The Tenth Annual Game Design Think Tank
Project Horseshoe 2015

Group Report:
Generative Systems, Meaningful Cores

Participants: *A.K.A. "Jazzhands"*
Squirrel Eiserloh, SMU Guildhall
Jake Forbes, Independent
Jonathan Hamel, Giant Sparrow
Thom Robertson, Incandescent Workshop LLC
Mike Sellers, Indiana University
Facilitator: Linda Law, Project Horseshoe
Will Emigh, Indiana University and Studio Cypher
Jason Grinblat, Freehold Games
Ray Holmes, GSN Games
Ian Schreiber, Rochester IT
Steve Swink, Cube Heart

Introduction

How do we build complex generative systems with meaningful cores? How do we reap all the benefits of procedural generation while constraining the system to sufficiently invoke a vision, make an argument, construct a narrative? Even more fundamentally, how do we talk about this stuff? Can we construct a vocabulary to articulate the dimensions of content generation in service of narrative? In this paper we summarize the workgroup's exploration of these and related topics.

Problem Statement

How do we as designers direct the deep possibility space of proceduralism with narrative intent? Can we provide an analysis of both narrative and procedural generation that answers these questions:

- What purpose(s) might procedural generation serve in the aid of narrative?
- What are the essential ingredients of a meaningful story?
- How can we use procedural design techniques to author content? How have previous designers used procedural generation to tell stories?

What Procedurality Offers Narrative

As designers we're primed to want to use systems to evoke experiences. But when we start to talk about using systems to construct narratives, the question of motivation arises. Do narratives need systems? What can systems, and specifically procedural systems, offer narrative beyond what can be accomplished by

traditional means? We've defined four categories of answers.

Personalization

We've had experiences where we've relayed, or have had relayed to us, stories of unexpected game play: harrowing near-death experiences, surprising results that arose from unconventional strategies, or embarrassing follies that followed from misreading situations. What allows us to tell or hear these "player" stories and be entertained is their uniqueness; they weren't scripted by an author who anticipated both the outcomes and our reactions to them. Instead, they were the result of a particular confluence of systems that produced a singular, and thus personalized, experience.

We value personalized experiences. We enjoy moving through a narrative in a way no one else has quite before. And we enjoy sharing these experiences with others. Procedurality allows us to set the stage for these kind of personalized player stories. We don't have to abandon all authorship, though. By engineering our procedural systems, we can outline the storyspace and let players tell *their* stories while exploring *our* themes.

Authenticity

Related to the personalization of procedural narratives is the concept of their authenticity. In some ways, we privilege stories without authors as more authentic because they mimic our experiences in the real world. When the minotaur's attack results in an unlikely fumble and we're able to roll away and kill him with a low percentage arrow shot, there's a sublime "realness" to the experience that transcends the same encounter had it been carefully scripted by a designer. Procedural systems can enable "realer" narratives because they don't directly provide narratives. Instead, they merely provide the components and the rules that act on them to *produce* narratives, just as reality merely provides the components and the rules that act on them to *produce* the stories that occur in our daily lives (Admittedly, this assertion takes for granted a certain set of philosophies, namely, those that assume there isn't a script for real life).

Authored stories can be powerful tools for evoking certain feelings in players, but the breadth of generative systems and the interactivity of games as a medium endow us with a different set of tools that can be used to set the stage for "authentic" experiences.

Story Volumes Rather than Storylines

Beyond producing personalized and authentic traditional stories, procedurality also enables a new class of insight into narrative. Games occupy a unique space in the way they allow us to experience similar, but not identical, content over and over again. We can use this characteristic to our advantage in setting the stage for procedural narratives.

In a traditional narrative, we follow a predefined storyline. When the traditional narrative is implemented in a game, we might retread portions of that line over and over again as we replay portions of the game, but often we do so without any new insights. In a system designed to allow for procedural narratives, the parameters of each playthrough can allow us to experience a new narrative that's different but still congruent with the previous narratives. Each narrative gives us a new insight into the themes that the designers want us to explore. In this way, the procedurality allows us to explore a *story volume* rather than a single storyline. Our understanding of the game's themes arises from the aggregation of our experiences playing through similar story lines. Playing a game with a robust procedural landscape for producing certain types of narratives might be more akin to watching several films in a genre rather than a single film.

