Integrating Plot, Character and Natural Language Processing in the Interactive Drama *Façade*

Michael Mateas[1], Andrew Stern[2]
(co-authors listed alphabetically)

[1] Dept. of Computer Science, Carnegie Mellon University,
michaelm@cs.cmu.edu
[2] InteractiveStory.net,
andrew@interactivestory.net

Abstract. Façade is an artificial intelligence-based art/research experiment in electronic narrative – an attempt to move beyond traditional branching or hyper-linked narrative to create a fully-realized, one-act interactive drama. We are completing a three year collaboration to engineer a novel architecture that integrates emotional, interactive character behavior, drama-managed plot and shallow natural language processing. In this architecture, authors can organize hierarchies of reactive behaviors and natural-language discourse contexts into entities called *story beats*, achieving rich, robust local interaction; a *drama manager* globally sequences beats chosen from a large pool in order to make story tension rise and fall to match an Aristotelian arc. Within this architecture we are building a dramatically interesting, real-time 3D virtual world inhabited by computer-controlled characters, in which the user experiences a theatrical drama from a first-person perspective. Façade will be publicly released as a free download in 2003.

1 Introduction

Interactive drama concerns itself with building dramatically interesting virtual worlds inhabited by computer-controlled characters, within which the user (hereafter referred to as the player) experiences a story from a first person perspective [1]. Over the past decade there has been a fair amount of research into believable agents, that is, autonomous characters exhibiting rich personalities, emotions, and social interactions [2, 5, 7, 8, 15]. There has been comparatively little work, however, exploring how the local, reactive behavior of believable agents can be integrated with the more global, deliberative nature of a story plot, so as to build interactive, dramatic worlds [16, 4]. We are currently engaged in a three year collaboration to build an interactive story world, Façade, integrating believable agents and interactive plot. Façade will be publicly released as a free download in 2003.

In Façade, you, the player, using your own name and gender, play the character of a longtime friend of Grace and Trip, an attractive and materially successful couple in their early thirties. During an evening get-together at their apartment that quickly turns ugly, you become entangled in the high-conflict dissolution of Grace and Trip's

marriage. No one is safe as the accusations fly, sides are taken and irreversible decisions are forced to be made. By the end of this intense one-act play you will have changed the course of Grace and Trip's lives – motivating you to re-play the drama to find out how your interaction could make things turn out differently the next time.

This paper discusses our approach and motivation for the design of the architecture, followed by a brief tour of the architecture. For more detail on our artistic design goals, authorial idioms and early observations of the successes and failures of the system, please refer to [12].

2 Approach and motivation

To date there have been two general approaches toward creating interactive narrative experiences. One approach is to hand-craft a structure of nodes, often in the form of a graph, network or flowchart, where each node is a finely-crafted chunk of content such as a plot event, information about a character, or a discrete location in an environment. The connections between nodes are often called paths or links. Typically a node connects with a small number of other nodes. The player is given the ability to traverse the graph, and the resulting sequence of nodes constitutes the experience of the narrative. Depending on the complexity of the interconnectedness between the nodes, the range of traversals through the structure can range anywhere from very limited and coherent to very numerous and fragmented, even cyclical and never-ending. Examples include the plot structure of action / adventure games, hypertext fiction, some text-based interactive fiction, and choose-your-own-adventure books.

Another approach is to create a procedural simulation – an open-ended virtual world containing a collection of independent elements, such as objects, environments and (often simplistic) autonomous agents, e.g., NPC's. Each element maintains its own state and has procedures governing its behavior – the different ways it can act upon and react to other elements in the world. The player is just another element in the world. As the simulation runs, all elements run in parallel, allowing many things to happen simultaneously. In its purest form, there is no particular pacing or explicit structure imposed on the experience; the possibilities are only limited by the combinatorics of the range of actions and reactions between the elements in the world. (A slightly more constrained form of simulation offers the player optional "goals" or "missions" to strive for, with a variety of ways to achieve them.) The player experiences a sequence of events over time, akin to how one experiences real life, which may or may not be interpreted as "narrative" by the player, perhaps depending on how closely the sequence of events happens to resemble a narrative structure. When this happens it is called "emergent narrative". Examples include levels in a first person shooter, sim games, "immersive simulation" games, virtual reality and virtual worlds (graphical or text-based, offline or online).

