Bringing Interactive Storytelling to Industry: Designing a Reactive Narrative Encounter System

Daniel Kline

Crystal Dynamics
1300 Seaport Blvd Suite 100
Redwood City, CA 94063
dankline@sbcglobal.net

The Author

Daniel Kline is a Lead Game Engineer at Crystal Dynamics. He's been an AI game programmer and designer since 2001, when he received a BA from Vassar College's Mathematics program. In that time, Dan has worked with companies across the games industry, including Activision, Blizzard North, LucasArts, and Midway. His work spans 9 titles, including Star Wars: Force Unleashed, Diablo 3, and Call of Duty: Finest Hour. Between his design and management roles, he's specialized in AI characters, AI-design collaboration, and drama management.

Introduction

The commercial games industry is struggling to find a form of narrative for single-player story-based games that takes advantage of the medium. Current methods are limited to structured narrative or simulation, both of which have serious game design constraints. Drama Management represents a potential new model. This talk will cover the author's research into bringing interactive storytelling to single-player story games. Focusing on the design and production needs of current titles, different groups of pen-and-paper RPG Game Masters were studied, and their interactive narrative approach was algorithmically replicated for video game storytelling on a per-encounter level. To verify these results, an "Encounter Manager" pen-and-paper algorithm was tested in place of a human Game Master, achieving similar storytelling results and exposing new conclusions. Implementing these techniques in several video game projects has shown diverse payoffs, not just in narrative, replayability, and pacing but also in risk-reduction, art production, open world and linear level design, and new game genres. However, the uncertainty and risk of a new model of game production combined with the lack of a driving need in commercial games has so far proven a major hurdle for this approach. Looking forward, similarities with other procedural narrative efforts are explored, and future steps are proposed.

Current Storytelling Methods

Video games have relied on two proven storytelling methods, structured narrative and simulation. Structured narratives, frequently called branching, rely on linear pre-authored scripts, sometimes broken up with limited player choice. These static tales can achieve strong drama and characterization, as exemplified in games like *Mass Effect* or *Metal Gear Solid IV*. However, branching sidelines the player's input into the narrative, which can only be overcome through large amounts of expensive content.

Simulation games take the reverse approach. They focus on simulating a world, and give the player maximal space to find their own stories (Wright 2007). Although they promote player creativity, simulation games like *Civilization 4* and *Mercenaries 2* lack the narrative characterization, thematic exploration, and reflection seen in branching games.

Neither of these approaches can effectively combine player interactivity and story. Open world games in particular need a narrative that can adapt to a wider variety of player actions in many locations. A simple example of this appeared as far back as eight years ago with *Grand Theft Auto III,* which spawned different gangs with different attitudes towards the player depending on the player's location and path through the story. Broadly put, there is a growing video game design need for both: to involve players more in stories alongside deeper and more meaningful narratives, all without requiring overwhelming amounts of content.

Games have long tried creative solutions to the problem. *Grand Theft Auto IV, Radical Dreamers,* and *Fallout 3* all employ such tremendous amounts of branching content that they can respond to a variety of player goals using world exploration. Alternatively, games like *Spore* or *Nethack* place predefined narrative "chunks" of content in arbitrary sequences to create a unique story. But all of these techniques are still fundamentally static and impersonal. There is a search for a *reactive* narrative solution.

Seeking New Inspiration

In 2005, lacking video game models, I started looking to other game media that had already tackled this problem for inspiration. Pen-and-paper role-playing games ask Game Masters to improvise, generate player content, and tell stories on the fly. Despite such games being an original inspiration for computer RPGs, video games like *Oblivion* and *Neverwinter Nights* have focused on replicating pen-and-paper gameplay, not the single-player reactive storytelling experience. By studying these improvisational techniques in action, I hypothesized that an algorithm could fill the Game Master's narrative setting role for a single-player story game, an "encounter sequencer".

