CARDINAL

Computer-Assisted Authoring of Movie Scripts

Anonymous

ABSTRACT

We present CARDINAL, a tool for computer-assisted authoring of movie scripts. CARDINAL provides a means of viewing a script through a variety of perspectives, for interpretation as well as editing. This is made possible by virtue of intelligent automated analysis of natural language scripts and generating different intermediate representations. CARDINAL generates 2-D and 3-D visualizations of the scripted narrative and also presents interactions in a timeline-based view. The visualizations empower the scriptwriter to understand their story from a spatial perspective, and the timeline view provides an overview of the interactions in the story. The user study reveals that users of the system demonstrated confidence and comfort using the system.

KEYWORDS

ACM proceedings, LaTeX, text tagging

ACM Reference format:
Anonymous. 2018. CARDINAL. In *Proceedings of IUI, Tokyo, Japan, March 2018 (IUI' 2018),* 12 pages.
https://doi.org/10.1145/nnnnnnn.nnnnnnn

1 INTRODUCTION

Movies tell complex and detailed stories to viewers through intricately planned narrative structure. These stories are traditionally authored in text editors. Existing applications allow for organization of material, but lack support for visualizations that highlight different aspects of a script.

Information extraction and visualization of the various elements of the story can help script writers analyze when and where actions and interactions between actors in the scenes of the script unfold. The lack of specialized visualizations hinder story writers from discerning if their description of the story matches their creative intent. Characters movements and interactions in the script may be complex and difficult to follow solely through the text.

Extracting information from scripts to visualize it in novel and helpful ways is challenging due to the lack of meta information about different parts of a script. Script writers, just like any other person, develop their own habits over time. While there is an industry standard format, scripts differ a lot in the implementation of this standard. New visualizations should not need manual input from script writers. These tools should not interrupt or change the work-flow of script writers, making this problem difficult to approach and solve. All data must be extracted from text alone, requiring sophisticated parsing capabilities.

Existing solutions like Final Draft [6] or Adobe Story [1] provide some tools that allow script writers to plan certain aspects of their script, but the main focus is on organizing scenes, locations and actors without visualizations of interactions and movement. These features require the application to be aware of details of the written script, which traditional script writing applications do not do in sufficient capacity.

We propose CARDINAL, a script authoring application taking a natural language processing based approach to provide new visualizations for script writers. CARDINAL provides line visualizations that allows users to properly plan parallelism in actions and clearly visualizes which actors are interacting with whom at which point in time in the script. Furthermore, CARDINAL also provides an automatically generated three dimensional preview of the written script that allows script writers to not only detect logical errors in their script, but also provides an easy way to keep in mind what the authored story currently could look like in a production setting. These different views in CARDINAL are possible by virtue of intelligent automated analysis of these natural language scripts, and generate intermediate representations of scripts, that allow us to view the scripts in different types of ways.To extract the information needed to automatically generate these visualizations from text, CARDINAL employs a natural language processing solution that translates text in the script into actions that are used by the application. While the application stays true to its roots and sticks close to the industry standard format of scripts, the functionally is extended in reasonable and automated ways. Other than in traditional

IUI' 2018, March 2018, Tokyo, Japan

ACM ISBN 978-x-xxxx-xxxx-x/YY/MM…$15.00
https://doi.org/10.1145/nnnnnnn.nnnnnnn

script writing tools, script writers can now follow the action in a script much more easily by simply looking at the interaction view or the generated preview.

We show how these new visualizations work and look and how they are implemented in a proof-of-concept prototype. We also mention the challenges that come with each feature. Our expectation of the usefulness and efficiency of our novel visualizations for extracting data from scripts is corroborated with a user study, which confirms that subjects take less time to extract information while also reducing the fraction of wrong answers when using our visualizations.

2 RELATED WORK

CARDINAL uses sophisticated natural language processing frameworks based on Stanford's Core NLP framework [14]. To get the best results for our interface, we also use dependency resolution [4] and coreference resolution [13]. Natural language is also used by other systems such as LyriSys, which uses it towards generating lyrics [23]. LISA [19] uses a similar natural language-based approach to extract information about stories and automatically flags logical problems within the story. There are various other attempts that focus on extracting narrative information, and some require further user input [5, 7], others focus on other aspects of the narratives such as affect [8].

Pavel et al. [16] propose a new way to visualize, search and browse in movie plots based on synchronized captions, scripts and plot summaries. They describe an interface that integrates information from different documents to provide access to an improved search that works at several levels of granularity. Newer visualization tools are being introduced in other domains as well, which make use of combined models, neural networks and other newer technologies [2, 20].

VidCrit [17] is a new system provides the ability to give asynchronous feedback on drafts of edited videos. The feedback viewing interface describes in this paper includes a time line that automatically segments a video into topical text sessions and labels them with additional contextual information. Ruben et al. even propose tools to work with audio based stories [18].

In [22] Tapaswi et al. present a new way to automatically summarize the storyline of movies and TV episodes by visualizing character interactions as a chart. In this work, the authors also present a time line based chart where each character of a story is represented by a horizontal line. These lines join, split and fade in our out based on whether a character is part of the scene. In CARDINAL, we try to take the time line one step further and do not only represent character interactions on a per-scene level, but instead go deeper to a per-action level. We visualize dialogues and actions and join or split lines within each scene based on those interactions.

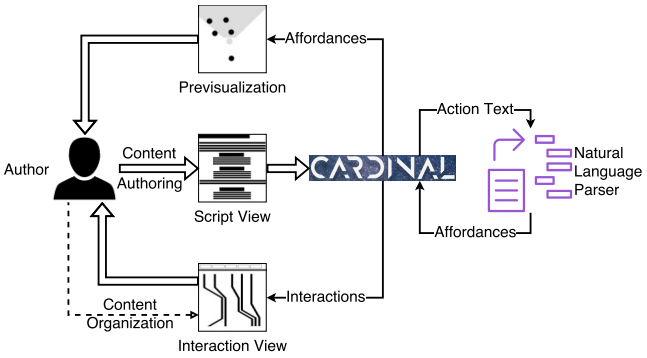

Figure 1: Conceptual framework of CARDINAL. The user authors a movie script mostly in a textual script view. The system takes the text and stores it in an internal meta-annotated representation format. Text of actions is further passed to a natural language processing server and transformed into affordances, which are then used in other visualizations.

CARDINAL uses affordances as atomic actions executable by smart objects. CANVAS [10] is an authoring system that is built on top of a smart object and affordance system as well. (Kapadia et al.) describe a computer-assisted visual authoring tool for synthesizing multi-character animations from sparsely-specified narrative events. Other than CARDINAL, CANVAS uses a story boarding approach to specifying a narrative that is then completed automatically by the system.

Storyboarding is also used by StoryCrate [3], where Bartindale et al. present a tangible tabletop prototype for live story boarding in film production. In their work on behavior trees, Kapadia et al. [11] describe a new approach to modeling multi-character behavior using behavior trees. This work is continued in [12] and [9] where a new approach to integrating interactivity into behavior trees is presented.