Practicality

Finally, there's the practical side of development. Procedural generation can be a great way to produce a

lot of content at a low cost. Narrative content is expensive, and procedurality can potentially provide much more narrative content than even large development teams can budget for. This isn't to say that procedural narratives should wholly replace authored content or that procedural content is effective even when it's poorly executed. Rather, even in naive implementations, procedural content can be used to complement authored content, expanding the amount of play at a fraction of the cost of a fully-authored game.

Designing for Maximum Narrative Potential

Apophenia with Intent

Humans are hardwired to look for patterns in randomness (apophenia). But patterns alone do not a meaningful story make. By weighting the elements in our generative systems in service to a theme, we can increase the chance that the player will piece them together into an experience as meaningful as an authored story. Prioritize theme and tone over plot. Allow for moments that exist solely to be emotional or evocative with little or no connection to story. Stay vague enough so that the player is able, even required, to do the heavy lifting about figuring out how the pieces fit together.

Characters and Conflict

In narrative rich games, it is often characters, not plot, that resonate most strongly with players. A player's relationships with NPCs may be directly measured, as in a dating sim, implied by who a player chooses to party with in an RPG, or exist outside of systems entirely in the player's head. When using procedural systems to create characters, allow for sufficient meaningful variance in NPCs to afford possibility of some which are special to player. Once a player has found characters to latch onto, we are primed to exploit the emotional ties produced by strong attachments and extreme relationships.

Persistence of NPCs across procedurally generated encounters creates a personal history that can complement a framing story or serve as a central narrative. In the *X-COM* games, for example, any of the procedurally-generated soldiers on the player's team has the potential to emerge a hero or die tragically. In *Hero Generations,* the actions of one playthrough inform the hero and scenarios of the next generation so that the procedural elements carry the weight of legacy.

Narrative Payload

Leverage emotionally-loaded scenarios to foster drama from fewer building blocks. Emphasizing themes of love and loss, revenge and redemption, friendship and family amplify narrative potential. Case-in-point, in the heavily procedural game *Road Not Taken,* the player is tasked with rescuing lost children from a freezing cold forest. The children could be replaced with bags of gold and it wouldn't impact the game mechanics at all, but the player's relationship to their gameplay objective would be far less emotional. *Fallout IV*, which utilizes procedurality to a lesser degree, maximizes the narrative potential of its emergent and procedural systems with an opening scenario that flavors everything that comes after. Murdered spouse, stolen child, world in ruins, hero out of time -- once the player has been given these lenses through which to see the gameworld, they are primed to find meaning outside of the authored story bits.

Mechanics themselves can be emotionally charged to powerful effect. While not a procedurally driven game, *Papers Please* is a game where every one of the player's actions has life or death consequences for the player's family and the families of those he meets. *Redshirt* uses generative systems to create a crew on a Star Trek like starship but limits the players' interactions to those of a social network, "Spacebook." Actions are zero sum and revolve around liking posts, organizing and attending events, and relationship status. Choices can advance the player's career or lead to defriending. Feelings are hurt. Even though the setting is whimsical and cartoony, experience with real life social networks makes these interactions

emotionally charged.

With the story volume primed, throw lots of darts. Accept that not every beat will line up the way an authored story would, but trust that the player will fill in the gaps.

Techniques for Producing Generative Systems with Meaningful Cores

Historically, designers have adopted a variety of philosophies, approaches, and methods with regard to the interrelation, interaction, and integration of procedural generation systems and human-authored content, with varying degrees of success. Some such techniques are highly contextual and game-specific; others are more general and may be applied more universally.