Over years of play, I sequenced and cataloged pen-and-paper game sessions of five Game Masters. Each story element was given a letter to represent the part of the story it was describing. A character might be assigned to "A", a location to "B", a plot to "C", another character to "D", etc. Each element was assigned either a lowercase letter if it was just a quick non-interactive mention, or a capital letter if it was a major interactive plot point. Connected elements that were part of the same scene were underlined. Elements that were tied to a location were denoted with a " ' ". This generated play sequences such as "abacdAcefBafAcDe'cE"', which could go on for many lines depending on the length of the session, style of game and Game Master, and speed of play.

Analyzing this data, it quickly became apparent that while games varied widely, all had similar internal structure. Different Game Masters and different source games had different flavors, say split-up story lines or long drawn out combats, yet many seemed to share common elements. Patterns emerged. For example, most all encounters focused around a single big event. Many of these elements were part of short sequences that could be pre-determined, say, a 3-part arc. Others would be a single element from a very long story arc, foreshadowing background plotlines. The mixture of these elements appeared semi-random, but post-questioning showed it was determined by game pace and player goal more than specific narrative need. The use of these element patterns fulfilled that need. Pace or tension was clearly the most significant factor, arcing the game towards a climax. Different themes were also frequently invoked, such as loyalty, comedy, or revenge, to better vary and explore the story. Surprisingly, many of these elements were also location-independent. In response to events, Game Masters would subconsciously move major events and characters around to maintain the game's pace and give the players the appearance of local agency. This immediate response to the player's actions created more agency then an accurate world simulation does. Game Masters would also unconsciously mix in "setting" elements – elements that provided local color and character to the world or to a particular faction or character.

Defining a Storytelling Algorithm

Discovering these patterns led to the creation of an AI algorithm called the "Encounter Manager" that could mimic a Game Master. Each game was defined by a sequence of "encounters", like scenes in a movie. The sequences of related encounters, separated by time, were termed "story threads". Each story thread represents independent plot lines in the narrative. Correspondingly, each story thread contained "story knots", pre-defined short discrete setups that would lead to one encounter. I discovered later these are similar to *Façade's* "story plots" and "beats".

The Encounter Manager began by taking these pre-authored knots and trying to sequence them together into a storyline to fit the designer's goals and the player's actions. For simplicity, each knot corresponded to one encounter. One story thread at a time would determine the current plot line and the next pool of knots the Encounter Manager could choose from.

While regular knots are tied to story threads, independent "detail" knots were interleaved into the sequence to represent setting elements that flesh out the world or foreshadow future plot lines. Other times, these detail knots represented long-term plot lines in a traditional narrative intersecting the local encounters. Detail knots mimicked story elements that don't directly advance any particular plot, but create a sense of world and character necessary to keep the story rich. The mix of available thread knots and detail knots allowed the Encounter Manager to break up a story thread when reactively desirable. Stylistically, all of the one-off details tended to be the easiest to author and some of the most creative, while thread knots showed the most narrative range, tending to start small and then grow into important narrative plot points and big moments.

In the data representation (Figure 1), each thread was given a narrative theme and each knot was given a pacing level from 1-10. The thread would contain an average pace to keep thread selection relevant. The algorithm would choose a thread based on a currently desired themes using a weighted pick, taking into account player and designer input. Then it would pick the next knot from the chosen thread and available one-off knots and apply it to an encounter, looking to gradually increase the desired pace of the game. Once that encounter finished, the game would update the designer's chosen narrative inputs and the algorithm would run again. Completed knots could be flagged by a designer to be available again after a certain encounter time-frame. Failed knots could be re-queued or aborted. When a thread finished, the algorithm would drop the current game pace by a level-designer tuned amount, automatically creating a narrative pacing curve with peaks and valleys.

