Façade: An Experiment in Building a Fully-Realized Interactive Drama

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

[1] Literature, Communication and Culture and College of Computing, Georgia Tech, michael.mateas@lcc.gatech.edu
[2] InteractiveStory.net, andrew@interactivestory.net, www.interactivestory.net

Introduction

Contemporary games are making significant strides towards offering complex, immersive experiences for players. We can now explore sprawling 3D virtual environments populated by beautifully rendered characters and objects with autonomous behavior, engage in highly visceral action-oriented experiences offering a variety of missions with multiple solutions, and interact in ever-expanding online worlds teeming with physically customizable player avatars.

At the same time, we are all acutely aware of the limitations of contemporary game designs and technologies. Game designers do a remarkable job innovating within these constraints, and judging by the industry's healthy revenues, gamers are willing to conform their expectations as well. But these constraints are significant nonetheless, restricting the possibilities of interactive entertainment. Most notably, games are unable to convincingly address many of the topics and themes of human relationships, thereby limiting both their mass market appeal and potential cultural value.

Instead, today's games remain almost exclusively oriented around physical action. Players are limited to operating weapons, running, jumping, fighting, controlling vehicles and so on; a few games exceed this by allowing players to construct new objects and environments and to modify the appearance of avatars. Although their visual fidelity continues to increase, computer-controlled non-player characters (NPCs) remain stubbornly simplistic, with a narrow repertoire of behaviors akin to insect-level intelligence. Conversations with NPCs do exist in some games, but in contrived, restricted forms; players are typically offered a short menu of canned phrases they can "speak", narrowly focused on progressing the game forward along pre-determined paths. These "conversations" are generally on the same order of simplicity as the player's limited set of physical actions.

In effect, game designs are restricted in their subject matter because players cannot yet speak in *natural language* to the game. Language would offer players a far greater range of expression than physical action alone; language is of course the primary way we communicate with one another in real life. Language allows us to talk about ideas and express

attitudes and emotions – key requirements for interactive experiences focused on human relationships, currently the domain of drama and literature.

Networked games and persistent online worlds allow human players to communicate in natural language with *each other*, but the game system itself does not understand the communication between players. It is unclear how easy or common it will be for average players to achieve meaningful interactive experiences about human relationships beyond what emerge from their own open-ended naïve interactions with each other. In order to help support more meaningful, dramatic experiences, the system needs mechanisms for recognizing the social effect of player utterances, e.g. agreement, disagreement, alliance, criticism, flirtation, etc., and deciding how to have the game react to these utterances.

Similarly, games will have difficulty addressing themes of human relationships until computer characters themselves are able to be more expressive. NPCs need a richer repertoire of behaviors than they currently possess, able to emote and express their personalities in a variety of ways, and engage in conversation about a variety of topics. This goes hand-in-hand with players speaking natural language; once players are allowed to be more expressive, the game needs to express more as well. Authoring NPCs that continuously and richly convey their personality and emotions while remaining fully responsive to player interaction requires moving beyond finite state machines and scripting languages. What is needed is a character authoring language that combines tight authorial control with the real-time capability to dynamically combine multiple behaviors and pursue multiple goals.

Game designs are moving towards rich, complex worlds offering open-ended, sandbox-style play. At the same time, we know that narrative structures (stories) have historically been an extremely successful way of representing human relationships. In order to create interactive experiences about human relationships, it makes sense to create a system that tries to shape open-ended play into narrative structures. That is, metaphorically speaking, to offer open-ended sandbox-style play in which the system knows how to make sandcastles, and tries to collaborate with the player to do so. Of course, interactive story has been a holy grail of game design since the advent of computer games; we are all well aware the fundamental tension between player freedom and story structure. What is needed is a drama manager [Bates 1992, Weyrauch 1997], that is, an artificial intelligence system that uses knowledge about how stories are structured to construct new story-like experiences in response to the player's moment-by-moment, real-time interaction.

In this paper we discuss our research and development towards creating an architecture, and a story design using this architecture, that integrates a broad and shallow approach to natural language processing, a novel character authoring language and a novel drama manager, in order to build an *interactive drama* about human relationships.

Façade: an experiment in interactive drama

In Façade, you, the player, using your own name and gender, play the character of a long-time 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 replay the drama to find out how your interaction could make things turn out differently the next time.

Figure 1. Real-time rendered characters Grace and Trip in Façade, with player-typed text and hand cursor.

Façade is an attempt to create a real-time 3D animated experience akin to being on stage with two live actors who are motivated to make a dramatic situation happen. Instead of providing the player with 40 to 60 hours of episodic action and exploration in a huge world, we want to design an experience that provides the player with 20 minutes of emotionally intense, unified, dramatic action. The player's actions should have a significant influence on what events occur, which are left out, and how the drama ends. The experience should be varied enough that it supports replayability; only after playing it 6 or 7 times should the player begin to feel they have "exhausted" its possibilities. In fact, full appreciation of the experience requires the drama be played multiple times. Change in the plot should not be traceable to distinct branch points; the player will not be offered an occasional small number of obvious choices that force the plot in a different direction. Rather, the plot should be smoothly mutable, varying in response to some global state that is itself a function of the many small actions performed by the player throughout the experience.

