Experimental AI in Games: Papers from the AIIDE Workshop
AAAI Technical Report WS-16-22

ClueGen: An Exploration of Procedural Storytelling in the Format of Murder Mystery Games

Andrew Stockdale
As834@kent.ac.uk

Abstract

In this paper I present ClueGen, a murder mystery game that generates its own narrative. I discuss the genre's suitability for combining gameplay and story using procedural generation techniques, before explaining the implementation and rationale behind ClueGen, utilizing the genre's common structure as an opportunity to define the scope of the generation requirements. In particular I detail a novel system for telegraphing deceptive characters in procedurally generated games with text-based dialogue, by using audible cues.

Introduction

Procedural generation as a content generation technique for computer games has been explored with growing enthusiasm in recent years. It is the practice of creating content not by hand, but algorithmically. The most obvious success stories of procedural generation involve level generation, and certain genres of game particularly benefit from this technique (such as platformers and sandbox games, e.g. Spelunky and Minecraft). There is a wealth of research related to these geographic generation techniques, ranging from labyrinths (Ashlock, et al, 2011) to entire landscapes and the placement of gameplay-specific objectives within them (Togelius et al, 2010). However, the impact of procedural generation techniques in more plot-driven genres is still relatively unexplored and thus attempts to create a game that can produce interesting and varied stories are rare by comparison.

Meanwhile in the field of computational creativity, recent years have seen a surge of interest in computational models for storytelling (Gervás, 2009). So far these attempts are mostly of academic interest as the stories produced are of a much lesser quality than stories written by contemporary authors. That being said, the stories are often coherent and reasonably interesting. When applied to the medium of computer games these story generation techniques could be invaluable in creating worlds that feel worthwhile exploring.

Background

Dwarf Fortress is an indie base-management game that generates new worlds each time it is played, complete with simulated historic events. These work very well at enriching the world so that it feels like more than simply geography, but they do not influence the gameplay at all. Procedurally generated levels have become such an effective mechanic because they require the player to engage with them. Therefore, it seems that the best way to approach procedural storytelling in computer games is to attempt to link the story with gameplay.

One notable game attempting to do this is Ultima Ratio Regum (Johnson 2016), which populates its worlds with an enormous amount of procedurally generated sociological and cultural details, that the player must become familiar with to be successful at the game. Symon (Fernández-Vara and Thomson, 2012) is another attempt to introduce puzzle-solving elements into a procedurally generated story, via a novel framework for adventure games. The Regen system (Kybartas and Verbrugge, 2013) presents an approach to evaluating and rewriting the structure of narrative-driven quests in games. These are very ambitious and versatile examples, so there may be value in exploring the use of similar techniques in a more restrained genre. ClueGen represents one foray into exploring such a genre, and the simplified techniques one may employ when working within these limitations.

The murder mystery genre is a good starting point to attempt this marriage between procedurally generated plot and gameplay. When looking at detective fiction, the genre's closest relative in literature, certain patterns emerge that make the overall structure easy to replicate in computer game form (one character has a reason to kill another, they do so, they hide the evidence, and an investigation occurs). The entire story is often set in one location, with a small cast of characters who all have varying opinions of each other. As well as this, the actual purpose of detective novels which is to intrigue the reader and get them to attempt to predict

the author's intent and determine the killer, means that there is an amount of 'gameplay' inherent in the plot, making this a perfect genre to attempt this marriage of gameplay and story.

One of the more common approaches to story generation is through creating characters that are permitted to act upon their own goals and motivations. Many narrative frameworks have used this model throughout the years (Meehan, 1977, Riedl and Young, 2010, Rank et al, 2012). This is a very promising model for the murder mystery game, as it will result in characters whose actions can be explained through their motivations (thus creating solvable crimes for the player to investigate).

Design and Implementation

Game Engine

To achieve the aims outlined previously, ClueGen was developed in Unity (unity3d.com) with C#. Unity is a cross-platform game engine, and using it meant that development time could be focused solely on the game logic rather than graphics and physics programming.

