tapraid4/z92-psen/z92-psen/z9200107/z922006d07a | sangreyj | S=9 | 11/17/06 | 22:33 | Art: z92-2006

Barry G. Silverman*
Michael Johns
Ransom Weaver
Josh Mosley
Electrical and Systems Engineering
School of Engineering and Applied Science
University of Pennsylvania
Philadelphia, PA 19104-6315

Gameplay, Interactive Drama, and Training: Authoring Edutainment Stories for Online Players (AESOP)

Abstract

This paper describes initial efforts at providing some of the technological advances of the videogame genres in a coherent, accessible format to teams of educators. By providing these capabilities inside an interactive drama generator, we believe that the full potential of educational games may eventually be realized. Sections 1 and 2 postulate three goals for reaching that objective: a toolset for interactive drama authoring, ways to insulate authors from game engines, and reusable digital casts to facilitate composability. Sections 3 and 4 present progress on those tools and an in-depth case study that made use of the resulting toolset to create a large interactive drama. We close with lessons learned to date and a look at the remaining challenges: the unpleasant reality that state- of-the-art tools are not yet able to boost the productivity of edutainment authors.

1 Introduction and Goals

We envision a future where many games exist that help people to cope with their health issues, child rearing difficulties, and interpersonal traumas. Further, these games will be so compelling and easy to revise that many players will feel compelled to contribute their own story to the immersive world—a contribution that is both self-therapeutic and that helps others who see some of their own dilemma in that story. This will be an industry that is consumer grown, since they will be the creators of new games for other consumers. As a few of many possible examples (1) parents will experience what other parents of handicapped children have struggled with and overcome, (2) children who are bullies will learn what their bullying does to other kids, and (3) people with chronic health issues (overeating, diabetes, heart disease, etc.) will learn what happens when self- denial and poor diets prevail. We envision that a single underlying game editing environment and alterable cast of digital characters can be used to facilitate such a variety of games with therapeutic value.

At present there are many obstacles to this vision: (1) the videogame industry offers addictive, immersive entertainment and provides most of the seeds for this industry to grow from; however, their games have little education fo-

Presence, *Vol. 16, No. 1, February 2007*, 000-000

*Correspondence to basil@seas.upenn.edu.

cus and they provide few if any tools directly reusable in this niche; (2) the computer-based education field does produce interactive training tools, however, these are heavily corporate and government training based and have almost no entertainment value and hence aren't spontaneously fueling much consumer interest; (3) the field of movies and TV show writing creates compelling characters that consumers care deeply about, but this medium offers no chance of interactivity that is vital to self-discovery and skill development; (4) the field of human behavior modeling offers innumerable models based on first principles of physiology and psychosocial dynamics, yet outside of a few experimental military simulators, these are rarely inserted into autonomous characters in videogames and interactive dramas; and (5) the successful edutainment offerings to date (e.g., Math Blaster, Reader Rabbit, Oregon Trail, etc.) are monolithic, nonalterable creations of their proprietors. We need a next generation of environments that takes the best from each of these fields and provides the needed capability. The elements of this environment mostly exist, but they haven't been properly put together yet.

We believe that one could take the important elements that exist today and synthesize them into the desired capability for Authoring Edutainment Stories for Online Players (AESOP). Provided the game authoring toolbox (what we call AESOP) is usable and useful, then game authors will be able to write about their situations and game players will benefit from immersively experiencing and seeing the problems that others have had to deal with. The first goal of this research is thus to explore alternative ways for a game generator to help authors introduce entertainment and free play into role playing games and interactive dramas that are training interventions.

This goal is compounded since learner oriented game designs are one of the most difficult areas in developing videogames. First off, although training requires players to progress through stories (pedagogically valuable scenarios), at its heart gameplay is not about interactive fiction though there are those who buy interactive fiction games. Interactive drama is all about storytelling from the author, while gameplay is much more about story creation by the player—and these competing aesthetics need to be resolved if pedagogical games are to achieve their potential in general.

More than any other mechanic of gameplay, a narrative in a game raises the idea of destroying the central aesthetic—that players create their own stories and that is what keeps them coming back. (Mechanics is a term used in game design. It most nearly means "functionality" on how game pieces work, or how the game functions.) Other gameplay mechanics more or less have a story inside them, in fact countless stories inside them. Further, many of these mechanics have built-in skill training functions at the same time that they permit unconstrained play and inquiry. For example, a racing and chasing mechanic includes lessons about how to chase down bad guys and cut them off from escape. Likewise a combat game has built-in weapon firing and target damage models, plus skill challenges such as room clearing, among others. If one invests in the realism of these mechanics, they provide useful training and transferable skills (Filipczak, 1997; Green & Bavelier, 2003). The same should be true if interactive dramas are well done, particularly if the goal of the drama is to learn, rehearse, and transfer skills in interacting with people; and/or to learn how to persuade people to change dysfunctional behaviors and by that to learn how to cope with one's own poor health behaviors before they become a real world problem. That is, dramas are essentially dialog games, and hence one must take extra precautions to avoid damaging the gameplay aesthetic.

It is also a fact that students learn the most and retain it the longest when they must teach a topic to others (Gibbons & Fairweather, 1998; Reigeluth, 1999). One always learns a subject better if one is confronted with being the teacher rather than the student. So a microworld could be quite a powerful training device if it affords teaching opportunities, or even better if it thrusts the player into roles where other characters will be vulnerable and dependent on the player to teach for a successful conclusion to be reached. Why should the player care to become a teacher? What can drive them to reach this level of learning? People reflect this kind of passion for videogames and at the movies. When game mechanics work and when characters are likable, players (viewers) achieve enormous empathy for the characters (Decker, 2002; Hopson, 2001; LeBlanc, 2002) and are willing to go to great lengths to save them and to help them work out their problems (e.g., as in "God" games such as The SIMS, virtual Petz, and Tamagotchi), and to go on quests on their behalf or assist them in shifting their behaviors to more successful models such as in role playing games. In these milieus, players reveal willingness to learn skills that will help the dysfunctional characters to cope.