In Façade, the player is given almost no direction or role to play; she simply tells the system her name and gender, allowing her to "play herself". The drama takes place in a small simulated virtual world, the apartment of the married couple Grace and Trip, in which the player, Grace and Trip can continuously move anywhere, use objects, speak dialog and gesture at any time. From the get-go, Grace and Trip attempt to engage the player in psychological head games, for example, posing situations to the player in an attempt to force her to choose sides in an argument. However, the player is not limited to playing these psychological games; she can say anything she wants at any time. The characters are designed to respond robustly to a variety of open-ended dialog from the player, including

questions and provocations. Grace and Trip attempt to focus the interaction on their psychological games but are able to engage in a certain amount of digression.

In fact, players are rewarded for being proactive and acting dramatically. Through dialog, gesture and action, the player has continuous effects on her affinity with Grace and Trip, the current level of tension, and the specific information revealed about their marriage. These changes in simulation state in turn modulate the behavior of Grace and Trip, carving a path through the simulation space, ultimately causing irreversible change in the characters, thereby enacting a drama. Just like other software simulations, the scope of possibilities in Façade is finite, but is unusual in its use of language and gesture to engage in interactions about human relationships, rather than physical action to engage in battles, missions, quests or environment construction.

Before diving into describing the architecture, we'd like to cut to the chase and briefly reveal what we believe Façade does and does not accomplish. In a nutshell, the architecture offers a new framework for authors to create structured hierarchies of behaviors, which when performed and modulated by the player's interaction (natural language, gesture and physical action), allows a theatrically dramatic experience to occur. However, having an architecture for structuring and performing an interactive drama is only half the battle – the behaviors themselves still have to be written, in Façade's case, by a human author. Within this architecture we are authoring a one-act interactive drama, to be publicly released as a free download in 2003. The Façade system is generative in the sense that it mixes and sequences behaviors in sophisticated ways, as this paper will describe, but it does not generate the individual behaviors. Hand-authoring behaviors is a time consuming process – by the time Façade is done, we will have spent two man-years on authoring alone, but even this results in only a 20 minute one-act play replayable 6 or 7 times before it is exhausted. Furthermore, Façade of course does not achieve general purpose natural language understanding; instead it listens for a large variety of word patterns and phrases focused on the context of its dramatic situation, which feed into a discourse management system, described in the "player interaction" section of this paper. More discussion of the success and failures of the system occurs in "early observations" section.

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., NPCs. 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 experience. 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 an 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 proce-

dures (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. Taken from the theory of dramatic writing [McKee 1997], a beat is the smallest unit of dramatic action that moves a story forward. 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 [Bernstein 1998, Ryan 2001] for examples of such diagrams); perhaps the only way to visualize them is to enumerate the thousands of possible orderings. 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.

Similarities and differences to existing game architectures

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. Further, beats change the overall behavior of the simulation to a greater degree than game levels tend to. 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. In contrast, beats each have 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. Finally, 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 [Mateas and Stern 2000] 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.

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. For more detail than this brief tour of the architecture provides, please refer to [Mateas and Stern 2002b, Mateas 2002].

The entire system was implemented from scratch on the Windows XP platform. The following authoring languages were developed:

- A Behavior Language (ABL), a reactive planning language based on Hap [Loyall and Bates 1991] that supports sequential and parallel behaviors, including joint behaviors. The compiler was implemented in Java and javacc, and compiles to Java.
- Natural Language Understanding (NLU) Template language, a forward-chaining template rule language for specifying templates that map player-typed surface text into discourse acts. The compiler was implemented in Java and javacc, and compiles to Jess [Friedman-Hill 1995-2002], a java implementation of the CLIPS rule language [NASA 1985-2002].
- Reaction Decider language, a forward-chaining rule language for proposing and selecting reactions to discourse acts, runs on top of Jess with Java support routines.
- Beat Sequencing language, a language that specializes in drama management. The compiler was implemented in Java and javacc, and compiles to Java.

Façade has several threads that run in parallel:

- the non-photorealistic real-time rendered 3D story world, including the environment, objects and characters' bodies (written in C++ with OpenGL)
- the believable agent Trip (written in ABL)
- the believable agent Grace (written in ABL)
- the Player's avatar agent (written in ABL – recognizes player actions)
- natural language processing (NLP) (runs template and reaction decider rules)
- the drama manager (sequences beats described in the beat sequencing language)

[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.

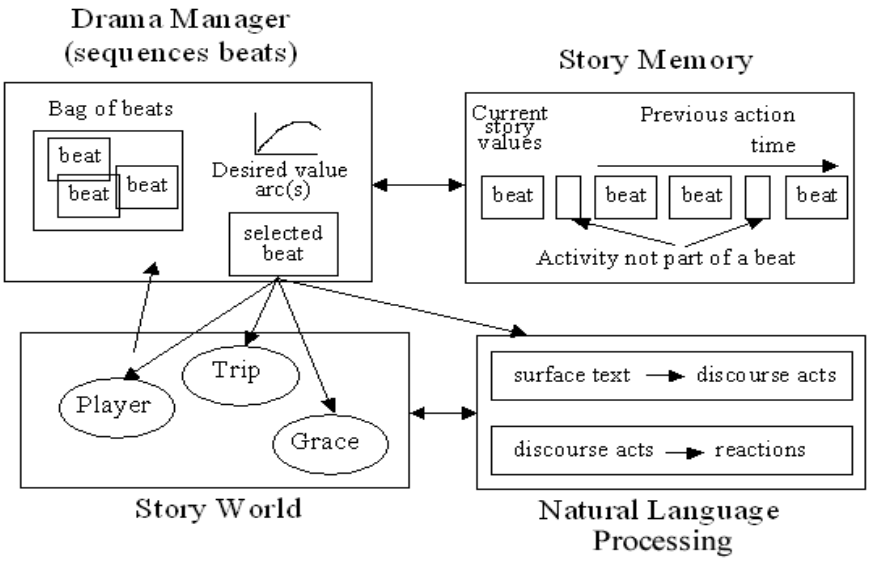

Figure 2. Façade interactive drama architecture

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.