Façade is an attempt to find a capable middle ground between structured narrative and simulation. We want to combine the strengths and minimize the weaknesses of each approach.

The strength of the structured narrative approach, and what is lacking from simulations, is that the system (if so designed) can offer the player a well-formed experi-

ence. This means the experience is unified, where all parts of the experience are necessary to contribute to a unified whole with little or no extraneous action, and the experience is efficient and well-paced, where the experience does not take an inordinate amount of time or labor for the player, stays interesting and never lags or gets boring (not everyone is willing to spend 40+ hours at the computer to get a complete experience). The tension of the experience may even be made to rise and fall at a pace to match a Aristotelian dramatic arc. These time-tested qualities of unity, efficiency and pacing are part of what makes good narratives so pleasurable, or at least can make them unpleasurable if missing.

The strength of simulations, and what is lacking from structured narratives, is that the player has a high degree of agency and freedom of expression at all times. Many things can happen at any time; the space of possibilities is often an order of magnitude or more larger than what is possible in even a complexly-tangled narrative graph structure. This degree of agency is part of what makes the best simulation games so pleasurable, or unpleasurable when missing.

On a moment-by-moment basis, Façade is a simulation. It has a simulated virtual world with objects, the behavior-based autonomous agents Grace and Trip, and the Guest character controlled by the human player. In the moment, the simulation offers a high degree of freedom and local agency to the player, and is where the *character* (personality, emotion, lifelikeness) of the believable agents is experienced first-hand, including varied and robust natural language discourse (dramatic conversation). Beyond what a pure simulation contains, however, is an additional invisible agent called the *drama manager*. The drama manager continuously monitors the simulation and proactively adds and retracts procedures (behaviors) and discourse contexts by which Grace and Trip operate. That is, the rules of the simulation are regularly being updated in an attempt to give the player a well-formed overall experience with unity, efficiency and pacing. These simulation updates are organized into *story beats*, each a collection of behaviors tailored to a particular situation or context but still offering a non-trivial simulation space. Beats are annotated by the author with preconditions and effects on the story state, instructing the drama manager when they make sense to use, in the interest of creating an overall dramatic narrative – a *plot*. These preconditions and effects serve to specify a *partial ordering* of beat sequences. So at a high level, Façade's collection of beats and sequencing rules implicitly define a complex narrative graph – a graph with a link structure complexity making it intractable to manually, explicitly, define. That is, the plot structures effected from the dynamic sequencing of a collection of beats would be intractable to capture in a single directed network or flowchart diagram (see [3, 14] for examples of such diagrams); perhaps the only way to visualize them is to enumerate all possible orderings, technically in the thousands. The more beats there are for the drama manager to work with, the more possible orderings that emerge, and the more global agency (plot control) the player will experience.

How are beats different than, say, levels in a first-person shooter or "immersive simulation" game? Don't game levels also update the rules of the game simulation – also a middle ground between structured narrative and simulation? The differences between Façade beats and game levels have to do with grain size, the number of possible coherent orderings, the degree of update per change in the simulation, and the

seamlessness between changes. In Façade, beats are changing every minute or so, are chosen from a pool of ~200 beats, and by design can occur in many different orders while still maintaining narrative coherence. Game levels typically change every 10-15 minutes at most, are chosen from a small pool of levels, and typically cannot occur in many different orders while maintaining narrative coherence. Also, beats change the overall behavior of the simulation to a greater degree than game levels tend to. Each beat has a custom *context* in which saying the same words or doing the same actions as before may now yield a very different result than they did in previous beats. In a typical game level, the environment may have changed from the previous levels, but shooting the same weapon or performing the same action tends to have the same general effect on the world as it did before. Furthermore, beats in Façade are sequenced together seamlessly allowing for continuous, uninterrupted immersion in the narrative, without the break in time or place typical between game levels.

