Camelot: A Modular Customizable Sandbox for Visualizing Interactive Narratives

Alireza Shirvani, Stephen G. Ware
Narrative Intelligence Lab, University of Kentucky
Lexington, KY 40506
{ashirvani, sgware}@uky.edu

Abstract

Camelot is a modular customizable virtual environment that is inspired by the needs of current and previous narrative generation research. Camelot is meant to facilitate interactive narrative prototyping, controlled comparisons of different systems, and reproducing and building on the works of others. It provides a 3D presentation layer that is fully separable from the narrative generation system that controls it. This allows any application, AI algorithm, or technology, written in any programming language, to connect and use Camelot to visualize their interactive narratives. In this paper, we introduce Camelot and its capabilities, and provide some details on how and to what extent it can be used to benefit the interactive narrative community.

1 Introduction

Camelot is a modular and customizable interactive narrative environment that provides a sandbox to act as a presentation layer for any narrative generation system. Camelot is a real-time 3D third-person virtual environment that takes place in a Medieval fantasy setting and includes customizable characters, places, and items. By using this environment, researchers can build and test prototypes faster and easier.

By providing a fully separate presentation layer, Camelot is independent of the programming language or technology used by the narrative generation system. This separation of concerns lets Camelot provide a standard of presentation that can be shared among the interactive narrative community. Through this standard, highly different AI approaches can be meaningfully compared to one another and evaluated in the same context and with the same subjects. Moreover, this standard can facilitate researchers to reproduce and build upon the works of others.

In this paper, we provide some details about the accessibility and capabilities of Camelot. We hope that it may reach and assist many researchers in their efforts to contribute to the interactive narrative and AI community. In section 2, we will discuss the design of Camelot to support research and simplify its application. Section 3 presents the potential applications of Camelot and several proof of concept games that are free to access and play. Section 4 discusses our previous attempts and future plans for community outreach, and finally, section 5 presents the conclusions.

2 Design to Support Research

2.1 Interoperability

To generate an interactive narrative, Camelot communicates with an *experience manager* (EM) (Riedl and Bulitko 2013). Experience managers, sometimes called drama managers, emerged early in interactive narrative research (Bates 1992; Weyhrauch 1997) and continue to be a popular architecture (see Roberts and Isbell (2008) for a survey). In contrast to some previous narrative control systems, such as *Mimesis* (Young 2001) or *Zócalo* (Young et al. 2011), Camelot provides both the presentation layer and the bridge that connects it to an EM.

A Camelot EM can be written in any programming language that has standard input and output capabilities. In fact, all communications between Camelot and the EM are transmitted via the standard I/O, e.g. `System.out.Println` in Java, `print` in Python, or `Console.WriteLine` in C#. Camelot has a large list of available commands that can be used to control its UI, characters, environments, etc. These commands are referred to as actions and have the following format[1]:

ActionName(Argument1, Argument2, ...)

e.g. Attack(Hero, Villain)

Sit(Tom, Room.Chair)

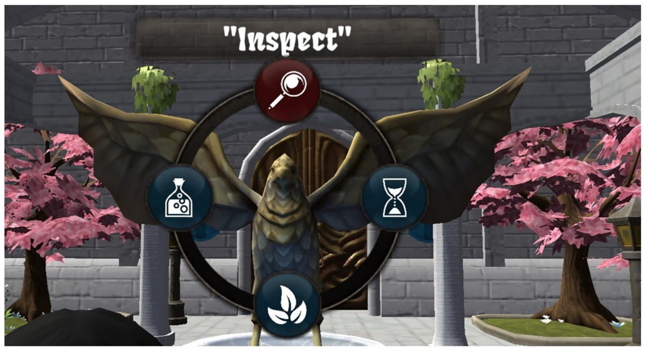
PlaySound(LivelyMusic)

Managing Sequences of Actions

To execute an action, an EM can append *start* to a command and sends it to Camelot. Camelot then attempts to execute that command and responds with the same command with a

[1]For a complete list of actions, the description of their function, and the details of their arguments, please refer to the actions page of the documentation website (link provided at the end of the paper).

Figure 1: The radial menu shows a list of enabled affordances enabled on an object (the eagle statue). The icon, title, and name of each affordance is specified by the EM.



succeeded prefix, when the execution is successful, or otherwise with an *error* or *failed* prefix. The response message starts with *error* if the action could not be started, due to, for instance, insufficient or incorrect arguments, or targeting characters, items, or places that were not instantiated beforehand. The response message starts with *failed* if the action execution fails after it was started, e.g. characters trying to walk out of a locked prison cell, player character walking interrupted by user input. Whether an action fails with an *error* or *failed* message, a short message is also appended that describes the reason for the failure. The EM can use these responses to properly sequence the commands its wants to visualize.

An EM can also append *stop* to a previously started command to stop its execution. In that case, Camelot responds with a *failed* message and does it best to revert back any changes made by the execution of that command. For instance, if a character is in the process of exiting a door after opening it, the door is closed as a consequence of stopping the *Exit* command.