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).

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 attempts to find a different behavior to accomplish its task, failing if no such alternative behavior can be found.. 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).

Behaviors are authored to cleanly inter-mix their actions in order to modulate the animation of the character's 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 [Mateas and Stern 2002a, Mateas 2002].

The paradigm of combining sequential and parallel behaviors, and propagating their success and failure through the ABT, are the foundation of the power of ABL as a language for authoring believable agents, versus purely sequential languages such as C++ or Java where parallelism has to be managed manually. ABL is effectively a multi-threaded programming language, making it very easy to author behavior mixing – a powerful feature, but one that can get out of control quickly. In this way, ABL is challenging to program in, even for experienced coders. An important feature of ABL (the primary feature that distinguishes it from Hap) is support for synchronized joint behaviors, which helps the author harness the power of multi-threaded programming.

Features of the ABL language:

- WMEs – working memory elements, which are data structures that can hold information (ie, variables), equipped for match tests in conditionals such as preconditions
- preconditions – an optional test attached to a behavior to determine if the behavior is available to accomplish a goal in the current context
- context conditions – an optional test that will fail a behavior if the test becomes false
- success tests – an optional test that will succeed a goal if the test becomes true
- acts – a step in a behavior that takes some sort of action in the world, typically an animated body action such as taking a walk step, gazing at an object, speaking dialog
- mental acts – a step that does some arbitrary internal computation, in raw Java code
- subgoal and spawngoal – activate a new goal as a child of the current behavior, or as a child of a some other specified behavior
- joint with teammembers – when subgoaling, synchronously start the same behavior in multiple believable agents, if possible
- priority – determines which parallel behaviors run before others
- persistence – a way to retry a goal if it succeeds or fails
- ignore failure – if a goal fails, do not fail its parent
- effect only – do not require this goal to succeed for its parent to succeed
- atomic behaviors – during this behavior, do not allow any other behaviors to run (shut off parallelism)
- specificity – specify the order in which to attempt different behaviors for the same goal
- number needed for success – how many children of a parallel behavior are required to succeed for the parent to succeed
- sensors – a way to have WMEs get automatically updated information about the 3D story world
- conflicts – a way to prevent certain parallel behaviors from running at the same time, if needed
- demons – a technique of creating a behavior that waits in parallel to all other behaviors for a condition to become true, and then takes action
- meta-abl – a way to reflect upon and affect the currently active behaviors themselves (kept track of in the ABT), to allow a behavior to directly alter, succeed, or fail other behaviors

API between ABL and the 3D story world

Grace and Trip's ABL behaviors sense the 3D story world (e.g., observe where the player is standing) and take action in the world (e.g., animate their bodies) by sending query and action messages to the world. Additionally, they can receive spontaneous event notifications from the world (e.g., the player just said something). The following is a summarized description of the API between the ABL agents and the 3D story world:

- GetObjectPosition
- SetObjectPosition
- GetObjectRotation
- SetObjectRotation
- GetObjectPickupPosition
- GetObjectPickupRotation
- GetObjectStagingPosition
- GetObjectState
- GetGazeTracking
- SetGazeTracking
- SetPerformanceInfo
- DoWalkStep
- AbortWalkStep
- DoDialogAnimation
- Abort DialogAnimation
- DoGestureAnimation
- AbortGestureAnimation
- DoFullExpressionBaseAnimation
- DoFullExpressionMoodAnimation
- DoMiscLittleAction
- GetAllHeldObjects
- SetObjectToHold
- PlaySoundEffect
- PlayMusic
- SetMiscWorldInfo
- EventNotificationAnimationCue
- EventNotificationPlayerGestured
- EventNotificationPlayerTypedDialog
- EventNotificationObjectActivation

In summary, ABL behaviors send simple parameterized action requests to the 3D story world such as "take an angry walk step towards the couch", "look at this object in a coy way", "speak this line of dialog", "do an anxious but smiling facial expression", "do an emphasis hand gesture and nod when I speak", "make my eyes quiver", and so on. ABL behaviors sense the world with queries such as "what is the location of the wedding picture", or "what am I holding in my hand", and receive automatic event notifications such as "you just finished speaking a certain word in your dialog", or "the player just spoke these words", or "the player just picked up a martini glass". The 3D story world is responsible for accomplishing basic performance tasks such as low-level motor control of the body, procedural animation of facial expressions and gaze, lip-sync to dialog, and pathplanning.

Player agent

The Player ABL agent does not take any action in the story world (the human user fully controls the Player character), but instead senses the Player's actions, extracts longer-term meaning from them ("compound sensing"), and supplies this information to the other agents in the system, e.g., Grace, Trip, the drama manager. This includes determining when the Player is making significant movements around the room, and when she has been looking at an object for a significant amount of time.

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) within the beat 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.

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.

- 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

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.

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 that 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 occurring in Façade. Instead, Façade makes use of specific (non-

general), a-theoretical, 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. 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).

