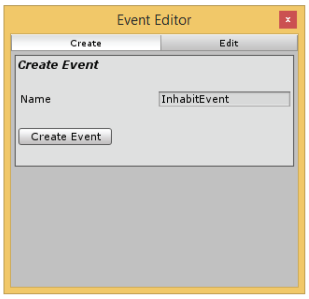
Comments...

▼ Ghost

AnimationLayer Ghost
Agent User

Gesture jump

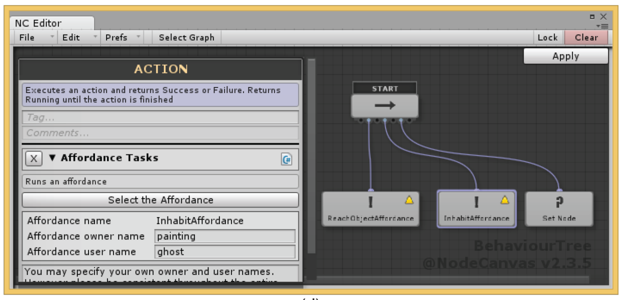
(b)



Create Event

Name InhabitEvent

(c)



Executes an action and returns Success or Failure. Returns Running until the action is finished

Tag...

Comments...

▼ Affordance Tasks

Runs an affordance

Affordance name InhabitAffordance
Affordance owner name painting
Affordance user name ghost

You may specify your own owner and user names.

(d)

Fig. 3: **Affordance and Event Editors**. (a) The Inhabit affordance is created, which is owned by a smart object and used by a smart character to inhabit the smart object. (b) The affordance behavior tree represents the series of actions. (c) An Inhabit event is created. (d) An event behavior tree specifies a sequence of affordances and condition nodes.

tative icon for use in the authoring system. Both events and affordance PBTs may be cloned and modified to support reuse.

Event-based planning. Additional semantics are specified at the event-level to enable reasoning about the logical connectivity of events. Preconditions and postconditions are defined as conjunctive normal form (CNF) expressions on the state and relations of the PBT participants. The event behavior tree in Figure 3 (d) includes a *Set Node*, which specifies a postcondition associated with the event. In our example, the `IsInhabiting` state of the ghost is set to true. In the context of our example, we may required that the smart object has role `IsInhabitable` or we may set an `IsInhabitedBy` relation between the smart object and character. The author may specify pre- and post-conditions within both Affordance and Event PBTs. The SWB will determine which conditions to use to support event-based planning and is compatible with the requirements of our example story authoring system [2].

Coupled Story World Building and Story Authoring. Traditional systems decouple the act of building story worlds and defining narratives, which are often executed by different users (story world builders and story writers). In this paper, we present a system that takes steps towards making story world building accessible to non-experts. However, in a traditional uni-directional workflow, a story world, once finalized, cannot be modified while authoring stories. This introduces certain limitations where users need to forecast all the foundational blocks (smart objects and events) that need to exist in a story world.

To mitigate this, we introduce a bi-directional workflow that couples story world building and story authoring. Our system naturally extends to facilitate seamlessly transitioning between these two acts. This workflow affords several benefits allowing traditional story authors to easily edit and modify existing story world definitions to accommodate new features, and introduce new smart objects or events, that may be necessary to realize their narrative.

5 Evaluation

Method. We conducted a user study to evaluate the usability and observed benefit offered by the Story World Builder (SWB). Eight computer science students participated in the evaluation. All participants had no prior experience with SWB, 63% had prior experience authoring stories with CANVAS and 38% reported prior experience using the Unity3D game engine. The task involved the iterative and bi-directional process of building a Haunted Castle story world with SWB and authoring stories with CANVAS. Text-based instructions provided an overview of the system and specified a list of story world elements to produce. This ensured that all experiment participants created similar set of affordance and events, although some variation was possible in the implementation.

The user study was conducted in four parts. Part 1 involved an introduction video, which demonstrated how to build a painting smart object, a scientist smart character and an experiment event. Test subjects were then provided with a story world containing elements described in the video. Part 1 concluded by guiding the participant to author a story from the story world produced during the video demonstration. Part 2 involved extending that story world. Experiment participants were asked to create an investigate event, which involved reusing a ReachObjectAffordance in addition to creating new affordances. After playing a story involving the new event, participants then created a ghost smart character and a spook event, in which a ghost spooks the scientist. A longer story was then authored. Part 3 involved updating events in the story world to enable planning-based computer-assisted authoring. An IsInhabiting state and IsInhabitable role were created and integrated into the existing current events. The pre- and post-conditions made it possible to author a subset of the events in the previously authored story. The authoring system automatically completed the story using the semantic event-level specification. Part 4 concluded the user study with a questionnaire.

