poster

a morphological approach to interactive storytelling

Dieter Grasbon (dieter@grasbon.com), Norbert Braun (norbert.braun@zgdv.de)
Digital Storytelling Department, Computer Graphics Center, Darmstadt, Germany

Abstract

Few attempts have yet been made to implement a story engine for guiding interactive drama. Most of them rely on the computer's ability to generate the story in full detail. By focusing on story guidance at the level of morphological functions as defined by Russian formalist Vladimir Propp, we take a different approach. We do not attempt to provide a model for generating stories in detail. Instead, we expect human authors to create specific interactive scenarios for each function in advance. Our primary concerns are high-level guidance of plot, as well as finding the best compromise between author input and machine generation. By providing access to the story model itself, we allow authors to control the story at all levels of detail. Our first prototype features a wide variety of plots being generated from a limited number of scenes.

Keywords: story engine, nonlinear, interactive storytelling, narrative intelligence, Propp

Project URL:
http://www.zgdv.de/zgdv/departments/z5

Introduction

Story engines are tools that tell interactive stories. It is evident that static narrative structures are in conflict with interactivity. Interactive storytelling is a live experience. Therefore, we need a run-time engine taking care of the user's experience as the story keeps unfolding.

Interactive drama is the class of narratives generated by story engines. In most cases, it allows the user to step into the role of the protagonist and witness the events from a first person perspective.

In recent years, several interactive storytelling systems have been developed. The Erasmatron, a story engine developed and implemented by C. Crawford, is based on a sophisticated world model. It seeks to balance character-based and plot-based approaches by using verbs as the basic components of action. Crawford does not believe in story generation through algorithms and therefore plans for the author to create a useful set of verbs that the engine can work with [1]. When designing a story engine for the DEFACTO project, N. M. Sgouros followed a different approach. Using a rule-based system, he aimed at shaping interactive plot into the structure of Aristotelian drama by modeling rising conflict between the characters [2]. M. Mateas and A. Stern are working on an interactive story world. They define beats as the basic units of plot and interaction [3].

All of these approaches control the plot at a very detailed level. The advantage of these designs lies in providing the user with frequent opportunities for influencing the plot. Although this is obviously an important goal for interactive storytelling, we feel technology does not yet provide us with means to create meaningful stories under this condition.

In this paper, we will describe a different approach to interactive storytelling. We have chosen to deal with interactive plot at a higher level – the level of morphological functions as defined by Russian formalist V. Propp [4]. Our story engine is designed for a mixed reality scenario in which the user wanders through physical space while wearing augmented reality devices.

Adaptation to user preferences

It is difficult for us to imagine the potentials of interactive narrative, because we grew up with linear media. One of the opportunities we envision is shaping the overall experience according to the needs of the user. If films can be classified in regard to violence, language and sexual activity, this should be possible for the elements of interactive stories, as well. We have implemented a system that chooses which scenes to show according to the age and preferences of the user. By not offending the personal tastes of the audience, we believe emotional immersion can be improved.

The story model

P. Sengers points out that narratives are always particular [5]. It is in the details that the shaping influence of the author becomes visible. We do not believe that machines are able to create convincing details. Therefore, we have decided to put the responsibility of generation at scene level and below entirely into the hands of the author. This is a critical decision in our design, since it has some major drawbacks. We cannot expect human authors to write more than a few dozen scenes, as well as the rules defining their flow. Basically, a small number of scenes are all that our engine can work with. However, we believe that even under these restrictions, our system features a sufficient variety of engaging plots.

The story model of our prototype is based on V. Propp's "Morphology of the folktale", which was written in 1928. As the basis of his study, he used an arbitrary sample of 100 Russian fairy tales. As mentioned by S. Bringsjord and D. Ferrucci [6], Propp's system of classification inspired research on story grammars, one of the fundamental approaches to story generation. Since our goal is not story generation, but rather guiding interactive drama, we use Propp's work to create rules and algorithms for the run-time engine instead of defining a grammar. The application of Propp's morphological approach to interactive storytelling was suggested and sketched by J. Murray [7].