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; after it completes, 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 beat's dramatic situation. For example, if the player refers to divorce during the "Fix_Drinks" beat, Grace and Trip mix in a short beat goal discussing their feelings about divorce, 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.

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.

Structuring the story as a bag of beats

The nature of Façade's drama – the story of a married couple inviting a friend over for drinks, ostensibly discussing how great their lives are while dodging and weaving around the fact their marriage is falling apart, as the tension builds to a breaking point a la *Who's Afraid of Virginia Woolf* – was chosen because it can be successfully broken apart into story beats that can be coherently resequenced in many different orders. Façade is purposely designed as a somewhat open-ended psychological situation with an array of topics to be discussed, secrets to be revealed and head games to be played, in which it is acceptable for only a subset of topics, secrets and head games to occur in any one run-through of the drama. Contrast this to a tightly-plotted drama such as *Casablanca*, where the order of events is carefully crafted and really cannot be altered without ruining the integrity of the narrative. It is safe to say that certain types of stories, such as character-oriented kitchen sink dramas, lend themselves better to interactivity than plot-oriented action dramas.

Beat Sequencing Language

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> – a static priority; 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 that may play a role in beat selection.
2. Evaluate the preconditions for all the unused beats. This computes the set **Satisfied**, consisting of beats with satisfied preconditions.
3. Evaluate the priority tests of each beat in **Satisfied**. Collect the beats in **Satisfied** that 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 story arc (for Façade, a 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.

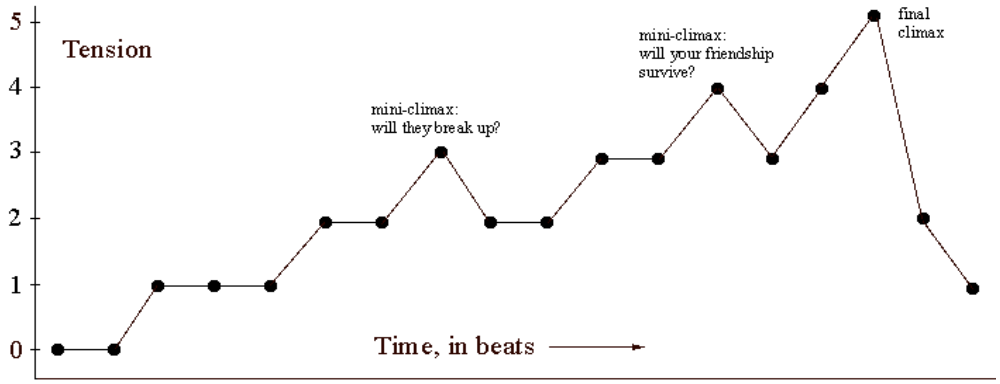

Figure 3. Aristotelian story tension value arc, used for scoring beats in beat sequencing.

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.

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.

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	$1 / e^{\| slope_{target} - deltaT_{beat} \|}$

Table 1. Scoring algorithm for beats, used in step 4 of the beat sequencing decision.

Early Observations

How well does this architecture work? That is, to what extent does it offer the player the experience of an interactive story? Although the authoring of Façade is still underway, this section includes a few early observations of what we already understand about the successes and failures of the system.

The success of an interactive story involves the combination of many factors, some of which of course are subjective and can only be measured empirically, if at all. However, we can try to quantify a few characteristics of Façade that we believe have a significant influence on the quality of the experience as a whole.

Player agency

We believe it is valuable to try to measure the degree of agency the player has in an interactive story, both local agency and global agency. Three ways to measure local agency include:

- on average, how many distinctly different expressions / actions can the player input to the system at any moment;
- on average, how many distinctly different immediate responses can the system express back to the player, in reaction to her action;
- on average, how often does the context of the narrative change, such that repeating the same action gives a different immediate response.

For all three of these, we believe Façade performs well relative to the state-of-the-art for interactive narratives (e.g., critically-acclaimed examples of hypertext fiction, interactive fiction, adventure games and immersive simulation games.) In Façade, the player is al-

[3] 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.

ways free to express any of the ~40 types of discourse acts, some of which have a range of parameters, e.g., topics that can be referred to. These discourse acts are mapped from millions of combinations of raw natural language surface text. The NLP system maps the player's input to one of anywhere from 3 to 10 different context-specific local responses (local beat goals), and/or one of least 30 different generic global responses (from a total pool of ~500 global mix-in beat goals). The context of the narrative changes frequently in Façade, with small updates on every player interaction, and larger updates once or twice a beat, about every 30 to 60 seconds. Furthermore, the system never repeats a reaction (except for generic deflection reactions) – as the player interacts, she works her way through a broad database of content, only seeing about 25% of the total content in any one session.

One way to measure global agency is to count the number of possible orderings of meaningful story events, assuming the player has strong influence on which ordering occurs[3]. Façade has a two-tiered hierarchy of chunks of meaningful content: beat goals (sentence-level chunks) and beats (plot-event-level chunks). Here we are concerned with the number of possible orderings of beat goals within any beat, and the number of possible orderings of beats over the entire story. A typical beat chooses from 10 local and 30+ global beat goals, and is about 6 beat goals in duration. Not all permutations are possible of course, because of the partial ordering imposed by beat goal preconditions and the particular selection logic of the NLP rules. Nevertheless, we can safely say that a typical beat has hundreds, if not thousands, of coherent possible orderings of its beat goals (phrases and sentences). Moving up the hierarchy from beat goals to beats, the beat sequencer will have ~200 beats at its disposal, with a full story about 18 beats in duration. Again, preconditions and effects specify a partial ordering; we estimate there will be thousands of distinctly different possible beat orderings (plot-events) in Façade. Just as with local content, the player can only see a minority of the total number of beats in any one session, hopefully motivating her to replay the story to see more, inevitably with a different ordering.