A common set of questions present themselves surrounding any approach:

- Which element(s) of the final product are generated by the system? Which are human-authored?
- For human-authored elements, when and how are they integrated into the experience?
- For procedural elements, to what degree (if any) are they influenced by context or player choice?
- Do procedural elements select, modify, or combine human-authored elements? If so, how?
- Do authored elements drive, influence, limit, or modify procedural content generation? If so, how?

While a formal or exhaustive taxonomy of such techniques far exceeds the scope of this paper, a few common approaches may be described.

Procedural baseline

One simple approach is to use a procedural content generation system - be it for character names, terrain generation, building creation, etc. - as a jump-start for hand-authored content: e.g. generate 500 random names, then hand-pick the best ones and use them manually. Generate procedural terrain, then hand-modify and save out the results. Generate 100 villages, then pick the best one to start with, tweaking it further, and shipping it as “computer-assisted authored content”.

Procedural detailing

Similarly, one can start with hand-authored content and then rely on procedural systems to populate it with details. For example, moss can be automatically applied to the north face of trees, rocks, and huts placed by the developer; an erosive algorithm may applied to a hand-crafted river. Artists and level designers can quickly “paint” procedurally-generated ground foliage into their terrain, without having to specify every bush, flower, and leaf.

Mad libs

With a “mad-libs” style approach, authored content provides discrete gaps which are filled procedurally. The simplest example of mad-lib content might be a text adventure that says “Nice to have you with us, Susan”, expressed by the content author indirectly as “Nice to have you with us, $Name$” or similar. Likewise, “She knows him from Homlet - they grew up there together” might be authored as “$Subject1$ knows $object2$ from $Village$ - they grew up there together”. As evidenced by this example, the devil in the details for mad lib systems often lies in the determination of what what variables exist, and how they are exposed to the designer.

Simple mad-lib techniques can easily be applied to non-textual content as well; for example, a hand-crafted castle or spacecraft can be rendered with its faction’s flag or insignia displayed on it (in a place reserved for

the insignia by the content author).

A mad-lib approach can also be applied to narrative plot and character elements as well; a human author may write specific hand-authored content for canned character relationships (sibling, spouse, lover, parent/child, master/apprentice, etc.) and apply these dynamically to in-game characters, which may in turn be hand-authored (or procedurally generated). Likewise, plot elements can be dictated in general terms - one party member will betray the others in Act 3 - but the betrayer's identity may be chosen at runtime.

Scripting

In a scripting-based approach, the designer essentially writes high-level procedures or logic for determining the use of dynamic content generation or deployment. IF you have talked to the old man AND you have found his granddaughter THEN he will tell you of the secret cave. Scripted approaches, while extremely powerful, can be tedious to author - and often puts programmatic burden on non-programmers, often without the advanced IDE tools available to programmers (debugger, breakpoints, watch), so in some ways it can easily become the worst of both worlds (design and code). Scripted approaches also sometimes suffer from linear (or even exponential) content authorship costs, as every possibility needs to be accounted for. Still, scripting remains a powerful tool, and is still popular with some developers. Also, it should be noted that any of the other techniques here may be expressed with scripting-like elements, such as conditional rules.

Blueprints

In this context, a "blueprint" is a hand-authored piece of meta-content which does not define each piece of content explicitly, but rather defines its generative possibility space; actual instances of content of that type may then be procedurally generated from that blueprint. For example, an oak tree blueprint might be defined as being 10-40m in height, having between 4 and 8 branches, with foliage starting between 1 and 2 meters off the ground, etc. A wizard's toward blueprint might be defined as having between 7 and 12 levels, with a single bedroom amongst the top three floors, a front door on the bottom level, and windows on all other levels. A feral orc blueprint might dictate range-variations for such creatures: minimums and maximums for height, weight, skin tone, arm length, walking speed, gait, as well as AI characteristics such as bravery or aggression, gear worn, loot carried, or distinctive features (scar, eyepatch, mohawk). Thus, an entire army of feral orcs may be created from a single blueprint can feel very organic, with each orc looking, moving, and acting with slight differences.