Game Concepts

Presentation

ClueGen is presented using 2d graphics drawn from an oblique perspective. The game is set entirely in a mansion with ten different rooms. The player can only see the room their avatar is currently in, encouraging exploration.

The player may converse with other characters via onscreen text by selecting dialogue options from a menu.

Items, Weapons and Containers

To give the NPCs (non-player characters) more methods to interact with the environment and thus create events for the player to discover, items were introduced. Containers such as cabinets, vases and chests are placed in rooms that fit their appearance, and are used to store items that currently aren't being held by characters. Items take the form of common household objects and can be picked up or stored by any character, including the player.

The scenario presented in the game assumes that the murderer does not arrive with their chosen weapon and so must find something suitable in the house. Many of the items strewn about the house are potentially lethal and are assigned to a damage type: sharp, blunt, poison, or gunshot.

All items in the game were deliberately placed during development, rather than procedurally. This is to ensure that their placement makes sense to the player and does not serve as a distraction, while also highlighting any unusual item placement as the result of an NPC's interaction. For example, the player should not be concerned to find several drawers full of knives in the kitchen, but should find a knife stashed in a vase in the washroom suspicious.

Plot Generation

The overall flow of the plot generation phase is as follows:

1. NPCs are instantiated and assigned a random name, gender, graphic, and starting position
2. Create families and randomly assign NPCs to them, changing their surnames as necessary.
3. Assign family members to spouse/child roles
4. Randomly select a motive (see *motives*)
5. Select a murderer and victim
6. Generate history according to the selected motive (see *histories*)
7. Manipulate the relationships of those involved in the murder
8. Spread knowledge of the history to other NPCs
9. Create and spread the pre-selected number of red herrings
10. Proceed to NPC Simulation phase

Relationships

Many of the NPCs' actions are governed by their relationships to the other NPCs. For example, one character may decide not to tell the player that they saw someone behaving suspiciously, if they are particularly close to that NPC. On the other hand, they may be more willing to put someone's name forward as a suspect if they strongly dislike them.

To store and access the various character relationships the game maintains an n*n matrix of integers ranging from -3 to 3, where n is the number of NPCs in the game. -3 represents a deep hatred of a character, whereas 3 represents love. At the start of the game, all the relationships that are vital to the plot (i.e. those concerning the murderer, the victim, and anyone else who has a motive to kill) are manipulated to create a coherent story, while all other relationships are assigned randomly. Random relationship values are weighted so that the extreme values on either end of the scale are less likely than neutral values. Alongside these relationships, NPC's also have a loyalty point which is set to an integer between 1 and 3. This is the minimum value relationship the NPC requires to have with another character before they will lie for them. The probabilities for this value are weighted so that 3 is the most common loyalty point, and 1 is the least common. This means that most NPCs will only lie for characters they love, but occasionally fiercely loyal NPCs will emerge who will lie to protect people they are only friends with.

Motives

There are four potential motives for the murder, these are revenge, lover revenge, jealous love, and inheritance.

- **Revenge** signifies a wrathful murder as payback for a past event.
- **Lover revenge** occurs when a character kills someone they yearn for who will not reciprocate.
- **Jealous love** is used when a character kills their rival in love.

- **Inheritance** can only occur when there is a family with 2 or more children, and signifies siblicide committed in order to earn more of the inheritance from other family members.

While at first the implementation of only four motives seems like it will result in a very limited set of stories, these motives are further expanded upon through the use of histories.

Histories

Histories are objects used to flesh out the overall theme set by the motive. These can be thought of as the defining event that set someone down the path to kill. Each instance of a history contains information about the NPCs involved and which of them were negatively affected by the event. For example a game with the jealous love motive might have a StoleLover history. If Anna had been in a relationship with Ben before he left her for Caroline, the History's fields would be filled accordingly:

NPC1	***NPC2***	***Lover***	***WhichNPCIs Victim***
Anna	Caroline	Ben	1

This example illustrates a History that could result in a game where Anna kills Caroline.

All NPCs keep a list of histories. This represents the collection of any histories they are aware of, either via rumour or involvement. The NPC refers to this list when questioned by the detective in order to form their testimony.

