Exploring Dynamic Interactive Narrative

Jason Merrin

Advisor: Norman Badler  
University of Pennsylvania

# Abstract

*In the game industry and in electronic entertainment at large, a question is being posed as to how best to deal with interactive storytelling. In the past, interactive stories have been built in a tree structure, with alternate stories written that branch wherever the player gets to make a choice. Now, however, new attention is being paid to true dynamic storytelling, and the possibility of constructing a great number of quality stories through artificial intelligence and large scale design. My intention is to explore the current field of interactive storytelling, and attempt to devise and implement a system in which the story is dynamically generated around the input of the player.*

*Project Blog: http://dynamicstorytelling.blogspot.com*

# INTRODUCTION

It is a serious priority in the entertainment industry right now to create a form of interactive storytelling so that the audience might somehow gain control over the narrative. Unfortunately, the current “branching tree” system utilized by most interactive stories – from video games to choose your own adventure books – are prohibitively costly and time consuming. Any decision which branches the story requires a doubled amount of work from that point forward. A choice made early on in a story means that the rest of the story is written twice – once for each possible decision. In a video game, it means the rest of the game is made twice - in essence doubling the budget of the game to create content most gamers will never experience. Still, the industry is obsessed with adding interactivity to their stories, so a new model apart from the tree structure is needed.

In theory, a dynamically building story would mean that for less cost and less effort, audiences would get a story with much more variation than they could ever hope for from a branching tree system. Stories would redefine themselves based on the user to give the optimized story experience. Furthermore, with added random elements and subtle game interactions, the audience could have a completely different experience with each play through. The replayability of games would skyrocket. In theory, a future application could also be dynamically changing movies that are different every time you watch.

My approach would be to compartmentalize elements of story and program for interaction with other elements. Much like different weapons in games must be programmed to interact with other objects in specific ways, I believe certain characters, objects, and locations should be able to interact with others in such a way that builds the story in real time. Non player characters could have their own histories and emotions, and should exist outside of their interactions with only the player. In this way, the player can have a butterfly effect on the story – things that are said or done to characters cause them to react and interact with other non-player characters in a way that creates conflict and therefore builds a story. An example of a potential scenario – the player asks a character to go and get something for them, but when they do they bump into another character, one who deeply hates them. They get into a fight, and if there is a weapon nearby one might kill the other, creating the central conflict for the story. If the player removed the weapon from the room, then the fight would only be verbal, and might somehow become a catalyst for the central conflict of the story. I believe it is essential that there only be one major driving event for the story, and perhaps several subplots. At a certain point, an artificial intelligence crafting a story must stop looking for conflict and start looking for resolution.

This project makes the following contributions:

• A compilation of research done into dynamic interactive storytelling to this point

• A proposal for a new system of dynamic storytelling.

• A proof of concept basic implementation of said system.

## Design Goals

The target audience for my project is game developers, and those interested in electronic interactive entertainment in general. I hope to design a method of electronic interactivity that has the potential to make fully interactive story experiences a viable option for developers.

## Projects Proposed Features and Functionality

In my design project, I hope to implement:

* A basic A.I. character system, in which characters “behave” and so affect the story apart from the player
* A storytelling A.I. that looks for a central conflict and a resolution, and stops after they have been achieved.
* A basic “game” that showcases the way in which even the subtlest of actions can radically change not just the outcome but every component of the story.

# RELATED WORK

*Adaptive Storytelling and Story Repair in a Dynamic Environment*. Richard Paul, Darryl Charles, Michael McNeill, David McSherry (UK). – An paper detailing the shortcomings of linear stories as they are used in games today, including the thousands of scripted storyline options one might find in an MMORPG. The paper suggests some radical alternatives to linear storytelling that it believes might work, including some ideas I plan to build off of, but largely avoids specifics of a new structure. Most importantly, the paper devotes a lot of time to highlighting the biggest potential pitfall to adaptive storytelling – breaks in the continuity of a shifting story – and how to avoid it. [PCMM11b]