Tags

A tag-based approach is similar to the mad-libs approach, but with an increased level of abstraction. The author tags items/plants/animals/characters/buildings/locations with different characteristics (often from a finite official list), which then allows those objects to be eligible for use in other procedural content generation. For example, if an evil temple is meant to be "scary", it can choose decorative elements from amongst those marked as being scary. A wizard's tower may have a requirement that it is situated in an "exotic" location, which may be defined as being any of: the top of a mountain, the heart of a volcano, the bottom of a lake, an island in the clouds, the center of a thick swamp, etc. Twelve different variations of "retail chair" can be chosen to populate different shops with different styles of seating.

More concretely, tags can be used to mark variations of foliage as being appropriate for appearance in various biomes, so that oak trees appear in deciduous and boreal forest biomes, but not in desert or savanna.

Seeding the data jar

A number of techniques require the author to supply examples of desired content, which are consumed by an algorithm which then attempts to produce more content "in that vein". Markov chains, for example, are

popular for use in this purpose; seeding the system with the phrases “I am wet and I live in a box” and “I am poor and hungry” might result in such generated sentences as “I am wet and hungry” or “I am poor and hungry and I live in a box”. Name generators may take lists of hundreds of “seed names”, mine them for letter and syllable use and word-shape, and recreate names using similar word-shapes and syllables. Markov chains and similar techniques may be applied to almost any form of generative content, including everything from progressive notes in a melody line to adjacency of rooms in a house.

Puzzle pieces

A content author may craft various “puzzle pieces” of content, with “connection points” at their extrema; other puzzle pieces may be selected which share compatible connection points with previously-chosen pieces. This can easily be applied to rooms (hallways connect to doorways or other hallways) as well as character relationships (the betrayer loves the affluent, who is in a relationship with the betrayed, who is a longtime friend of the betrayer) or level generation (the gold key must be reachable without having the gold key) or even narrative lines (the murder’s revelation must come after the murder).

Constraint satisfaction

In a constraint-based approach, the author dictates criteria which must be met by a procedurally generated element (location, item, character, building, plot); the procedural content generation system then attempts to satisfy all of the required (and many/most of the desired) traits requested. For example, we may define a typical house as having: exactly one kitchen, at most one garage, between 2 and 4 bedrooms and between 1 and 3 baths, etc.

Constraint satisfaction as a content-generation method is a broad field presently and previously studied in depth by academic researchers. Implementation approaches may vary dramatically, from trial and error (keep generating totally random content until you satisfy all constraints) to evolutionary (generate thousands of attempts, then mix and match sections of more-successful attempts) to winding/unwinding (keep laying puzzle pieces until you find a solution, backing up to try a different variation any time you get stuck).

Player-observational and self-observational systems

One technique that deserves much more attention is the use of systems which pay attention to, and are “aware” of, what the player has (or hasn’t) done yet in the game, as well as what sort of content the system itself has (or hasn’t) yet served up. Simple examples of this technique include dynamic player instruction which skips moot hints/tips (e.g. “Build archer towers to defend from a distance!” when the player has already built dozens of archer towers), or the selection of music or zingers to play which avoids those recently played. More nuanced examples might include systems which make statistical observations about player actions or previously-served content and move to cater to, or move away from, emerging trends, or systems which attempt to ascribe meaning or intent to player action, and act on those observations.

AI and narrative directors

Another high-level technique involves the use of so-called “AI directors” which observe and/or dictate content to engineer a desired mood, tension, pacing, or challenge. Human content authors can direct these systems at a high level, providing desired curves for each of several factors (similar perhaps to Vonnegut story shapes) which the system tries to fulfill by adjusting the “temperature” of each of several “knobs” exposed to it (type, frequency, number, and capability of enemies, randomization of loot drops) based on various metrics observed (player character health and inventory, skill use), comparing observed levels to desired levels. Currently a popular subject of game-related AI research.