Action Abstraction Levels
Many Camelot actions are comprised of smaller units that are managed by Camelot without concerning the EM. For instance, when the EM calls the *Exit* command, Camelot makes the specified character walk to the specified door, open the door, and go through the door. Camelot then closes the door and makes the screen fade out. In doing so, Camelot does not burden the EM with small units of work that can be combined into a single action. In this case, walking to, opening, and closing a door, as well as having the screen fade out are all also available to the EM to execute individually.

Furthermore, the EM is free to define any level of action abstraction by managing the execution of a sequence of Camelot commands. For instance, in an EM, we can define a *Shop* function that when called, runs a sequence of Camelot commands that make a character walk to a merchant and take an item from them.

Asynchronous Execution
Camelot manages simultaneous actions that use the same assets. Camelot locks characters, furniture, and items when an action starts using them. All other starting actions that target those objects need to wait for the release of that lock. For instance, assume that an EM simultaneously asks both Tom and Jane to go to a merchant to take an item by calling *Take(Tom, Item, Merchant)* and *Take(Jane, Item, Merchant)*. If Tom reaches the merchant first, they start taking the item, while Jane walks to the merchant and waits. At this point, the EM could, for instance, decide to stop the command *Take(Jane, Item, Merchant)* upon receiving *started Take(Tom, Item, Merchant)*. Otherwise, when *Take(Tom, Item, Merchant)* succeeds, *Take(Jane, Item, Merchant)* resumes and since items cannot be in two places at once, the item disappears from Tom's hand and is placed in Jane's.

User Input
When the player interacts with the environment, Camelot sends messages with an *input* prefix to notify the EM. These commands include any interactions with the objects or non-player characters in the environment, dialog choices, keyboard inputs, or specific changes in the position of the player character.

e.g. *input arrived Hero position Castle.Door*

input Draw Hero Sword

input Key Inventory

Since the EM is fully separate from Camelot, it is not dependent on any specific algorithms or technologies, and it does not even have to be deployed on the same physical machine (following the architecture used by Young's *Mimesis* (2001), and many other similar systems). The only connection between Camelot and an EM are the simple strings of texts that are human readable and easy to understand.

2.2 Camelot Gameplay Logs

While a user is playing an interactive narrative, Camelot generates a list of all its communication messages with the EM, as well as their time stamps. Camelot gameplay logs capture all the events that occur during a playthrough via the user input or the EM, including when actions start, succeed, or fail. These files can then be used to almost accurately reproduce and analyze a user's playthrough and the story that unfolds based on their choices. A log file's information and file size make it very efficient to transfer and collect a data set of user gameplay, which can benefit data-driven storytelling systems. We also provide an application that can be downloaded from Camelot's website and used as an EM to almost accurately recreate a playthrough from a log file.

2.3 Modular and Customizable

Camelot comes with a set of characters and places that can be customized as intended. To create various characters, the EM can choose from different body-types, hair styles, hair colors, eye colors, skin tones, and outfits. Figure 2 presents some examples of these characters. Camelot also provides many small, contained, pre-built environments, named *places*, that can be instantiated to create the story world. Figure 3 presents some examples of these places.

Each place comes with a set of interactive furniture, such as shelves, chairs, tables, or cauldrons, that can be hidden or shown depending on the context of the story.

Camelot does not impose any restrictions on where the doors of each place lead to. This enables Camelot's world creation to be modular and allows any configuration of the space. More specifically, every door leads to an area outside the place obstructed by white clouds. When a character exits through a door, they stay behind that door and wait for the EM to change their position. The EM creates the illusion that doors are connected by having a character enter through one door immediately after leaving through another.

2.4 Stateless Presentation

Camelot only acts as a presentation layer to an EM. Since different AI technologies, e.g. planning and machine learning, have very different representations of state, Camelot does not require the EM to use any particular state representation. In fact, to a large extent, Camelot does not keep track of the state of the world.

For example, Camelot has a UI element named *the List* that can be used to represent character inventory (Figure 4). It is in fact *the List* and not *a list*, as in there are not different instances of it for different characters. Again, it is the EM that decides what to put in the list when to display the list (e.g. to show the inventory of a specific character).

Furthermore, Camelot also has no notion of the player. More specifically, any of the instantiated characters can be controlled by mouse and keyboard as long as they are the *camera focus*. We will discuss the camera focus later in the Camera Control subsection. Any references to the player character in this paper refer to the one currently being controlled by the user.

There are some exceptions to the stateless nature of Camelot, specifically the physical position of characters. For instance, if a character is sitting on a chair, an action that attempts to make another character sit on that chair will fail with a message stating that the chair is already occupied by another character.

Moreover, many character actions require the character to first walk to the target. Since places are independent contained environments, the corresponding actions will fail if a targeted character moves to a different place.

However, this is not true about items. All actions that target items will teleport the specified item to the position required by the action. For instance, if an item is on a shelf and the EM asks a character to take the item out of their pocket, the item will instantaneously disappear from the shelf and appear in their hand. In addition, the *SetPosition* command can be used to instantaneously teleport a character or item to any other position within any place. Therefore, Camelot can also be adopted in interactive storytelling systems with a weak or non-existent sense of permanent state, such as purely language-based interactive narratives (e.g., neural language model based storytelling systems such as (Martin, Sood, and Riedl 2018).