From the related work discussed in this section CARDINAL sets itself apart as a unique application which uses natural language understanding to extract affordance-based information from the script, and simultaneously performs previsualization to provide the author with multiple representations of the narrative they are creating with the application.

3 OVERVIEW

CARDINAL distinguishes itself from traditional script writing tools by focusing on assisting scriptwriters during the process of authoring stories. Figure 1 depicts an overview of the structure of components and features of CARDINAL. This section gives important terms and then briefly describes the various views shown in the framework.

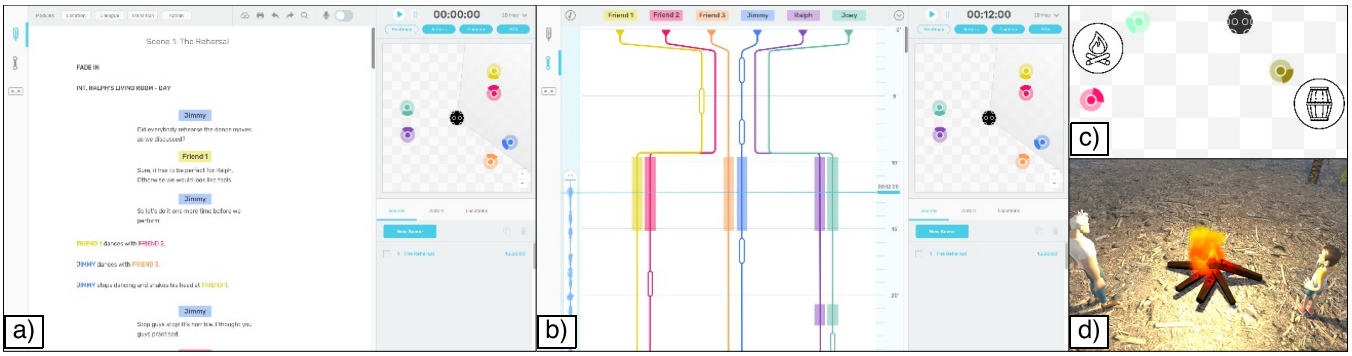

Figure 2: The various views of the CARDINAL system: a) Script View, b) Interaction View, c) 2D-Preview, d) 3D-Preview.

Symbol	Definition
$\mathbf{A}_{\text{all}}$	The set of all affordances.
$\mathbf{A}_{\mathbf{P}}$	The set of all affordance instances in project P.
$\mathbf{A}_{\mathbf{S}}$	The set of all affordance instances in scene S.
$\mathbf{S}_{\mathbf{P}}$	The set of all scenes in the project P.
$\mathbf{S}_{\mathbf{O,P}}$	The set of all smart objects in project P.
$\mathbf{S}_{\mathbf{A,P}}$	The set of all smart actors in project P.
$\mathbf{S}_{\mathbf{C,P}}$	The set of all smart cameras in project P.
$\mathbf{S}_{\mathbf{O,S}}$	The set of all smart objects in the scene S.
$\mathbf{S}_{\mathbf{A,S}}$	The set of all smart actors in the scene S.
$\mathbf{S}_{\mathbf{C,S}}$	The set of all smart cameras in the scene S.

Table 1: Definitions for affordance sets, project sets and scene sets.

Terminology

Special Sets. CARDINAL stores sets of smart objects, affordances and affordance instances in different places, each representing different structures in the application. The most frequently used sets are shown in Table 1.

Smart Objects. Smart objects $\mathbf{S} =< \mathbf{A}, \mathbf{s_s}, \mathbf{s_d} >$ are objects that have a static significance value and a dynamic significance value that are both used to compute how much that object adds to the significance of a position in a scene. Finally, each smart object has a list of affordances $\mathbf{A} \subseteq \mathbf{A}_{\text{all}}$, which represent the actions and interactions offered by that smart object. It is also important to note that smart objects also, at any time, have a position and an orientation , which are represented using position, forward and up vectors.

Affordances. Affordances describe both dialogs and actions such as walking, cheering and waving. The direction of initiation in affordances is usually reversed. The user of an affordance initiates the affordance, while the owner of the affordance controls the affordance. An affordance can be described as a four value tuple: $\mathbf{A} =< \mathbf{O}, \mathbf{U}, \mathbf{v} >$. $\mathbf{O}$ denotes the owner, $\mathbf{U}$ denotes the set of users for the affordance. Every affordance also has a vector $\mathbf{v}$ associated to it. This vector is used to map the verbs from the script text to the closest affordance, which is described in Section 4.

One example that showcases the reversed flow of control particularly well is a ball. A ball offers an affordance to be picked up, which uses the hands of an actor. The ball knows how it is supposed to be picked up and controls the actor trying to pick up the ball. For simple affordances, this reversed flow of control is not easily seen.

Interactions. Interactions $\mathbf{I} \subseteq \mathbf{A}_{\mathbf{S}}$ are sets of affordances that describe how a set of actors interacts with each other and with objects in the scene. Interactions can span a long time and can contain affordances that happen in sequence. Interactions can not span across scene boundaries.

Views

In CARDINAL, we introduce a variety of views that focus on different aspects of the script: the text-based script view, the interaction view highlighting exact timings and parallelism in actions, and the preview that comes with different filters and focuses on planning spatial properties of a story. These views are shown in Figure 2.

Script View. The script view is the text-based view of CARDINAL. Just like in traditional scripts, it allows script writers to create, edit and remove parts of the script. It sticks to the ndustry standard for scripts.

Interaction View. The interaction view shows the evolution of each actor in a scene as a graphical 2D representation of the script. In order to be fully distinguishable, every actor has its own color and line. The interaction view is created automatically from the parsed script.

2D/3D Preview. Keeping track of movement and locations in 360 degree movies purely based on a textual representation of a story is difficult. CARDINAL provides a solution to these problems by automatically generating a 2D as well as a 3D preview of the authored story. Text entered in actions is passed to a natural language processing subsystem, which in turn maps the action text to an action vocabulary and produces animations for the actors in the preview scene. This automatically generated preview comes with a set of predefined filters that each highlight different features of the story.

The different views allow the user to interact and view the script in multiple ways, which current scripting tools lack. The views are not only for viewing but also allow for editing. The interaction view allows the user to make edits in terms of interactions as opposed to the raw text in script view. This novel approach is the core of CARDINAL and its focus on assisted authoring of scripts.

4 SCRIPT VIEW

The main tool of every script writer is a script view. Just like in traditional script writing applications, CARDINAL allows authors to write their scripts in a well known format. Unlike traditional scripts, however, CARDINAL improves upon existing tools by further highlighting actors. Each actor is assigned a unique color upon creation that identifies him throughout the application in actions, the automatically generated preview and the interaction view.

The four different elements commonly found in movie scripts, i.e., dialogs, actions, locations and transitions, are represented in CARDINAL by modules that an author can drag and drop into the script view. Each module has a start- and an end-time that is automatically computed based on its relative position in the existing script view:

- **Headings** are the titles of a scene and are written in all upper case letters.
- **Locations** start with a shortcut for exterior (EXT.) or interior (INT.) scenes.
- **Actions** describe things happening that aren't dialog. These include movement, interactions, but also circumstantial descriptions.
- **Dialogs** have a heading with an all uppercase actor name, followed by the dialog text. Both the heading as well as the dialog text are centered, unlike other parts.