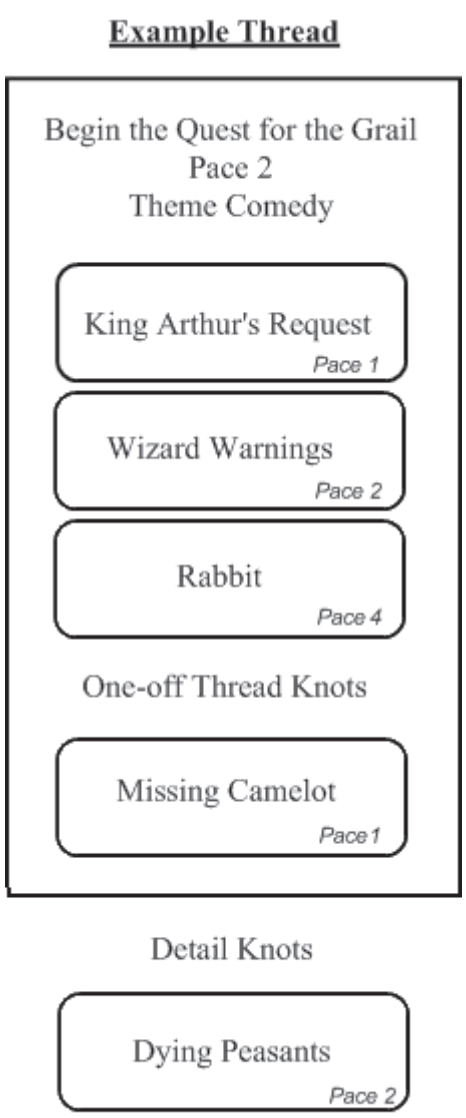

Figure 1

This Encounter Manager is *reactive* because it takes external input and uses it to create the next step in the narrative at each point. Players who give the same input and share the same random seed would see the exact same story result. The Encounter Manager shares parts of both the structured and simulated narrative forms, but is distinct from each. The pieces of each encounter are authored similar to structured narratives, but these pieces are sequenced together based on the game state similar to simulated narratives. This reactive storytelling occupies a granularity between the two – not so large as to lay out the entire story, but not so small as to be completely defined by game actions and systems.

Pen-and-Paper Playtests

Early playtests of the Encounter Manager algorithm replacing a live Game Master in specially designed pen-and-paper games led to many iterative improvements. We found longer threads were usually unnecessary and ran the risk of blocking the algorithm, so story threads were kept deliberately short – 3-4 knots. Often a designer would just come up with a dramatic climax and then work backwards, choosing one or two setup knots that built to the climax and then one subtle entry knot that would kick the thread off. Intriguingly, broken-up long threads did not need to be explicitly tied together. They could simply be indirectly grouped by theme and faction, using the pacing arc of the level to automatically spread them out. Testing showed different designers could even independently author short threads with the same theme and characters in mind and have them work together naturally, each replay of the test feeling unique. The Encounter Manager made all these thread knots interweave with the detail knots and appear to be part of one authored narrative with multiple complex thematic layers happening simultaneously.

This 3-level layering of long plots from detail knots, story threads, and local setting detail knots created lots of depth and kept the story fresh. This meant the overhead of executing simultaneous story threads wasn't needed to create complex stories. Additional refinement showed it was very powerful to combine several detail-style knots with each story thread, or "one-off knots". Like thread knots, these un-sequenced knots would build on the thread's characters and themes, and were only allowed to run while that thread was active. One-offs gave the Encounter Manager supporting material, so as extra encounters were needed the algorithm could easily interweave one-offs with the detail knots to build up the theme and pace simultaneously.

Informal questionnaires from the role-players in the pen-and-paper playtests showed player satisfaction with the Encounter algorithm was very high – most of the pen-and-paper players who tried both approaches said they thought the sequencing was a successful imitation. In particular, difficulty balancing and pace management were much improved over human Game Masters. Surprisingly, no one noticed the lessened interplay of story elements in one encounter, or that during the games the algorithm had several times run out of content and left encounter spaces empty or generic. In some ways the Encounter Manager seemed better then a human director – a smoother experience, more focused. In other ways it was worse - there was less encounter variety and the model could get caught without enough content in for the space. But these limitations actually fit modern video game design well. Video game play encourages more gameplay repetition then pen-and-paper. Players will naturally avoid boring or empty areas and move to more interesting ones. Furthermore smooth play is known to be a major factor in player flow and player retention. Overall, the study of pen-and-paper role-playing encounter sequences for algorithmic imitation showed promise, and more rigorous analytic study in this area could prove fruitful. These results encouraged us that Encounter Management could

be a major step towards better stories in single-player video games, and led to several different in-game implementations of the concept.

