An Unconventional Approach to Procedurally Generated Narratives

Prepared by:
Jean-Luc Burhop
Jared McArthur
Chris Bero

Introduction:

We started this project because as avid tabletop gamers we were interested in creating a programmatic framework in which we could easily generate large dynamic stories that wouldn't feel pre generated. This project was also a way for us to explore how fluid story generation works in real, human conversation,how artificial intelligence could be introduced to a computer system to imitate this generation, and a test of our current skill building a path finding system.

Overview:

Video games, in particular roleplaying game (RPGs), have always faced one terrible flaw when you compare them to their table top counterparts. Video games are traditionally limited to what has been programmed into the game. This is an issue in two ways: firstly it means that the writers need to explicitly spell out all the possible combinations of events to execute the story, and secondly it severely handicaps the aspect of replayability in such games. It was the goal of our project to research one potential method to resolve this problem. We quickly realized a procedurally generated narrative was the best possible chance for success. This solution was modeled after methods implemented by the popular tabletop RPG *Pathfinder* by Paizo Publishing.

Pathfinder consists of one Game Master and one or more additional players. The Game Master (GM) is in charge of generating a fluid narrative in which the players influence the state of the game through both studied decisions and probabilistic operation. When presented with a scenario, the players in the game interact with the GM by verbalizing their actions as the GM follows a set of rules to modify the state of the game. For example, a play may wish to physically attack a creature that is near; to do this they must tell the GM that they wish to attack, as well as roll a dice to see if the attack was successful. The element of probability through the rolling of a dice adds an artificial element of surprise to the game as each player's actions may not always go as intended. The main draw to most tabletop RPGs such as Pathfinder is the element of true intelligence being introduced into the progression of the players' story. Many GM's use pre generated stories called *Modules*. Modules are written in a way to allow the GM to inject true intelligence into a story to maintain the fluidity of a tabletop while also giving the players a clear goal. While most moderns games attempt to create a sense of freedom, this is heavily restricted by hard coded decisions implemented by developers that expose the game's underlying artificial intelligence.

The system we are proposing is largely based off the idea that most narratives are predictable and fit repeatable patterns. We can show this by comparing Star Wars and Shrek, movies that present very different environments and themes; one a space opera, the other a comedic fairy tale. When you analyze these movies deeply, a common underlying structure reveals itself, one of the hero's journey. The Hero's Journey is a narrative pattern identified by Joseph Campbell.[5] This pattern consists of twelve steps the each hero must undergo to reach the end of the journey. These events that the hero must undergo include and the order in which they typically happen are:

1. The Ordinary World
2. The Call to Adventure
3. Refusal of the Call
4. Meeting with the Mentor
7. Approach to the Innermost Cave
8. The Ordeal
9. Reward
10. The Road Back

5. Crossing the Threshold to the Special World
6. Tests, Allies and Enemies

11. The Resurrection
12. Return with the Elixir

The individual explanations of each step are not really important in the context of the paper. What is important is the framework for which stories can be created. In fact, the underlying structure is hidden with different types and characteristic of storytelling; Things like genre or writing style effect how to tell a story. To explain this we will look at the fourth step "Meeting the mentor." This is described by thewritersjourney.com as :

> "The hero comes across a seasoned traveler of the worlds who gives him or her training, equipment, or advice that will help on the journey. Or the hero reaches within to a source of courage and wisdom."[5]

In the popular movie Star Wars episode IV Luke Skywalker find his mentor in the form of Obi Wan, a wise old Jedi. Obi Wan teacher Luke of the power he has within him.[13] In the movie Shrek, the protagonist, Shrek find his mentor in Donkey, a loveable donkey who has seen much of the world. Donkey aids Shrek in his quest while simultaneously teaching Shrek how to have a friend.[12] Both Donkey and Obi Wan are perfect examples of a mentor for the hero of the story, however they could not be more different character wise. The underlying structure says that mentor aids the protagonist, it does not state the exact character type This change in how the mentors are presented gives two stories with the same structure a very different impression to the audience, illustrating that the skeleton of the narrative is much less important than they way the story is told. In our system we utilise this flexibility in storytelling. We do this by adding selecting a group of event that we want to have in our story(i.e. 12 steps of the hero's journey) and then building a story that correctly links the various steps together.