Propp defines function as follows: "Function must be taken as an act of dramatis personae, which is defined from the point of view of its significance for the course of action of a tale as a whole" [4]. Propp uses capital letters to denote functions. For example, "A" represents a function in which the villain causes harm or injury: In Star Wars, Luke's foster parents are killed. Later, the protagonist is tested by a potential donor (function D). Depending on his reaction (E), he is either given a magical agent (F) or not. It is important to note that one and the same action can have different functions depending on the context of the surrounding plot. Thus, functions are independent of the specific characters, as well as of the particular actions that perform them.

We have chosen to implement Propp's system for a variety of reasons. B. Linane-Mallon and B. R. Webb argue that the elements of stories are highly interrelated [8]. P. Sengers [5] points out that the cause and effect of narrative events are more important than the events themselves. Therefore, it seems that stories cannot be divided into discrete modules. We believe Propp has solved this problem by defining functions in the context of the surrounding story.

Sengers also notes that narratives are specific to their culture. By starting out with a model based on fairy tales, we believe to have minimized cultural barriers of understanding, since fairy tales and myths are known to be very similar across different cultures [9]. However, we encourage the author to reinvent the story model according to the specific set of stories that are to be told.

Comparing Propp's classification of plots to other approaches, e.g. [10], we see its primary advantage in the fact that it integrates all possible variants in one unified model, making it especially applicable to interactive storytelling. Instead of viewing different plots as distinct and linear entities, Propp's classification is continuously aware of the storyteller's branching possibilities between morphological variants.

Architecture

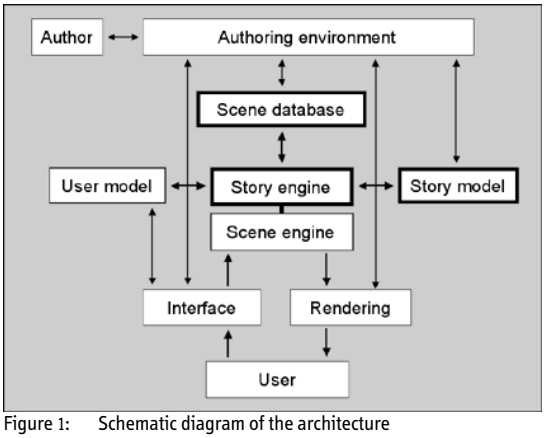

Figure 1: Schematic diagram of the architecture

Figure 1 gives an overview of our system. Modules in bold borders have been implemented in the prototype. It consists of the story engine, a story model and a small number of scenes. In the system we envision, interface and rendering modules separate the story engine from the user, making the engine independent of input (keyboard, tracking, ...) and output (text, animation, ...) modalities. The interface module translates user interactions into semantic abstractions that the engine can work with. The user model stores static, as well as dynamic data, on the player. Static information includes age, gender and preferences of the user. Dynamic information is derived by processing user interaction and then concluding if the player is bored, entertained or overloaded by the material presented. The engine uses both kinds of data when choosing the next content to be shown. N. Szilas [11] has suggested a similar user model.

The author exerts direct influence on every part of the system except the story engine and user model. First, the author modifies the story model according to the set of stories that are to be told. Then, detailed descriptions of a number of scenes are created. Finally, the author designs the appearance of the interface and influences the way in which characters and scenes are rendered. Each scene has to correspond with a function in the story model and is annotated with information about its minimum and maximum duration, context requirements, characters, setting, levels of violence, etc., unless these factors are generic.

The scene engine plays scenes that the story engine has selected. User interaction is analyzed and mapped to a story act, which is handed back as a scene result to the story engine. In case of the prototype, scene content is displayed by text output only. Here, the player directly chooses the desired story act from a set of multiple choices.

Levels of abstraction

The story engine works with two levels of abstraction. At the upper level, a sequence of functions is selected in real-time. We use the term function in the same way as Propp does, as an abstraction from characters, setting and action while emphasizing the function's role in the surrounding plot. User interaction and other constraints can result in functions being skipped, as well as in the choice of a different variant of a specific function. At the lower level, a particular scene is executed for each function. They are either being generated or have been authored in advance.