At this point, let us restate the first goal as researching a generator that permits authors to create interactive role playing games that preserve the central aesthetic of gameplay, that utilize stealth learning and self-discovery in microworlds for training and behavior change purposes, and that incorporate learning by teaching. A second goal is to provide a high level graphical user interface for the generator, and by that to insulate authors from having to learn a game engine's details. A corollary to that goal is that the generator must itself be kept fairly simple if the laity is to succeed in using it.

2 Creating Stories with Free Play

As already mentioned, we are seeking to set up a generator that can expose constructs and parameters of a storyworld so that new interventions may be more readily authored that promote free play and entertainment within a narrative structure. To support this research, we are attempting to produce a cast of animated puppets and sets (introduced in what follows) in a way that they can be reused for many stories this is our (third goal). This is the idea of a composable and reusable storyworld, including digital sets, cast members, and Campbellian archetypes that can be adapted and extended for further sequels not even yet anticipated. Our ideas for reusable casts and archetypes follow from work such as Campbell (1973), Decker (2002), and Propp (1968), as well as how they are used in franchise games, comics, and serials. We include characters of different ages, genders, and backgrounds/ethnicities, and in the roles of hero, sidekick, allies, opponents, tricksters, lovers, and so on.

It is worth pointing out that, for now, we made a conscious decision to base this cast and sets around 2D, hard-edged cel-based animation since research has shown that subjects with health behavior change issues often allocate little cognitive processing to health messages, and feel greater confidence about being able to process and conquer message sets introduced in cartoon formats (Green & Brock, 2000). However, the underlying technology also supports 3D animation, as is used in our Unreal Tournament game for military training.

In addition, we chose a finite state machine (FSM) approach as the basis for our dialog model and our scriptwriting application. The FSMs may be represented visually within a directed graph or tree. Edges represent the various dialog choices available to the user after a given node plays out. Each node contains both dialog and animation instructions for the avatar and non-player characters (NPCs) to carry out and that may be activated in parallel. This approach allowed our writers to choreograph the animation of multiple characters to occur simultaneously. The AESOP generator is currently implemented to help authors with the simple FSM approach and so that it can encapsulate and deliver the interactive game to other devices that display and track gameplay. Section 3 will explain this structure in more detail. Before that, however, it is important to further explore how AESOP seeks to satisfy the first design goal.

We freely admit to several design biases in our approach and make no attempt to justify these. They include:

- a desire to try (2D) graphics rather than text only;
- a belief that the tension between entertainment and education can be resolved;
- a belief that it is possible to author highly branching, interactive stories that are coherent and sensible.

2.1 Narrative Intelligence

The field that some refer to as narrative intelligence has recently produced a number of rich ideas for incorporating narrative into game worlds without totally sacrificing gaming's central aesthetic. In this research, we synthesize, adapt, and extend several of these ideas as this section will note. None of the literature to date has directly addressed the topic of learning by teaching, so this places us in a new realm that drives our inquiry. Also, very little if any of the narrative intelligence research to date has addressed how to assist storyworld authors, so several original contributions are needed here as well to truly realize our research goals. We are creating the Authoring Edutainment Stories for Online Players (AESOP) generator as part of this research and are seeking to have it assist with authoring constructs as F1 portrayed in Figure 1 and as further described in what follows.

From the player's perspective, when they encounter a storyworld such as in Figure 1a, they do not wish to be placed on metaphorical on rails—a storyline forcing the player down a narrow path that is author specified. For example, Figure 1a shows four areas of town where scenes (and tracking objectives) exist. One can linearize the world or allow the user to meander in and out of these freely. The best narrative solutions found in the game field to date tend to approach this concern by interspersing free play/inquiry and player story creation with player-selected choice points for advancing the story. At these choice points, the player approaches characters or other devices that reveal more of the author's story and that advance them to the next scene of the drama. In this manner, a drama eventually unfolds. Some successful examples of this blending of story and game are Grand Theft Auto and Deus Ex, among others.

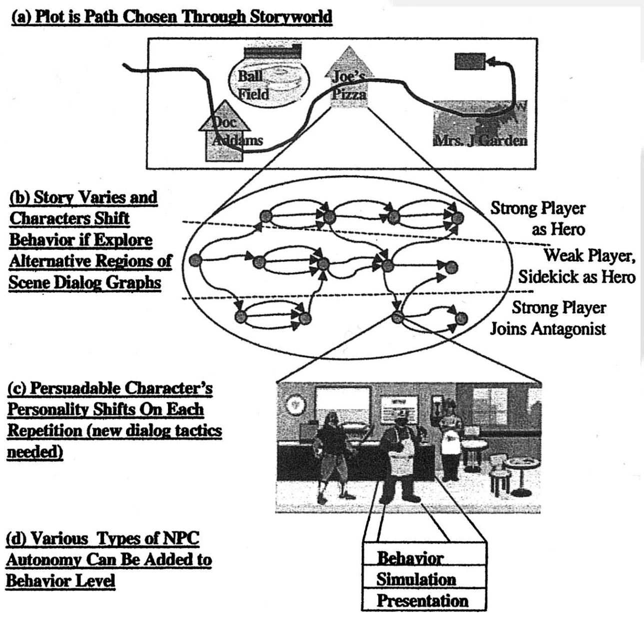

(a) Plot is Path Chosen Through Storyworld
Ball Field
Joe's Pizza
Doc Addams
Mrs. J Garden
(b) Story Varies and Characters Shift Behavior if Explore Alternative Regions of Scene Dialog Graphs
Strong Player as Hero
Weak Player, Sidekick as Hero
Strong Player Joins Antagonist
(c) Persuadable Character's Personality Shifts On Each Repetition (new dialog tactics needed)
(d) Various Types of NPC Autonomy Can Be Added to Behavior Level
Behavior
Simulation
Presentation