2.5 Simple Description of Affordances

Affordances are the actions a player can choose from in an interactive narrative (not to be confused with Camelot commands also called actions). In Camelot, affordances can be simply described by the *EnableIcon* command. *EnableIcon* can be used to describe an affordance that can be performed to a character, furniture, or item. For instance, it can be used to allow the player to click on a chair to sit on it.

There are several important things to note about *EnableIcon*. When *EnableIcon* is used for a character, furniture, or item,

- The object will be highlighted when the user hovers the mouse over it.
- If the user right-clicks on the object, a radial menu is shown that presents all available interactions that can be performed on that object. Each option can be presented with a title and an icon. Camelot provides a large variety of icons that can used for this purpose. Figure 1 presents an example of a radial menu.
- When the user chooses to interact with an object, Camelot only responds by sending an *input* message to the EM. Camelot does not start any action unless directly instructed by the EM. This grants the EM full control over what to do next in response to user interactions and whether to accommodate or intervene (see Riedl, Saretto, and Young's (2003) discussion of mediation).
- The affordances can also be removed by simply calling the *DisableIcon* command.

As an example if Camelot receives *EnableIcon(SitDown, Chair, Room.Chair, "Sit on the chair")*,

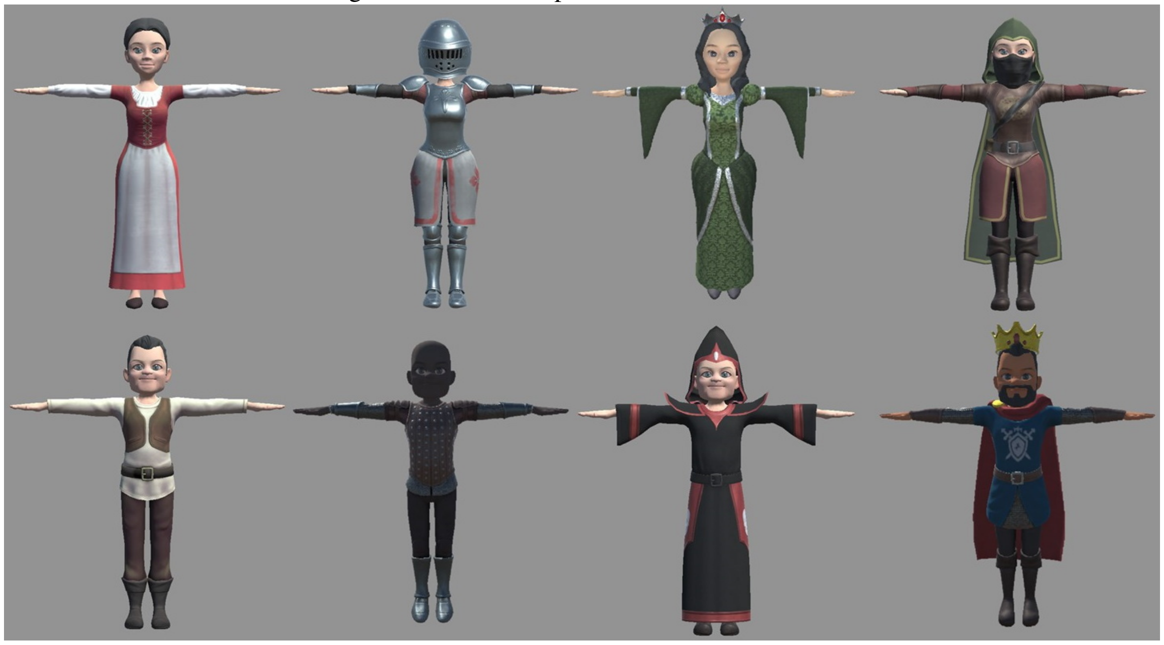
- When the user right-clicks on the chair, they see an option with title *"Sit on the chair"* and icon *Chair*.
- If the user clicks on the chair, Camelot sends the following message to the EM: *input SitDown Player Room.Chair*. This notifies the EM that the *Player* has chosen *SitDown*.
- The EM can then choose to make the player character sit on the chair by sending *start Sit(Player, Room.Chair)* to Camelot, or it can choose to show a message to the user like "I am not tired right now!"

2.6 Animations and Expressions

A large set of available Camelot commands can be used to animate characters. For instance, characters can open doors or chests, sit on chairs, or sleep on beds. These animations can be used as a visual response to user interactions in form of player character actions, as well as non-player reactions to those actions, e.g. clapping, laughing, waving, etc.

In addition to these animations, there are several expressions that can be used to express character emotions, which are happy, sad, angry, scared, surprised, and disgusted. The *SetExpression* command changes a character's facial expression as well as their idle animation to reflect that emotion. Character expressions can also change during dialog to display their reactions to dialog choices. These expressions can be used by believable agent research that model affect (Arellano, Varona, and Perales 2008; Marsella and Gratch 2009;

Figure 2: Some examples of Camelot characters



Neto and da Silva 2012; Alfonso Espinosa, Vivancos Rubio, and Botti Navarro 2014; Shirvani and Ware 2020). Figure 5 presents some examples of these expressions.