Since the script view has no way of showing parallelism in actions, modules are inserted in the script view in a way that all modules following the new module are shifted forward in time by the duration of the new module.

Affordances extraction

In the script view all information used by other visualizations is extracted. Text in action modules is sent to a remote natural language processing (NLP) service, responsible for parsing the text and sending it back in a form of subject-verb-object triples. These triples are then interpreted and checked for correctness. The parsed subjects and objects are searched for by name in the list of all available objects in the current project. If the subject is found, the list of affordances offered by it is searched for an affordance that represents the relation in the parsed action.

If neither the subject nor the object is found, the script view shows a warning, explaining that no object with the specified parsed name could be found. If the relation is not found, then a warning is displayed that the subject does not have an affordance of the specified name. This functionality allows for an efficient work-flow where actions are automatically parsed and translated into affordances that can be then used by the preview and the interaction view to reflect a current content of the script.

Natural Language Processing

Scriptwriters are able to provide an input in natural language in the form of a main story plot, i.e., interactions between characters, and dialogs (locations and transitions are chosen from predefined options). With the help of a Stanford CoreNLP framework [14] a semantic analysis of the action text is performed, automatizing a process of creating affordances.

A story is assumed to be divided into separate modules (paragraphs), consisting of logically connected sentences. A first step involves applying co-reference resolution to the whole text in one paragraph – pronouns, such as 'he', 'they', 'it', are assigned the real names of actors, objects and places based on previously analyzed sentences. Then, single sentences and words (tokens) are extracted. Relations between tokens are analyzed using dependency graphs [4].

The main parsing task involves extracting subjects (nouns), relations (verbs) and objects (often nouns), i.e., `relation triples`, as well as relation modifiers (mostly adverbs), which describes a way affordances are performed (walking could either fast or slow). Also more complicated structures of sentences are handled, containing, e.g., auxiliary verbs and continuous forms of verbs, i.e., a sentence "Adam is gone" would be recognized as a state of the person, in contrast to sentence "Adam is going to Isa", being a proper action.

Nominal modifiers ("talk **with** someone"), and relational objects ("talk with **someone**") are extracted as well, but they are analyzed in the way that the final output of the parser consists of only triples and modifiers, i.e., a relation would be "talk with" and an object would be "someone". If more

than one object is specified in the sentence, then for each of them an independent triple is created. Each module and each sentence in the module are indexed which is then used to provide proper timings – as a result all triples generated from one sentence are assigned the same time and resulting actions are executed simultaneously. AN exception from this rule, when timing words are used, i.e., in a sentence "Before Isa shakes Marcel's hand, she cries." the second action is performed first.

Creating new affordances requires defining at least possible owners O and users U, associated relations (verbs) and animations. As a result, a set of affordances set up in CARDINAL is finite, requires much time to be extended, and thus it is intractable to create a complete library of affordances corresponding to every word possibly used by a script writer. We resolve this problem by assigning to every created affordance a vector v, which is an embedding of an associated relation (verb) extracted using Word2vec [15]. Each extracted relation triple is mapped to one of previously defined affordances by looking for a most similar verb to an extracted relation in a set of relations associated with previously defined affordances based on a cosine similarity of vectors.

5 INTERACTION VIEW

The interaction view visualizes a movie script as a set of lines, one for each actor. This visualization shows the interactions between actors and groups them accordingly. A user is able to immediately extract information crucial for each scene, such as participating actors, their interactions over time, and the timing relative to other affordances in the scene. Professional production of movies involves a lot of people not equally familiar with the script – the interaction view will offer a lead-in in any stage of the production.

A system that helps creating a video content should not only help understand the script and detect flaws and inconsistencies, but once the problems are apparent one should be able to change and correct the script. The interaction view meets this requirement as it is fully interactive – a user is able to add and remove actors and affordances, as well as change the timing of each affordance separately by dragging and scaling a graphical element to the desired form. Changes made here are consistent over all views.

The visualization is built dynamically based on the intermediate representation produced by the script view:

- Every **actor** has its dedicated column, serving as a beginning of a character's line. A line of an actor always starts at y_0, but can move in x direction dependent on other actors he interacts with. The lines move back to their dedicated columns as soon as the interaction finishes.
- Each **affordance** is represented as a graphical element drawn on the line of its owner. In order to provide a clear overview we distinguish dialogs and other interactions by using different graphical elements. The relative position on the y-axis gives information on the order, while the length gives insight into a relative duration of the affordance. A more detailed description of the affordance appears while hovering the mouse over the graphical element. The dialog elements then show the name of the affordance, all involved users and (if applicable) a content of the actual dialog.

Figure 3 shows an example way a user can interact with CARDINAL, by providing a text in the script view, which is later automatically split into submodules and translated into affordances. The interaction view then visualizes these affordances and allows a user to change their order or duration, what is reflected back in the script view.

Interactions

In order to visualize a context of the individual affordance we designed **interactions**. An interaction i groups related affordances (e.g., a dialog interaction consists of multiple talk affordances) and is visualized by joining corresponding lines for the time of interaction. The main difficulty in creating the interaction view is to identify related affordances. We came up with several definitions for a relation between affordances and different approaches to group them. In the following we present these approaches and discuss their respective benefits and trade-offs. We introduce following nomenclature: $a_1 \backsim a_2$ means that affordances a_1 and a_2 are related to each other, $c_1 \vdash c_2$ means that characters c_1 and c_2 are close to each other. Both aforementioned relations are reflexive and symmetric, but not transitive.

Furthermore, we use the set $\mathbf{A_S}$ of all affordances in the current scene and define $\mathbf{I_S}$ to be the set of all interactions in the current scene. We define a function $\mathbf{F_P}$ to get a set of all owners $\mathbf{O}$ and users $\mathbf{U}$ of an affordance or interaction. Between affordances there could be:

(1) **No relation** Affordances are not related to each other.

$$a_1 \nsim a_2$$

(2) **Relation by locality** Affordances are related based on the location of their owner. Affordances with proximate actors are related to each other.

$$O_1 \vdash O_2 \implies a_1 \backsim a_2$$

(3) **Relation by participation** Affordances are related based on common actors.

$$\mathbf{F_P}(a_1) \cap \mathbf{F_P}(a_2) \neq \emptyset \implies a_1 \backsim a_2$$