We connect our stories by treating the events has a block, similar to terrain in the game *Minecraft*. The modern sandbox action game *Minecraft* is a game that uses a very common technique for terrain generation in a 3-dimensional environment. In the game, there are very smooth transitions between each piece of terrain to give the illusion of true nature. While the visual implementation of our project is different, the idea is the same; we will leverage common procedural generation techniques to create a directed graph of interconnected nodes each requiring very specific entries and having outputs that affect the state of the program. Each node of the graph will represent an event in the player's story. The nodes will have strict requirements to be traversed to and will affect the global state of the story when entered. For example, there may be an adjacent node on the structured graph that has the potential to be traversed to. The story generation tool we will implement will take into consideration anything and everything the player has done up to this point (to be referred to as the *Story State*). By checking the Story State, the driving intelligence in the story creation software will choose the most appropriate node to traverse to on the directed graph. Then, based on a volatile list of attributes that the player has acquired throughout the story, appropriate dialog for traversing to the chosen node will be modified then displayed to the screen. This will give the story an illusion of linearity. Next the node will list the ways that it changes the story (i.e. "Your mentor dies!"). Essentially we are turning each node into a function that takes a set of history events '*H*' and returns a new set equal to the union of *H* and the story blocks outputs. This means that given a story we can represent each point in the story mathematically as a collection of past events. The nodes are essential a formal grammar or a set of words (i.e. finite strings of letters, symbols, or tokens.) In

Computer Science formal languages are used for the precise definition of data formats and the syntax of programming languages. [3][6]

It would be really easy to just build a collections of blocks and then build a random story, but this will not necessarily be an engaging narrative. Using Legos as an example, if we were to say only blue legos can connect to red legos, then start building things according to the rules we could possible get a good looking structure but more often than not we will get some random tower of blocks. This is because color is not the only attribute to a lego, they also have dimensions(2x2,2x4, etc...) If we wanted to make sure we build something good every time, we can add another set of rules, or better yet blueprints. So let's say we wanted to build a tree every time, we would know simple things like go from a 2x2 lego to a 1x1, because then the trunk of your tree is getting thinner, or after some amount of trunk legos start looking for a bigger lego piece that will allow us to start making branches, while still fitting the color combinations. This is the same idea behind the blocks. Our proof of concept attempts to build a Hero's Journey. As mentioned above, we know certain events need to happen need in the journey, so we add those blocks to the story first, and then find all the possible paths to use the available story blocks. We find the paths using the formal grammar rules decided with each block. For example, we want to build a story that has an event B, C, E. if the Story State looked like:

Current State = "(A)(B)(C)"

Two Possible Acceptable Blocks:

1. [(C) -> (D)]
2. [(B) -> (E)]

Both of these possible are acceptable because the current state contains the required event types

So at this point the Story system could make multiple stories. First it could end the story after using block 2, or it could use block one, check for new possible blocks, and keep building from there, eventually adding the E it needs to complete. It is important to keep in mind, that the best story is not always the one with the shortest path. The story system will build several different versions of the story and then using a fitness function, detrimen which is the best story to use. It is also useful to generate multiple story lines because then the Story system could present the user with a choice of path, while knowing that whichever choice they choose, the story has an end. This is a very similar to a problem that a maze generator needs to solve. If the maze is procedurally generated you need to be sure that the maze has an exit, otherwise it is unsolvable. Building a lot of possible stories does have several potential problems. If the software builds all possible story orders then this quickly becomes a task that is almost not feasibly computable. The depth of the story could be limited with a hard cap or a function could be put in place that penalizes the Story system for not "getting to the point fast enough." There are a lot of possible techniques that could be use to limit the computational explosion, but but some methods would work better than others for different types of stories. For example, the Story system is not only applicable to a hero's journey, it could be catered towards types of users. Maybe the user currently playing like a lot of side quests which means the catered Story system would have a lot of loosely related events. A user who wants to get write to the point would have a story that builds without doing much branching. The story system as a whole needs to answer a lot of smaller questions about the types of stories to be built in order to answer questions like "How do we limit the computational requirement growth?" or "What is the best path?".

Once we find the path that the story will take, we want to add some fluid dialog to the output to generate a dynamic story. For example if the story path has taken a player through a node that adds a modifier to increase the 'fame' of the player, there may be dialog options within other nodes scattered throughout the graph that mention this. The modifiers added to a collection of information about the player's state can even allow or deny them entry into other nodes. The dynamically dialog is very important to a fluid story because this can create the illusion that the world the player is in is 'real' and not some linear pre generated storyline. Part of the reason a tabletop story is believable is due to its ability to adapt its dialog based on player's past actions. By implementing this feature into our system we blur the line between pre generated narrative and true random narrative, hopefully creating something comparable to real human interaction.