Taking Reactive Storytelling In-Game

In-game testing showed the concept was both flexible and avoided many of the traditional designer fears about interactive storytelling. While heavily scripted location-based encounters were still difficult to author and limiting for the player, Encounter Management made other encounters simpler to pull off and achieve more narrative significance, particularly agent-focused, audio, or player-reactive events. Furthermore, Encounter Management created opportunities for new gameplay. Encounter Management allowed for new types of event triggers such as time-based and player-performance-based triggers in addition to the typical location-based triggers. To maintain game pace an encounter could fire if a player hung out in one place for too long, or hadn't had a conflict for a while. Another encounter could fire if the player had been in an encounter for a certain period of time. This kind of new gameplay helped break up what is traditionally a trigger-bound experience, and combined with the playtest content examples helped convince designers encounters would not be too difficult to author.

In another example, knots could sometimes require a particular location or precondition to be selected. If this knot was next in line, the algorithm would just delay or abort its thread. Uniquely however, the system could also attempt to push the player towards a goal location, using encounters that move the player like chases, escapes, follows or empty space. These features helped overcome design concerns that the Encounter Management wouldn't be able to handle running out of content or encourage players not to explore. As playtests had shown, empty space in particular felt like a clue to game players, and encouraged them to avoid that direction.

Encounter Management also had other benefits. The Encounter Manager's dynamic spawning and pace tracking lent itself naturally to dynamic difficulty adjustment. Each play-through became unique – both tailoring the game to the player's play style and pace and encouraging replayability. This also made level backtracking quite possible – a space would be different every time it was visited. Interesting backtracking made the world feel much less canned and more alive and natural, similar to simulations. This new way of designing and authoring led multiple times to more creative and exciting approaches to addressing a traditional linear level space.

Having the Encounter Manager interpret player input in an otherwise traditional video game was a challenge. The range of possible inputs varied widely depending on a particular game's design and the type of encounters used. The best suited designs have transparent “game-y” player input into the Manager. Other cases used clever design or world setting to make player feedback easy to incorporate. In the worst, and unfortunately common case, the game only lets the Encounter Manager treat the player's actions as pass/fail/abort, limiting reactiveness. However, even then Encounter Management was worthwhile because it was still more reactive then the obvious alternative model, branching, retains many of its other gameplay and workflow benefits, and was easily extensible to designs more ambitious later in the project, even after content had been produced.

Player modeling AI is an obvious addition to the Encounter Manager to help solve this player input problem. But in commercial game production it still seems most effective to simply track common player actions and use design sense to predict what those inputs mean, followed by playtesting and QA verification to find places where this breaks down. Debugging tools would are also a huge help. Recording playthroughs and player inputs allows the Encounter Manager to easily recreate bugs. Going a step further, randomized player styles can generate large numbers of encounter sequences in text or the content author's tool, rapidly identifying narrative holes, authoring mistakes, or system bugs. In fact, this is a great unit test for the Encounter Manager as a whole. Clever state tracking and player simulation in the unit test can even support large-scale automated testing.