Figure 1. *Sample insertion points for narrative intelligence to minimize perceptions of limiting free play.*

Similar to this is the approach being taken in the Army's Mission Rehearsal Environment (MRE; Swartout et al., 2001); however, unlike the popular titles, this approach is concerned with imparting doctrinally correct training objectives. The MRE approach requires authors to (1) deconstruct the story into the smallest parts (scene nodes) where autonomous character and player freeplay can be permitted, and (2) to identify graph transitions that are gate conditions for triggering scenes and/or for allowing scenes to be omitted without loss of training value. This approach permits the player to explore a node repeatedly, getting better with each try and through exit node feedback. However, this approach is for doctrinally correct training that requires repetition for improvement.

In the current research we are interested in learning by teaching, in mental model transfer, and in behavior shifting as mentioned earlier. This relaxes the need to repeat the identical scene, and affords the opportunity for scenes to hold surprising plot reversals if you play them differently. For example, Figure 1b zooms us in on a plot or dialog graph for a sample scene or quest of the storyworld. Here the nodes and edges are as described for the FSM in the prior section. Depending on the player's personal goals, entertainment objectives, style of play, and confidence, among other factors, they may decide to pursue very different avenues through a scene's dialog graph and in fact through the entire story world. This can be both educational and entertaining. Monkey Island is an example of a title that incorporates argumentation tactics in this manner, though in that title there is only one outcome and path out of a scene regardless of dialog choices. Popular games such as Civilization, Black & White, and EverQuest, in turn, have richer outcome possibilities and let the player learn about the world's emergent nature and how they have to live with the worlds they create and the personas they project. In all these games, lines of intellectual inquiry have consequences and discoveries result from exploration.

The same types of exploration can exist in pedagogically oriented dialog graphs. Earlier efforts introduce such possibilities by providing side characters that coach and cajole a player back to the pedagogically preferred path (Marsella, Johnson & LaBore, 2000; Silverman et al., 2001) of a dialog graph. In the types of storyworlds we currently envision, however, the learning can be just as effective if a companion, sidekick, or "window character" regrets aloud the player's decisions and then on its own directly performs the dysfunctional character training and persuading (Decker, 2002). In theory, the player should even be able to adopt a potential storyworld antagonist's causes and be entertained by helping to support the antagonist's objectives throughout the storyworld, yet suffer no loss of learning as a result. Using this technique of gameplay, we are finding that such dialog plot regions and the resulting opportunities to try out "both sides" of a conflict can enhance the drama and help boost replayability (see Case Study of this paper and Silverman, Johns, & Weaver, 2003).

The reason players can adopt the cause of either side and potentially suffer no loss of learning is tied to how we learn in stories. In storytelling theory, the listeners, or participants, are viewed as containing significant understanding and know-how already, and the story is but a fuse to ignite the recipients into synthesizing a new conceptualization for themselves. This is also consistent with persuasion theory, such as in the Persuasion Likelihood Model (Petty & Cacioppo, 1986) which states that rational arguments are *unlikely* to persuade. Rather it is the peripheral cues that are modeled that convince the audience. Thus the many movie scenes of actors smoking during or after a significant activity have far more power than all the public health media campaigns laying out the rational arguments about adverse health effects. As long as an interactive drama successfully models the desired health behaviors, the theory suggests the player will pick up on it.

There have been a number of investigations into conversations with autonomous agents outside of stories and/or in fixed plot graphs and their findings have potential here. Some of the earliest work dealt only with simple animation and kinesthetic issues such as breathing and blinking, lip synching, and facial expressions—things we label as the presentation layer in Figure 1d (e.g., see Lasseter, 1987). More intriguing, however, is work on the higher layers of Figure 1d and how they might be integrated into dialog plots as suggested in Figure 1c. We define the simulation layer of an agent as how it performs in the world, for example, navigation, collision avoidance and collision damage, and physiological needs and health needs. In some characters, we embedded a number of validated reservoir models of body organs and functions (Silverman et al., 2002). Less realistic models are widely used in popular God games; however, there is never any story designed into it, and players merely ascribe story when uncorrelated events arise. In our work these eventually are intended to provide many potentially engaging training dialog opportunities, particularly with characters that attempt to deny or mislabel their symptoms, risk factors, and lifestyle habits.

Likewise, the behavior layer involves characters' emotions and motivations, planning/choosing style, and general personality variables including coping modes. One idea here is to allow the moods and personalities of the non-player characters (NPCs) to respond dynamically and emergently to direct player interaction. The NPCs include parameterized models of autonomous, emotive behavior and different types of responses to player actions or dialogs. The Virtual Theatre Project (Hayes-Roth & Rousseau, 1997) has explored this concept for fixed plot graphs and shown that players perceive significant dramatic variability and story-creating potential, even though the plot is fixed. An early prototype of Heart Sense Game has likewise deployed an autonomous coach/companion that alters its mood, emotion-directed utterances, and physical expressions as a function of where the player strays in the plot or dialog graph and found this reduces player difficulties (Silverman et al., 2001). In the current research we have eliminated overt coaching, but are considering this idea for key characters so they alter their personality each time you play. This in turn could further the perceived player variability and sense of free play. It also would mean players must use a different persuasion strategy each time they reenter a scene.