We see functions as classes, and scenes as instances of those classes. A scene possesses unity of time, space and action, thus facilitating temporal, spatial and emotional immersion of the user. Between one scene and the next, leaps in time are possible and even desirable, since we do not want to bother the player with less exciting events.

Polymorphic functions

User interaction cannot change the morphological function of most scenes after their beginning. However, we have implemented a few polymorphic functions. Our conception of polymorphic functions was inspired by the notion of polymorphic beats, introduced by M. Mateas and A. Stern at a different level of detail. They refer to the most basic elements of action, which can have different outcomes depending on user interaction [3]. We are referring to scenes composed of a number of actions at a specific location. When the user enters a scene that corresponds to a polymorphic function, its outcome is not yet determined. It is important to note that by outcome, we do not mean the content of the scene (which we allow to be flexible in case of non-polymorphic functions as well), but rather its function for the overall story. Thus, user interaction and other dynamic factors influence which morphological variant is chosen for the scene after its execution.

We have implemented polymorphic functions for the outcome of attempts of villainy (function "A" in Propp's system) and for the reaction of the hero to riddles and tests (E). User interaction decides if these functions succeed or fail, which has direct consequences for the remaining plot. If we need these functions to succeed (as in case of villainy), we either repeat them with different scenes or choose a non-polymorphic variant with the desired outcome.

Scene Selection

Selecting the relevant parts and leaving out the boring ones is the essence of storytelling. In our case, we assume the author is mindful that every written scene is relevant. Therefore, the engine needs different criteria for selecting the scenes to follow at any given time.

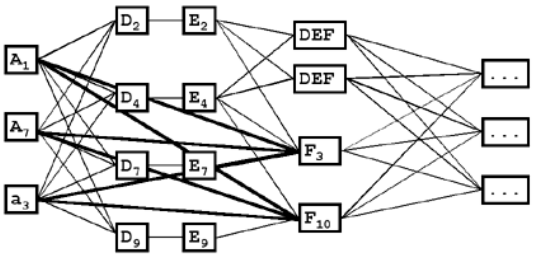

Figure 2: A section of the story space

First of all, some constraints are encoded with the functions that the scenes perform. Figure 2 shows a section of the story space. The section is traversed from left to right while each square represents a function. It is important to note that the connections between them are not fixed, but rule-based. Propp discovered that any function D can follow function A, but each D requires a specific E. Furthermore, D and E can be repeated in some cases or even skipped altogether. We intend to use this kind of visualization for the authoring environment. The squares could be filled with sketches of scenes fulfilling the function and be connected by the author in an intuitive way. Using the sketches, a variety of (linear) storyboards could be generated according to different user interactions.

Our second criterion is time. The user chooses a certain time frame for the overall experience that has to be met. If the player's pace of interaction is very slow, many functions will be skipped. User timing, therefore, has an indirect effect on the remaining plot. Before selecting a scene, the engine checks whether it can be played in the remaining time. If the function of the scene requires other functions to follow, the durations of their corresponding scenes have to be taken into account, as well. With each scene description, the author has to encode an interval of its shortest and longest possible duration.

Concerning our augmented reality scenario, the number of possible scenes is further reduced by the current location of the user in physical space (unless scenes can be generated for each setting from templates). The user will feel immersed in the story, because it keeps unfolding wherever (s)he goes.

Yet another criterion is the current context of the story. The author encodes a list of context requirements with each scene description. Scenes can create new context (i.e. if they introduce characters to the story). They can require context to be present (i.e. a certain type of misfortune) and they can remove context (i.e. if that misfortune is liquidated).

Furthermore, the user model's current state is taken into account. If the player is a kid, the engine will avoid showing violent scenes. If the user seems bored, the system will prefer scenes labeled as exciting. On the other hand, if the player seems overloaded by the material presented, the engine will select less demanding scenes. If the system is still left with a set of choices after processing these criteria, a random decision is made.

Implementation

Our current prototype is written in Prolog, which we have chosen due to its rapid prototyping capabilities. The story model is implemented as a meta-program and a set of rules. Morphological functions are executed as operations of the meta-program. A repeat-until instruction is supported as well. It can be used to ensure that the attempt of villainy eventually succeeds. Rules define the interrelationship between different functions, such as implication and exclusion.