Camelot does not support graphic depictions of strong violence[2] or inappropriate nudity in order to make interactive narratives designed with it easier to approve by groups like university Institutional Review Boards (IRBs). Several Camelot games have been used in IRB-approved studies (Ware et al. 2019; Shirvani and Ware 2020).

2.7 Flexible UI

Camelot provides several general UI elements to use in a narrative. In addition to the radial menu, the list window can be used to present a list of items to interact with, e.g. displaying the inventory of a character or container, RPG character statistics, a set of skills to purchase, etc., and the narration window can be used to present simple text.

The dialog window provides interactive dialog that can be configured with character portraits and links embedded in the text that the user can click (Cavazza and Charles 2005; Endrass et al. 2013; Ryan, Mateas, and Wardrip-Fruin 2016). Dialog links are parts of the text that are highlighted in blue and can be clicked to represent dialog choices or advance the dialog tree. Figures 1, 4, and 6 present examples of these UI elements.

2.8 Camera Control

Camelot provides different options for controlling the camera, which can be used via *SetCameraFocus*, *SetCameraMode*, and *SetCameraBlend* commands. At each moment, the camera can be focused on a character, furniture, or item using *SetCameraFocus*. If the focus of the camera is a character and the input is enabled (via the *EnableInput* command), the camera follows that character's movements, and that character can be controlled by mouse and keyboard. We must note that the input can also be disable at times, for instance, during cutscenes.

SetCameraMode can be used to switch between three camera modes in real-time. The *follow* camera mode displays a third-person over-the-shoulder view of the character that is the focus of the camera, e.g. as in action RPG games. This mode can only be enabled if the camera focus is a character.

In *track* mode, a top-down view of the place is displayed, e.g. as in point-and-click adventures. As the character moves, the camera changes rotation to keep the character at the center, and if the character moves too far, the active camera switches to a different camera of that place that has a better view of the character.

Finally, the *focus* camera mode, presents a front close-up of the camera focus. This mode can be used to display character expressions or temporarily shift the focus of the user to a specific item or furniture that can be interacted with.

When the EM changes the camera focus or mode, the view transitions from the active camera to the new focus or mode. The duration of this transition can be controlled via the *SetCameraBlend* command. This command gives the EM more freedom to control the camera and create dramatic shifts or cuts during cut-scenes.

2.9 License and Availability

Camelot is published under the Non-Profit Open Source License 3.0. This license allows Camelot to be used for per-

[2]Characters can attack using the *Attack* command presented by swinging their arm while holding an item such as a sword or hammer.

Figure 3: Some examples of Camelot places

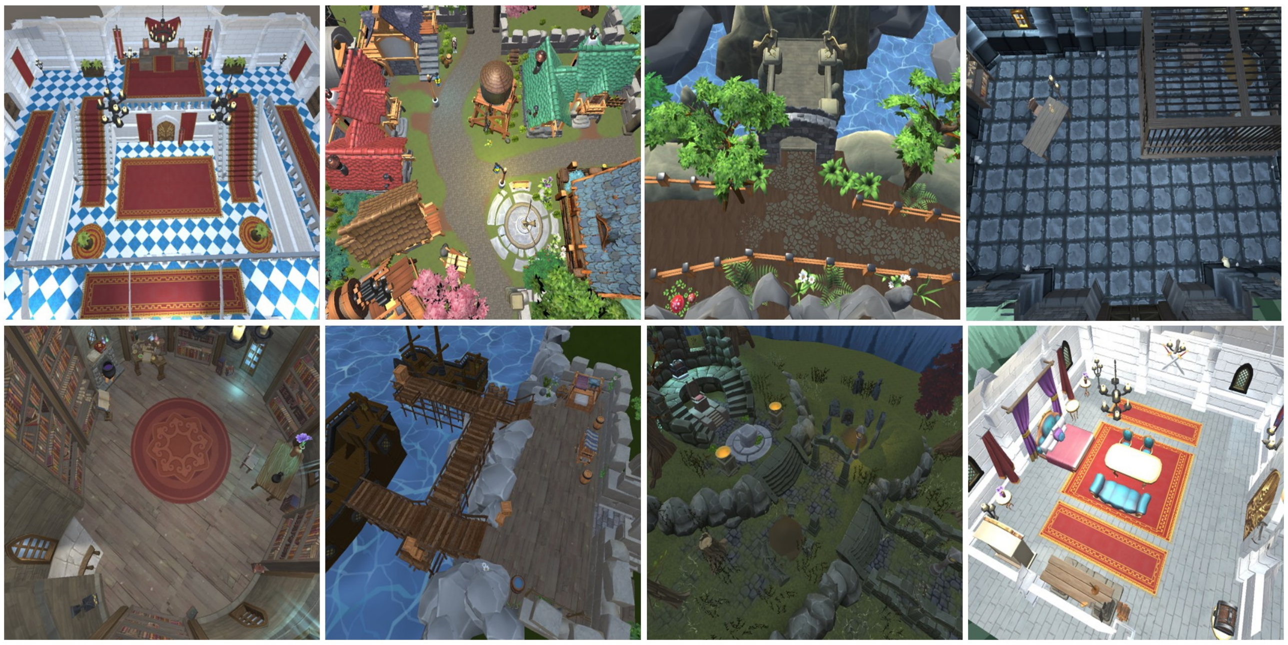


Figure 4: The list shows the current owner of the list (on the left) and a list of items (at the center) that can be selected to interact with or view their details (on the right).