Silverman et al. (2002) demonstrate how a number of models from the psycho- physiological literature relevant to these two layers (simulation and behavior) can contribute to making agents autonomous and need-reservoir driven in their coping styles and emotive decision making. This is an attempt to move beyond Bates' believable and broad agents (Mateas, 1997) into the realm of reliable models of human performance calibrated against field data—an area where learning systems must depart from entertainment.

A final issue facing storytelling in simulated worlds is that it forces the player into what is arguably the worst side of human-computer interaction, that of the computer's poor conversational capabilities. As with voice menu systems on the telephone, the machine-generated voices are stilted, their ability to handle nuances is poor, and they often misunderstand the speaker. Up to now we have used a text to speech system during authoring but replace it with actor voice-overs once the parts are finalized. We completely avoid the speech recognizers and instead rely on dialog menus, which raises several difficulties. Specifically, the risk of dialog menus is that the designer has neglected to include options the player would like to see voiced, or if they are voiced, hasn't included mechanics in the other characters to support the idea in the player's head. So far in the case study, however, we have not encountered this difficulty and believe the large degree of free play mentioned in this section tends to minimize the dialog menu risks.

3 The AESOP Generator

AESOP is intended as a front end authoring aid that includes plot and dialog editing GUIs (graphical user interfaces), storyworld templates, palletes of reusable parts, digital cast members, autonomous behavior modules, and reusable art/animation assets. Its output is automatically parsed into XML instructions for each agent in the storyworld in the form of FSMs that are sent to the game engine. With the use of an XML interface, the AESOP editor suite becomes engine independent. Its FSMs could in theory be played by any of a variety of game engines that run NPCs and avatars. Figure 2 overviews that architecture, and the discussion that follows provides further details. F2

When building a piece of edutainment, the generator must support the entire group, including training content developers, story writers, and game authors. A goal of this research was to study one or two such groups as they attempted to create an edutainment system, elicit their design protocols and intermediate game representations, and to try and craft a generator environment that might better support their mutual and collaborative efforts. In the latter, we were hopeful of placing various tools and versions of a generator in front of them to further the requirements of the observation and elicitation process and to study environment design concerns. Where and when we did not have a specifically needed tool, we intended to support the need manually, by directly programming the authoring need and/or game mechanic. In this fashion we are engaging in a traditional spiral software development of the AESOP generator.

In terms of specifics, the lead author of this paper serves as principal investigator of both the AESOP generator (Silverman, Johns, et al., 2003) and two edutainment projects that are making use of it—the Heart

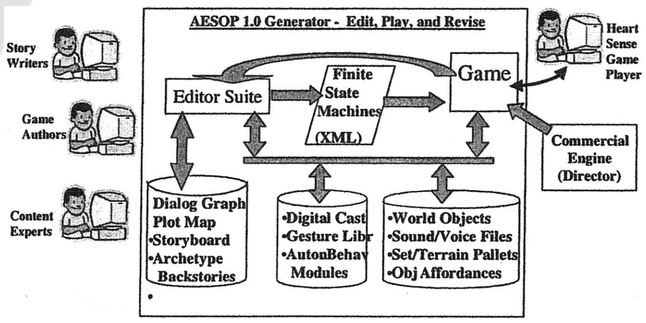

AESOP 1.0 Generator - Edit, Play, and Revise
Story Writers
Game Authors
Content Experts
Editor Suite
Finite State Machines (XML)
Game
Heart Sense Game Player
Commercial Engine (Director)
Dialog Graph Plot Map •Storyboard •Archetype Backstories
•Digital Cast •Gesture Libr •AutonBehav Modules
•World Objects •Sound/Voice Files •Set/Terrain Pallets •Obj Affordances

Figure 2. *Architecture of the AESOP generator that supports the three goals of creating pedagogical interactive dramas with reusable casts/objects and insulates authors from commercial systems.*

Sense Game (HSG) role playing drama (Silverman et al., 2001) and a recreation of Mogadishu/Black Hawk Down (BHD) crowd scenes for a first person shooter scenario (Toth et al., 2003). This paper focuses primarily on the HSG and Director version of the AESOP generator; though discussion at times will mention features and lessons of the other applications.

The HSG started last quarter 2001, and since that time the HSG development group met at times weekly and at other times bimonthly for 90 minute face-to-face design brainstorming and feedback discussions. The content of these interactions is described in the case study (Section 4). This group included six faculty investigators, two graduate student researchers, at times up to nine undergraduate digital media design and systems engineering students (helping with art, animation, sound, voice-overs, etc.), and one junior and one senior screenwriter (part-time, freelance). They were supported in between meetings via a variety of collaborative tools including threaded chat, web-based ftp repository (organized into sub-team memory bins), email listserv, and general email. Threaded chat was highly useful at the outset for discussing learning objectives and game mechanics, but, as the threaded conversations grew, people found them cumbersome and resorted to email and direct meeting instead. The ftp repositories followed a similar pattern.