The motivation to find a middle ground between structured narrative and simulation manifests itself in Façade in another key way: the architecture offers direct support for coordinated activity between the autonomous agents in the simulation in the form of *joint goals and behaviors*. This allows the author to more readily create sophisticated, lifelike coordinated behavior between dramatic characters than is otherwise possible in a strongly autonomous agent architecture. See [10] for further discussion of this issue.

Finally, we are motivated by the belief that building a *whole system* containing all the required pieces to achieve a complete user experience – a short but "fully-realized" interactive drama – forces us to address issues that otherwise get ignored or swept under the rug when developing only a piece of an architecture. Furthermore, building a complete experience allows us to release the work into the world for people to play with and critique, compelling us to put significant energy into the quality of the writing and story design.

3 Architecture and authorial idioms

The Façade architecture integrates story level interaction (drama management), believable agents, and shallow natural language processing in the context of a first-person, graphical, real-time interactive drama. To our knowledge this is the first published architecture to integrate all these pieces[1]. Façade's primary architectural contribution, besides achieving the integration itself, is architectural support for authoring dramatic beats, an architectural level which combines aspects of character and story.

---

[1] Zoesis, the company founded by former Oz Project leader Joseph Bates and Oz Project members Bryan Loyall, Scott Neal Reilly, and Peter Weyhrauch, demonstrated an unpublished interactive drama architecture integrating believable agents, gestural language, adversary-search-based drama management, and reactive musical score, at the 1999 AAAI Fall Symposium on Narrative Intelligence.

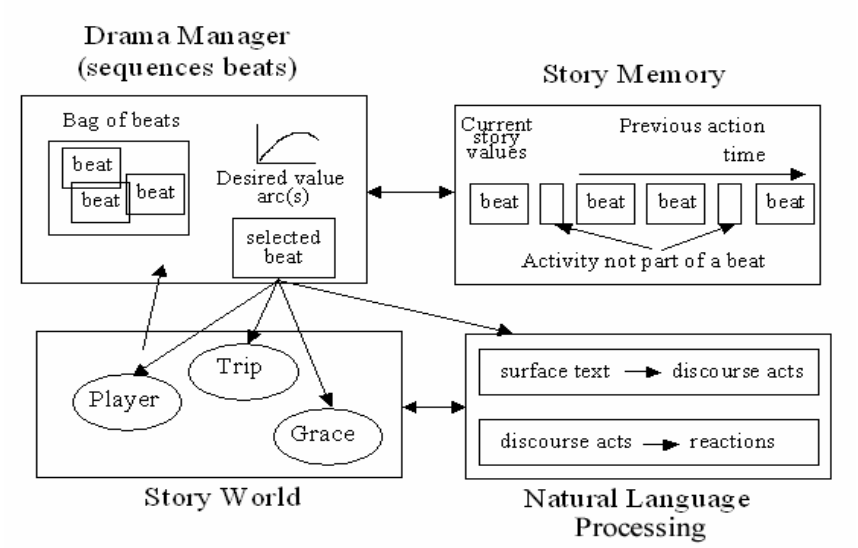

Fig 1. Façade interactive drama architecture

3.1 3D story world and first-person user interface

Façade's real-time rendered 3D story world is implemented in C++ with OpenGL. Character animation (body and face) is achieved through a mixture of procedural animation and layered keyframe animation data. We implemented a simple scripting language that can sequence together individual frames of keyframe body animation, and annotate pre-recorded lines of audio dialog with lip sync, timing and emphasis cues.

