Dynamic Plot Generating Engine

Maria Arinbjarnar
University of York
Heslington, York, YO10 5DD
maria@cs.york.ac.uk,

Abstract

When creating interactive drama computer games then it is frequently useful to be able to generate a plot to use as a background for the drama and to use as a knowledge base for Non-Player Characters.

In this article a new engine is proposed that dynamically creates new game plots for a murder mystery based game. The game engine uses Bayesian networks to create a new plot based on a probability map of a typical murder mystery novel. To abstract plot elements for the construction of the Bayesian network a morphology, similar to the morphology of the Russian structuralist Vladimir Propp, is created for murder mysteries. The engine creates a complete and consistent murder mystery that is solvable with logical inference. For each new game the engine generates a unique plot, a murder mystery with the necessary attributes that are needed for such a plot to be logically consistent, coherent, and complete, and make sense from the human perspective. This game engine is responsive to preset constraints rather than a pre-authored narration, which opens up numerous possibilities for integration in computer games where the constraints can be manipulated by interactions from game events and player interaction.

1 Introduction

Quite a large portion of the game market focuses on producing games with a high degree of narration. These games, especially Role-Playing Games, have very advanced game engines offering all the best in combat, graphics and sounds that is available on the market. These games are specifically trying to meet a demand for non-linear game interaction. Their answer has been to offer increasingly multi-linear narrations. The problem with that is that the games are still linear: The player is still forced to follow a set line or lines of preset prerequisites to advance or complete the game. This increases the danger of the player becoming stuck because she did not follow the set storyline. More importantly, it ruins the replay ability of the games because the narration is not very responsive to player interactions and the average player does not care to play the game more than ones.

I propose a Dynamic Plot Generating Engine (DPGE) as a first step in solving this problem. The DPGE utilizes a Bayesian net, which is a probability-driven causally connected network [9] to generate a new murder mystery plot for each new game initialized.

Any narrative will contain a plot that details its main events distinct of the characters involved in them or the themes illustrated by them, and may contain subplots to describe the structure of narratives within the main narrative that are not a direct or necessary part of the main plot [10, 2]. The plots generated by the DPGE describe the main events that are necessary to describe a murder mystery and provide the initial conflict for a murder mystery based game.

The construction of the plots is based on a morphology similar to the morphology of the Russian structuralist Vladimir Propp [11]. This morphology abstracts necessary actions and events to create complete and consistent murder mystery plots. The plots are complete in such a way that they include a victim, a murderer, motive, clues, suspects, weapons and scenes. The plots are consistent which means that expected causality applies e.g. the murderer will have opportunity and motive to commit the murder. Evidence will be with respect to the cause e.g. male murderer will leave a male sized footprint and if the wound was open then it is likely that there will be blood on the murder weapon.

In section 4 I discuss other work that uses Bayesian networks or Propp's Morphology. In section 2 the morphology that is created is described and its relation to Propp's work is discussed. In section 3 a run of the engine is detailed step by step. In section 5 some experiments and implementations of the engine are described and possible usability is discussed. Section 6 summarizes the findings. Section 7 discusses possible future extensions or developments to the engine.

2 Morphology

Vladimir Propp (St Petersburg, April 29, 1895 - Leningrad August 22, 1970) a Russian structuralist observed that the Russian fairy tales, classified by Aarne index 300 to 749 [4], are constructed of repetitive themes where actions and functions stay the same but the names and attributes of the

persons change. In this case a function is defined as an act of character and viewed for its significance in the course of an action. This means that each function is independent from the rest of the narration and is only recognized for its effect on characters each time and for its part in completing an action. From his careful observations Propp was able to propose the following 4 rules for the makeup of the set of Russian fairy tales:

1. Functions of characters serve as stable, constant elements in a tale, independent of how and by whom they are fulfilled. They constitute the fundamental components of a tale.
2. The number of functions known to a fairy tale is limited.
3. The sequence of functions is always identical.
4. All fairy tales are of one type in regard to their structure.

Propp numbers the functions and states correctly that no function may happen after a function of a higher number has happened. This is to ensure a coherent time line. Propp managed in this way to map a sort of unintentional logical time line for the fairy tales. Some of Propp's functions are paired such that one will logically follow another. Yet the functions are not causally dependent such that the former has to have come before the latter. The latter function can happen independently from the former [11].

Example 1 (The Proppian function.) *Fairy tales frequently start with an interdiction that is addressed to the hero:*

- *you dare not look into this closet or*
- *don't pick the apples*

This is followed by a violation, the hero does what he had been warned or forbidden to do. If there is no prior interdiction then the hero is seen either violating some common norm, like being late or oversleeping, or he ventures out because of need or because of the urgings of some other character.