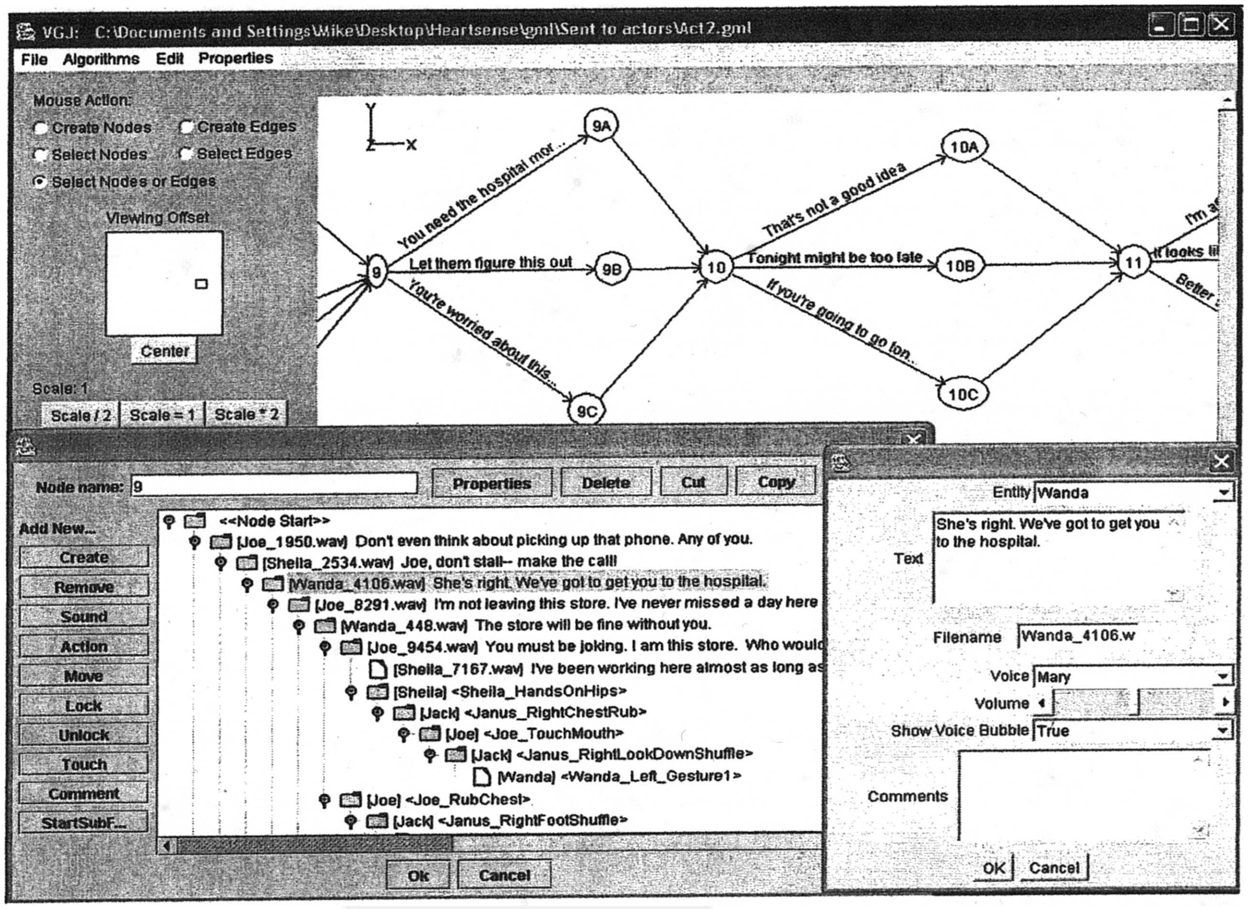
Figure 2 shows two boxes labeled Editor Suite and Engine. Various tools were placed into these boxes and evaluated/improved over time, as subsequent sections of this paper suggest. As an overview of that discussion, the plot map (acts, scenes, etc.) and character backstories started as text- only descriptions, evolved to a manually filled in multimedia set of webpages (www.seas.upenn.edu/~barryg/heart/index.html), and is now targeted to become an interactive editor that will assist in merging learning objectives with story writing goals. The earliest versions of the branching, interactive dialog script were table-based which was then replaced by a directed-graph editor (Section 3.1). The earliest versions of the game engine existed prior to the artwork/Flash movie stores and utilized stick figures (with text to speech) to act out the roles and dialog from the script. Subsequent versions included a library of sets and characters replete with growing stores of gestures (Flash movies) and actions one can assign with a mouse-click to the character puppets (see Section 3.2). One authors dialog and action in the graph editor tool. This produces scripts in text and graph markup language (GML or XML) that are instruction sets or FSMs that the engine can run on top of Director with the help of a text to speech (TTS) processor and the library of Flash movies for each character's gestures and actions. Thus there is no need to program in Director, and developers author role playing dialog scenes and watch them acted out with the push of a button, provided they do not expect gestures and actions that are not yet in the Flash movie stores for each character.

At times we have included autonomous emotive agents in earlier versions of HSG, agents capable of emergent behavior (Silverman et al., 2002; Silverman, Johns, et al., 2003), while the BHD and other applications make substantial use of such autonomy. These are NPC agents that operate with their own behavior goals, standards, and preferences, and that can react to and effect the drama and the player. The current article omits discussing these characteristics, but we have numerous papers on this topic (e.g., see Silverman et al., 2002 and Silverman, Johns, et al., 2003, among others), and we continue to work on the challenges of integrating author-driven vs. agent-driven story elements.

3.1 GraphEdit Tool

Our FSM editor is a modified version of Visual Graphs for Java, developed at Auburn University. To facilitate our particular needs, the second author of this paper added custom dialog boxes for the data we manipulate, and added support for the XML output required by our game engine.