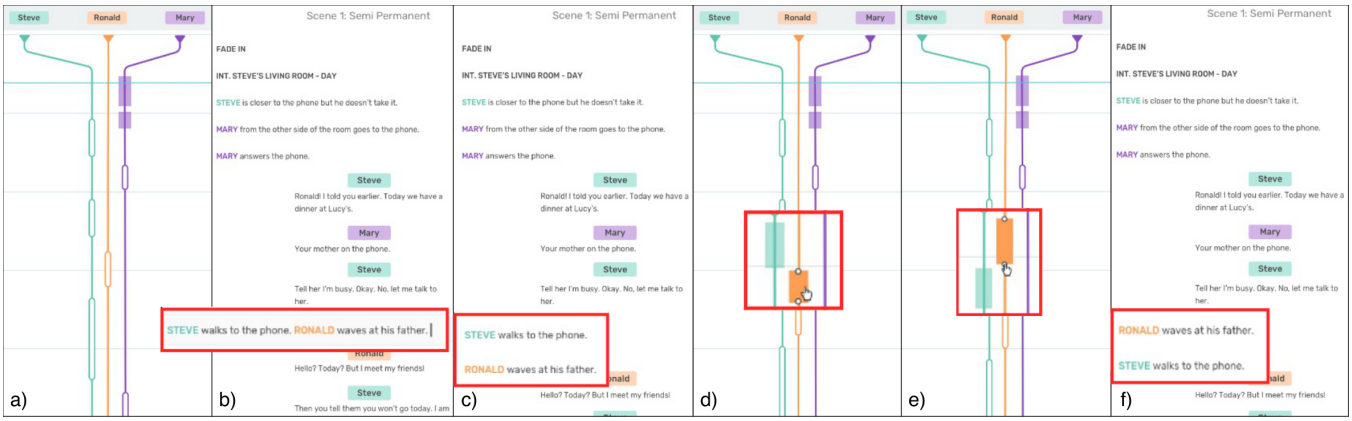

Figure 3: Figures a) to c) – CARDINAL automatically parses action modules into affordances that are animated in the preview and shown in the interaction view, and splits the text into two separate modules. Figures d) to f) – the interaction view allows users to arrange affordances happening in parallel by providing tools necessary to modify their start times and duration (here a user increases a length of the movement affordance to make Ronald move a bit slower).

Grouping. An interaction consists of a group of pairwise related affordances. These groups can be:

(1) **Exclusive** Actors can only be in one interaction at a time.

$$\forall i_1, i_2 \in \mathbf{I_S}, i_1 \neq i_2 \implies \mathbf{F_P}(i_1) \cap \mathbf{F_P}(i_2) = \emptyset$$

(2) **Inclusive** No restriction on the actors of an interaction, they can be part of one or more interactions.

Spatial interactions. Scripts use location relative to other actors or objects in the scene rather than absolute location. This allows us to determine each actor's location at any time based on the initial location of the actor and spatial affordances like *go to someone* and *go to some object*. Therefore, groups created by locality are always exclusive. The main benefit of this approach is it's simplicity, while reflecting our intuitive understanding of interactions in most scenarios: people interact with their immediate surroundings. However there are scenarios where interactions created by locality ignore certain affordances. *Point at someone* or *greet someone* don't require their users to be proximate to the owner. Nevertheless, one could visualize these affordances by connecting the actors with dotted lines.

Exclusive interactions by participation. The *related* relation is not transitive. Therefore, actors might very well be in two different groups of related affordances. For exclusive interactions we don't allow this behavior. We have to select affordances which we won't take into account for creating the interactions. We propose the following selection criteria:

(1) **Occurrence count:** We count the number of occurrences of actors in each group of related affordances. For every affordance we go through its actors and look for the group having the highest occurrence count. That's the group that "wins" this affordance, in every other group this affordance is discarded. To prevent equal counts we weight the occurrences by the importance of each affordance.

(2) **Affordance importance:** Affordances are assigned a weight between 0 and 1. We propose the following ordering from highest to lowest importance: dialogs, affordances with small range like affordances *touch* or *hold*, affordances with hight range like *point at* or *wave at*, affordances without specified targets like *cheer* or *clap*.

Inclusive interactions by participation. Inclusive interactions are created from the related affordances as it is without further selection necessary because we allow actors to be part of multiple interactions. This approach is the most correct in the sense that we do not ignore any affordances. However, there is a significant increase in visual complexity. Multiple parallel lines are needed to visualize an actor being part of multiple interactions.

Figure 4 presents adding a new affordance in the interaction view and automatic grouping of existing affordances. Figure 5 visualizes different grouping approaches described above: exclusive and inclusive interactions by participation as well as exclusive interactions by location.

6 PREVISUALIZATION OF MOVIE SCRIPTS

CARDINAL uses the natural language text in the script view and translates it into an intermediate representation (IR). This intermediate representation is made of modules, which describe affordances and their respective timings in the scene, as described in the script view section. The IR is used throughout the system to automatically generate visualizations. For

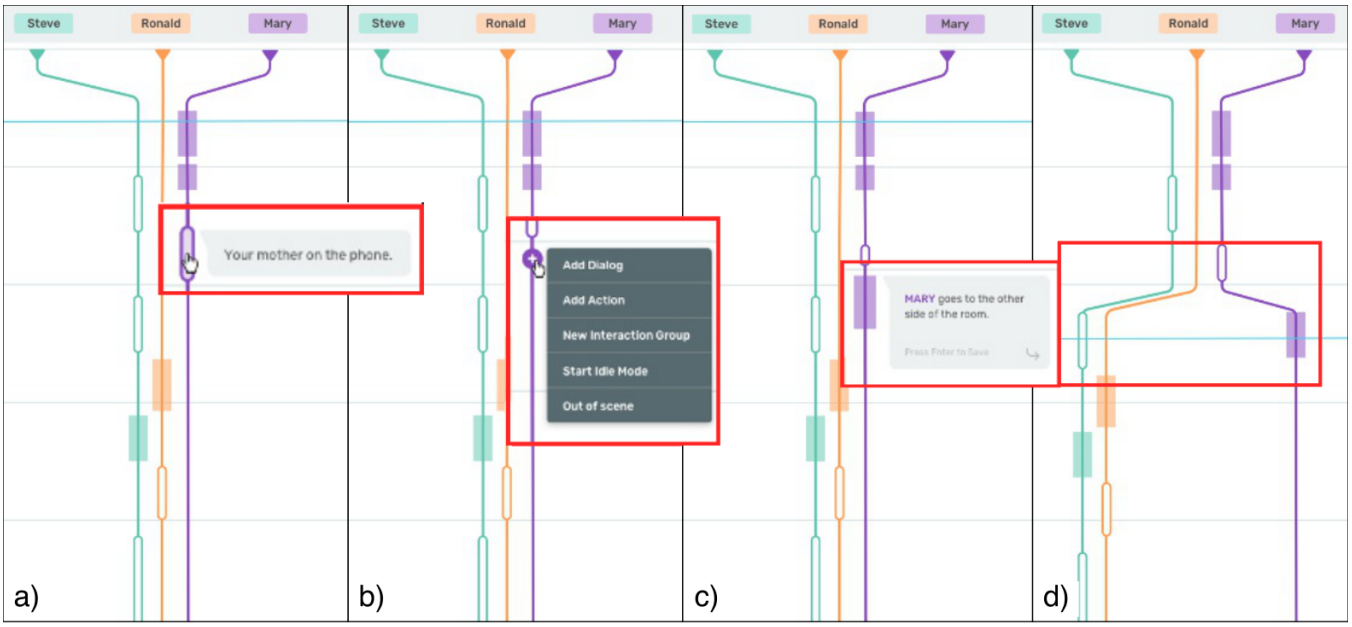

Figure 4: The interaction view can also be used to author affordances for actors. This gives script writers the flexibility needed to fine-tune their story. CARDINAL automatically splits actors into interaction groups: a) the user looks at a dialog in the interaction view and realizes that the actor should be moving away from it's current interaction group, b) the user adds a movement affordance directly in the interaction view, c) the movement affordance appears in the interaction view and all other visualizations, d) the movement of the actor away from it's current group causes the interaction view to split the actors in two interaction groups.