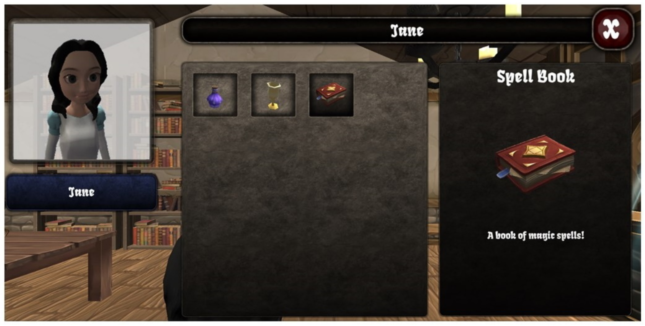


Figure 5: Examples of Camelot emotions, from left to right, surprised, angry, and scared

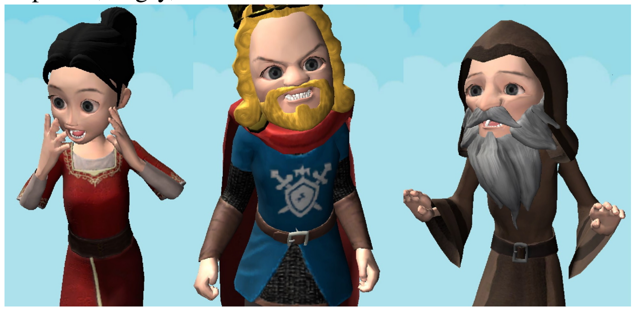
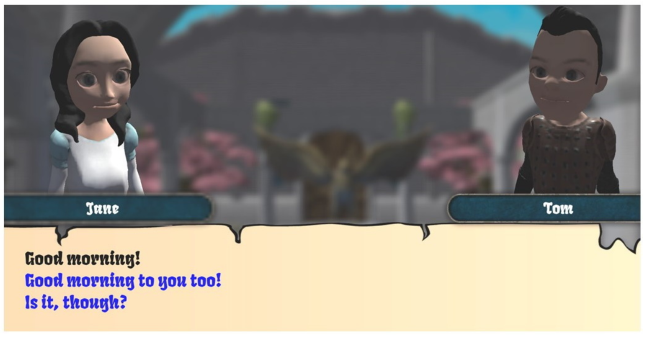


Figure 6: Interactive dialog can show up to two characters and any number of clickable links. Clickable links are highlighted in blue.



Good morning!
Good morning to you too!
Is it, though?

sonal, professional, and academic projects at no cost. It is only necessary to acknowledge the original project and creators in any derivative works[3]. Currently, the executable can be downloaded and used on Windows and Mac operating systems. The source code is also available to download. However, the copyrighted assets are not distributed with the source, and can be purchased from the Unity Asset Store at additional cost. The link to Camelot's documentation and download are presented at the end of this paper.

3 Applications and Practices

Camelot can benefit a wide range of AI research including but not limited to:

[3]We ask users to cite this paper in any published works that use Camelot.

- Automatic story generation and agent simulations using neural networks, reinforcement learning, and other machine learning algorithms (Rowe and Lester 2013; Harrison, Purdy, and Riedl 2017; Wang et al. 2017; Martin et al. 2018; Tambwekar et al. 2018).
- Strong-story and strong-autonomy systems using narrative planning (Young et al. 2013), with goals (Riedl and Young 2010; Teutenberg and Porteous 2013; Ware and Young 2014; Shirvani and Ware 2019a) and beliefs (Teutenberg and Porteous 2015; Shirvani, Ware, and Farrell 2017; Eger and Martens 2017; Shirvani, Farrell, and Ware 2018).
- Generating believable behavior by modeling agents with emotions (Gebhard 2005; Marsella and Gratch 2009; Shirvani 2019; Shirvani and Ware 2020) or personality (Bahamón and Young 2017; Berov 2017; Shirvani and Ware 2019b; Shvo, Buhmann, and Kapadia 2019).
- Dialog generation in interactive narratives (Cavazza and Charles 2005; Endrass et al. 2013; Ryan, Mateas, and Wardrip-Fruin 2016).
- Social simulations and interactive dramas using rule-based systems (El-Nasr, Yen, and Ioerger 2000; McCoy et al. 2012; 2014) and beat-based architectures (Mateas and Stern 2003).
- Intelligent camera control for virtual environments (Drucker and Zeltzer 1994; Ferreira, Gelatti, and Musse 2002; Jhala and Young 2010; Markowitz et al. 2011)

So far, Camelot has been used to create four different interactive narratives. *The Relics of the Kingdom* and *Murder in Felguard* were developed respectively in C++ and Python by two different teams of undergraduate students at the *University of Kentucky*. *The Three Kings* was developed in C# and best showcases different features of Camelot and the use of its UI in creating branching narratives. In contrast to the last three mentioned interactive narratives that were hand authored, *Saving Grandma*, also developed in C#, is a story graph interactive narrative that was generated using narrative planning (Ware et al. 2019). *Murder in Felguard* and *The Three Kings* are both free to access on Camelot's documentation website.