The story world consists of the animated bodies of Grace and Trip and their apartment, primarily a large furnished living room where the action of the drama is designed to take place. The interface is first-person in 3D space, navigated with arrow keys. The mouse controls a small hand-shaped cursor with which the player picks up and uses objects. The player can click Grace and Trip on their shoulders to comfort or hug them, or click on their lips to kiss them. To speak dialog, the player types text which appears on the lower part of the screen, like subtitles. Discourse is continuous and real-time, not turn-based; if the player enters text while Grace or Trip is speaking, it tends to interrupt them at the moment the enter key is pressed.

3.2 Believable agents

A key requirement for making lifelike believable agents is to endow them with the ability to do several intelligent activities in parallel – for example, to gaze, speak,

walk, use objects, gesture with their hands and convey facial expressions, all at the same time. The believable agents Trip and Grace are each composed of a large collection of parallel, sequential and joint behaviors written in A Behavior Language (ABL), a reactive planning language based on Hap [9]. The ABL compiler was implemented in Java and javacc, and compiles to Java. The ABL agents communicate to the story world across a Java - C++ API via dll's.

In ABL, an activity (e.g., walking to the player, or speaking a line of dialog) is represented as a goal, and each goal is supplied with one or more behaviors to accomplish its task. An active goal chooses one of its behaviors to try. A behavior is a series of steps, which can occur sequentially or in parallel. Typically, once a behavior completes all of its steps, it succeeds and goes away. However if any of its steps fail, then the behavior itself fails and the goal is forced to choose a different behavior to accomplish its task, if it has one. Furthermore, a behavior may have subgoaled its own set of goals and behaviors. To keep track of all the active goals and behaviors and subgoal relationships, ABL maintains an *active behavior tree* (ABT).

Believable agent behaviors are authored to cleanly inter-mix their actions to modulate the animation of their body and face in the animated story world, to sense information about the world, and to perform local, context-specific reactions to the player's actions. For more detail on ABL, including code examples, refer to [11].

The support code for the player avatar is also implemented in ABL. These behaviors do not take any action in the story world but instead sense the player's actions, extract longer-term meaning from them ("compound sensing"), and supply this information to the other agents in the system, e.g., Grace, Trip, the drama manager.

3.2 The beat and dramatic performance

At any given moment, the current story beat provides Grace and Trip with a context-specific collection of ABL behaviors, effectively determining the scope of the simulation during that beat. Only one story beat is active at a time. A beat's behaviors are tailored to focus the activity of the characters in a particular narrative direction, while keeping them broadly reactive to other narrative directions.

For example, the "Fix_Drinks" beat has a collection of behaviors whereby Trip and Grace verbally spar as they ask the player what she wants to drink, revealing hints about the conflict in their marriage. This collection of behaviors has a range of ways to perform this activity, and the flexibility to briefly diverge from this activity if the player tries to do something else, with an attempt to coax the player to return to the beat's intended activity. If the player persists on not choosing to participate in a beat's intended activity, the beat may abort, and the system moves on to a different beat, hopefully more aligned to the player's interactions.

To achieve this level of flexibility, a beat's collection of behaviors are organized into a set of *beat goals*, which potentially can occur in different orders, some of which are optional. In the authorial idioms we've developed for Façade, there are generally 5 types of beat goals in a beat:

- transition-in beat goals – the characters express their intentions for this beat
- body beat goals – pose a dramatic question or situation to the player

- local/global mix-in beat goals – react to the player before the beat completes
- wait-with-timeout beat goal – wait for the player's reaction to the situation
- transition-out beat goals – final reaction to the player's action (or inaction) to this situation

A beat goal is typically implemented as a series of steps in a sequential behavior. Each step is a subgoal to a coupled pair of *joint parallel behaviors,* one for Grace and one for Trip. It is here where the asynchronous autonomous agents Grace and Trip synchronize their behavior and coordinate their dramatic performances – for example, where one speaks and the other reacts with a look and gesture, or where they banter lines of dialog back and forth in quick succession. Each joint behavior in the pair subgoals one or more of the following *performance behaviors* in parallel:

- Staging (where to walk to, where to face)
- Dialog to speak (one or more pre-recorded phrases that form a sentence)
- Where and how to gaze
- Arm gestures to perform (e.g., raise arms to indicate enthusiasm)
- Facial expression to perform (a composite of parameters, e.g., angry smile)
- Head and face gestures to perform (e.g., eyes wander and head tilts down)
- Small arm and head emphasis motions, triggered by timing cues from the dialog (e.g., little head nods, hand flourishes)

Performance behaviors make use of goal priorities and lower-level body resource management behaviors to avoid conflicting actions and behavior thrashing.

3.3 Player interaction

Player interaction alters the performance of a beat (local agency), and can have longer term effects on future beats (global agency). The system attempts to interpret player action into one or more *discourse acts*. A discourse act is a concise representation of the general meaning of the player's action. Any dialog typed by the player, any discrete gesture made by the player, and some patterns of player movement through the environment are interpreted as one of the discourse acts in Table 1, most with the option to be directed towards Grace or Trip.

Each beat is designed to be able to respond to any of these discourse acts at any time, in a manner appropriate to its currently active *context(s)*. A context is implemented as a set of forward-chaining mapping rules defining how to react to player action. The current beat activates multiple individual contexts, typically a single unique custom context and one more global contexts, each at their own priority. Global contexts tend to be reused among beats, so an author usually only has to create one new unique context per beat.

The same discourse act in one total context (beat) may elicit a different response in another total context (beat). The sophistication of the response varies from discourse act to discourse act, from beat to beat. This variation in response to the player's actions from beat to beat, in conjunction with the varying behavior collections (i.e., tailored activities) from beat to beat, are the fundamental ways that the Façade simulation changes over time to achieve an interactive narrative.

- agree
- disagree
- positive exclaim
- negative exclaim
- express happy
- express laugh
- express sad
- express angry
- maybeUnsure
- dontUnderstand
- thank
- apologize
- referTo <*topic*>
- praise
- ally
- criticize light
- criticize harsh
- oppose
- flirt
- pacify
- provoke
- greet
- goodbye
- getAttention
- intimate
- judgment
- suggestion
- misc-custom
- manipulation
- jokeTestLimits
- inappropriate
- hug
- comfort
- kiss
- physicallyFavor
- wanderAway

Table 1. List of discourse acts, which encapsulates the player's range of expression

3.3.1 Broad and Shallow Natural Language Processing (NLP)

Here we briefly describe how the player's typed text ("surface text") and discrete gestures are mapped into discourse acts, and how the discourse acts gets mapped into reactions. For example, if the player types "Grace isn't telling the truth", the NLP system is responsible for determining that this is a form of criticism, and deciding what reaction Grace and Trip should have to Grace being criticized in the current context. General natural language processing is of course a notoriously difficult problem. Building a system that could understand open-ended natural language utterances would require common sense reasoning, the huge open-ended mass of sensory-motor competencies, knowledge and reasoning skills which human beings make use of in their everyday dealings with the world. While Façade is a micro-domain, a dramatically-heightened representation of the specific situation of a couple's marriage falling apart, not the whole world, there are still no general theories, techniques or systems that can handle the syntactic, semantic and pragmatic breadth of the language use which occurs in Façade. Instead, Façade makes use of specific (non-general), atheoretical, author-intensive techniques to understand natural language typed by the player.

Phase I of NLP involves the recognition of discourse acts from surface text, accomplished by rules written in a custom NLU Template Language, which compiles to Jess [6], a java implementation of the CLIPS rule language [13]. This custom rule language looks just like Jess with the addition of an embedded template description language that allows compact descriptions of surface text patterns to appear on the left hand sides of rules. Template rules map "islands" of patterns in the surface text into intermediate meanings, which are then chained together to produce the final discourse act(s), capturing the pragmatic meaning of the surface text. The template rules for Façade err on the side of being overly permissive, mapping a large number of inputs, including ungrammatical ones, to discourse acts.