Rumours and Red Herrings

Additional challenge is introduced to the game by way of red herrings. In this case red herrings are histories related to the victim but unrelated to the murder. These are disseminated randomly among the NPCs and inserted into their history lists through rumour. As a result, when the player asks an NPC if they know anyone who may have a motive to kill, the NPC may suggest someone who is ultimately uninvolved.

Well-placed red herrings can quickly create intricate webs of character relationships that are enjoyable and challenging for the player to unravel.

NPC Simulation

During the NPC simulation phase each character goes about their business based on their motivations and pre-determined characteristics. It is here that the specifics of how the murder is committed are decided. Time moves forward in five minute increments, and each character may act once per increment.

Each character has the option to move to an adjacent room, take an item from the room they are in, drop an item from their inventory, or not act. The performed action is selected randomly, and the probability of each action being selected is determined for each individual NPC when they are first created. There is also a 0.05 chance that a given NPC will have the kleptomania trait, which results in a much higher rate of item pickups. These very simple variations in probabilities result in characters that feel like they have a distinct personalities or behave suspiciously, acting as a 'soft' red herring. If a character enters the same room as the body of the victim, they call for the detective. This initiates a one hour (game-time) countdown before the detective arrives and the investigation phase begins. This character waits with the body until the player's arrival, to serve as a reliable starting point for them to begin their investigation.

The NPC who is playing the role of the murderer follows different rules to the other NPCs as they have a specific aim they need to achieve. Initially they will seek out a suitable weapon to commit the murder with, moving from room to room and checking various containers. Once they have a weapon, they will search for their victim, and strike the moment they are alone with them.

Once they have killed their target the murderer then hides the weapon. To do this they simply place it in an available container in the room they are in, or if no containers are available they move to another room to try there in the next five-minute increment. NPCs can say that they saw another character pick up or drop an item, but not which item or which container it was placed in. This, combined with the possibility of other characters picking up or dropping items at will, means that someone witnessing the act of hiding the weapon is rarely enough for the player to solve the case.

Once the murderer has completed all their objectives, they act exactly as a normal NPC does, with the exception that they cannot 'discover' the body and call for the detective.

Investigation

Examining the Body

One of the first things a player should do to begin their investigation is find and examine the body. This provides an estimate for the victim's time of death, as well as which type of weapon was used.

NPC Introductions

To ensure the player is not completely overloaded with information at the outset of the game, the characters are introduced to them organically. The player gathers knowledge of their identities simply by initiating dialogue and asking who they are. The NPC will then reveal their name and any family members present.

NPC Testimony

One of the most crucial abilities the player has is the ability to interrogate NPCs about their whereabouts throughout the evening, events they witnessed, or anyone they potentially suspect.

All questions are answered using an instance of a Testimony. Each Testimony is a representation of an NPC's

personal account of a memory. NPCs can alter the details of a memory when they relay it to the player (making it untrue), or omit it altogether. These decisions are stored as booleans so they can be checked when the player accuses the NPC of foul play.

When asked about their whereabouts throughout the evening, the NPC first polls the timeline for a list of every event where they moved from one room of the mansion to another. They then iterate through the list, creating Testimonies for each event, omitting or altering information as they see fit.

The testimony is then translated to grammatically correct sentences and relayed to the player as dialogue between them and the NPC. This same process is used when an NPC is asked to describe everything they saw throughout the evening, with the difference that rather than polling the timeline for all events where the NPC switched rooms, they ask for a list of all events that the NPC witnessed.

A SuspectTestimony is a different type of testimony that is used when the player asks an NPC if there is anyone they believe might be responsible the murder. These are formed by the NPC looking through their list of histories they aware of, and seeing if there are any involving the murder victim. The NPC will randomly select from these relevant histories to determine which name they will put forward. One crucial nuance to this random selection of suspects is that an NPC will be more likely to put forward the name of someone they dislike as a suspect.

Lies and Omissions

The crux of the game's challenge is that when questioned by the player, NPCs have agency over what information they provide.