4 Community Outreach

Our hope is to encourage researcher to adopt Camelot in their relevant research. In previous years, Camelot was introduced in the Playable Experiences track of AIIDE (Samuel et al. 2018). A tutorial on Camelot was also held at AIIDE to showcase the capabilities and use cases of Camelot. This tutorial featured several invited demonstrations of experience managers that used interactive behavior trees (Martens and Iqbal 2019), multi-agent reinforcement learning (Busoniu, Babuska, and De Schutter 2008), the *Ensemble* engine (Samuel et al. 2015), multi-agent narrative planning (Ware et al. 2019), and murder mystery generation (Mohr, Eger, and Martens 2018). A showcase of Camelot will also be presented at AIIDE 2020's Intelligent Narrative Technologies (INT) workshop.

Our focus for the future of Camelot is to organize the Interactive Narrative Challenge (INCH). The purpose of INCH is to solicit AI EMs from many interactive narrative researchers and present their interactive narratives to human judges for qualitative and quantitative evaluation. INCH provides a practical context for controlled comparisons of interactive narratives across different systems. INCH will feature awards for many contributions in various aspects of a narrative, including use of narrative devices, e.g. flashbacks, foreshadowing, suspense, etc., story coherence, player freedom, replayablility, character richness, and so on. As a result of INCH, researchers will have access to free evaluation of their work by human players, as well as the dataset of the logs of all playthroughs. These logs can be further used to analyze user experience or to train a data-driven AI narrative system.

5 Conclusions

Translating AI algorithms and technologies into a user-friendly, visual interface is almost always a step in evaluating narrative generation systems via a human audience. The purpose of Camelot is to provide a modular, customizable, and easy-to-use virtual environment for researchers to visualize their stories. Camelot is fully independent of the experience manager that controls it, which allows any programming language or algorithm to easy connect and take advantage of Camelot. In addition, it enables the controlled comparison of drastically different narrative generation systems and allows researchers to reproduce and build on the work of others.

We plan to take advantage of Camelot in the Interactive Narrative Challenge to encourage researchers to submit their interactive narratives and to provide them with access to qualitative and quantitative evaluation of their work by human judges.

Camelot is an ongoing project and we plan to improve and expand it to support future interactive narrative authoring techniques.

Downloading Camelot

You can view a comprehensive interactive documentation website for Camelot at:

www.cs.uky.edu/~sgware/projects/camelot

The documentation provides details on how to use Camelot and its commands, as well as showcasing its characters, places, items, affordances icons, visual effects, and sound effects. You can also download Camelot for Windows or MacOs from the documentation website.

The website provides several applications that can be used as example EMs for Camelot. First, *CamelotReplay* is an application that reproduces a playthrough from a log file. Next, there are simple EMs that give beginners a place to start working with Camelot. They showcase a character moving from one place to another, trying out different outfits, and buying an item from a merchant. Finally, there are also two full interactive narratives, *Murder in Felguard* and *The Three Kings,* that showcase the wide range of things you can do in Camelot.

Acknowledgments

The development of Camelot was supported by the University of New Orleans and the University of Kentucky. We thank Edward T. Garcia, Rachelyn Farrell, and Porscha Banker for their insights and assistance with the project.

References

Alfonso Espinosa, B.; Vivancos Rubio, E.; and Botti Navarro, V. J. 2014. Extending a BDI agents' architecture with open emotional components. Technical report, Department of Information Technology, Universitat Politeócnica de Valeóncia.

Arellano, D.; Varona, J.; and Perales, F. J. 2008. Generation and visualization of emotional states in virtual characters. *Computer Animation and Virtual Worlds* 19(3-4):259–270.

Bahamón, J. C., and Young, R. M. 2017. An empirical evaluation of a generative method for the expression of personality traits through action choice. In *13th AAAI International Conference on Artificial Intelligence and Interactive Digital Entertainment*, 144–150.

Bates, J. 1992. Virtual reality, art, and entertainment. *Presence: Teleoperators & Virtual Environments* 1(1):133–138.

Berov, L. 2017. Steering plot through personality and affect: an extended BDI model of fictional characters. In *Joint German/Austrian Conference on Artificial Intelligence (Künstliche Intelligenz)*, 293–299. Springer.

Busoniu, L.; Babuska, R.; and De Schutter, B. 2008. A comprehensive survey of multiagent reinforcement learning. *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)* 38(2):156–172.