Phase II of the NLP chooses a potential reaction to the discourse act for the current beat to mix in to its ongoing performance. A local, beat-specific set of custom rules (a context) are authored for each beat, written in a Reaction Decider language, also built on top of Jess. Additionally, a beat typically activates reusable, shared, global rule sets, called global contexts. Each context has rules to map some or all of the ~40 types of discourse acts to a proposed *reaction*, which if selected, the beat will mix in to its performance. At a minimum, local contexts typically can react to the player's mild or strong agreement, disagreement or non-direct answer to the question or situation posed in this beat. If the discourse act falls outside the domain of what the local context was listening for, the active global context(s) are always ready with a general reaction to propose, chosen from a pool of hundreds of reactions, including reactions to player's affinity moves (e.g., praise, criticism, flirts), references to objects and related topics (e.g., marriage, infidelity).

3.3.2 Interaction handlers

Once the NLP system has chosen a reaction to a player action, it is now up to the current beat to mix in a performance of the reaction. This is accomplished by the beat's *interaction handler* behaviors, which typically abort the current beat goal and insert a new high priority beat goal into the ABT, corresponding to the chosen reaction type. The newly inserted beat goal is prioritized to immediately begin executing, and later on the previously aborted beat goal will re-run if needed, using alternate repeat dialog. Early on in the beat, the NLP-decided reaction to player action is typically a mix-in beat goal – a reaction authored to respond in some way to the player's action, but not resolving the situation of the beat. For example, if the player refers to divorce during the "Fix_Drinks" beat, Grace and Trip mix in a short beat goal about their feelings about divorce – and then resume where they left off with their "Fix_Drinks" beat goals. Later in the beat the chosen reaction may be a transition-out beat goal, where the player's action has been interpreted to resolve the outcome of the beat.

Mix-ins and transition-outs, besides performing local reactions to player action, can be annotated with side effects on global story state. For example, a mix-in that reacts to "praise Grace" may also shift the player's affinity towards Grace. Or, a reaction to "referTo infidelity" may increase story tension. Altering story state will affect future beat selection, and may even cause the current beat to abort.

3.4 Drama management

So far we have focused on how the player experiences local agency in Façade, that is, how the player is able to see clear, immediate reactions to her interaction. The smarts to handle local interaction – the logic of beat goals plus interaction handlers – reside within beats. The smarts to incorporate interaction into a larger scale story structure, to incorporate not just the most recent interaction, but in some sense the entire history of interaction into the future direction of the story, and thus to provide global agency,

resides in the drama manager. In Façade this is the beat sequencer, which selects the next beat in the story based on the previous interaction history.

In the Beat Sequencing Language developed for Façade, the author can annotate each beat with selection knowledge, can define actions that are performed at various stages in the beat selection process, and can define beat variables that are accessible by all tests and actions within a beat. Selection knowledge consists of:

- precondition {<test>} – if the precondition test is true, the beat is a candidate for selection.
- weight <float> – a static weight modifying the probability of this beat being selected. If no weight is specified, the default weight is 1.0.
- weigh_test <float> {<test>} – if the test is true, the probability that the beat will be selected is multiplied by the weight; the associated weight overrides the static weight.
- priority <float> – beats are selected for sequencing by a weighted random draw from the beats with satisfied preconditions in the highest priority tier.
- priority_test <int> {<test>} – if the test is true, then the beat has the specified priority; the associated priority overrides the static priority.
- effects <story value changes> – the changes to make to story values if the beat completes successfully. The effects contribute to the probability of selecting the beat depending on how well they match the desired story arc.