In our graphs, nodes contain uninterruptible segments of storytelling, and edges correspond to the choices given to the user after each node plays out (see Figure 2). Within each node is a set of behaviors assigned to various characters, arranged as a tree. There are eleven possible behaviors, the most common being: 1) *SOUND,* which causes the specified character to lip-synch a line of text either defined by a wav file or, failing



Node name: 9

<<Node Start>>
[Joe_1950.wav] Don't even think about picking up that phone. Any of you.
[Sheila_2534.wav] Joe, don't stall-- make the call!
[Wanda_4106.wav] She's right. We've got to get you to the hospital.
[Joe_8291.wav] I'm not leaving this store. I've never missed a day here
[Wanda_448.wav] The store will be fine without you.
[Joe_9454.wav] You must be joking. I am this store. Who would
[Sheila_7167.wav] I've been working here almost as long as
[Sheila] <Sheila_HandsOnHips>
[Jack] <Janus_RightChestRub>
[Joe] <Joe_TouchMouth>
[Jack] <Janus_RightLookDownShuffle>
[Wanda] <Wanda_Left_Gesture1>
[Joe] <Joe_RubChest>
[Jack] <Janus_RightFootShuffle>

Entity: Wanda
Text: She's right. We've got to get you to the hospital.
Filename: Wanda_4106.w
Voice: Mary
Volume
Show Voice Bubble: True
Comments

Figure 3. *Illustrative screens of the graphEdit tool for constructing branching dialog and adding choreographic and multimedia instructions to FSMs.*

that, a text-to-speech generator; 2) *ACTION,* which causes the character to perform some specific animation; or 3) *MOVE,* which causes the character to physically move from one position on the screen to another. When one behavior finishes, all of its direct children are executed in parallel. This allows for authors to specify the timing of various components of a scene without knowing specific details about the art or voice assets that will eventually be put in place.

Once the tool is utilized, one can save the graph out to XML format which will be used by the game application and engine.

Suppose we want to create a scene in which Jack says "Hello," and Joe replies "Hi" while waving. We would begin with a speech behavior for Jack, and then add two child nodes: one for Joe's reply and one for his gesture. Since both begin after Jack stops speaking, they will execute at the same time and Joe will speak while waving.

Figure 3 shows our dialog graph authoring tool. The main screen shows the structure of the graph, with edges representing the options presented to the user and nodes, as described above, containing the behaviors that result. Overlapping this window in the lower left is the node editing dialog, in which the behavior tree for a given node is created. Finally, in the lower right is the behavior properties dialog, which gives control over specific aspects of each behavior. For example, a gesture behavior dialog contains among other things a list from F3

tapraid4/z92-psen/z92-psen/z9200107/z922006d07a | sangreyj | S=9 | 11/17/06 | 22:33 | Art: z92-2006

which to choose the character performing the gesture, and a list of gestures available to that character.

Once the tool is utilized, one can save the graph out to XML format which is then passed to a subsequent module for parsing and linking with the game engine.

3.2 Gesture Builder

The fourth author of this paper, and his Digital Media Design (DMD) and Fine Arts students, have created all the artwork for the reusable casts as well as the sets and terrain objects for the HSG version of AESOP. The Flash artwork was developed in tandem with the story development using a stylus pen and Adobe Illustrator. Each body part was drawn on a separate layer to aid the construction of the Flash-animated "puppets." To provide the maximum flexibility, it was essential to build the animations so they could be run independently and simultaneously.

In addition, they created a Director-based demonstrator in order to test how the Flash animation segments would flow into each other and in order to build gestures. The third author then used this as a starting point to build a browser application that would output, as XML files, sequences of animation that could be utilized by the scriptwriters. These are the animation Macros which are referenced within an ACTION behavior (e.g., `the action = "Jack_RightChestRub" in <ACTION entity="Jack" starttime="166846" endtime="15394585" action="Jack_RightChestRub" framespersecond="24"/>`). So the Macros are just composites of some of the individual animations within the character puppets. A simple Macro might just be "`Jack_Blink`". A more complicated Gesture, such as "`Jack look angry,`" might include a change of facial expression, a shift of weight and movement of arms, forearms, and hands so they raise to the hips. The resulting coordination of this animation would be exported as the Macro "`Jack_LookAngry`" for use in an ACTION. A few complicated but common animations, such as walking, might be created such that they could be called by a single animation reference in the Macro editor. Figure 4b gives an overview of the library of macros added to date for a given character of the cast. F4

This structural approach is fairly common to the game industry; however, in this particular case it was necessary to provide a simple interface and upgradeable characters that could accept new animations on a need-by-need basis as the story development team authored stage direction. Macromedia Flash was a simple and inexpensive 2D animation tool well understood by the undergraduate DMD students. Motion capture, and post-capture editing, would be another technique of generating animation fragments that could be integrated with this Macro editor.

3.3 Engine/Wrapper

A traditional approach to a game such as Heart Sense would be to create it using the multimedia development program Macromedia Director. The final product would be a self-contained application, royalty free and not requiring Director to run, and suitable for running from a CD or placing on the web in an html wrapper. The drawback is that the creation of subsequent games would require access to the expensive Director software, as well as access to (an even more expensive) programmer knowledgeable in Director's powerful and sophisticated internal scripting language, Lingo.

The third author foresaw that Director could be used to create an engine that alone is not a game, but instead is designed to accept instructions for assembling and presenting a game. Consequently, part of this project has been developing the presentation tool engine in Macromedia Director. Director was the first choice for this due to the ease of producing executable programs for Windows and Macintosh, and browser-embedded applets that are platform independent.