A narrative director system can also use observe player experience as feedback into generation systems. For example, if the game observes that the player had the greatest overall difficulty with swamp areas, the

final “boss” level can be chosen to take place in a swamp area, populated with a context- appropriate enemies.

Evolutionary / machine learning techniques

Machine Learning and evolutionary techniques (such as genetic algorithms) can also be an extremely interesting and powerful source of procedural content generation, but many of them require human evaluation/intervention at some level in order to define, or evaluate, what is “good” from what is “bad”, such that the algorithm can continue to progress more toward “good” content generation. Yet another deep topic of extensive academic research.

Layering of techniques

One of the most useful methods is the use of one or more of these techniques in combination. A particularly good, and easy, way to do this is to take a layered approach which applies one technique for high-level authored/procedural content integration (e.g. plot generation), and another technique for lower-level content integration (e.g. item generation).

Outputs as inputs

Another key strategy to the layering of authored and procedural content integration systems is to establish a content generation pipeline which produces concrete, human-readable, human-editable output at each stage (e.g. in a text/XML form), which is then used as the input for the next (usually, lower-level) content generation/integration stage. For example, a human-directed (via use of constraints) procedural generator which produces blueprints (as XML files), which in turn can be used to generate instances of content. This is powerful for a number of reasons: first, it allows designers and programmers insight into each stage of the generation pipeline, demystifying the sometimes inscrutable black box of procedural content generation; second, it affords human content authors to inject hand-crafted content at any level of the system, from macro to micro.

Eastwood: a Procedurally-Driven Card Game for Telling “Western”-style Stories of Redemption

In order to test some of the basic ideas we uncovered in our exploration of generative systems, we decided to paper prototype a simple game with a generative narrative. In order to count as a generative system, we required that our design have a strong theme (narrative design) but allow the player to play through a variety of non-authored scenarios (procedural generation).

Our design began with the themes of Westerns, particularly *Unforgiven*. In that movie, the main character is a gunfighter who has given up alcohol and killing but comes out of retirement to provide for his family. The primary themes of that work that we wanted to carry over into our game were:

1. the costs of violence
2. the trade-offs between “doing right” in the short-term and long-term

Summary

In *Eastwood*, the player takes the role of a retired gunfighter who has given up wickedness and killing. Your daughter has run off with the antagonist, a man whom you know to be trouble. The game consists of your attempts to chase after her in the hope of convincing her to return with you. In a more strongly authored version, additional story elements can be described before or revealed during gameplay. For example, unbeknownst to your daughter, the antagonist was the one who killed her mother.

This is a single-player card game, although it is more fun with many people discussing the story as it unfolds.

Components

3 Tracks (1-6), one each for ‘wickedness’, ‘information’, ‘infamy’
12 Location cards
24 Person cards, 2 each of 12 unique People
1 Farmhouse (starting location)
1 d6
12 Tombstones

Goal

The game ends when the player has explored the full Location deck, at which point they’ve found their daughter. To win, they must use the information they’ve gathered to convince the daughter to return.

Mechanical details are in the End Game section below.

Setup

The player starts at their farmhouse, which is played directly in front of them. The Locations are shuffled and 8 are placed together to form the Location deck. The other 4 are not used in the current game. The Person cards are shuffled to form the Person deck. The players wickedness, information, and infamy are all set to one.

Gameplay

In *Eastwood*, the player travels to a new location each turn where they meet new people and attempt to discover information that they can use to later on. In keeping with the themes of desperation and revenge, whenever the player goes to a location, someone dies (at the player’s choice, but not necessarily directly because the main character killed them).

In each turn, the player will:

1. Travel to a new Location
    1. Encounter bounty hunters
    2. Reveal the Location
2. Meet 2 People
3. Mark one Person for death
4. Resolve effects

Travel

To travel, the player must first encounter bounty hunters. The player rolls a die and compares its value to their current infamy. If they roll higher or equal, they successfully avoided them. If they roll lower than their infamy, then they encounter bounty hunters, whom they kill, which increases their wickedness by 1.