The functions and roles of Propp's Morphology are suitable to create a Russian fairy tale. A murder mystery has little in common with a Russian fairy tale and so a completely new set of functions and roles needs to be defined with respect to Propp's rules. I will now describe the roles implemented and then go through the rules one by one as they have already been implemented in the engine.

There are three roles defined in the murder mystery morphology; *victim, murderer* and *suspect*. The murderer alone will satisfy the constraints that define a murderer. These constraints are: all evidence match, has motive, is connected to the victim and had opportunity. There is only one murderer in the generated plot. Each suspect that is not the murderer will fail to satisfy one or more of the constraints that define a murderer. All of the characters are either related to the victim or have a motive to kill the victim or both.

There is no detective role defined because the detective commonly enters the scene after the murder takes place and this solution does not address that specifically.

Rule #1 The first rule of Propp's Morphology asks for functions that are stable, constant elements in a tale, independent of how and by whom they are fulfilled. A murder mystery has a different set of functions than a Russian fairy tale. I have defined the following necessary functions for a murder mystery: *motive, means, opportunity* and *murder*.

The *motive* function joins motives with the murderer and some suspects. Below is a description of each of the motives implemented in the engine.

The *means* function joins the murderer and some suspects with a some weapons. Such that that the murderer and some suspects will be connected to a weapon through evidence such as their hair or fingerprint being on it.

The *opportunity* function joins the murderer and some suspects with on scene evidence, such that that the murderer and some suspects will be connected to the scene by for instance a footprint or some hair found. This would show that they where indeed on the scene and thus had opportunity

Finally the *murder* function ascertains that only the murderer has motive, means and opportunity to commit the murderer and that each of the suspects will be found lacking in one or more of these conditions. How this is done is better explained in section 3

Rule #2 It is clear from the description of functions for rule number one that the functions are limited in number.

Rule #3 The DPGE creates the past for the game and thus the sequence of functions do not follow a time-line but it still holds that a murderer would first need to have a motive then means and opportunity before being able to kill the victim. This sequence of functions is readily apparent if the resulting plot is recounted. Meaning that the plot can always be used by a human to create a narrative.

Rule #4 All the mysteries are created from the same Bayesian network with the same set of function and thus the mysteries are all structured in the same way.

In order to have a set of probable motives I used some of Agata Christies murder mystery novels for good ideas. The result from a light review are the following motives.

Swindle. Either the murderer is swindling the victim and the victim threatens to tell or the victim was swindling the murderer and the murderer wanted revenge. Either way the swindler should be rich.

Blackmail. Either the murderer was blackmailing the victim and feared that the victim would reveal it or the victim was blackmailing the murderer. In either case it is necessary that the one being blackmailed has some dark secret or else he would not be blackmailed.

Wedlock. In Agatha Christie's time it could sometimes be difficult and or bad publicity to get a divorce. This can also apply today that people do not wish to go through the divorce process and lose half their belongings and possibly custody of the children. The constraints in the net for wedlock is that the murderer is the victim's spouse.

Inheritance. This is self explanatory and has the necessary constraint that the victim must be rich and the murderer must be an heir.

Adultery. The victim was having an affair with the murderers spouse and the murderer kills because of pure jealousy.

Revenge. The murderer believes that he has just cause for revenge. The victim must have harmed the murderer in some sense in the past.

Debt. The murderer owed the victim lots of money and could not pay. This has the necessary constraint that the murderer is not rich.

3 The Engine

Step 1, Reading data The engine reads in a text file that details all possible names of persons that can be used by the engine. I picked out names of people that appear in Agatha Christies novels. The engine then reads a text file that details all variable types that should be in the net and their connection to other variables.

Step 2, Drawing the net Each variable in the net is drawn, with the aid of SMILE [3], with respect to it's description in the text file. One node could have the following description:

```
var:V_victims_size,Victims size
states:Tall,Medium,Small
parents:V_victims_sex
values::0.4,0.4,0.2,0.2,0.5,0.3
```

In this example the variables name is *Victims size* and its identity is `V_victims_size`. It has 3 states `{Tall, Medium, Small}`. It has one parent and the identity of the parent is `V_victims_sex`. This means that an arc will be drawn from `V_victims_sex` to `V_victims_size`, with the arrow pointing to `V_victims_size` as sex has a causal affect on a person size. Finally the values of the variables are detailed. Table 1 shows the probabilities (values) for this variable.