Problems Faced:

The main issue we faced was ensuring everybody had the same understanding of the project. We later realized that the issues stemmed from a flexibility in the software that allowed a lot of interpretations. The software is inherently broad as to mimic the flexibility in modern tabletop RPGs. This problem is what caused us to realize how much the implementation depends on the type of story being built. This issue also caused us to abstract the problem as far as we could in the code. These issues caused us to engineer a framework that can be used by someone to create these stories; we made the tools for more creative people than us to build dynamic narratives.

Secondly, we ran into issues developing the final GUI. The GUI required too much time to implement as originally imagined. After meeting with Dr. Rochowiak, it was decided to focus more on the framework, then the GUI for the application. However, we are still able to show the general concepts.

Future Works:

Possible works include research into limiting the growth of the possible stories. This research will need to include a recommendation for when to use each limiter. This would require a profile of many more story types. This is a job for someone with a greater combinatorics background then the one our team currently possess.

I literary analysis of other story profiles including possible methods of abstraction. The Hero's Journey was easy because it already was in a 12 step format. The future works would allow us to make more complex underlying structures. These story profiles would also consist of ways to make more bread stories versus more in depth systems.

Lastly, it would be possible to build a neural net to analysis the text in books to also map out the stories framework. However, that requires something on the level of IBM Watson.

Conclusion:

The system we built has a lot of possible paths to move to move forward. As discussed in the overview, the story system proposed is heavily dependent on the domain it was built for. The general mechanics of building a story are pretty universal, like the block classification or the formal language. Whether is was built to tell wide expansive stories, or linear stories heavily change the exact implementation. This is why we only built the Hero's Journey creator. People who use the framework will need to cater the exact specifications to their needs. Just like the modules in Pathfinder, it is not our goal to replace all of the creativity, we just aim to lighted the creative load.

Bibliography:

1. *A, G. Toward Supporting Stories with Procedurally Generated Game Worlds (n.d.): n. pag. Georgia Institute of Technology. Web. 26 Mar. 2016.*

2. *Bourg, David M., and Glenn Seemann. AI for Game Developers. Sebastopol CA: O'Reilly, 2004. Print.*

3. *"Formal Grammar." Index of /apache2-default. N.p., n.d. Web. 19 Apr. 2016.*

4. *"Generating a Directed Acyclic Graph from Predefined Elements with Connection Requirements." Algorithm. N.p., n.d. Web. 26 Mar. 2016.*

5. *"INTRODUCTION." Hero's Journey. N.p., n.d. Web. 27 Mar. 2016.*

6. *"L-system." Wikipedia. Wikimedia Foundation, n.d. Web. 26 Mar. 2016.*

7. *Laitch123. "Infinitale - A Method for Procedural Randomly Generated Narratives in Video Games." YouTube. YouTube, 15 July 2013. Web. 16 Apr. 2016.*

8. *Lynch, Sarah. "The Point - Tabletop vs. Gaming." GameSpot. CBS Interactive Inc., 6 Nov. 2013. Web. 18 Apr. 2016. <http://www.gamespot.com/videos/the-point-tabletop-vs-gaming/2300-6415949/>.*

9. *Mark, Dave. Behavioral Mathematics for Game AI. Boston, MA: Charles River Media, Course Technology, Cengage Learning, 2009. Print.*

10. *Or How To Achieve Graph Based Nirvana In Java. JUNG 2.0 Tutorial (n.d.): n. pag. Http://www.grotto-networking.com/JUNG/JUNG2-Tutorial.pdf. Web. 26 Mar. 2016.*
11. *Shiffman. "8.5 L-Systems (The Nature of Code)." YouTube. YouTube, 11 Aug. 2015. Web. 16 Apr. 2016.*

12. *Adamson, Andrew, Vicky Jenson, Aron Warner, John H. Williams, Jeffrey Katzenberg, Ted Elliott, Terry Rossio, Joe Stillman, Roger S. H. Schulman, Mike Myers, Eddie Murphy, John Lithgow, Cameron Diaz, Vincent Cassel, and William Steig. Shrek. Glendale, CA: DreamWorks Animation, 2006*

13. *Star Wars: Episode IV - A New Hope. Screenplay & Dir. George Lucas. Lucasfilm & Twentieth Century Fox, 1977. Film.*