The in-game implementations of the Encounter Manager have varied depending on the game's vision and team's skill set. For example, an open world encounters could be defined by player state, locale, and input, smoothly integrating into an already procedural development pipeline. In linear levels, such as actor/object-based combat or exploration games, a different approach is needed, where each space is typically divided up into rooms or “zones” that define where an encounter can occur. Upon entering these zones, the Encounter Manager could pick an encounter and dynamically spawns the actors and objects, activating any necessary behaviors or scripts. Special care needs to be taken in the Encounter Manager to make sure objects are streamed in ahead of time and to spawn things within memory limits and out of sight. In one such use case, spawned encounters were left fairly generic, leaving the general AI and player to make the scenario interesting. In another use case, each location was filled with knots specifically authored for that spot, and streaming was only done between larger areas. This allowed the zones to be much closer together. While on the surface this space-driven authoring would seem to fall back into the content explosion problem, it critically felt comfortable to designers used to old trigger-script style systems, while still getting the narrative and workflow benefits of an established Encounter Manager. Plus, the design could then manageably encourage the player to travel through the space many times. Testing showed that, because some knots can be moved around and players are used to poor pacing and sequencing, teams could get away with much less overhead with this approach then initially feared. There are many other possible implementations possible depending on the game's event triggering mechanism, including using time or experience-focused

models rather than spacial models.

Unexpected Benefits

Surprisingly, in development encounter construction was generally faster with Encounter Management. Laying out an encounter was simpler because it was independent of what was around it. Encounters meant the game could still feel locally full during iteration without already having large amounts of interconnected linear content. And because encounter creation was not necessarily dependent on level layout, environment art construction could be largely separated from in-game events or development changes. If a room needed to be cut or changed, there could be no significant re-work – knots would automatically move and the game could be tested immediately. Whole layouts could be iterated on with ease. Encounter Management can thus sharply reduce level art production times, the largest production hurdle in most modern games. Game productions could naturally scope based on how much art is completed, greatly reducing production risk. This is likely the biggest success of the Encounter Manager. Not the improved narrative and player experience but the ability for content-driven game stories to adapt to the *developer's* requirements procedurally. Recent level design work in *Far Cry 2* represents this new approach well. Level designers were still effective at creating world spaces without guarantees of what events might happen within or around them (Morin 2009).

Once the algorithm is implemented and the game design adjusted, these development time and risk reductions can be huge. Combined with the new gameplay opportunities afforded there is ample reason to employ Encounter Management techniques in modern video games. Replayability, dynamic difficulty, player reactiveness, unpredictability, encounter layering, time and player-based events, even novelty makes Encounter Management ideal for single-player story game designs. But this is not to dismiss the narrative benefits. While in-game playtests have shown that the basic Encounter algorithm can seem invisible to players, the player experience is positively affected. The pacing controls, in particular, give a more natural ebb and flow to the play. Events move and shift based on the player's actions. Themes can be subconsciously communicated to the player, where scripts and cutscenes were the only solution before. Reactive difficulty can be measured and tied to pace, and the player's tracked inputs into the system give an easy hook for training. Storytelling and game making is, frankly, simpler.

But players assume this is a linear experience unless the difference is communicated or demonstrated to the player. This becomes an exercise in game design – if the vision is to expose the system then design mechanics and dynamics need to be created that incorporate the player's deliberate input into the thread and knot selection structure. Games that encourage replayability, not surprisingly, have had the most success with these procedurally created narratives. This has the nice side effect of both reducing the minimum game length and encouraging gameplay study – the repetition of game systems is a play that encourages learning. This variety also encourages player-to-player word of mouth, particularly when players can actively manipulate the storylines or the game design encourages cross-player story pollination. Regardless, even in games where Encounter Management is not obvious to the player, the improved experience can act like a marketing or sales driver. *Left 4 Dead*, using a similar sort of system, did this very cleverly with its marketing for its "AI Director", with breakout success.