Given a collection of beats represented in the beat language, the beat sequencer selects beats for sequencing. A sequencing decision is initiated when the current beat successfully terminates or aborts. Beat behaviors are responsible for monitoring their own success or failure criteria and informing the beat manager that the current beat has terminated. The steps for making a beat sequencing decision are as follows:

1. Initialize any beat-specific state which may play a role in beat selection.
2. Evaluate the preconditions for all the unused beats. This computes the set **Satisfied** of beats with satisfied preconditions.
3. Evaluate the priority tests of each beat in **Satisfied**. Collect the beats in **Satisfied** which are in the highest priority tier into the set **HighestPriority**.
4. Score each beat in **HighestPriority** using the effects to compare the beat with the desired tension arc, producing the set **ScoredHighestPriority**.
5. Multiply each beat's score by its weight, producing the set of weighted beats **WeightedScoredHighestPriority**.
6. Randomly select a beat from **WeightedScoredHighestPriority** according to the probability distribution defined by the weighted score.

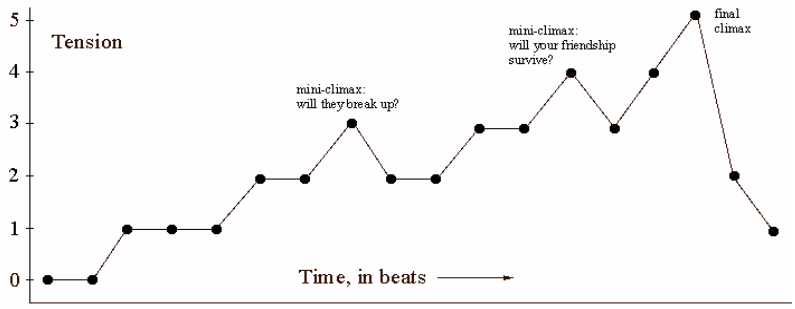

Fig 2. Aristotelian story tension value arc, used for scoring beats in beat sequencing.

The available beat whose story tension effects most closely match the near-term trajectory of the ideal story tension arc in Figure 2 will score the highest in step 4 of the sequencing decision. The scoring algorithm is described in Table 2 below.

Definition	*Expression*
Current tension value	T_{cur}
Number of beats remaining in the current linear segment of the arc (e.g., 3)	numBeats
The tension target of the current linear segment in the tension arc	T_{targ}
Desired adjusted slope for the current tension trajectory	$slope_{target} = (T_{targ} - T_{cur})/numBeats$
Delta tension value change for a candidate beat, from its effects knowledge	$deltaT_{beat}$
Candidate beat score	$1 / e^{\lvert slope_{target} - deltaT_{beat} \rvert}$

Table 2. Scoring algorithm for beats, used in step 4 of the beat sequencing decision.

One can imagine that the collection of beats might not allow the beat manager to do a good job tracking the desired value trajectory – at any point in time the candidate beat set just doesn't offer beats with good value increments. Regardless of how far off the value increments are, there will be some beat(s) in the candidate beat set that have the best (albeit small) score. As these low-scoring beats are chosen, the cumulative error[2] between the actual value arc and the desired value arc will steadily grow. Furthermore, beat management could fail by having no candidate beats available (beats with satisfied preconditions) given the current story state. When an overly large error term or an empty candidate set occurs, this means that the beat collection is not rich enough to approximate the story arc given the player's interaction. These failures can be used while developing the story to determine that beats must be changed or new beats created.

[2] The beat manager keeps track of $error_n = sqrt(\sum_{i=1 \text{ to } n} (A_i - A_{iopt})^2)$ (the standard square root of sum of squares error), though any error term could be used.

4 Conclusion

The Façade architecture offers the author a framework for organizing the content of a dramatic story into a hierarchy of pieces: beats, each containing beat goals, each containing behaviors that the system dynamically organizes into an active behavior tree. Each piece in the hierarchy is annotated with conditions on global story state and/or local simulation state, and each piece can potentially cause effects on such state. It is the author's job to define the conditions, priorities, scoring weights, content meaning, and effects of each piece. As the player contributes her own continuous effects on the simulation and story state through her actions, the resulting sequencing of the content – the construction of the narrative – occurs.