*Game A.I. as Storytelling*. Mark Riedl, Vadim Bulitko, David Thue. – A basic but still helpful presentation on how to use A.I. as architecture for storytelling. The system used here is more simplistic than the one I hope to implement, in that it still has a baseline “ideal story” which the player is able to take control of and change (rather than a largely randomized and generated from scratch story that might surprise even the game creator). Even so, there are certain elements of my planned game management A.I. that I could build off of theirs, such as how to have the A.I. keep track of what story changes make sense next (though I’d need to devise my own way of coming up with how to handle major story beats). [RBT11]

*“Behind the Façade”Guide*. Procedural Arts. – A behind the scenes “how it was done” guide to the popular interactive story Façade, which is known for the depth and subtlety the user interaction has on the game’s story. Useful, because I am still uncertain as to the ways in which I might be able to integrate user interaction.

*MIST: An Interactive Storytelling System with Variable Character Behavior*. Richard Paul, Darryl Charles, Michael McNeill and David McSherry. – A paper on “stable but dynamic environments.” Perhaps the most useful to my research, this paper advocates an unscripted experience shaped not only by user interaction but by autonomous NPCs. A framework architecture for the world is provided, along with a partial implementation and results. My plan is to study their architecture as a jumping off point for my own, trying to push past just dynamically generating a story to a place where the game is aware if that story is any good, and is able to adjust its direction if not. [PCMM11]

*Supporting Rereadability Through Narrative Play.*  
Alex Mitchell, Kevin McGee (SG). – An exploration into what makes a story interesting to go through again, and the differences between replaying and rereading. One interesting element is that with the game implementation at the heart of this paper, the authors found that people replayed not to find better narrative closure but to “do better than last time.” I actually plan on using this paper as a guide for what to avoid… My goal is not to construct a game where one replays to get a new story, and not to do better than before. I think a good way to handle this is to take out many “losing” scenarios for the player, in which the player is more of a catalyst for what happens to NPC characters in the game.

*Why Paris Needs Hector and Lancelot Needs Mordred: Using Traditional Narrative Roles and Functions for Dramatic Compression in Interactive Narrative.*Janet H. Murray (US). – A more specific exploration into abstracting character types and immediately recognizable story threads in such a way they can be parameterized and controlled. More useful to the actual building of my story world and character/plot thread interactions than the MIST paper, which focuses more on A.I. and architecture than on how to make the story itself work in a meaningful way. [Mur11]

*Back-Leading through Character Status in Interactive Storytelling.* Jichen Zhu, Kenneth Ingraham, J. Michael Moshell (US). – A fascinating paper on dynamic, real time storybuilding that equates the “best of” scenario for interactive storytelling with improvisational theater. At the end of the paper, a technique is presented for handling status shifts to the story, and while it isn’t the most robust method presented in one of my resources, it comes from a radically different and intriguing angle. [ZIM11]

*Hooked! – Evaluating Engagement as Continuation Desire in Interactive Narratives.* Henrik Schoenau Fog (DK). – A really interesting paper that would honestly only factor into my own implementation if I finished the demonstration component significantly ahead of schedule. This paper proposes ways to measure user engagement with the story during the actual progression, giving the program the ability to shift gears and change the story if it seems like the user is losing interest. While this could obviously have huge benefits to a dynamically generated story system such as my own, it also not intrinsic to the system I am hoping to implement. [Sch11]

# PROJECT PROPOSAL

Once more, the intended goal of this project is to discover an alternative means to the tree method of interactive narrative, and to implement a basic system capable of dynamically generating original and decently well-crafted stories. The way I hope to approach this is by crafting multiple A.I.s (particularly characters) capable of interacting with one another as well as with the audience controlled character/elements, and having these elements governed by an A.I. that keeps track of potential story conflicts, subplots, and resolutions.

## Anticipated Approach

I plan on creating A.I.s that are built to interact with each other in such a way that the “audience” can follow the interactions. Each character A.I. will be given a set of emotions that will cause them to act towards other characters in a certain way, which will affect the emotions of the characters involved in the interaction.