NPCs have two methods of deceiving the player: lies, and omissions. Lies are defined as testimonies that have had their details altered. Omissions are testimonies that are true, but that the NPC decided not to tell the player about.

The decision to lie about or omit an event is made at the moment an NPC is creating a testimony for it, while they are iterating through their memory, and depends on what kind of event they are describing, and who was involved. For example, if an NPC is describing an event involving someone they are loyal to, they will omit it if it revolves around them picking up or putting down an object. While these two actions might not appear inherently suspicious, they create many opportunities for characters to deceive, giving the player more mysteries to solve, and making the task of tracking down the murder weapon much harder.

NPCs may also opt to omit testimonies that would name a character they are loyal to as the one they most suspect of committing the crime. If this is the case they will simply provide the default response, that is that they don't particularly suspect anyone. This creates interesting drama where characters strongly suspect the actual murderer, and precisely know their motive, but will stay silent to protect them.

Accusations

The player may accuse an NPC of lying (i.e. altering the details of an event) after any testimony. If the NPC was lying, they will shamefully admit it and reveal the truth of the event. In order to accuse an NPC of omission, the player may select the "I think you're hiding something" dialogue option. This checks if the NPC has omitted any testimonies and if so randomly selects one of them for the NPC to reveal. Discovering omissions is normally a very dramatic moment during gameplay, as they will often reveal both crucial evidence about the murder, and where the loyalties of the character who hid evidence truly lie, throwing all of their previous testimony into question.

To ensure that simply accusing characters of lying after each testimony is not a valid tactic, they will become frustrated and impatient with the player after being falsely accused three times. After this, they will refuse to speak to them, effectively locking out a valuable source of information.

Stress

As the core difficulty of the game stems from the way facts are distorted or hidden from the player, it is important that they have some method of identifying exactly when this is happening.

An aim for this project was for the player to be able to arrive at correct answers not just from collecting evidence, but by also getting a feel for the characters and their motivations. This is an important aspect of detective fiction and it requires players to use their intuition as opposed to just logical deduction.

LA Noire (published by Rockstar Games, 2011) is a noir detective game that involves a similar combination of evidence analysis and character interaction. The player must question suspects and decide whether or not to accuse them of lying based on whether they sound and look believable. This was achieved by utilising a technology called MotionScan to record the facial movements of real actors, and then mapping those movements to 3d models in-game. However, this approach has several limitations. For one, it is hugely expensive to implement, with LA Noire being one of the most expensive games ever developed at an estimated cost of $50m[1]. More importantly, it is an approach completely incompatible with procedural generation. It is

[1] Will L.A. Noire change the game for actors? - BBC News. BBC News. 2016. Available at: http://www.bbc.co.uk/news/entertainment-arts-13507355. Accessed February 10, 2016

impossible to write a script and record a performance for every event that a procedurally generated plot could present.

The chosen solution to this issue is stress, which is quantified as a float from 0 to 1. A character's stress increases every time they tell any single lie the first time, every time they omit a single testimony the first time, when they are correctly accused of lying or omitting testimony, and when they commit a murder. Therefore stress can be used to make reasonable assumptions about how much any one character has to hide.

Initially the intention was to present stress to the player through dialogue by way of filler words and hesitation. Stressed characters' dialogue would have a chance to have words such as 'um' and 'uh', and ellipses dispersed throughout, to give the impression that they were uncomfortable with speaking to the player. This was effective in describing the character's emotional state, but wasn't very subtle. Once the character's dialogue is displayed onscreen, players may spend as much time as they'd like reading over it. This means they have time to analyse the frequency of these filler words to form an estimate of the character's stress, and thus use their deduction once again rather than intuition.

The final solution to this problem again involved how the dialogue is displayed, but incorporated audio to approximate the hesitation in a stressed character's voice. When an NPC is speaking to the player, their dialogue is displayed in text, with each letter appearing one after another. As each letter appears, it is accompanied by a tone, which represents the characters voice. The pitch of this tone is randomly assigned per character within a range specified by their gender (male characters will tend to have lower voices than females, although there is some overlap between the ranges). Letters are ordinarily revealed at a constant rate, with the exception of commas and full stops, which have a slight delay added to them to introduce a sense of realism and flow to the dialogue.