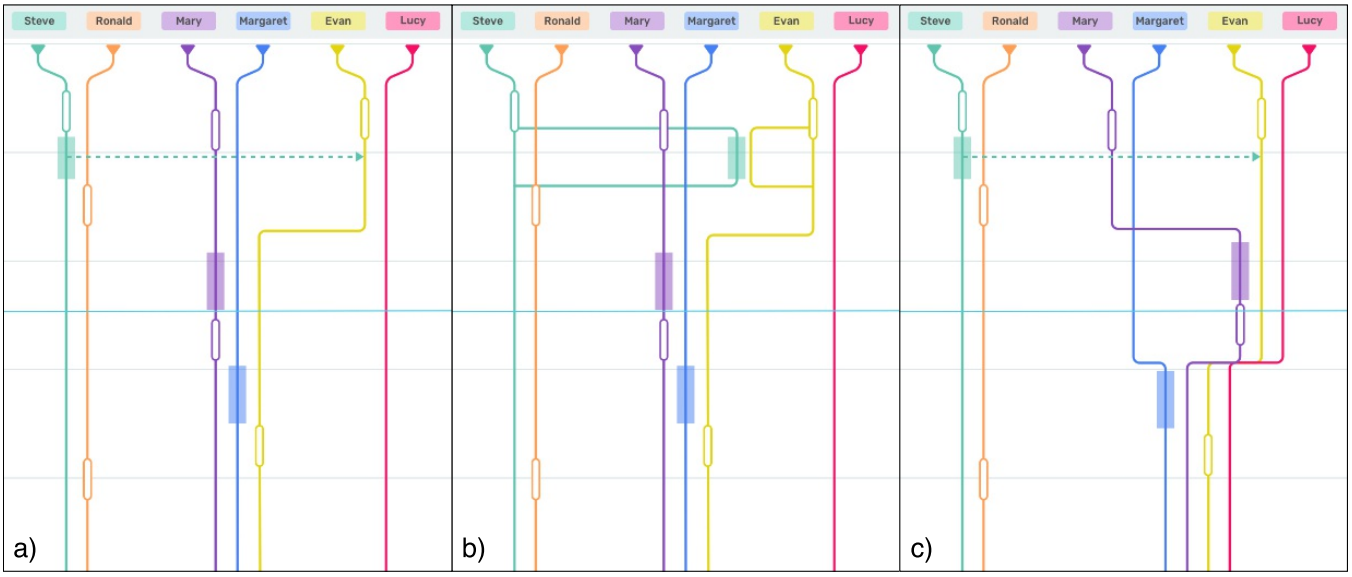

Figure 5: The different definitions for relations between affordances and different approaches for grouping them together result in different looking interactions views for the same story: a) exclusive interactions as grouped affordances related by participation, b) inclusive interactions as grouped affordances related by participation, c) exclusive interactions as grouped affordances related by location.

previsualizations, we generate a behavior tree based on these affordances which plays animations and handles movements. Our system uses the ADAPT [21] framework to generate the behavior trees.

Behavior Trees for Script Visualization

Behavior trees have been used before in ADAPT to display animations and sequences of animations. For CARDINAL, behavior trees have the additional benefit of providing an easy structure to encode dependencies among different affordances. Our implementation is based on the same framework as ADAPT and comes with the basic building blocks of behavior trees, such as the action nodes, sequence nodes, selector nodes, loop nodes and wait nodes. CARDINAL uses these node types to encode the movie script in a behavior tree by forming blocks of overlapping affordances, effectively handling dependencies in a way that keeps all potential errors local to each block. The behavior tree in CARDINAL is constructed by a dispatcher that maps affordance descriptions to their corresponding instances:

$$d(a) : \mathbf{A}_S \rightarrow \mathbf{N}_S \cup \varnothing \tag{1}$$

The function d maps each affordance instance of a scene to either a node in that scene, or to nothing, $\varnothing$. If an affordance instance is mapped to nothing, there is no behavior tree node implementation available for that affordance, which makes it a virtual affordance. Each behavior tree node in CARDINAL is annotated with the start and end time of the corresponding affordance. Additionally, each node has access to its affordance instance, and therefore to its owners and users, allowing nodes to affect the participants of their coupled affordance instance in terms of a predefined, shared behavior. The nodes are represented in the equation below, where t_s and t_e are start time and end time.

$$\begin{aligned} \mathbf{N}_S &= \{d(a) \mid a \in \mathbf{A}_S\} \\ n \in \mathbf{N}_S &=< a, t_s, t_e >, t_s \leq t_e \\ a \in \mathbf{A}_s &= \text{Source affordance instance.} \end{aligned} \tag{2}$$

An example behavior tree is presented in Figure 6.

Tree Construction

CARDINAL groups overlapping affordance instance nodes $n \in \mathbf{N}$ into parallel sequences and prepends them with a wait node with access to a shared watch. If a new node overlaps multiple existing nodes, the existing nodes are first grouped into a single sequence node and that node is then grouped in parallel with the new node. This approach keeps errors in check by making sure they stay block-local.

For example, A block b_2 will only start executing once its predecessor block b_1 has finished executing because the behavior tree effectively encodes a dependency between block b_2 and block b_1: $b_1 \leq b_2$. If an affordance in block b_1 is

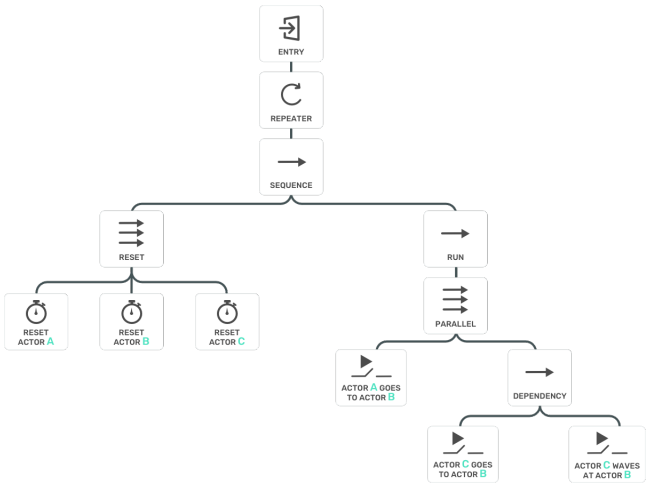

Figure 6: An example behavior tree used in CARDINAL

out of control and takes longer than planned, the behavior tree ensures that the story will be correctly displayed again as soon as it enters block b_2, effectively keeping an error block-local. This approach has the downside that there is a degenerate case where each inserted affordance overlaps with the previous one, resulting in one giant block without encoded dependencies. In the end, the result of this dispatching process is a behavior tree that encodes the written story.