Division of responsibility between the author, the system and the player

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. Pieces can also be annotated with priorities, context-conditions, success-tests, and in the case of beats, scoring weights. It is the author's job to define the conditions, tests, 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 pre-

[3] This analysis does not address the more subjective question of how meaningfully different the various event orderings are from one another. The perceived differences between event sequences is strongly influenced by the particular meaning of each event.

sumably 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.

We should not forget the contribution of the player's expressions themselves to the quality of the story as a whole. By virtue of the open-ended natural language input in the interface, the player can type any dialog they want at any time; this dialog is performed (displayed) on-screen. In this way, the player is doing more than influencing the plot; the player is also interacting to produce dialog to be included in the text of the story overall. In a stageplay trace of an experience of playing Façade, the player's language would be equally intermixed with the system's language. Furthermore, the system always tries to maintain the player's suspension of disbelief, to create the experience of a "real" dramatic play; the characters never utter "illegal command", "system does not understand", or "does not compute", even when the system does not actually understand. Assuming that the player isn't purposely being absurd, a trace of the experience should be coherent, containing at worst the occasional non-sequitur.

Failures of the system

Natural language understanding. Given the kind of story we want to tell, an adult domestic drama, the use of language in the story is unavoidable. But just as the open-ended natural language input offers the player a true-to-form way of participating in a drama, this interface will inevitably be the least effective and most aesthetically failure-prone part of the system. We expect three types of failures in Façade regarding natural language input: non-understood utterances, false positives, and an asymmetrical range of expression.

As mentioned earlier, Façade makes use of specific (non-general), a-theoretical, author-intensive techniques to understand natural language typed by the player. But even with thousands of hand-coded template rules, and the limiting of the number of words in any one utterance, players will be able to (accidentally or not) input text that the system cannot understand. When this occurs, the best the system can do is fail gracefully. Non-understood utterances trigger behaviors that employ an array of deflection-type responses, such as "hold on, hold on" or "well, what I was trying to say was...", or by abruptly changing the subject, as if they have something urgent they need to say and can believably ignore the utterance. As mentioned earlier, the system never outright rejects an utterance with an error message, which would unnecessarily break the player's sense of immersion in the drama and suspension of disbelief that Grace and Trip are intelligent adults.

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. This is based on the design approach that it is more interesting for the characters to eke some meaning out of a broad set of utterances, and thus have some interesting response for this broad set, than to only have interesting responses for a narrow set. While players will sometimes try to

break the NLP by seeing what kinds of ludicrous sentences they can get the characters to respond to, the templates are designed not to robustly support this meta-activity, but rather to extract meaning from a broad collection of "natural" utterances likely to arise during the course of playing Façade. This permissiveness will inevitably result in false positives, where the system thinks the player meant something that she did not. This could potentially be very confusing to the player, forcing her to try to understand why the characters reacted the way they did.

Finally, although Façade achieves some symmetry in the interface beyond what it is typically offered in interactive narratives – that is, both the characters and the player speak in natural language – the player may notice an asymmetry between the range of expressions she can perform and the range of responses the system can perform in return. With open-ended natural language input, at any moment the player can express millions of different meanings valid within the domain of the story. However the system has, at best, only hundreds of different meanings it can express back at any moment. If noticeable, this incongruity will further impair the player's suspension of disbelief.

Authorial labor. While the architecture affords the author an expressive framework for creating a character-oriented interactive drama, in practice it is time consuming to create all of the behavior to achieve what we consider a satisfying level of lifelikeness, responsiveness and player agency. A future research direction to pursue is one where the system can help generate beats, beat goals and behaviors themselves, helping relieve the authorial burden of creating enough content that achieves a satisfying level of agency for the player. If the system is to do more sophisticated reasoning and therefore generation, this would probably require, for example, annotations on story content more descriptive than preconditions and effects.

Authorial complexity. As described earlier, the languages developed for this architecture, particularly ABL with its direct support for parallel behaviors and behavior mixing, offer the author the expressive power to achieve lifelike behavior in fewer lines of code than traditional sequential languages such as C++ and Java. However this power can be difficult to harness; one can quickly write programs that are difficult to debug. The learning curve for taking full advantage of the authorial power of an architecture like Façade may be steeper than we would like.

Sufficient degree of global agency. While the Façade architecture can support a non-trivial degree of story agency for the player, achieving this requires creating a critical mass of story content for the system to work with. As previously described, Façade's idioms for authoring real-time interactive lifelike dramatic performance put much of the burden of content creation (e.g., behavior coding) on the human author, versus having the system itself share more of the load by being more generative. Even with our plan to spend upwards of two man-years on authoring on Façade, it is unclear whether we will achieve the critical mass of story content that allows for the plot to feel as mutable as we originally hoped for.

Player as protagonist. The player in Façade is more than an interactive observer, but is not the story's protagonist. The story of Façade is primarily about the dissolution of Grace and Trip's marriage; the player plays a central role in the outcome of this dissolution, but is not the "main character". At most the player shares an equal role with Grace and Trip (which, as authors, we are happy with, and we hope players will be satisfied with as well).

Originally we had intended to author additional story content that would have put the player more front-and-center in the drama, but due to time constraints have had to cut that aspect of the potential story.