One element of my intended approach is to give characters emotional ranges that affect their interactions with other characters, items, and locations. Ideally, I will start with just the emotions of “love” and “anger” – the two most common instigators of conflict in narrative storytelling. Certain character behaviors will be governed by their emotions or could trigger heightened emotions in other characters based on prewritten backstories. A character in a heightened emotional range might do something extreme, which would serve as an obvious inciting incident for our story. I can also then “jitter” the starting emotional range, adding an element of randomness to which character might trigger the story, how, and why.

The governing A.I. will be potentially the most challenging element to implement, as it must detect “story beats,” like inciting incidents, subplots, and resolutions, and jettison any developments that derail the flow of the story before the user is made aware of them.

## Target Platforms

The A.I.s will be written in C# so that they can be used in a Unity based game demo.

## Evaluation Criteria

If my project is successful, playing the game should give the user a feel that they are impacting the narrative in some significant way. Each play through the game should yield different results, and the story elements of the game should vary enough that even after several playthroughs they still feel fresh and unpredictable. Comparing this implementation to already existing work, I hope my system will provide more variation with fewer assets. Which is to say, I hope my system demonstrates how wildly the story can vary even with only a few very limited characters, items, and locations.

# RESEARCH TIMELINE

**Project Milestone Report (Alpha Version)**

* Completed all background reading
* Constructed an in depth model for a hypothetical narrative A.I. capable of discriminating and guiding a dynamically generated story.
* Proposed software framework is functioning with simple base case

**Project Final Deliverables**

* Basic game with greatly variable story created in real time using narrative A.I.
* Demonstrational video
* Documentation

**Project Future Tasks**

* Build out the storytelling A.I. into a full storytelling engine that supports stories in all forms and genres, that can have story conditions set by users (instead of hard coded in for demonstrational purposes).
* Expand out A.I. of characters. In particular flesh out character emotions and emotional responses.
* Make locations more substantial, so that setting can affect genre and character moods.

# Method

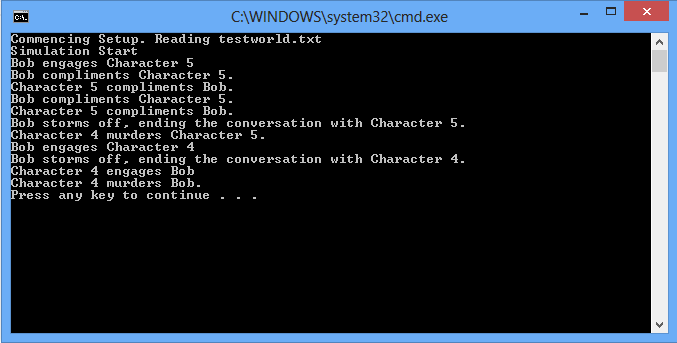
The approach I took is built around this core idea: story is plot, plot is driven by action, action is driven by motive, motive is driven by emotion. As a result, the system I decided to implement is built around action and emotion. In screenwriting, a story is defined by three plot points – the inciting incident, plot point one, and plot point 2 – each defined by a major action that defines the story going forward. I decided that I would use this model as my jumping off point, creating a story that cycled through three “plot points” and then ended.

**Figure 2:** *Story Graph*

I set about trying to implement this concept in text based form. First I created a series of characters, defined by name, gender, and a set of emotions that character held. Then I made it so that on each timestep of the program, characters would search for other unoccupied characters and begin interactions with them, performing an action based on their emotional state that affects their own emotional state, their partner’s emotional state, and often times more (“murder” removes a character from the game, “walk away” ends the interaction, etc.).