The story engine consists of a meta-interpreter that processes the meta-program encoded in the story model. If the current instruction is a morphological function, the system checks if it can be played in the remaining time. Using a forward propagation algorithm, it takes into account the duration of each implied function as well. If enough time is left for the current function, all matching scenes are screened according to the other criteria described above. If no scene meets the requirements, the function is skipped and the interpreter proceeds to the next instruction of the meta-program. Otherwise the selection is handed over to the scene

engine for execution. After the scene has been played, polymorphic functions are instantiated with the story act of the player. Then, the story engine updates dynamic context as well as a list of required functions in the database. Finally, the meta-interpreter steps to the next in-struction of the meta-program. A detailed description is given in [12].

Shortcomings and limitations

Our current prototype has a variety of limitations that we hope to overcome in the future. Several modules have not yet been implemented. However, the story engine can be tested using text in- and output. Unfortunately, the author has to write each scene by hand. It would be desirable to generate scenes on demand. Due to our split-level approach, this issue does not affect the overall design: It does not change the corresponding function if a scene like villainy(abduction, dragon, princess, castle) has been pre-authored or is generated from scratch.

Conclusions and future work

We consider high-level guidance of plot to be the primary concern of our design. Morphological functions provide us with means of abstraction that respect the story context. Our approach allows for authorial control at all levels while generating a large variety of plots. We have adapted Propp's model for interactive storytelling. This was achieved by introducing polymorphic functions, dynamic context and time management. Players evaluate the system's output as coherent, engaging and varied while being well adapted to their interaction. Future work will focus on the implementation of the remaining modules.

References

[1] C. CRAWFORD, Assumptions underlying the Erasmatron interactive storytelling engine, in Proceedings of the AAAI Fall Symposium on Narrative Intelligence, 1999.

[2] N. M. SGOUROS, Dynamic Generation, Management and Resolution of Interactive Plots, in Artificial Intelligence 107(1), 1999, pp. 29-62.

[3] M. MATEAS AND A. STERN, Towards Integrating Plot and Character for Interactive Drama, in K. Dautenhahn, (Ed.), Proceedings of the 2000 Fall Symposium: Socially Intelligent Agents: The Human in the Loop, AAAI Press, Menlo Park, CA, pp. 113-118.

[4] V. PROPP, Morphology of the Folktale, in International Journal of American Linguistics, Vol. 24, Nr. 4, Part III, Bloomington, IN, 1958.

[5] P. SENGERS, Narrative Intelligence, in K. Dautenhahn, (Ed.), Human Cognition and Social Agent Technology, Advances in Consciousness Series, John Benjamins Publishing Company, Philadelphia, PA, 2000.

[6] S. BRINGSJORD AND D. FERRUCCI, Artificial Intelligence and Literary Creativity: Inside the Mind of Brutus, A Storytelling Machine, Lawrence Erlbaum, Mahwah, NJ, 1999.

[7] J. MURRAY, Hamlet on the Holodeck: The Future of Narrative in Cyberspace, MIT Press, Cambridge, MA, 1998.

[8] B. LINANE-MALLON AND B. R. WEBB, Evaluating Narrative in Multimedia, in DSV-IS'97, 4th International Eurographics Workshop, Granada, Spain, 4-6 June, 1997, pp. 83-98.

[9] E. FROMM, The Forgotten Language: An Introduction to the Understanding of Dreams, Fairy Tales and Myths, Grove Press, New York, NY, 1957.

[10] R. B. TOBIAS, 20 Masterplots: Woraus Geschichten gemacht sind, Zweitausendeins, Frankfurt a. M., 1999.

[11] N. SZILAS, Interactive Drama on Computer: Beyond Linear Narrative, in Proceedings of the AAAI Fall Symposium on Narrative Intelligence, 1999.

[12] D. GRASBON, Konzeption und prototypische Implementation einer Storyengine: Dynamisch-reaktives System zum Erzählen nichtlinear-interaktiver Geschichten bei größtmöglicher Spannung, gedanklicher Immersion, Identifikation und Motivation des Spielers, diploma thesis, Technical University of Darmstadt, 2001.