Scalability. This architecture is designed to support intimate interactive experiences. That is, it supports relatively rich and varied expression for a single player among a small number of complex characters in a small environment, versus relatively shallow expression between multiple players and large number of simple characters in a large environment. It is not clear how well this architecture would scale up to the epic scope of a typical contemporary game, if desired.

Related Work

This section provides brief descriptions of systems that explicitly employ AI techniques to maintain/generate a narrative arc in response to interaction, followed by descriptions of systems that apply AI techniques to generate non-interactive stories.

Interactive Story

Laurel [Laurel 1986] provides a seminal description of a dramatic virtual world in which an AI system is used to structure interaction as a dramatic arc. While she did not implement a system, she did describe the dramatic requirements for interactive drama and explored what an architecture for such a system might look like.

Sharp [Sharp 1989] created King of Chicago, an ambitious interactive fiction game in which the player chooses between courses of action for Pinky, a young gangster trying to take over Al Capone-era Chicago, by clicking on alternatives presented in thought balloons and playing arcade action sequences. Story episodes, akin to Façade's beats, are annotated with precondition values that get matched by an episode selector to current story values, such as characters' happiness, reputation and morale level. The selector does a least square fit matching. Within an episode, interaction is controlled by if-then-style branching. Episodes are grouped into story sequences, defining the current set of selectable episodes. The system contained 270 total episodes, representing up to 8 hours of gameplay.

Script-and-demon story systems combine the use of a script to specify linear or branching sequences of events with demons that are associated with events. The demons won't let an event happen until certain preconditions on the state of the world have been satisfied. Plot graphs [Kelso, Weyhrauch & Bates 1993] are one example of such a system. A plot graph lays out scenes in a directed acyclic graph (DAG). The arcs represent the must-precede relationship. Only after all preceding plot points have happened can the next plot point be entered. Associated with the arcs are hints and obstacles; these are ways that the drama manager can influence the world. Hints make it more likely that the user will move into the next scene; obstacles slow the user down. Demons recognize when a user has completed a scene. Another example, Pinhanez's Interval Scripts [Pinhanez 1997], represents the script by using a temporal calculus to represent temporal relationships among intervals. Some of these intervals are connected to sensors (demons) that wait for events to occur in the world; others are connected to actuators that make events happen in the world. A constraint propagation mechanism is used to determine the state of each interval

(now, past, future, or some mixed state). When a sensor has the value now, it begins looking for its associated event to happen in the world. When an actuator has the value now, it makes its associated event happen in the world. In Galyean's Dogmatix [Galyean 1995], an analogy is made between the action selection problem in behavioral agents and the event selection problem in plot control. At each point in time, a behavioral agent must select one (or in general, some small subset) behavior from its pool of possible behaviors. This selection is accomplished as a function of the internal state of the agent and the external state of the world. Analogously, at each point in time a plot selection mechanism must select an event to make happen out of the set of all events it could make happen. In Galyean's system, this selection is a function of story state variables (history), sensors (demons watching for events in the world), and temporal relationships. The temporal relations hierarchy, before, xor, and must-happen place a partial order on the possible sequences of events chosen by the selection mechanism. At each point in time, the event that has the highest "fitness" is chosen for execution.

Weyhrauch [Weyhrauch 1997] developed a game-tree search technique for drama management in an interactive world. His system controls a story at the level of plot points, or major scenes in a story. The system has available to it a collection of system "moves" that, by manipulating characters and the physical state of the world, modify the probability of certain plot points appearing in the future. Whenever the system detects that a plot point transition occurs (the player has done something in the story world to make a plot point happen), it projects all future combinations of system moves and plot point occurrences (to the end of the story). Each total projected story (past plot point sequence plus future projections) is evaluated by an author-written function that computes the "goodness" of the total story. The system then makes the system move that maximizes this story "goodness". Weyhrauch's system is intended for the sequencing of plot points, relatively large granularity story units that often correspond to scenes. Architectural components called recognizers (not implemented in Weyhrauch's dissertation) are responsible for guiding activity within the plot point.

The Erasmatron [Crawford 1999] is an authoring system designed to allow writers with a non-technical background to write interactive stories. Characters are represented through collections of verbs (actions) that the character can execute. Action choices are made based on the current state of the world and the state of internal traits within an elaborate character trait system. The player selects actions from changing menus of available actions. There is no explicit representation or control over the story arc; the story arc is implicitly determined by the character actions and the actions made available to the player. Like most adventure games (whether text or graphical), the Erasmatron uses a discrete time model, in which the player is offered distinct interaction opportunities and in which both player and character actions execute instantaneously.

The DEFACTO interactive story environment makes use of a rule-based system for story guidance [Sgouros 1999]. Generation rules encode knowledge about role related norms, social actions, and personal goals and relations. Based on the role and goal specifications of the cast of characters (including the player), the generation step generates possible character actions. For the player character, possible actions are presented in a menu. The evaluation step makes use of dramatic significance rules to select specific actions for the non-player characters from among the set of possible actions computed by the generate step. The cycle of generation and evaluation continues until no dramatically interesting

situations are found. At this point the resolution rules compute the final outcome of the character interactions to complete the story.

The Mimesis architecture [Young 2001] constructs story plans for real-time virtual worlds, currently the Unreal Tournament world. The story plans generated by the planner are annotated with a rich causal structure. The system monitors for player actions that might threaten causal links in the current story plan. If a threat is detected, the system either generates a new plan that accommodates the player action while still accomplishing the story objectives, or intervenes by causing the player action to fail and thus protect the threatened causal link.