So part of the engine software is supplied by the Director program in the creation of a stand-alone player application. The heart of the engine, however, is an algorithm written in Director's Lingo scripting language that essentially parses the FSM markups or instruction sets contained in an XML data file and translates them into instructions native to the Director multimedia presentation environment. To refer back to the beginning

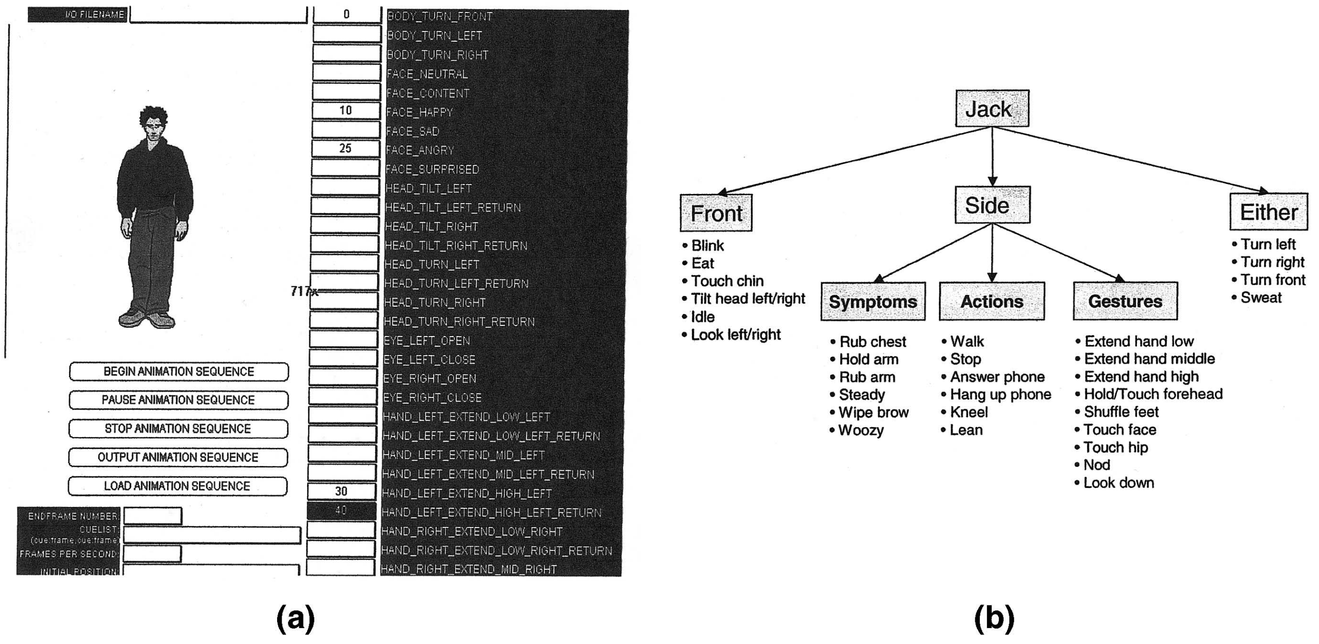


Figure 4. *(a) Overview of the Flash macro editor (demonstrator tool) for building gesture procedures and movies for the various characters of the reusable cast. (b) illustrative macros created for a character (Jack) using the demonstrator tool.*

of this section, this is the environment in which a Director programmer could work to create a game such as Heart Sense. Without AESOP, if one desired to create another, different game, one would have to basically start from scratch. With our engine, a writer/artist could create a game using AESOP to specify the structure and content of the game, then distribute it with a copy of the engine for playing by the user.

The engine's algorithm has some very simple low level requirements:

1. An XML datafile called Statemachine.txt must be present. Within the FSM represented in XML, choices made or nodes reached may cause other FSMs to be loaded into the engine (e.g., Act2.txt), but the game will always start from Statemachine.txt.
2. Upon completion of the behaviors contained in an individual node of the FSM, all out-edges (transitions) will be represented as textual buttons in the lower-left corner of the screen. Therefore, what the user clicks on determines to what node of the FSM the game progresses. There is a limitation of six out-edges.
3. All elements referenced within the XML datafile (such as the graphic table.pct, the flash animated character Jack.swf, the Macro `Jack_WalkLeft.txt` or the audio file Jack_567.wav) must be present and correctly named in the appropriate sub-directories (Audio, Media, Macros).

Beyond this, assuming validity of the XML, the engine basically enforces agent/object turn taking, assigns resources and procedures from libraries (voice files, animation movies, etc.) to puppets, and handles input from the user. The algorithm for this is described as follows:

1. On launching the engine opens the XML file Statemachine.txt. It looks at each node of the FSM in turn and appends the contents of the element `<BEHAVIORS>` onto a variable behavior. There are 11 kinds of sub-elements in the element `<BEHAVIORS>`. The five most commonly used are `<CREASE>`, `<ACTION>`, `<SOUND>`,

`<MOVE>`, and `<STARTSUBFSM>`

2. After the XML is parsed the engine will search through the "`behavior`" looking for "behaviors" with a `starttime` of 0 (e.g., `<CREATE entity="Charlie" starttime="0" endtime="7806641" x="1000" y="200" z="16" scale=""filename="charlie.swf" ink="trans" blend=""rotaton"" skew="" />`).
3. The engine will wait for the completion of the `behavior` described in (2) and then search "behaviors" for any behaviors beginning with a starttime equal to the endtime of the behavior just completed. Puppets can be directed to do several types of behavior simultaneously this way, such as move, animate, and talk; and action can be easily sequenced to the appropriate moment, such as stopping the walk animation when the puppet has reached the destination location on the screen. Also, since sound is time dependent and animation is frame dependent, this method bridges the problem of variable processor speeds from machine to machine.
4. If the chain described above eventually leads to a `<SUBFSM>` element (e.g., `<STARTSUBFSM starttime=13859719" endtime ="9887273" filename="A1S2.txt" />`), the cycle will repeat as if from (1) using the named file.