However, in between each word there is also a chance for a delay based on the NPC's stress value, to indicate them wavering or pausing. The probability of a delay being added is equal to the NPC's stress, so a character who is completely stressed at a value of 1.0 will stumble on every word.

These delays combined with the audio cues mentioned earlier result in a surprisingly effective approximation of speech. Characters who are not stressed will sound very composed and fluent, with natural sounding pauses between sentences and a consistent speaking pace. Characters who are under a lot of pressure on the other hand, will sound as if they are struggling to maintain their composure, with awkward stumbles and an overall uneven pattern to their speech. One of the main successes of this method is that once a character has finished speaking, the information about how they delivered their speech is gone. Players must therefore pay close attention to the character they are speaking to and the intricacies of their speech patterns to determine if they are lying, just as they would have to in real life.

Study

Methodology

At the end of development ClueGen was tested to determine its effectiveness at generating and presenting a story. The game was played by ten volunteers – all university students, studying a variety of courses - with varying experience with computer games, with the developer in the room guiding them through the experience. Open dialogue was established and participants were encouraged to provide a 'stream of consciousness' so that it would be easy to determine the immediate effects of their interactions within the game.

Results

Overall, response to the game was very positive. Participants were immediately engaged once presented with a mystery to solve and enjoyed exploring the mansion and meeting its inhabitants.

A particular highlight expressed by all participants was mapping out the family trees and thus making guided assumptions as to who each character would lie for. A notably absent feature discovered from hearing this is that there is no method to determine the relationships between characters who are not in a family, other than by which characters are willing to put them forward as their suspects. A relatively subtle way to include this feature would be to give characters access to a dictionary of phrases and words that convey their closeness to other characters, which they can use when mentioning them in their testimonies. The stress mechanic was very successful in subtly helping players determine when they were being lied to, and in keeping them engaged even after they had spoken to all the NPCs multiple times. Players were quick to identify that there were some differences in the way that certain characters were speaking and that it was probably linked to their testimonies. On repeated plays, most (7/10) players eventually discovered the true link between vocal stammering and stress.

One interesting finding was that over time testers became much more efficient with their investigations. On their first play, typically players would exhaust all available dialogue options with each NPC as they met them chronologically, before beginning to focus their investigation around key suspects. However, on repeated plays, dominant strategies emerged. The fastest tactic was usually to search for the body and identify the kind of weapon used first. From there the players would seek out all the known motives and histories in the game by asking each character who they

suspect, and then pay particularly close attention to the movements of anyone involved in those histories. While this tactic seems obvious given a knowledge of how the game selects the murderer and victim, it is important to note that the players don't have this knowledge and so will only discover over time through pattern recognition that the murderer cannot be someone that no other character suspects. It remains to be seen how the perceived difficulty changes after it has been played a great number of times, as the most tests any one participant took part in was five.

A recurring criticism of the game was that certain dialogue did not feel realistic, or did not have the necessary gravity to portray the drama of the situation. For example, when a character tells the detective that they suspect their spouse, the default dialogue for accusations is used. While it makes sense in terms of both the relationship system and in creating dramatic stories that it should be possible for spouses to dislike and suspect each other, characters need some way to acknowledge that the situation is unusual. A similar problem arose where the same type of history occurred multiple times in one game. For example, when the victim had fired multiple NPCs from their company. Although this coincidental repetition adds credibility to the idea that the victim was in a position of power, the impact was lessened due to the identical dialogue used to describe each individual occurrence. It seems much more written dialogue for use in specific situations is required to flesh out NPCs and make them more believable.