After encountering any bounty hunters, the player draws the top Location card and places it just above their current location. If this Location has special rules listed on the card, they may override the rules as written. For example, the Saloon requires the player to meet 3 People and Mark 2 (instead of 2 and 1).

Meet

Now, the player draws two Person cards and places one to either side of the current Location. Each Person card has a name or role, an age, and two consequences, one for killing and one for saving.

For example:

Sally (age 23)
MARK: Burn Location
SAVE: No effect

Person cards may also have unique rules that override the written rules.

If the player reveals a Person card that has already been Marked, that card represents the player being tormented by the Person they killed. They rotate the Person card sideways to indicate that they are a Ghost, then draw and place a replacement Person side. This will mean one side of the Location will have a Person card on it and the other will have a Person card and a Ghost (a Person card rotated 90 degrees). This may have an effect during the Resolve phase.

Mark

With the Location and 2 Person cards revealed, the player has to mark one for death by deciding which will die in this scene.

To do this, player player places a Tombstone marker on that Person card to indicate the Mark. Players are encouraged to discuss what just happened. This might be character-directed (the main character kills them in a shoot-out) or it could be ancillary (they get shot accidentally when the main character gets run out of town).

Resolve

The player now resolves the effects, in this order:

1. Any special effects on the Location. For example, the Graveyard gives 1 wickedness to the player for every two people they killed in previous Locations.
2. Follow the MARK effect listed on the Person card who has just been Marked. For example, Sally requires the player to Burn a Location (discard a Location from the Location deck). If there is a Ghost next to this Person, the player gains 1 Wickedness or 1 Infamy (their choice).
3. Follow the SAVE effect listed on the Person card who was not Marked. For example, the Sheriff gives +1 information.

If the player's wickedness ever increases above 6, then they lose the game, having gotten sucked back into the life of violence that they had forsworn.

End Game

The game ends as soon as the player has to Travel to a new Location and the Location deck is empty (usually after 8 Locations, but sometimes less).

There were several variants tested for this section of the game depending on how important we wanted the "scene" with the daughter to be.

For a longer scene, the daughter had a deck of arguments that were revealed one at a time, each with a symbol. The player had to play an information card with a matching symbol to each argument as it was revealed in order to stay in the game. If any revealed argument couldn't be matched, then the player lost. If all of the arguments were matched, the player won.

For a shorter scene, the player adds their infamy and wickedness values to create an untrustworthiness rating. From there, the player subtracts their information rating. The player then has to roll higher than the remaining untrustworthiness rating in order to win and convince the daughter to return home.

Cards

These are some samples of the cards created during prototyping.

Locations

Name	Effect
Farmhouse	The starting location
Saloon	Meet 3 People and Mark 2 (instead of 2 and 1)
Church	Double any wickedness earned at this Location
Graveyard	Gain 1 Wickedness for every two People Marked at earlier Locations.
Brothel	Burn 1 Location

People

Name/Role/Age	MARK Effect	SAVE Effect
Sally (age 23)	Burn 1 Location	No effect
Cowboy Joe (age 18)	+1 Infamy	+1 Information
Patches the Horse (age 7)	Burn 1 Location	No effect
Sheriff (age 48)	+2 Wickedness	+1 Information
Deputy Alive (age 19)	Burn Location	+1 Information, +2 Information if the Sheriff has been Marked
James (age 5)	+1 Information, +1 Infamy	No effect
Preacher (age 50)	No effect	No effect
Sally's Baby (age 4 months)	+2 Wickedness	No effect
Martha (age 62)	Next Location, MARK one extra Person	Next Location, Meet one extra Person

Lessons Learned

Creating this simple card game allowed us an opportunity to very quickly and easily test the impact of some of the strategies we explored in our discussion. Through this practice, we discovered some guidelines that improved the generative narrative aspects of this game.

Relationships

One of the most fruitful areas of change was creating visible or obvious relationships between characters and other characters and between characters and locations. For example, meeting Sally's baby after marking Sally for death has a much stronger impact than meeting another baby. Similarly, meeting a Preacher in the Brothel immediately gets players imagining why this happened, since they have a very strong negative relationship.