CARDINAL uses two subtrees that are looped. The first subtree, the reset-tree, resets all actor positions and orientations to their original values. The second subtree, the story-subtree, contains the affordance- and compound-nodes.

Affordance Nodes

Affordance nodes are behavior tree nodes that are instantiated by the dispatcher, which turns an affordance instance into an affordance node that has access to its source instance. These affordance nodes are then inserted into the behavior tree as described in the previous section. Something that is not obvious is that these affordance nodes themselves have children. Affordance nodes are therefore compound nodes: $n \in \mathbf{N}_S =< a, t_s, t_e, C >$, where $\mathbf{N}$ is the set of all BT nodes, and $\mathbf{C}$ is the set of the BT nodes that are child nodes.

To provide the functionality needed to generate character visualizations by animating 3D models, CARDINAL uses a similar approach to ADAPT [21], where each character is controlled by an animation controller that is split into three layers: Head, Hands and Body. Animations for these layers are cleverly blended together, but only one animation per layer is allowed at any time. If a new animation is played for the same layer, the old animation will stop and the controller

blends into the new animation. These animation changes are available as behavior tree nodes and are the building blocks of affordances. All of this processing done to produce behavior trees can then be used for 2D or 3D preview.

Previews

2D-Preview. The 2D-preview is a top view of the scene. This preview excludes all detail apart from locations of actors and important objects and hence is perfect for planning movement and spatial relationships between actors. The 2D preview is based on the state managed and modified by the behavior tree. Each actor is assigned a symbol in the color of that actor, that shows the the location and the direction they are looking. Another feature for which the 2D-preview provides functionality is the camera. The camera is a smart object type with its own set of affordances that deal with camera movement and focus. The 2D-preview helps you visualize the field of view of the camera and allows users to verify whether all desired actors are in view. This preview is shown in Figure 7.

3D-Preview. The 3D-Preview allows authors to watch their stories as they are authored. Affordances that are added to a scene are immediately translated into animations represented by behavior tree nodes and can then be viewed in this preview. This view allows users to visualize the locations of actors as well as how they interact with each other. It shows all parts of an interaction: When, Where and How. This makes the preview less focused than other, more specialized visualizations, but provides an animated version of the story. The 3D-Preview uses the ADAPT framework [21] and the example is presented in Figure 8.

7 DEMONSTRATION AND USER STUDY

We demonstrated CARDINAL and its efficacy of authoring scripts, which is shown through this paper. Figure 2 demonstrates the various views of CARDINAL used to author a script. Figures 3, 4 and 5 show the usage of the interaction view of CARDINAL for editing the script. Finally, Figure 7 demonstrates the visualization features of the CARDINAL system. Please look at the supplementary video for more detailed demonstration.

We also conducted a user study to assess whether the visualization possibilities of the CARDINAL system help the user to understand stories better. Each subject was asked to answer comprehension questions about a given script using either the CARDINAL system or a control system. As control system we used trelby, an open source script writing system similar to final draft.

Method

Each subject was provided with a short introduction. The introduction included a tutorial for every system we used in the study. All the subjects carried out the experiment on a similar PC with two monitors. One monitor showed the script the other one showed the questions. To prevent any learning effect we prepared two different stories (A and B). The stories had the same structure e. g. the same number of actors, interactions and objects. We prepared questions which we adopted to the respective story, not differentiating on the logic. Every subject answered the questions about one story using the traditional scripting tool and then answered the questions about the other story using the CARDINAL system. We used the number of concurrent interactions in the script to control the difficulty of the task. Befor every set of questions we asked a control question to ensure that the adjustment to the new system does not influence our results. We assigned story A and story B evenly to the two systems in order to decrease the systematic error. The answers to the questions were typed into an editor window where a script was running that enabled us to measure the time used to answer each question precisely. After the experiment the subjects were asked standard usability questions about the CARDINAL system. The subjects were all Disney Research associates or their friends. We rewarded each of them with a cinema ticket at the end of the experiment.

Metrics

Factors. The system used to answer the questions and the story A and B were treated as factors in our experiment.

Dependent variables. We logged two main metrics to capture the performance of each subject. The first measure is the time needed to answer the questions. We measured it in seconds. The second measure was correctness of the answers. We handled correctness strictly and marked the questions either correct or incorrect. These metrics are treated as dependent variables (DV).

Results

We recorded 16 subjects from the age of 22 up to 39, 37.5% females. The subjects didn't have any prior experience in scriptwriting. All subjects used both scripting systems with two different stories. The results showed a decrease in the time used to answer question when the CARDINAL system was used.

The statistical analysis was conducted as follows: First we verified with a paired t-test that the system used had no influence the correctness of the answers. Then we conducted a 2-way mixed model ANOVA to confirm that users are significantly faster to find content in a story when using the CARDINAL system compared to a traditional system.

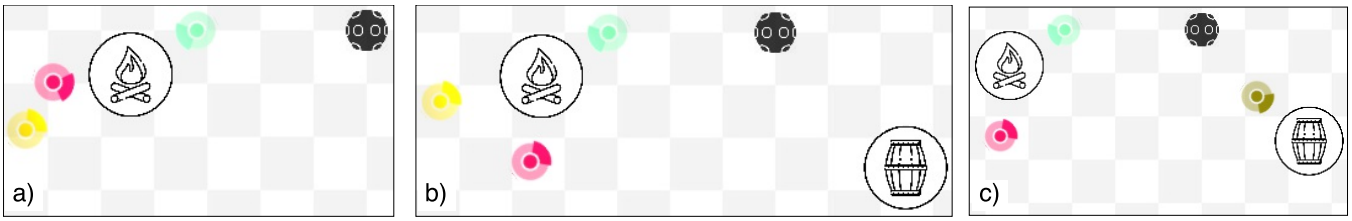

Figure 7: The 2D-Preview is mainly used to resolve conflicts and issues regarding positioning and movement of characters. Authors can resolve conflicts by making actors move to different positions: a) the user sees that three characters that are not supposed to like each other are standing around a campfire, b) the user adds another object, a barrel, c) the user makes one of the three characters walk to the barrel instead of the campfire to resolve the situation.