The Mission Rehearsal Exercise Project [Swartout et. al. 2001] is building a widescreen, surround-sound, immersive training environment for military peacekeeping operations. The narrative training scenarios make use of a combination of scripted and autonomous characters. Story structure is managed by the story net, a structure similar to the plot graphs described above. Nodes define contexts within which the player is free to act. To progress the story, conditions associated with the arcs of the story net must be satisfied in order to traverse the story to another node. Node traversals (movement along an arc) are communicated through non-interactive cut scenes.

Story Generation

AI work in story generation has focused on building architectures that non-interactively generate stories. The stories are most often expressed as narrative prose or in some high-level natural-language-like formalism. Each story generation architecture makes a different set of commitments regarding the fundamental units out of which stories are composed, and provides a different set of answers to the questions “What are the authorable units?”, “How can they be combined?”, and “What knowledge guides and constrains their combination?” For this reason, story generation provides a fertile ground for thinking about architectural commitments in any story-based system, including interactive drama.

Tale-Spin [Meehan 1976] generates stories in the domain of Aesop’s Fables. Tale-Spin models the goals and problem solving processes of a set of characters. The stories generated essentially consist of the problem-solving traces of selected characters given initial goals. The major insight of Tale-Spin is that the problem-solving traces of a set of characters doesn’t necessarily make a story. Additional knowledge about what constitutes a good story is needed.

Around the same time as Tale-Spin, people were exploring an alternate approach to story generation based on story grammars [Colby 1973; Rumelhart 1975]. Story grammars encode story structure for a given domain using formal grammar rewrite rules. However, it turns out that syntactic rewrite rules are not enough; the rules must be given semantic (extra-structural) annotations in order to appropriately constrain story recognition and generation. Modern work on story grammars attempts to better understand the relationship between syntactic and semantic knowledge within the grammar [Lang 1999].

Universe [Lebowitz 1985] generates soap-operas, entangling its characters in endless episodic action. Characters are represented as collections of traits. Hierarchical story plans directly coordinate characters in story situations. While these plans have no notion of exe-

cution or run-time interaction, they are similar to beats in that they directly coordinate multiple characters.

Minstrel [Turner 1994] is a case-based model of the human authoring process, writing stories in the King Arthur domain. The system solves a series of authorial goals (e.g. “Make Lancelot run into Guinevere in the forest”) by searching for “similar” situations and analogically mapping them to the current situation. Ani [Kahn 1979] is the only story generation system we know of that creates animated output. Given a formal story description from the Cinderella domain, Ani uses a constraint satisfaction process to design a 2D animation (characters are represented as geometrical shapes) that conveys the story.

Brutus [Bringsjord & Ferrucci 2000] generates stories of betrayal. Thematic knowledge is used to guide the generation of a plot via forward simulation of characters in a simulated world (a la Tale-Spin). Characters are represented by beliefs, goals, plans, and reactive behaviors (production rules). The sequence resulting from forward simulation is turned into a plot outline using story grammars. The plot outline is then turned into natural language using a grammar augmented with knowledge of literary effects.

Conclusion

This mid-project paper described Façade’s project design goals and motivations underlying the design of the architecture, and provided a tour of the architecture, some early observations about the successes and failures of the system, and related work. We look forward to completing the authoring process in 2003 and releasing Façade to the public to play.

The proof is in the pudding. Future papers will detail the lessons learned from the authoring process, and attempt to evaluate how well Façade achieves a middle ground between open-ended simulation and structured narrative.

References and Bibliography

Aarseth, E. 1997. Cybertext: Perspectives on Ergodic Literature. Baltimore: Johns Hopkins University Press.

Bates, J. 1992. Virtual Reality, Art, and Entertainment. Presence: The Journal of Teleoperators and Virtual Environments 1(1) 133-138.

Berne, E. 1968. Games People Play: The Psychology of Human Relationships. New York: Random House.

Bernstein, M. 1998. Patterns of Hypertext. In Shipman, F., Mylonas, E. and Groenback, K. (eds.), Proceedings of Hypertext '98. New York: ACM.

Blumberg, B. and Galyean, T. 1995. Multi-level Direction of Autonomous Creatures for Real-Time Virtual Environments. In Proceedings of SIGGRAPH 95.

Bringsjord, S., and Ferrucci, D. 2000. Artificial Intelligence and Literary Creativity: Inside the Mind of Brutus, a Storytelling Machine. Mahwah, NJ: Lawrence Erlbaum.

Colby, K., Weber, S. and Hilf, F. 1971. Artificial Paranoia (Parry). Artificial Intelligence, 2:1, 1-25.

Colby, B. 1973. A partial grammar of Eskimo folktales. American Anthropologist 75:645–662.

Crawford, C. 1999. Assumptions underlying the Erasmatron storytelling system. In M. Mateas and P. Sengers (Eds.), Working Notes of the 1999 AAAI Spring Symposium on Narrative Intelligence. AAAI Press. Forthcoming in M. Mateas and P. Sengers (Eds.), Narrative Intelligence. Amsterdam: John Benjamins, 2003.

Friedman-Hill, E. 1995-2002. Jess, the Rule Engine for Java. Sandia National Labs. http://herzberg.ca.sandia.gov/jess/

Galyean, T. 1995. Narrative Guidance of Interactivity. Ph.D. Dissertation, MIT Media Lab, MIT.