Appropriate Details

One major difficulty in designing Eastwood was deciding which details were important and which could be left out. Including more details made it easier for players to create an understandable story but left less room for exploring a story and sometimes included contradictory results. If *Eastwood* had gone this

direction, the player would have moved directly from location to location following the plot of *Unforgiven*, with only the characters from those locations showing up. Including fewer details allowed players to explore more of the story-space but reduced the strength of the theme when taken too far. For example, an early version of *Eastwood* included more generic locations (Canyon, Town, City). These locations were too vague to give players a starting point at a story. Choosing elements that immediately brought to mind stories in the Western genre worked much better (Church, Brothel, Saloon, Graveyard).

Mixing 'hard' and 'soft' narrative

We describe a 'hard' narrative as a narrative that is enforced by the systems of the game, whereas a 'soft' narrative is one that may contain designed narrative elements but that may not occur or that may easily be interpreted by the player. On the 'hard' end of the spectrum would be a cut-scene or, in more systemic fashion, a linear progression through set locations. The 'soft' end of the spectrum would include objects and systems that allow for emergent player stories to form, like the classic stories players have of the disasters that befall their settlements in Dwarf Fortress. Soft narratives are valuable in allowing players to explore a story-space rather than just one story but hard narratives allow the designer to enforce the story that they want to tell.

We found that the best experience happened when we mixed the hard and soft narratives in the right proportions. The actual proportions will be different for different games, of course, but our guiding principle was to include as much hard narrative as was required to make sure that our themes were present while making the rest of the narrative as soft as possible. This is what lead to the system when the player has to mark a person for death at each location, since that forces the idea that this mission has a significant cost as well as giving the player a framework to think about the short-term versus long-term costs of their actions.

Making story elements visible

This guideline seems obvious but is often overlooked. Players are able to tell much better stories if they can see, at least in summary form, what happened before. For example, the impact of seeing Sally's baby after marking Sally for death would be significantly diminished if the player didn't remember that they'd killed Sally. Our solution was to create a sort of story chart through gameplay where the locations, people, and tombstones remain out to give the player a visual reminder of where they've been and what they've done. A digital game might have to be more creative with their solution, since it will include a lot more detail than this simple card game, but it will be even more important, since players may have significant breaks in between play sessions.

Repeating story elements

People tend to interpret repetition as meaning. If a character or object shows up more than once (particularly if they have a name or some easily identifying feature), they're assumed to be important. Designers can take advantage of that by identifying characters as important (if the player happens to see them a lot) or by making characters important (by making sure they show up a lot). In *Eastwood*, the ghosts are one way of repeating story elements to increase importance. Having the ghost appear after the character is killed makes the player think more (and hopefully feel more strongly) about their death. Relationships are also a good way to repeat elements. Seeing Sally's baby acts like foreshadowing for Sally (or vice versa), which gives the latter character more impact.

Results

By playing through *Eastwood*, players will create a unique story of an attempt at redemption and what that costs. The choices the player makes at each location have long-term mechanical consequences but may also have narrative consequences as well. The arc of the game is well-defined, with wickedness, infamy,

and information ratcheting up as players get closer and closer to their daughter and the end of the game. Within that arc, players create their own small stories to answer the questions posed by the unique configuration of the cards they come across. Why is the Preacher in the Saloon? Perhaps he is preaching temperance or sacrifices himself so that the other people in the Saloon might live. Why are Cowboy Joe and Sally's Baby in the Church? Perhaps Cowboy Joe has brought the baby here in the hope that the church will take care of her now that Sally is dead or maybe Cowboy Joe is the father and is praying for Sally. The fluidity and breadth of possibility allow the player's actions in each location to ripple across the game as a whole and create a redemption arc that feels different every time.

Copyright 2000-2015, Fat Labs, Inc., ALL RIGHTS RESERVED