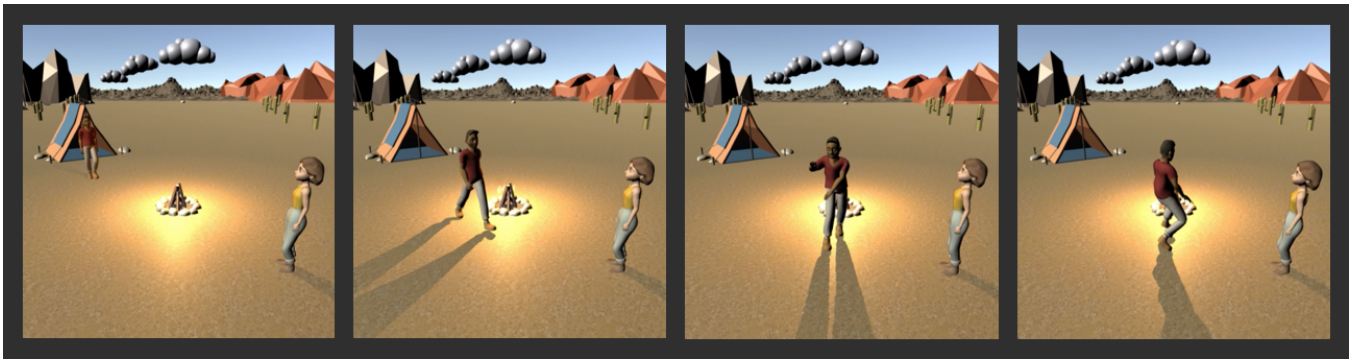

Figure 8: An example sequence of images from the 3D-Preview

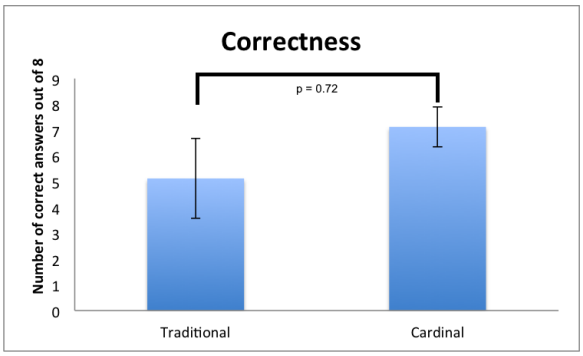

Figure 9: Average number of correct answers for every system used. Paired t-test showed no significant difference in average number of correct answers depending on the system used (p=0.72).

Paired t-test. A paired-samples t-test was used to determine whether there was a statistically significant mean difference between the sum of all correct answered questions using the CARDINAL authoring tool compared to traditional scripting. Data are mean ± standard deviation, unless otherwise stated. No outliers were detected that were more than 1.5 box-lengths from the edge of the box in a boxplot. The assumption of normality was not violated, as assessed by Shapiro-Wilk's test (p >.5). Participants did not differ significantly in the amount correct of answers, whether they were shown the CARDINAL scripting tool (3.75 ±1.3) or the traditional script writing tool (3.94 ±1.2), t(15) =3.6, p = .72, d=0.009.

Thus, we ensured that the difference were not due to the quality of answer but rather due to efficiency. Accordingly, all the following analysis were on time the task required.

2-way mixed model ANOVA. We treated the system as within subject and the two stories as between subject. There were no outliers, as assessed by examination of studentized residuals for values greater than 3. The data was normally distributed, as assessed by Shapiro-Wilk's test of normality ($p > .05$), but only trending for CARDINAL story for those who started with story b ($p = .04$). As the normal Q-Q plot showed approximate normality, and ANOVA can tolerate data that is non-normal (skewed or kurtotic distributions) with only a small effect on the Type I error rate, we proceeded without any transformation of data. There was homogeneity of variances ($p > .05$) and covariances ($p > .05$), as assessed by Levene's test of homogeneity of variances and Box's M test, respectively. There was no statistically significant interaction between the authoring tool and the plot of the story, $F(1, 14) = .567$, $p < .46$, partial $\eta 2 = .04$), and also no

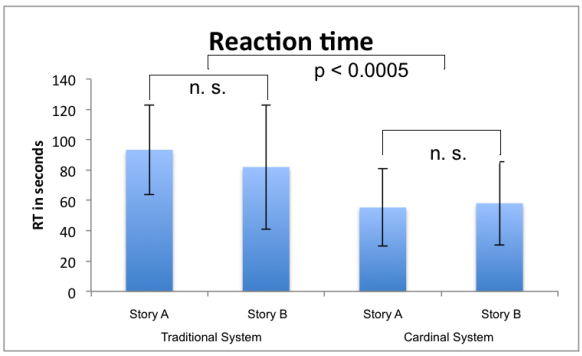

Figure 10: Average of time used to answer a question for every system used and every story analyzed. While the 2-way mixed ANOVA showed no significant difference in the time the task required between the two stories it showed a significant decrease in time the task required when the CARDINAL system was used compared to the traditional system (p<0.0005).

main effect of story plot was significant ($RT, F(1, 14) = .23$, $p < .64$, partial $\eta 2 = .16$. Thus any systematic effect of story content on extracting relevant information from script writings can be discarded.

The main effect of tool, however, showed a statistically significant difference in mean RT for different script writing tools ($F(1, 14) = 29.5$ $p < .0005$, partial $\eta 2 = .68$). We did find similar results when further tested whether main effect for script writing tool was different for the interaction view ($F(1, 14) = 9.48$, $p < .008$, partial $\eta 2 = .4$), or the preview ($F(1, 14) = 25.37$, $p < .005$, partial $\eta 2 = .64$), and neither a main effect for nor interaction with story plot ($p > .05$).

Usability. The usability user study revealed that users feel both more comfortable as well as more confident using our provided visualizations. Some users noted that having a summary of actions and interactions for each actor in the scene would help tremendously to identify dominating figures in the story. Furthermore, being able to smoothly move through time increases the value of the three dimensional preview immensely. Some users notified us about more subtle animations being really hard to see, raising the need for much more intricate camera controls for the visualizations. Fortunately, all features apart from the actor summary are already planned for the final product.

8 CONCLUSION

Limitations and Future Work

In this paper, we demonstrate the benefits of using CARDINAL as an authoring system for movie scripts. The user study shows that the scripting industry is in dire need of innovative authoring systems. Almost all the subjects mentioned that it was laborious to work with the traditional system. They were surprised, when we told them that this is still state of the art in the industry. By providing the user with various visualizations CARDINAL offers authoring possibilities that enable the user to narrate complex stories, limited only by his own creativity. The user study confirms the premise that users spend less time and effort and rate it subjectively easier to find specific information in a story using the CARDINAL system compared to a traditional one. In terms of correctness of the answers, the study showed no significant difference between the two systems.

In order to make the two authoring tools comparable (which is difficult as they have different features), we tried to focus on efficiency and response time while keeping the difficulty of finding the correct answer constant, which was successful (t-test). However, future studies using different story plots especially in terms of complexity can confirm whether CARDINAL also helps in increasing the success rate of finding correctness and quality of answers. Additionally, one should test, whether CARDINAL is particularly useful the more complex the plot gets.

The study was designed to extract relevant information during the production of a movie, rather than on script writing itself. However, the demand of extracting correct information in an efficient way is a demand within the movie industry as confirmed by qualitative feedback from scriptwriters that were not included in the study, and thus demonstrates external validity of these results. Future studies including different approaches and study subjects could help to explore in what area the tool is most helpful to use. The user study has motivated several features to integrate into the prototype. The subjects have requested functionality to selectively activate or deactivate content in the different visualizations. They explicitly asked for the possibility to focus on interactions for a particular actor in the interaction view.

In the user study we compared CARDINAL to an industry standard script. There are systems with a limited degree of visualization possibilities. To provide an extended analysis of the performance of CARDINAL compared to existing systems one would have to include this visualizations into a study. However these visualizations don't obey industry standards which makes it difficult to compare. Additionally most of them visualizations require additional user input and are not straightforward to use.