**Figure 3:** *Interaction Model*

For my earliest implementation, I simply defined my story by timeframe. After 10 interaction cycles, the game would end. This gave me a pretty decent simulation system, but didn’t highlight the story the way I wanted. In an effort to simulate the plot point model, I decided to go through and assign different actions “Action Importance Values” (AIVs) on a scale of 0-5 (5 being most important). For examples, one scale might have chat = 0, insult = 1, shout = 2, fight = 3, murder = 5. Different actions would be deemed important to the story by a “storyteller” of sorts, and then the program would keep going until three important actions had been reached (inciting incident, plot point 1, plot point 2). I found I could make the text story better (more interesting, at least), by suppressing the text output of actions below a certain AIV. For best results, I assigned one of the characters (usually at random) to be the “main character”, and kept track of who they interacted with and what degree that character was (if character A is the main character and A talks to B and B talks to C, A is degree 0, B is degree 1, C is degree 2. If C then talks to A, C is also degree 1). I included all text output for A, and then text output for all characters of degree over 5 if for the action on that turn (AIV-degree > 0). I only ended the “story” if the main character had been involved in 3 important actions. This method was actually able to generate some pretty decent stories with some reliability (like the story about the man who walked around looking for someone to talk to, was rejected then insulted by everyone, and murdered everyone, finally wandering around the world he’d emptied still looking for human contact).



**Figure 4:** *Early Text Based Generated Story*

At this point I received a request from my senior design advisors to make the simulation extendable. What I was building was a simulator with all of the actions and emotions hard coded in, and what they wanted was for me to build an engine of sorts – one that proved that it could perpetuate a story regardless of which actions and emotions were chosen. Making it so that actions and emotions could be added in separately in .txt form would not only make it easier to build and run actions and emotions for testing purposes, but would mean that just about anyone could come in and design a scene file as well as different actions that would radically change the nature of the simulation. The format I ended up going with is a config file that is read by the code at the very beginning. An example is included here:

emotions

5

love

hate

boredom

sleepy

happy

actions

7

converse.txt

stormoff.txt

insult.txt

sleepwith.txtflirt.txt

compliment.txt

wanderoff.txt

characters

6

Charlize

f

Maggie

f

Charles

m

random

end

The above txt file defines 5 emotions the characters can have and names them, then defines 7 actions and 7 txt files (one for each action) where the actions themselves are defined. Finally 6 characters are defined, with the first three specified with name and gender and the rest generated randomly by the program. An example of an action txt file is as follows:

Fight

4

fights with

being\_cocky

1

hate

50

.3

.3

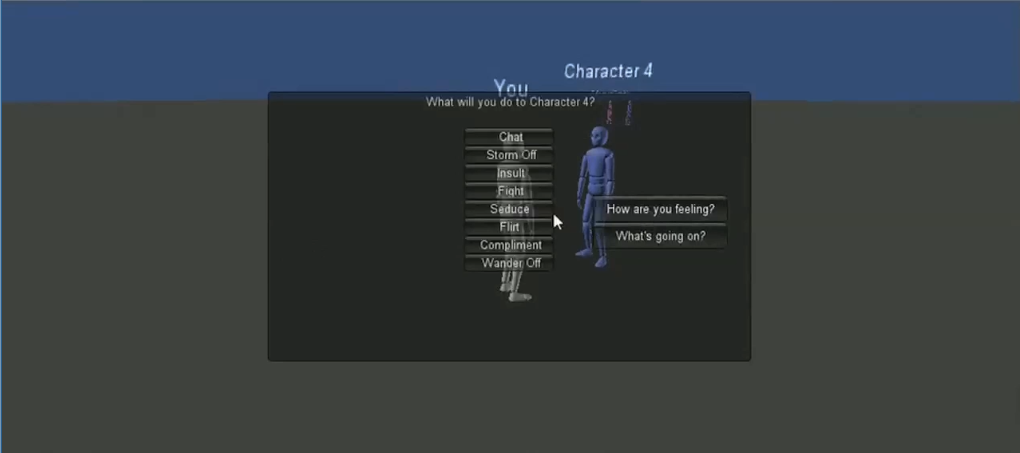
partner

end

The action file defines the name of the action, the AIV, the string of text used for the output, the name of the animation to play (placeholder in the text based version), the emotions and how they should affect the character, and finally a string of lines like “partner” (if the interaction requires a partner) or “kill” (if the interaction kills the interaction partner). For how emotions affect the scene, we have the name of the emotion (which should match an emotion found in the config file), followed by an ideal value. Character emotions are represented as values between 1 and 100, and a character’s decision is based on which action’s ideal circumstances are closest met by its emotions. For the example above, we care only that our character has a level of “hate” close to 50. For another action, we might want the character to have “hate” of 50 *and* “love” of 10. At each time step the character cycles through potential actions, picks the best fit (with very slightly jittered emotions to avoid complete predictability) and performs that action. The two “.3” values in the action file above represent how the action affects the emotional state of itself and then the emotional state of its partner for that distinguishing emotion. In this case, hate rises in both interaction partners by 30%. In other cases, it might get reduced. I chose to do emotion fluctuations by percentage to mirror heightened emotional states. Someone calm (low emotional value) is unlikely to get very suddenly very angry. Someone already fairly angry is in a good position to get much angrier or to calm down much more. So the amount the action affects the emotions of the characters scales with how emotional the characters are.

Once I had .txt based config files and action files setup, it was time to make a graphics version of my code. It seemed clear at this point that to really prove that this methodology could work for video games, I would need to show it intuitively – with actual characters walking around in 3D space.

A lot of time at this point in the project was spent learning how to use Unity. Eventually, though, I was able to create a 3D version of the text based code. There were a few alterations that had to be made, to be sure. For one thing, characters now walked around in an actual 2D plane, looking for another character within a certain radius of themselves that was not yet in an interaction. Now animations actually played, so that a character performing an action had a certain gesture linked to it. The output text of actions was still used, but now the most recent action performed by a character floated above his or her head at all times. The biggest question to deal with now was how to tell the story. I could no longer pick and choose what lines of action I wanted shown, since all actions were taking place in real time on screen. An early thought was to have the camera cut to whatever was happening on screen at a given time with the highest “AIV”, but since that could change second to second it didn’t seem like a very rewarding experience for someone watching. Finally I realized that this question was actually moot. I was designing a system for video games, in which the player has control over the game. By adding the character that the player controls and letting them interact with the world, I wouldn’t need to address what the player gets to see – he or she would decide that for themselves. I could still use the system for suppressing “boring” or unimportant information in the personal histories of characters. I had the characters remember the last 10 “weighted” actions – each time a new action is performed, it is added to the front of the list at position zero. If the list is full, the action with the highest value of (Position in List – AIV) gets removed, ensuring that when you ask the character what they’ve been doing they tell you their most important actions as a function of how recently the action happened and its own inherent importance. I also made it so that you could see character emotions if you asked for them, and could choose to ignore or interact with any character your avatar got close to. When interacting, the player has a choice to implement any actions declared in the config file. Other characters continue walking around and interacting while the player is in a conversation (even while they’re making their decision). Non player characters can get bored or angry and decide to end the interaction with the player character.



**Figure 5:** *Final Stage Unity Simulation*

This left me with a pretty good playable simulation, but it still felt like a simulation and not a game. Perhaps more importantly, it didn’t yet feel like a story. I’d taken out the “three actions and then it ends” rule, because once I’d made it playable the player could just choose three important actions in a row and end the game immediately with no real story told. I quickly realized that the action/emotion simulation might be a good engine for driving games with dynamic narratives, but any real game story will need to have a set start condition and end condition. This is no different from games like Minesweeper that I was trying to emulate in terms of “different every time you play” – you always start at the same place, and you always win or lose one of two ways. I decided to make a murder mystery game as a proof of concept where the start state (inciting incident) is defined as “there is a murder” and the win/lose state is “you accuse someone of murder and are right/wrong”. By specifying what condition defines the beginning and end, but leaving context open ended, I figured I could pretty simply turn the Unity simulation into a game with constantly changing story. I picked a murder mystery because the game is plot based, and to figure out who dunnit one must first figure out who had motive – motive that would be different every single time you play.