Results. All experiment participants successfully created the intended Haunted Castle story world and authored stories before proceeding to the questionnaire. The questionnaire began with the ten question System Usability Scale (SUS) [29]. The SUS was selected because it is easy to administer and can provide reliable results for small sample sizes. We applied the ten standard questions to

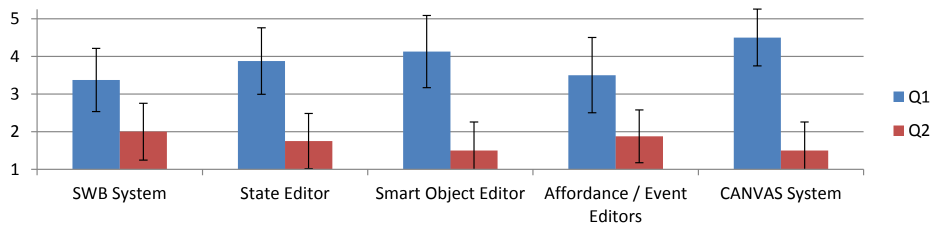


Fig. 4: Subject responses to the following statements: (Q1) "I thought the [fill-in with column label] system was easy to use." (Q2) "I think that I would need the support of a technical person to be able to use this [fill-in with column label]." Likert Scale: Strongly Agree (5) to Strongly Disagree (1). Error bar is 95% confidence interval.

evaluate the SWB system. The resulting SUS score of 66.88 demonstrates that we have a usable system.

Two SUS questions were reused to evaluate subcomponents of the system to edit states, smart objects and events, as well as the story authoring system (CANVAS). One-sample t-Tests were computed on the Likert scale responses to determine if agreement significantly differed from a neutral response. The questions and results are provide in Figure 4. For all systems and subcomponents, participants disagreed that they would require support of a technical person. We interpret this to mean novice users can independently use the system. Participants agreed that the CANVAS System was easy to use and tended to agree that the State Editor and Smart Object Editor were easy to use. The SWB and Affordance/Event Editors had more neutral responses, likely due to the process of specifying affordance and event PBTs, which may be less intuitive and more challenging for users. Further improvements in our system should focus on this aspect of domain specification.

We concluded the questionnaire with the following three questions about the benefits of specific aspects of our system (average Likert response score in parentheses): "I benefited from the ability go back and change the story world (changing state, smart objects, affordances, and/or events) as I was authoring the story."-(score: 4.38) "I benefited from reusing existing contents from the story world."-(score: 4.75) "I benefited from automatically completing stories based on the event conditions provided in the story world."-(score 4.38) Participants observed the greatest benefit in the ability to reuse content, and also agreed that bi-directionality and automation were beneficial.

6 Conclusions

This paper motivates the importance of accessible metaphors for domain specification, as a precursor to computational narrative generation. We describe the representation of domain knowledge, which utilizes an event-centric layer of abstraction. We demonstrate a graphical platform for event-centric authoring of story worlds, which includes specification of smart objects, states, affordances and events. Our system enables a bi-directional, iterative story world and story authoring design process. The principle of reuse is integrated to reduce the cost of content creation. The system streamlines the process of condition specification to enable computational reasoning and assistance in the story authoring process, which importantly reduces authoring complexity. Our user study demonstrates the efficacy of our system, accessible to novice users, to build story worlds.

Limitations and Future Work. The user study has motivated several future improvements. Affordance and Event PBT specification is the most challenging aspect of the system. Users have requested that the system provide more assistance with PBT specification. The formalism for affordance specification is unintuitive in some situations. Users also requested that the story world domain specification be directly edited within the 3D story world. We intend to extend the platform to give more creative freedom to the author and also to assist in the development of interactive narratives [30, 31]. The far-reaching goal of our research is to democratize story world building and digital story generation, by providing accessible metaphors for narrative content creation.

References

1. Mark O. Riedl and Vadim Bulitko. Interactive narrative: An intelligent systems approach. *AI Magazine*, 34(1):67–77, 2013.
2. Mubbasir Kapadia, Seth Frey, Alexander Shoulson, Robert W. Sumner, and Markus Gross. Canvas: Computer-assisted narrative animation synthesis. In *Proc. of SCA '16*, pages 199–209, Aire-la-Ville, Switzerland, 2016. Eurographics Assoc.
3. Alexander Shoulson, Max L. Gilbert, Mubbasir Kapadia, and Norman I. Badler. An event-centric planning approach for dynamic real-time narrative. In *MIG*, 2013.
4. Steven Poulakos, Mubbasir Kapadia, Andrea Schupfer, Fabio Zund, Robert Sumner, and Markus Gross. Towards an accessible interface for story world building. In *Interactive Narrative Technologies 8, AIIDE Technical Report*, 2015.
5. Mubbasir Kapadia, Alexander Shoulson, Funda Durupinar, and NormanI. Badler. Authoring Multi-actor Behaviors in Crowds with Diverse Personalities. In *Modeling, Simulation and Visual Analysis of Crowds*, volume 11, pages 147–180. 2013.
6. Aaron Bryan Loyall. *Believable agents: building interactive personalities.* PhD thesis, Pittsburgh, PA, USA, 1997.
7. Ken Perlin and Athomas Goldberg. Improv: a system for scripting interactive actors in virtual worlds. In *Proceedings of ACM SIGGRAPH*, pages 205–216, New York, NY, USA, 1996. ACM.
8. Eric Menou. Real-time character animation using multi-layered scripts and spacetime optimization. In *ICVS*, pages 135–144, London, UK, 2001. Springer-Verlag.
9. Michael Mateas and Andrew Stern. Integrating plot , character and natural language processing in the interactive drama facade. In *TIDSE*, volume 2. 2003.
10. Michael Mateas and Andrew Stern. A behavior language for story-based believable agents. *IEEE Intelligent Systems*, 17(4):39–47, 2002.
11. Michael Mateas and Andrew Stern. A behavior language: Joint action and behavioral idioms. In *Life-Like Characters*, pages 135–161. Springer, 2004.
12. Andrew Gordon, Michael van Lent, Martin V. Velsen, Paul Carpenter, and Arnav Jhala. Branching Storylines in Virtual Reality Environments for Leadership Development. In *AAAI*, pages 844–851, 2004.