Table 1: Victims size

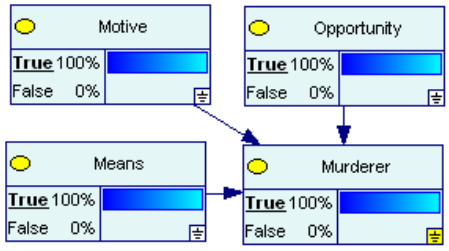
Victims sex	Male	Female
Tall	0.4	0.2
Medium	0.4	0.5
Small	0.2	0.3

The reason for two :: when defining values in the text files is because a flag can be entered to indicate certain types of probability distribution such as: twoRows, normalized, nomatch, etc.

- `values:twoRows:0.4,0.2` makes the engine automatically fill in 0.6 and 0.8 in the lower row, so that it adds up to 1 when the variable has only two states.
- `values:normalized:1,1,3,3,4,2` makes the engine consider the input as part of a whole and transform it to the equivalent part of 1.
- `values:nomatch:male` is used for variables that are defining spouses and indicates that a male can not have a male spouse etc.

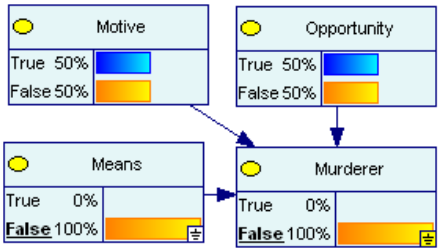
The text file also indicates whether the variables should be nested in a subnet and how many subnets should be created. As an example to create 4 subnets that detail weapons then the text file would read:

Figure 1: Unique murderer



Step 3, Instantiating specific variables The next step is to specifically decide on a murderer by randomly choosing one character. A murderer needs to have motive, means and

Figure 2: Suspect



opportunity so these variables in the murderer subnet need to be specifically instantiated for the murderer to guarantee that they hold true see Figure 1. Similarly to eliminate a suspect as a possible murderer then the suspect can not have motive, means and opportunity for the murder, so one of these variables needs to be instantiated to false for each of the suspects. In Figure 2 the variable *means* has bean instantiated to false. The variable that is instantiated to false is chosen stochastically. The graphical tool GeNIe [1] is used to show the variables of the net.

The result of instantiating the variables in this way is that the rest of the Bayesian net needs to recalculate to compensate for these new "constraints". When the net is recalculated then the variables that connect to the means, motive and opportunity variables are affected by how they where instantiated, so if motive is now set to true then that person must have a motive or the Bayesian net would not compute. Step 4 describes how the rest of the net is calculated.

Step 4, Instantiating the plot Finally all variables that have not been instantiated are now instantiated from the root of the directed acyclic graph, that a Bayesian net is, and to the leafs so as not to get an synchronization error in the net. The instantiation is done by generating a random number between 0 and 1 and instantiating with respect to the probability distribution of the variables states.

4 Related work

There are a some other solutions that use Propp's Morphology or Bayesian networks and here I name some of them.

The Story Engine [12] is based on Propp's Morphology [11] and assists in writing multi linear stories for Role-Play Games. It uses the Propp's Morphology to avoid deadlocks and to assist in creating a literary sound narration.

The Open ended Proppian Interactive Adaptive Tale Engine [8] creates interactive narrations and is in a large part based on Propp's Morphology [11]. An interesting part of this engine is a gossip system that connects the Non-Player Characters (NPCs) together and spreads news and opinions about the current player character in respect to his actions.

In the non-linear interactive storytelling game engine (NOLIST) [6] they utilize a Bayesian network to determine the culprit of a murder mystery; the Bayesian network is dynamically changing in response to actions and observations made by the player. It is not preset but rather it uses the player's moves and the logical inference of the net to determine the culprit. For example if the player finds a body and a gun lying beside the body then the probability that the murder victim was shot with the gun increases. The developers of NOLIST do this to be able to construct an engine that will create a dynamic storyline with respect to player actions and choices. The plot and the culprit are not known by the game engine in the beginning but are determined in the course of the game. Thus NOLIST creates the past in reaction to player interaction. Although NOLIST is highly adaptive to player interaction it might actually be too reactive. Players are likely to play games in similar manners each time and are thus not good at surprising themselves.

The DPGE that I propose to create a murder mystery plot and past life for characters differs from the above solution in one significant way. The DPGE is used to decide the past, that is, what has happened in the world and for the characters up to the point in time that the game starts. The DPGE is not intended to form future actions.

5 Tests

The engine has been tested in various contexts and proves to be easily connected to other applications and flexible to divers needs.

The first test of the applicability of the engine was to create a limited application that showed the plot created each time and provided a simple "who dunnit?" game. That game showed clearly that each new plot generated was distinct and complete.