Comparisons

In the past year, several games have attempted to use procedural techniques to sequence content. Perhaps the best example of Encounter Management so far is *Dungeons & Dragons: Tiny Adventures*, a Facebook application developed in 6 weeks that used Encounter Management to generate little 10-12 step story quests for the player to undertake, and briefly became one of Facebook's most popular games. But it is most insightful to take a step back and compare the Encounter Manager approach to that taken by another Drama Manager, namely, *Façade*. There are differences in approach. For one, Facade's Interactive Drama pushes towards highly interactive scenes with lots of player verbs whereas Encounter Management pushes towards fewer verbs with longer narrative arcs, Yet both independently developed approaches share remarkably eerie similarities (Mateas and Stern, 2005). Both operate on story knots, which have a local context that can change the meaning of the player's actions. *Façade* uses conversation direction much like encounters use story threads. Each requires a new way of thinking about game structure. Each arguably has similar limitations as well as additional freedoms. Both approaches structure themselves around pace or tension levels oriented towards a narrative arc, and strive to create interesting variety in one discrete "chunk" of the game. It seems there are a variety of chunks other then encounters or conversation beats – characters themselves, character traits, quests, level or object generation, even story moods and themes might be chunked and directed towards a designer's optimal player experience.

This points to a generic model of goal-directed procedural content, content that is constructed or sequenced with a procedurally determined purpose. So far, such purpose-driven content generating models have been relatively simple compared to other procedural methods. Many games remain too linear and tightly scripted to truly see benefits from managed encounters. Others, such as the top tier open world games, still rely on managed encounters to deliver narrative outside linear missions. Yet when used the incentive to switch appears significant – by directing even just the sequence of the content, the experience as a whole achieves a direction that sharpens

the moments and builds narrative form within the gameplay. The Encounter model, in particular, is a promising step. Working on the encounter level provides a bridge from the strict structured narratives to the broad intensive character-driven narratives that *Façade* represents. The larger scale of encounters as compared to *Façade's* beats allows Encounter Managers to build on well developed moment-to-moment gameplay. This further allows significantly less story content to be perceived as significantly more meaningful to the storyline. Encounter Manager narrative content itself can also be less complex because at this higher scale of narrative the number of player inputs and game states are much simpler, and can be constructed in a similar way to traditional video games. This makes the narrative content of Encounter Management more authorable. Compared to *Far Cry 2*, which tried using even grander yet still tightly controlled character-driven plots as chunks, Encounter Management avoids authoring nightmares by instead focusing on independent pieces of plot that run on abstract representations of theme, history, and game state (Hocking 2009).

Conclusions

Some have argued that this level of designer directorial control actually fights player creativity and expression by limiting the player's options and reducing the player's ability to predict and manipulate the world (Hecker 2009). In a sense they are correct - Drama Management systems doesn't seem to promote those kinds of play and shouldn't be judged that way. Their focus is different – on stories, smoother experiences, simpler and more focused development pipelines, better paces. Yet there is clearly a market hunger for the stronger stories and directed experiences procedural narrative brings. One could argue narrative itself is the dominant market, if movies, books, and television are considered. Player expression design is just a parallel track. Ingrained design approaches to narrative and fear of risk have so far kept Drama Management out of industry. By addressing these beliefs head on, and challenging designers with experimental data and production examples, Drama Management can be accurately weighed by development teams seeking narrative solutions.

References

Hecker, C. 2009. "Meaning, Aesthetics, and User Generated Content". *Game Developers Conference (GDC '09)*, San Francisco, CA, USA, March 28-April 1, 2009.

Hocking, C. 2009. "Postmordem: Ubisoft Montreal's Far Cry 2". *Game Developer Magazine,* March 2009.

Mateas, M. and Stern, A. 2003. "*Façade*: an experiment in building a fully-realized interactive drama". *Game Developers Conference (GDC '03),* San Jose, CA, USA, March 4 – 8, 2003.

Mateas, M. and Stern, A. 2005. "Procedural Authorship: A Case-Study of the Interactive Drama *Façade"*. *Digital Arts and Culture (DAC).* Copenhagen, Denmark, November, 2005.

Morin, J. 2009. "Player's Expression: The Level Design Structure Behind Far Cry and Beyond?" *Game Developers Conference (GDC '09)*, San Francisco, CA, USA, March 28-April 1, 2009.

Wright, Will. 2007. "Will Wright Keynote Speech". *South By Southwest (SXSW 2007).* Austin, Texas, USA, March 9-13, 2007.