# RESULTS

Success! Relative to writing the engine, creating a game out of it was easy. It only took a few hours to define the start and end conditions and draw up a few extra game screens. The majority of the game was fuelled by the emotions, characters, and actions I defined in the text files. Writing those up took less than half an hour, at which point I suddenly had a playable game that was a surprising amount of fun and took roughly 20 minutes to play through each time. And, as was desired, playthrough was different each time. While the basic type of game remained – someone kills someone else, you find the killer – all the details changed so that I, as the game’s creator/writer, had no idea each time what the story would be. Even so, the story made sense each time. I’d get elaborate stories such as two people would fall in love, sleep together, drift apart, find each other again, get in a fight, and then one would murder the other and try to lie about it to me, the detective. That’s not a story I wrote – that’s just an example of a generated story. While the stories, graphics, and gameplay are all pretty simplistic, there’s no doubt that the system works. I created a game that has a new story each time in less than a day using my engine. The possible applications for a larger game are staggering.

# CONCLUSIONS and FUTURE WORK

Games don’t need story trees to generate different story paths. Your story is defined by characters, who are defined by their actions and emotions. If you specify the actions and emotions those characters get to choose from (along with the setting, a start condition, and a win/lose state) and set them loose on a virtual world, you can create rich and infinitely variable stories to play through that will surprise even you. It’s a field of video game narrative that hasn’t been nearly explored enough, and which I feel I’ve proven holds incredible possibilities.

My next steps would be twofold. The first would be to select a new game idea/genre, with a new start state, win state, and gameplay mechanics that builds off of the same storytelling engine in a new way. I want to prove that the engine can work for any type of game that involves narrative, and not just this one. My next step after that would be to take the game I like better (for now, we’ll say the murder mystery) and expand it into a richer, deeper, more complex and graphically pleasing experience that would not only “prove” the idea on a basic level, but would serve to impress (and hopefully convince) game companies that this is a viable, cheap, and exciting future for storytelling in video games.

# References

|  |  |
| --- | --- |
| [PCMM11] | Paul R, Charles D, McNeill M, McSherry M. MIST: An Interactive Storytelling System with Variable Character Behavior. *Lecture Notes in Computer Science, 2011, Volume 7069/2011,* 219-230. |
|  |  |
| [PCMM11b] | Paul R, Charles D, McNeill M, McSherry M. MIST: Adaptive Storytelling and Story Repair in a Dynamic Environment. *Lecture Notes in Computer Science, 2011, Volume 7069/2011,* 128-139. |
| [RBT11] | Riedl M, Bulitko V, Thue D.: Game A.I. as Storytelling. *Artificial Intelligence for Computer Games 2011,* 125-150. |
| [ZIM11] | Zhu J., Ingraham K., Moshell J.M.: Back-Leading through Character Status in Interactive Storytelling. *ICIDS'11 Proceedings of the 4th international conference on Interactive Digital Storytelling,* 2011, 31-36. |
| [Sch11] | Schoenau Fog, H. (DK): Hooked! – Evaluating Engagement as Continuation Desire in Interactive Narratives. *Lecture Notes in Computer Science, 2011, Volume 7069/2011*, 219-230. |
| [Mur11] | Murray J. H. (US). Why Paris Needs Hector and Lancelot Needs Mordred: Using Traditional Narrative Roles and Functions for Dramatic Compression in Interactive Narrative. *Lecture Notes in Computer Science, 2011, Volume 7069/2011,* 13-24. |

|  |
| --- |
|  |
| |  |  |  |  |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | | **#** | **Task Name** | **Oct** | **Nov** | **Dec** | **Jan** | **Feb** | **Mar** | **Apr** | **May** | | 1 | Initial Research |  |  |  |  |  |  |  |  | | 2 | Formal system outline |  |  |  |  |  |  |  |  | | 3 | Basic sytem framework implementation |  |  |  |  |  |  |  |  | | 4 | Project milestone review |  |  |  |  |  |  |  |  | | 5 | Meetings |  |  |  |  |  |  |  |  | | 6 | Basic game implementation/demo preparation |  |  |  |  |  |  |  |  | | 7 | Final review preparation |  |  |  |  |  |  |  |  | |

**Figure 1:** *Intended Schedule*