Cavazza, M., and Charles, F. 2005. Dialogue generation in character-based interactive storytelling. In *Proceedings of the First AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment*, AIIDE'05, 21–26. AAAI Press.

Drucker, S. M., and Zeltzer, D. 1994. Intelligent camera control in a virtual environment. In *Graphics Interface*, 190–190. Citeseer.

Eger, M., and Martens, C. 2017. Character beliefs in story generation. In *Thirteenth Artificial Intelligence and Interactive Digital Entertainment Conference*.

El-Nasr, M. S.; Yen, J.; and Ioerger, T. R. 2000. FLAME—fuzzy logic adaptive model of emotions. *Autonomous Agents and Multi-agent systems* 3(3):219–257.

Endrass, B.; Klimmt, C.; Mehlmann, G.; André, E.; and Roth, C. 2013. Designing user-character dialog in interactive narratives: An exploratory experiment. *IEEE Transactions on Computational Intelligence and AI in Games* 6(2):166–173.

Ferreira, F. P.; Gelatti, G.; and Musse, S. R. 2002. Intelligent virtual environment and camera control in behavioural simulation. In *Proceedings. XV Brazilian Symposium on Computer Graphics and Image Processing*, 365–372. IEEE.

Gebhard, P. 2005. ALMA: a layered model of affect. In *Proceedings of the fourth international joint conference on Autonomous Agents and Multi-Agent Systems*, 29–36.

Harrison, B.; Purdy, C.; and Riedl, M. O. 2017. Toward automated story generation with markov chain monte carlo methods and deep neural networks. In *Thirteenth Artificial Intelligence and Interactive Digital Entertainment Conference*.

Jhala, A., and Young, R. M. 2010. Cinematic visual discourse: Representation, generation, and evaluation. *IEEE Transactions on computational intelligence and AI in games* 2(2):69–81.

Markowitz, D.; Kider, J. T.; Shoulson, A.; and Badler, N. I. 2011. Intelligent camera control using behavior trees. In *International Conference on Motion in Games*, 156–167. Springer.

Marsella, S. C., and Gratch, J. 2009. EMA: A process model of appraisal dynamics. *Cognitive Systems Research* 10(1):70–90.

Martens, C., and Iqbal, O. 2019. Villanelle: an authoring tool for autonomous characters in interactive fiction. In *Proceedings of the International Conference on Interactive Digital Storytelling*, 290–303.

Martin, L. J.; Ammanabrolu, P.; Wang, X.; Hancock, W.; Singh, S.; Harrison, B.; and Riedl, M. O. 2018. Event representations for automated story generation with deep neural nets. In *Thirty-Second AAAI Conference on Artificial Intelligence*.

Martin, L. J.; Sood, S.; and Riedl, M. 2018. Dungeons and dqns: Toward reinforcement learning agents that play tabletop roleplaying games. In Wu, H.; Si, M.; and Jhala, A., eds., *Proceedings of the Joint Workshop on Intelligent Narrative Technologies and Workshop on Intelligent Cinematography and Editing co-located with 14th AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment, INT/WICED@AIIDE 2018, Edmonton, Canada, November 13-14, 2018*, volume 2321 of *CEUR Workshop Proceedings*. CEUR-WS.org.

Mateas, M., and Stern, A. 2003. Façade: An experiment in building a fully-realized interactive drama. In *Game developers conference*, volume 2, 4–8.

McCoy, J.; Treanor, M.; Samuel, B.; Reed, A. A.; Wardrip-Fruin, N.; and Mateas, M. 2012. Prom Week: designing past the game/story dilemma. In *Proceedings of the International Conference on the Foundations of Digital Games*, 235–237.

McCoy, J.; Treanor, M.; Samuel, B.; Reed, A. A.; Mateas, M.; and Wardrip-Fruin, N. 2014. Social story worlds with Comme il Faut. *IEEE Transactions on Computational intelligence and AI in Games* 6(2):97–112.

Mohr, H.; Eger, M.; and Martens, C. 2018. Eliminating the impossible: a procedurally generated murder mystery. In *Proceedings of the Experimental AI in Games workshop at the 14th AAAI international conference on Artificial Intelligence and Interactive Digital Entertainment*.

Neto, A. F. B., and da Silva, F. S. C. 2012. A computer architecture for intelligent agents with personality and emotions. In *Human-Computer Interaction: The Agency Perspective*. Springer. 263–285.

Riedl, M. O., and Bulitko, V. 2013. Interactive narrative: An intelligent systems approach. *AI Magazine* 34(1):67–67.

Riedl, M. O., and Young, R. M. 2010. Narrative planning: Balancing plot and character. *Journal of Artificial Intelligence Research* 39:217–268.

Riedl, M.; Saretto, C. J.; and Young, R. M. 2003. Managing interaction between users and agents in a multi-agent storytelling environment. In *Proceedings of the second international joint conference on Autonomous Agents and Multiagent Systems*, 741–748.

Roberts, D. L., and Isbell, C. L. 2008. A survey and qualitative analysis of recent advances in drama management. *International Transactions on Systems Science and Applications, Special Issue on Agent Based Systems for Human Learning* 4(2):61–75.