The "who dunnit?" game provided the player with predefined logical conclusions based on the plot and showed that each plot was consistent such that the murderer and murder weapon could be discovered by deduction.

The DPGE has been connected to two other more extensive applications that show its usage better.

Hexia The engine was also connected to the Hexia dialog control system [13]. Hexia provided means to interact with the engine through MSN Messenger. This provided an interactive mystery where the player could talk with a NPC named Theresa and ask her to perform actions on scene such as "open drawer", "read letter" and "look at desk".

At the beginning of each game Theresa greeted the player and described the scene. The player then needed to tell Theresa what to do next and she then told the player the results of her actions.

As an example the player could have asked Theresa to look at a desk and Theresa then informs the player that there is a drawer in the desk. The player can then ask Theresa to open the drawer. The player does not need to indicate which drawer as the Hexia dialog control system is contextual, it is able to preserve context in a dialog. Theresa will then attempt to open the drawer and if she is successful then she will describe the content of the drawer, otherwise she will describe the problem, e.g. "the drawer is locked!"

At any given time the player could frame a suspect and get informed by the engine whether she was correct or not. The player can only frame suspects that she has become aware of in the game.

This implementation of the engine was demonstrated at the Reykjavik Artificial Intelligence festival (2006). Those who tried it where successful in playing the game and deducting who the murderer was after a few attempts. It should be stated that no formal tests where done that would prove this.

Authoring the dialog for NPC Theresa took less than three 40 hour work weeks.

Rational Dialog in Interactive Games The engine has also been used to provide a knowledge base for autonomous NPCs [5]. It created a background and common understanding of the world for the NPCs created and defined relations to other characters.

When creating the NPCs then first the plot is created and then the plot is used as the initial configuration of the world. Each NPC is created from the plot. The NPCs knowledge can then be configured such that the murderer knows everything about the murder but that other NPCs are fairly clueless but still have the sufficient knowledge base to realize that if they find a bloody knife then that knife is likely to be the murder weapon, especially if they know that the wound looks like a knife wound. This is essential so that the characters can decide on what to say when chatting with other NPCs or the player. If the NPCs want to make sense that is.

Possible applicability The engine is very useful to create a background for computer games. The background can be used in various ways as the tests show. Since the input is through text files then the engine can be easily controlled and new plot elements can be added to it simply by changing the text files.

Although the engine is now set up to produce murder mystery plots then it can very well be used to construct other types of narrative plots by changing the input such as for instance to create plots for fantasies or epic stories. It would be good to start by determining some morphology for these stories first.

The engine can be responsive to player action after the game has started even though the engine creates the background and is not intended to create narration for future actions. The engine could for instance be used to to create subplots in an interactive game on the fly. The text files could be manipulated by the game to better represent a players profile and thus provide subplots that would better suit the players interests.

In the same way the engine could be used to introduce new characters or scenes to the ongoing narration of an interactive role-play game.

6 Conclusions

Based on the tests that have been performed on the DPGE then it is clear that it is capable of producing complete and consistent plots. The plots are coherent to human testers and the testers are able to deduct correctly who the murderer is. The DPGE can to a large extent be manipulated through a text file and is thus fairly flexible and can be easily extended. It can be used to create new plots or subplots during gameplay.

7 Future Work

The DPGE could be used to construct subplots in the course of a game and it could then be responsive to actions that the player has made when tackling other mysteries that the DPGE has generated. It could for instance adjust difficulty levels of future plots with respect to the players competency in solving previous mysteries.

It would be good to create a more user friendly interface than a text file to manipulate the DPGE. It is possible to create a tool that could be used by people that do not have programming or specific computer science knowledge, such as screenwriters for instance.

It would also be good to use the DPGE to create plots for other types of stories, such as adventure or fantasy stories like the ones that are common in Role-Play Games. In order to do that some of the logic in the DPGE needs to be extracted and made manipulable through a tool or text file.

The Rational Dialog for Interactive Games [5] is part of an ongoing research into rational NPCs for computer games and the DPGE will continue to be developed alongside that work as needed. It will also be useful to connect the DPGE and the work on rational NPCs with other modules such as the dilemma module that encourages structure in emergent drama [7]

8 Acknowledgments

I would like to thank Yngvi Björnsson and Luca Aceto for their advise in building the engine. I would like to thank Helga Waage and Hexia.net for testing the engine and connecting it with the Hexia dialog control system. I would

also like to thank Finn Verner Jensen for his input and expert advise on Bayesian networks and structuring murder mysteries. I would also like to thank Daniel Kudenko and anonymous reviewers for reviewing the article and giving constructive feedback.