13. Damian Isla. Handling Complexity in the Halo 2 AI. In *Game Developers Conference*, March 2005.
14. A. Shoulson, N. Marshak, M. Kapadia, and N.I. Badler. ADAPT: The Agent Development and Prototyping Testbed. *IEEE TVCG*, 20(7):1035–1047, July 2014.
15. Daniel S. Weld. An Introduction to Least Commitment Planning. *AI Magazine*, 15(4), 1994.
16. R. M. Young. Notes on the use of plan structures in the creation of interactive plot. Technical report, AAAI Press, Cape Cod, MA, 1999.
17. M. Kapadia, S. Singh, G. Reinman, and P. Faloutsos. Multi-actor planning for directable simulations. In *Digital Media and Digital Content Management (DMDCM)*, pages 111–116, May 2011.
18. M. Kapadia, S. Singh, G. Reinman, and P. Faloutsos. A Behavior-Authoring Framework for Multiactor Simulations. *IEEE CGA*, 31(6):45–55, Nov 2011.
19. Mark O Riedl and R Michael Young. From linear story generation to branching story graphs. *IEEE CGA*, 26(3):23–31, 2006.
20. Brian Magerko, John E Laird, Mazin Assanie, Alex Kerfoot, and Devvan Stokes. AI Characters and Directors for Interactive Computer Games. *Artificial Intelligence*, 1001:877–883, 2004.
21. Peter William Weyhrauch. *Guiding interactive drama.* PhD thesis, Pittsburgh, PA, USA, 1997. AAI9802566.
22. Mei Si, Stacy C. Marsella, and David V. Pynadath. Thespian: An architecture for interactive pedagogical drama. In *Proceeding of the 2005 Conference on Artificial Intelligence in Education*, pages 595–602. 2005.
23. David Thue, Vadim Bulitko, Marcia Spetch, and Eric Wasylishen. Interactive storytelling: A player modelling approach. In *AIIDE*, 2007.
24. Robert Mosher and Brian Magerko. Personality Templates and Social Hierarchies Using Stereotypes. In Stefan Göbel, Rainer Malkewitz, and Ido Iurgel, editors, *Technologies for Interactive Digital Storytelling and Entertainment, TIDSE 2006*, pages 207–218. Springer, Berlin, Heidelberg, 2006.
25. Alexander Shoulson, Mubbasir Kapadia, and Norman Badler. PAStE: A Platform for Adaptive Storytelling with Events. In *INT VI, AIIDE Workshop*, 2013.
26. James Skorupski, Lakshmi Jayapalan, Sheena Marquez, and Michael Mateas. Wide ruled: A friendly interface to author-goal based story generation. In Marc Cavazza and Stéphane Donikian, editors, *Proceedings of IVCS'07*, pages 26–37, Berlin, Heidelberg, 2007. Springer-Verlag.
27. Marcelo Kallmann and Daniel Thalmann. Direct 3d interaction with smart objects. In *Proceedings of the ACM Symposium on Virtual Reality Software and Technology*, VRST '99, pages 124–130, New York, NY, USA, 1999. ACM.
28. Alexander Shoulson, Francisco M. Garcia, Matthew Jones, Robert Mead, and Norman I. Badler. Parameterizing behavior trees. In *MIG*, pages 144–155. Springer-Verlag, 2011.
29. J. Brooke. SUS: A quick and dirty usability scale. In *Usability evaluation in industry.* Taylor and Francis, London, 1996.
30. Mubbasir Kapadia, Jessica Falk, Fabio Zünd, Marcel Marti, Robert W. Sumner, and Markus Gross. Computer-assisted authoring of interactive narratives. In *Proceedings of the 19th Symposium on Interactive 3D Graphics and Games*, i3D '15, pages 85–92, New York, NY, USA, 2015. ACM.
31. Mubbasir Kapadia, Fabio Zünd, Jessica Falk, Marcel Marti, Robert W. Sumner, and Markus Gross. Evaluating the authoring complexity of interactive narratives with interactive behaviour trees. In *Foundations of Digital Games*, FDG'15, 2015.