Rowe, J. P., and Lester, J. C. 2013. A modular reinforcement learning framework for interactive narrative planning. In *Ninth Artificial Intelligence and Interactive Digital Entertainment Conference*, 57–63.

Ryan, J.; Mateas, M.; and Wardrip-Fruin, N. 2016. Characters who speak their minds: dialogue generation in Talk of the Town. In *Twelfth Artificial Intelligence and Interactive Digital Entertainment Conference*.

Samuel, B.; Reed, A. A.; Maddaloni, P.; Mateas, M.; and Wardrip-Fruin, N. 2015. The Ensemble engine: Next-generation social physics. In *Proceedings of the Tenth International Conference on the Foundations of Digital Games (FDG 2015)*, 22–25.

Samuel, B.; Reed, A.; Short, E.; Heck, S.; Robison, B.; Wright, L.; Soule, T.; Treanor, M.; McCoy, J.; Sullivan, A.; et al. 2018. Playable experiences at AIIDE 2018. In *Proceedings of the Fourteenth Artificial Intelligence and Interactive Digital Entertainment Conference*, 275–280.

Shirvani, A., and Ware, S. G. 2019a. On automatically motivating story characters. In *Proceedings of the Experimental AI in Games workshop at the 15th AAAI international conference on Artificial Intelligence and Interactive Digital Entertainment*.

Shirvani, A., and Ware, S. G. 2019b. A plan-based personality model for story characters. In *Proceedings of the 15th AAAI international conference on Artificial Intelligence and Interactive Digital Entertainment*, 188–194.

Shirvani, A., and Ware, S. G. 2020. A formalization of emotional planning for strong-story systems.

Shirvani, A.; Farrell, R.; and Ware, S. G. 2018. Combining intentionality and belief: Revisiting believable character plans. In *Fourteenth Artificial Intelligence and Interactive Digital Entertainment Conference*, 222–228.

Shirvani, A.; Ware, S. G.; and Farrell, R. 2017. A possible worlds model of belief for state-space narrative planning. In *Proceedings of the Thirteenth Artificial Intelligence and Interactive Digital Entertainment Conference*, 101–107.

Shirvani, A. 2019. Towards more believable characters using personality and emotion. In *Proceedings of the AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment*, 230–232.

Shvo, M.; Buhmann, J.; and Kapadia, M. 2019. An interdependent model of personality, motivation, emotion, and mood for intelligent virtual agents. In *Proceedings of the 19th ACM International Conference on Intelligent Virtual Agents*, 65–72.

Tambwekar, P.; Dhuliawala, M.; Martin, L. J.; Mehta, A.; Harrison, B.; and Riedl, M. O. 2018. Controllable neural story plot generation via reinforcement learning. *arXiv preprint arXiv:1809.10736*.

Teutenberg, J., and Porteous, J. 2013. Efficient intent-based narrative generation using multiple planning agents. In *Proceedings of the 2013 international conference on Autonomous agents and multi-agent systems*, 603–610. International Foundation for Autonomous Agents and Multiagent Systems (IFAAMAS).

Teutenberg, J., and Porteous, J. 2015. Incorporating global and local knowledge in intentional narrative planning. In *International Conference on Autonomous Agents and Multiagent Systems*, 1539–1546.

Wang, P.; Rowe, J. P.; Min, W.; Mott, B. W.; and Lester, J. C. 2017. Interactive narrative personalization with deep reinforcement learning. In *Proceedings of the 26th International Joint Conference on Artificial Intelligence*, 3852–3858.

Ware, S. G., and Young, R. M. 2014. Glaive: a state-space narrative planner supporting intentionality and conflict. In *Tenth Artificial Intelligence and Interactive Digital Entertainment Conference*.

Ware, S. G.; Garcia, E.; Shirvani, A.; and Farrell, R. 2019. Multi-agent narrative experience management as story graph pruning. In *Proceedings of the fifteenth Artificial Intelligence and Interactive Digital Entertainment Conference*, 87–93.

Weyhrauch, P. W. 1997. *Guiding interactive drama*. Ph.D. Dissertation, Carnegie Mellon University.

Young, R. M.; Thomas, J.; Bevan, C.; and Cassel, B. 2011. Zócalo: A service-oriented architecture facilitating sharing of computational resources in interactive narrative research. In *Working Notes of the Workshop on Sharing Interactive Digital Storytelling Technologies at ICIDS*, volume 11. Citeseer.

Young, R. M.; Ware, S. G.; Cassell, B. A.; and Robertson, J. 2013. Plans and planning in narrative generation: a review of plan-based approaches to the generation of story, discourse and interactivity in narratives. *Sprache und Datenverarbeitung, Special Issue on Formal and Computational Models of Narrative* 37(1-2):41–64.

Young, R. M. 2001. An overview of the Mimesis architecture: Integrating intelligent narrative control into an existing gaming environment. In *Working notes of the AAAI spring symposium on Artificial Intelligence and Interactive Entertainment*, 77–81.