References

[1] *GeNIe modeling environment for graphical probabilistic model*. Decision Systems Laboratory, University of Pittsburgh (dsl.sis.pitt.edu), 2007.

[2] Oxford English Dictionary. Online, Oxford University Press, 2007.

[3] *SMILE, reasoning engine for graphical probabilistic model*. Decision Systems Laboratory, University of Pittsburgh (dsl.sis.pitt.edu), 2007.

[4] A. Aarne. Verzeichnis der märchentypen. *Folklore Fellows Communications No. 3*, 1911.

[5] M. Arinbjarnar. Rational dialog in interactive games. In *Proceedings of the AAAI 2007 Fall Symposium on Intelligent Narrative Technologies*, 2007.

[6] O. Bangsø, O. G. Jensen, F. V. Jensen, P. B. Andersen, and T. Kocka. Non-Linear Interactive Storytelling Using Object-Oriented Bayesian Networks. In *Proceedings of the International Conference on Computer Games: Artificial Intelligence, Design and Education*, 2004.

[7] H. Barber and D. Kudenko. Dynamic Generation of Dilemma-based Interactive Narratives. In *Proceedings of the Artificial Intelligence and Interactive Digital Entertainment conference (AIIDE)*, 2007.

[8] C. R. Fairclough and P. Cunningham. AI Structuralist Storytelling In Computer Games. *Proceedings of the International Conference on Computer Games: Artificial Intelligence, Design and Education*, 2004.

[9] F. V. Jensen. *Bayesian Networks and Decision Graphs*. Springer-Verlag, 2001.

[10] G. Prince. *A Dictionary of Narratology*. University of Nebraska Press, 2003.

[11] V. A. Propp. *Morphology of the Folktale*. University of Texas Press, 2nd edition, 1968.

[12] O. Schneider, N. Braun, and G. Habinger. Storylining Suspense: An Authoring Environment for Structuring Non-Linear Interactive Narratives. *Proceedings of the International Conference in Central Europe on Computer Graphics, Visualization and Computer Vision*, 2003.

[13] H. Waage and B. Ingimundarson. *Hexia agent system design overview*. Hexia, 2004.

StoryAssembler generates dynamic choice-based narratives, similar in output to hyperfiction or Choose Your Own Adventure (CYOA) books. These types of narratives have been with us in digital form for quite some time, with early systems like HyperCard [11] and Eastgateâ€™s Storyspace [2] creating a fertile initial ground.Â For player-driven, it was all about state management of variables set before. StoryAssembler: An Engine for Generating Dynamic Choice-Driven Narratives FDG â€™19, August 26â€“30, 2019, San Luis Obispo, CA, USA. the scene even started. To generate procedural terrains with a high level of complexity, at interactive frame rates, we look to the GPU. By utilizing several new DirectX 10 capabilities such as the geometry shader (GS), stream output, and rendering to 3D textures, we can use the GPU to quickly generate large blocks of complex procedural terrain. Together, these blocks create a large, detailed polygonal mesh that represents the terrain within the current view frustum. Dynamic NavMesh Generating. Unreal Engine Feedback for Unreal Engine team. MaxPower42. September 28, 2016, 12:13am #1. I wish UE4 would have better support for procedural level design. And one thing you could improve rather easily (I assume) is how NavMeshes are generated.Â I know way too little about how the engine works internally to judge if simply â€œstoppingâ€ the main thread would speed up NavMesh generation. From my amateurish point of view it just â€œfeelsâ€ like it could be done quicker when smoothly running a game at the same time is not a prerequisiteâ€¦ Since I have a large non-linear level that requires all AI units to be able to move freely, why wouldnâ€™t I â€œwant to be using one giant navmeshâ€? My Dynamic Plot Generating Engine implements a morphology similar to Proppâ€™s morphology discussed. previously and thus satisfies the need for a literary connection and a call for conflict. Additionally it can supply a. new plot at each new game or, if it were a larger continuous game, at each new scene. This makes it highly. responsive to changes in the game and to player interaction without losing connection to literature and the. essentials of a good story. Murder she programmed: Dynamic Plot Generating Engine for Murder Mystery Based Games. Automatically generate a story plot for film or paperback using key words of your choice. Select from a variety of styles and either publish them online or destroy them forever.Â Plot Generator. Inspiration for your next novel, film or short story.Â Plot Generator. Our aim is to inspire you to write your own stories, using common genres and themes. We'll help you set the scene then build characters, describe them, name them, and work out how they fit together in an interesting story. We draft a compelling blurb to get you started. Quotes About Plot Generator. "This is the best thing to exist ever." "So I used a generator to create a random story and it turned out to be hilarious."