Additionally, future work can also focus on extending visualization of movie scripts to visualization of other textual narratives. Scripts provide the system with a structure of text and dialog that stories in other forms do not; however, other work has been done to understand information from stories [8]. The ability of visualization quality depends upon the ability of the natural language parser, which acts as a

limitation for CARDINAL. However, as seen in the demonstration video, CARDINAL clearly still can be used as a state of the art tool that can be used for authoring of movie scripts.

REFERENCES

[1] Adobe Story 2012. Adobe Story. (2012). https://story.adobe.com.

[2] Jordan Barria-Pineda, Julio Guerra, Yun Huang, and Peter Brusilovsky. 2017. Concept-Level Knowledge Visualization For Supporting Self-Regulated Learning. In *Proceedings of the 22Nd International Conference on Intelligent User Interfaces Companion (IUI '17 Companion)*. ACM, New York, NY, USA, 141–144. https://doi.org/10.1145/3030024.3038262

[3] Tom Bartindale, Alia Sheikh, Nick Taylor, Peter Wright, and Patrick Olivier. 2012. StoryCrate: Tabletop Storyboarding for Live Film Production. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI '12)*. ACM, New York, NY, USA, 169–178. https://doi.org/10.1145/2207676.2207700

[4] Danqi Chen and Christopher D Manning. 2014. A Fast and Accurate Dependency Parser using Neural Networks. In *Empirical Methods in Natural Language Processing (EMNLP)*.

[5] David Elson. 2012. *Modelling Narrative Discourse*. Ph.D. Dissertation. Columbia University.

[6] Final Draft 1990. Final Draft. (1990). http://www.http://finaldraft.com.

[7] Mark Mark Alan Finlayson. 2012. *Learning narrative structure from annotated folktales*. Ph.D. Dissertation. Massachusetts Institute of Technology.

[8] Amit Goyal, Ellen Riloff, and Hal Daumé III. 2010. Automatically producing plot unit representations for narrative text. In *Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, 77–86.

[9] Mubbasir Kapadia, Jessica Falk, Fabio Zünd, Marcel Marti, Robert W. Sumner, and Markus Gross. 2015. Computer-assisted Authoring of Interactive Narratives. In *Proceedings of the 19th Symposium on Interactive 3D Graphics and Games (i3D '15)*. ACM, New York, NY, USA, 85–92. https://doi.org/10.1145/2699276.2699279

[10] Mubbasir Kapadia, Seth Frey, Alexander Shoulson, Robert W. Sumner, and Markus Gross. 2016. CANVAS: Computer-assisted Narrative Animation Synthesis. In *Proceedings of the ACM SIGGRAPH/Eurographics Symposium on Computer Animation (SCA '16)*. Eurographics Association, Aire-la-Ville, Switzerland, Switzerland, 199–209. http://dl.acm.org/citation.cfm?id=2982818.2982846

[11] Mubbasir Kapadia, Nathan Marshak, and Norman I. Badler. 2014. ADAPT: The Agent Development and Prototyping Testbed. *IEEE Transactions on Visualization and Computer Graphics* 99, PrePrints (2014), 1. https://doi.org/10.1109/TVCG.2014.251

[12] Mubbasir Kapadia, Fabio Zund, Jessica Falk, Marcel Marti, Robert W. Sumner, and Markus Gross. 2015. Evaluating the Authoring Complexity of Interactive Narratives with Interactive Behaviour Trees. In *Foundations of Digital Games (FDG'15)*. 9.

[13] Heeyoung Lee, Angel Chang, Yves Peirsman, Nathanael Chambers, Mihai Surdeanu, and Dan Jurafsky. 2013. Deterministic Coreference Resolution Based on Entity-Centric, Precision-Ranked Rules. *Computational Linguistics* 39, 4 (2013), 885–916. https://doi.org/10.1162/COLI_a_00152

[14] Christopher D. Manning, Mihai Surdeanu, John Bauer, Jenny Finkel, Steven J. Bethard, and David McClosky. 2014. The Stanford CoreNLP Natural Language Processing Toolkit. In *Association for Computational Linguistics (ACL) System Demonstrations*. 55–60. http://www.aclweb.org/anthology/P/P14/P14-5010

[15] Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Efficient Estimation of Word Representations in Vector Space. *CoRR* abs/1301.3781 (2013). http://arxiv.org/abs/1301.3781

[16] Amy Pavel, Dan B. Goldman, Björn Hartmann, and Maneesh Agrawala. 2015. SceneSkim: Searching and Browsing Movies Using Synchronized Captions, Scripts and Plot Summaries. In *Proceedings of the 28th Annual ACM Symposium on User Interface Software & Technology (UIST '15)*. ACM, New York, NY, USA, 181–190. https://doi.org/10.1145/2807442.2807502

[17] Amy Pavel, Dan B. Goldman, Björn Hartmann, and Maneesh Agrawala. 2016. VidCrit: Video-based Asynchronous Video Review. In *Proceedings of the 29th Annual Symposium on User Interface Software and Technology (UIST '16)*. ACM, New York, NY, USA, 517–528. https://doi.org/10.1145/2984511.2984552

[18] Steve Rubin, Floraine Berthouzoz, Gautham J Mysore, Wilmot Li, and Maneesh Agrawala. 2013. Content-based tools for editing audio stories. In *Proceedings of the 26th annual ACM symposium on User interface software and technology*. ACM, 113–122.

[19] Rushit Sanghrajka, Daniel Hidalgo, Patrick P. Chen, and Mubbasir Kapadia. 2017. LISA: Lexically Intelligent Story Assistant. *Proceedings of the 13th Artificial Intelligence and Interactive Digital Entertainment Conference* (2017).

[20] Shilad Sen, Anja Beth Swoap, Qisheng Li, Brooke Boatman, Ilse Dippenaar, Rebecca Gold, Monica Ngo, Sarah Pujol, Bret Jackson, and Brent Hecht. 2017. Cartograph: Unlocking Spatial Visualization Through Semantic Enhancement. In *Proceedings of the 22Nd International Conference on Intelligent User Interfaces (IUI '17)*. ACM, New York, NY, USA, 179–190. https://doi.org/10.1145/3025171.3025233

[21] Alexander Shoulson, Nathan Marshak, Mubbasir Kapadia, and Norman I Badler. 2014. Adapt: the agent developmentand prototyping testbed. *IEEE Transactions on Visualization and Computer Graphics* 20, 7 (2014), 1035–1047.

[22] Makarand Tapaswi, Martin Bäuml, and Rainer Stiefelhagen. 2014. StoryGraphs: Visualizing Character Interactions as a Timeline. *2014 IEEE Conference on Computer Vision and Pattern Recognition* (2014), 827–834.

[23] Kento Watanabe, Yuichiroh Matsubayashi, Kentaro Inui, Tomoyasu Nakano, Satoru Fukayama, and Masataka Goto. 2017. LyriSys: An Interactive Support System for Writing Lyrics Based on Topic Transition. In *Proceedings of the 22Nd International Conference on Intelligent User Interfaces (IUI '17)*. ACM, New York, NY, USA, 559–563. https://doi.org/10.1145/3025171.3025194