In Façade, it is to the extent of the evaluation of this author-annotated knowledge about behaviors and beat goals and beats that the system is "reasoning" over the story content and "generating" a story. The system's role is that of an editor, assembling a story from an array of story fragments, where each fragment was created by a human author, who presumably possessed some original intent on how the fragments could ultimately fit together. (Generally speaking, once the grain size gets very small and the number of pieces gets very large, the line between assembling and generating content becomes blurry.) In Façade, the degree of player agency – how "interactive" the story is – is proportional to the number of possible coherent orderings, which is proportional to the number of pieces of story content. The potential "goodness" of the story is still very much in the author's hands, encoded in the quality of the content meaning of each piece on its own, and the narrative quality of the potential ordering of the pieces, encoded in the preconditions and effects.

References

1. Bates, J.: Virtual Reality, Art, and Entertainment. Presence: The Journal of Teleoperators and Virtual Environments 1(1) (1992) 133-138.
2. Bates, J., Loyall, A. B., and Reilly, W. S.: Integrating Reactivity, Goals, and Emotion in a Broad Agent. Proceedings of the Fourteenth Annual Conference of the Cognitive Science Society, Bloomington, Indiana (1992)
3. Bernstein, M.: Patterns of Hypertext. In Shipman, F., Mylonas, E. and Groenback, K. (eds.), Proceedings of Hypertext '98. New York: ACM (1998)
4. Blumberg, B. and Galyean, T.: Multi-level Direction of Autonomous Creatures for Real-Time Virtual Environments. In Proceedings of SIGGRAPH 95 (1995)
5. Blumberg, B.: Old Tricks, New Dogs: Ethology and Interactive Creatures. Ph.D. Dissertation. MIT Media Lab (1996)
6. Friedman-Hill, E.:. Jess, the Rule Engine for Java. Sandia National Labs. http://herzberg.ca.sandia.gov/jess/ (1995-2002)
7. Hayes-Roth, B., van Gent, R. and Huber, D.: Acting in character. In R. Trappl and P. Petta (Eds.), Creating Personalities for Synthetic Actors. Berlin, New York: Springer (1997)
8. Lester, J., Stone, B.: Increasing Believability in Animated Pedagogical Agents. Proceedings of the First Int'l. Conference on Autonomous Agents. AAAI Press (1997) 16-21
9. Loyall, A. B. and Bates, J.: Hap: A Reactive, Adaptive Architecture for Agents. Tech Report CMU-CS-91-147. Carnegie Mellon University (1991)

10. Mateas, M. and Stern, A.: Towards Integrating Plot and Character for Interactive Drama. In Working notes of the Social Intelligent Agents: The Human in the Loop Symposium. AAAI Fall Symposium Series. AAAI Press (2000)
11. Mateas, M. and Stern, A.: A Behavior Language for Story-Based Believable Agents. In Working notes of the Artificial Intelligence and Interactive Entertainment Symposium. AAAI Spring Symposium Series. AAAI Press (2002)
12. Mateas, M. and Stern, A.: Architecture, Authorial Idioms and Early Observations of the Interactive Drama Façade. Tech report CMU-CS-02-198, Carnegie Mellon Univ. (2002)
13. NASA.: C Language Integrated Production Systems (CLIPS). Originally developed at NASA's Johnson Space Center. http://www.ghg.net/clips/WhatIsCLIPS.html (1985-2002)
14. Ryan, M.: Narrative as Virtual Reality. Baltimore: Johns Hopkins University Press (2001)
15. Stern, A., Frank, A. and Resner, B.: Virtual Petz: A hybrid approach to creating autonomous, lifelike Dogz and Catz. Proceedings of the Second Int'l. Conference on Autonomous Agents. AAAI Press (1998) 334-335
16. Weyhrauch, P.: Guiding Interactive Drama. Ph.D. Dissertation, Tech report CMU-CS-97-109, Carnegie Mellon University (1997)