The motives and histories seen to be most compelling were those that involved more NPCs than just the murderer and victim, e.g. love triangles and family feuds. Purely subjectively speaking, the most interesting story was formed from a combination of the two. A plot was generated with two families (who will henceforth be referred to as family A and family B) who had a longstanding family feud. The victim was a son of family A, so naturally the rest of family A accused various members of family B. It eventually came to light that the victim was in love with a daughter from family B, but so was his brother, who killed him to try and keep the daughter to himself. The participant who played through this story was particularly impressed with the intensity of stories ClueGen was capable of generating. That being said, even very simple stories involving only the most basic histories were still considered intriguing during testing. The participants expressed that they felt this was due to the fact that they had to work to uncover all aspects of the story, and that the mere act of hiding information from them increased its value. This highlights a possible advantage in generating story content for games rather than the usual novella format commonly seen in computationally creative story generators. Games allow the developer to deliberately hide information from players with no guarantee that an unskilled player will ever uncover it, whereas in novels all information (with the exception of subtext) will eventually be revealed as long as the reader keeps reading. As discovered through testing, this added challenge piques the player's interest, adding intrigue and mystery to even the simplest plots.

A surprising finding from playtesting was that once players were impressed with the NPCs' abilities to lie, they were willing to attribute unusual behaviour to this deviousness. For instance, when one player realised that several innocent characters had unwittingly moved the murder weapon as far from the scene of the crime as possible, he assumed that they were conspiring with the murderer to hinder the investigation. As well as suggesting a potential feature for further development, this highlighted that if the AI's capabilities are demonstrated but not overtly explained, players are able to fill in the gaps and embellish on the plots presented them.

Conclusions

Overall the game functions well and creates interesting combinations of puzzles and stories, that are solvable when enough focus is devoted to the task. Further work is required to allow ClueGen to create characters that consistently feel believable, but the model of histories and motives that inform their actions works effectively and is easily expanded. While players' efficiency at determining the plot improved over repeated plays, no obvious patterns emerged and the stories still felt distinct, despite mainly being composed from a relatively small pool of motives and histories. The techniques applied here were chosen for the very specific task of generating murder mystery plots so the motive and red herring selection techniques are unlikely to be useful outside of these story types, but deception, character relationships and testimony are important facets of many genres, so the implementations of these could be potentially applied to future games that generate their own stories within said genres.

Acknowledgements

The research for this paper was carried out at the University of Kent. The author would like to thank Dr. Colin Johnson for his continued input and supervision throughout the development of ClueGen.

References

Ashlock, D., Lee, C., and McGuinness, C. 2011. Search-based procedural generation of maze-like levels. *IEEE Transactions on Computational Intelligence and AI in Games*, 3(3), 260-273.

Fernández-Vara C., and Thomson A. 2012. Procedural Generation of Narrative Puzzles in Adventure Games. In *Proceedings of the third workshop on Procedural Content Generation in Games*

Gervás P. 2009. Computational Approaches to Storytelling and Creativity. *AI Mag*. pp.49-62.

Johnson, M. 2016. Towards Qualitative Procedural Generation. Computation Creativity and Games Workshop. Forthcoming.

Kybartas, B., and Verbrugge, C. 2013. Integrating Formal Qualitative Analysis Techniques within a Procedural Narrative Generation System. In *Ninth Artificial Intelligence and Interactive Digital Entertainment Conference*.

Meehan, J. 1977. Tale-Spin, an Interactive Program that Writes Stories. In *Proceedings of the Fifth International Joint Conference on Artificial Intelligence.*

Pérez, y Pérez, R. 2015. A computer-based model for collaborative narrative generation. Cognitive Systems Research.

Rank S., Hoffmann S., Struck H. G., Spierling U., and Petta P. 2012.Creativity in Configuring Affective Agents for Interactive Storytelling. In *International conference on computational creativity* (pp. 165).

Riedl, M. O., and Young, R. M. 2010. Narrative Planning: Balancing Plot and Character. *Journal of Artificial Intelligence Research 39*.

Theune, M., Faas, S., Nijholt, A. and Heylen, D. 2002. *The virtual storyteller. SIGGROUP Bull.*

Togelius, J., Preuss, M., and Yannakakis, G. N. (2010). Towards multiobjective procedural map generation. In *Workshop on Procedural Content Generation in Games*, PC Games 2010.