Kahn, K. 1979. Creation of computer animation from story descriptions. Ph.D. Dissertation. MIT. AI technical report 540.

Kelso, M., Weyhrauch, P., Bates, J. 1993. Dramatic Presence. Presence: The Journal of Teleoperators and Virtual Environments, Vol. 2 No. 1, MIT Press.
Lang, R. 1999. A Declarative Model for Simple Narratives. In M. Mateas and P. Sengers (Eds.), Working notes of the Narrative Intelligence Symposium, AAAI Spring Symposium Series. Menlo Park:: AAAI Press.
Laurel, B. 1986. Towards the Design of a Computer-Based Interactive Fantasy System. Ph.D. Dissertation, Ohio State University.
Laurel, B. 1991. Computers as Theatre. Reading, MA: Addison-Wesley.
Lebowitz, M. 1985. Story Telling as Planning and Learning. Poetics 14, pp. 483-502.
Loyall, A. B. and Bates, J. 1991. Hap: A Reactive, Adaptive Architecture for Agents. Tech Report CMU-CS-91-147. Carnegie Mellon University.
Mateas, M and Sengers, P (Eds.). 2003 (Forthcoming). Narrative Intelligence. Amsterdam: John Benjamins.
Mateas, M. and Stern, A. 2000. 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.
Mateas, M. and Stern, A. 2002a. 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.
Mateas, M. and Stern, A. 2002b. Architecture, Authorial Idioms and Early Observations of the Interactive Drama Façade. Tech report CMU-CS-02-198, Carnegie Mellon University.
Mateas, M., Vanouse, P., and Domike S. 2000. Generation of Ideologically-Biased Historical Documentaries. In Proceedings of AAAI 2000. Austin, TX, pp. 236-242.
Mateas, M. 2002. Interactive Drama, Art and Artificial Intelligence. Ph.D. Dissertation. Tech report CMU-CS-02-206, Carnegie Mellon University.
McKee, R. 1997. Story: Substance, Structure, Style, and the Principles of Screenwriting. NY: HarperCollins.
Meehan, J. 1976. The Metanovel: Writing Stories by Computer. Ph.D. Dissertation. Yale University.
Murray, J. 1998. Hamlet on the Holodeck. Cambridge, MA: MIT Press.
NASA. 1985-2002. C Language Integrated Production Systems (CLIPS). Originally developed at NASA's Johnson Space Center. http://www.ghg.net/clips/WhatIsCLIPS.html
Pinhanez, C. 1997. Interval Scripts: a Design Paradigm for Story-Based Interactive Systems. In Proceedings of CHI97. Atlanta, GA, pp. 287-294.
Rumelhart, D. E. 1975. Notes on a Schema for Stories. In Bobrow, D. G., and Collins, A., (Eds.), Representation and Understanding. Academic Press. 211–236.
Ryan, M. 2001. Narrative as Virtual Reality. Baltimore: Johns Hopkins University Press.
Sgouros, N. M. 1999. Dynamic generation, management and resolution of interactive plots. Artificial Intelligence 107: 29-62.
Sharp, D. 1989. Story vs. Game: The Battle of Interactive Fiction. Computer Game Developers Conference. http://www.channelzilch.com/doug/battle.htm
Stern, A. 2003. Creating Emotional Relationships with Virtual Characters. In R. Trappl, P. Petta and S. Payr (Eds.), Emotions in Humans and in Artifacts. Cambridge MA: MIT Press.
Stern, A. 1999. Virtual Babyz: Believable Agents with Narrative Intelligence. In M. Mateas and P. Sengers (Eds.), Working Notes of the 1999 AAAI Spring Symposium on Narrative Intelligence. AAAI Press. Forthcoming in M. Mateas and P. Sengers (Eds.), Narrative Intelligence. Amsterdam: John Benjamins, 2003.
Stern, A., Frank, A. and Resner, B. 1998. Virtual Petz: A hybrid approach to creating autonomous, lifelike Dogz and Catz. Proceedings of the Second Int'l. Conference on Autonomous Agents. AAAI Press. 334-335
Stern, A., and Frank, A. 1998. Multiple Character Interaction Between Believable Characters. In Proceedings of Computer Game Developers Conference.
Stern, A. 1998. Interactive Fiction: The Story is Just Beginning. IEEE Intelligent Systems, 13:5, 16-18.
Swartout, W., Hill, R., Gratch, J., Johnson, W. L., Kryakakis, C., LaBore, C., Lindheim, R., Marsella, S., Miraglia, D., Moore, B., Morie, J., Rickel, J., Thiebaux, M., Tuch, L., Whitney, R., Douglas, J. 2001. Toward the Holodeck: Integrating Graphics, Sound, Character, and Story. In Proceedings of the 5th International Conference on Autonomous Agents.
Turner, S. 1994. The Creative Process: A Computer Model of Storytelling and Creativity. Lawrence Erlbaum.
Weizenbaum, J. 1966. Eliza. Communications of the ACM, 9, 36-45.
Weyhrauch, P. 1997. Guiding Interactive Drama. Ph.D. Dissertation, Tech report CMU-CS-97-109, Carnegie Mellon University.
Young, R. M. 2001. An Overview of the Mimesis Architecture: Integrating Intelligent Narrative Control into an Existing Gaming Environment. In The Working Notes of the AAAI Spring Symposium on Artificial Intelligence and Interactive Entertainment.