Each type of behavior also has its own algorithm within the engine to determine how to execute the desired behavior, and how to determine when the behavior has finished. For example, in the case of `CREATE`, the behavior names the visual media element that is to be used. The engine then imports it from an external library, and instantiates it on the stage according to the instructions in the `CREATE` behavior. The behavior is finished when the code instantiating the visual media element has finished and returns control to the function that called it, so "`behaviors`" is then searched immediately using the endtime number of that `CREATE`.

Each node's sounds are imported en masse at the beginning of the node, to avoid odd delays in conversation. When the `SOUND` behavior is called, the sound is played in an available sound channel (up to 8 simultaneously) and if it is directed at a particular entity (i.e., speech), amplitude data is dynamically generated and sent to the puppet for lip-synch animation. The engine must watch the sound channels that are occupied with playing a sound, and when they finish, search "`behaviors`" using the appropriate endtime number.

For `ACTION`, the `ACTION` element contains a reference to the Animation Macro, which is itself an external XML file contained in a library. The engine must first import and parse the Macro, then send the animation information contained therein to the puppet in question. The animation macro also contains the information as to what frame the animation is considered to be over. This is set by the choreographer using the Animation Macro Creator tool (shown earlier in Figure 3). Every time an `ACTION` behavior is sent to a puppet, that puppet's internal frame counter is set to 0 and restarted. The engine then monitors that puppet's frame number and, when that number is passed, searches `behaviors"` using that action's endtime number.

For `MOVE`, of course, the engine must watch the position of the puppet and declare the `MOVE` to be complete when it is in the correct location.

As previously discussed, the behaviors assigned to characters within a given node are arranged in a tree, with direct children executed in parallel upon completion of parents. The role of the engine, then, is to examine this tree of behaviors, determine whether any currently executing behaviors have completed in the last frame, and if so begin its children. When all behaviors in a node have completed, the engine presents all outgoing edges as choices to the user. When one is selected, the next node begins.

The engine does not restrict user interaction to merely choosing transitions between nodes. Clickability of objects in the game is supported; clicking a clickable object can launch a specified FSM, taking the game in another direction. Also, one of the major supported media elements is Flash animation; we have used this inexpensive and widely used multimedia animation program to create all our character puppets as well as non-static props in the game, such as an ambulance. Flash is commonly used for interactive online games, and thus could be used to create a custom game-within-a-game that allows more action-oriented gameplay than the basic narrative fiction game functionality of this writeup.

The engine will also keep a record of the user's choices (keystrokes) and time taken in the game, and record this to a data file when the game exits. If the player exits prematurely or an abort occurs, this file, along with cookie information, may be used to reconstruct where in the process the player was just prior to abort.

4 In-Depth Case Analysis: Results

The AESOP generator has been developed in parallel with the creation of the HSG, and with the goal of supporting the authoring of that game. To create HSG, three part-time creative writers and several content experts (co-investigators) assisted this paper's authors. This section reports some observations of that usage.

To initiate the case study, the first author of this paper came up with a short description of the intended game and a paper-based version. No game is likely to succeed if its appeal cannot be summarized in a few sentences. Specifically, for HSG, this description is (after some massaging from HSG co-investigators):

Heart Sense Game is a role-playing game in which you help the hero try to solve a crime and simultaneously rescue his career and find romance. However, as the hero, some of the many characters you might get clues from need your help to deal with heart attacks before they or others can help you. Since, for their own reasons, they often don't believe they are having a heart attack or don't want to take care of it promptly, there are significant obstacles to helping these characters to help themselves. And if you prefer to harm these characters, you are free to do so, but watch out, your own future will be affected as well!

The three-act, character-driven soap or adventure story is a well understood formula both in the movies and on television. The writers immediately recognized this format and could relate to its formulaic conventions to drive the player and his or her avatar through the story summarized above. Likewise the training content developers could identify with hero's journey as well. Jointly, writers and content experts began to make passes over the story to preserve its engagement (ENG) aesthetic which we believe is a function of overall transport (T) as well as factors concerning the plot (P1), people (P2), and places (P3). Keeping these in mind, the various authors provided brainstorming ideas and interactively deepened the script. The writers tended to form narrative descriptions of the scenes and the training developers began to allocate their learning objectives to these quests and scenes. A negotiation went on where the dramatically inclined attempted to limit the learning objectives entailed in any given scene, while the trainers tried to ensure that their full set of goals was covered somewhere in the overall journey.

The extent of the training objectives determines in part the length of the story, and the number of quests that must be included. Thus for example, in the heart attack domain there are multiple types of heart attack presentations, and three main categories of behavioral delay. After brainstorming, the goal of limiting the length to that of a television sitcom (50 minutes for once through) eventually ruled the day and limited the journey to three quests in total during Act 2. Further, writers insisted that each scene should move along quickly and not sacrifice dramatic pace for the sake of training detail. The negotiation landed on the side of less-is-more, though discussions continued about implementation details for quite some time. In both gaming and storytelling, it is vital that each line of dialog potentially has three purposes: move the plot along, reveal some aspect of the speaker's backstory, and set up any local effect (e.g., joke, action, lesson, etc.). This required a number of discussions and rewrites about dialog, plot, scenes, and beats.

The result of the negotiation between writers and trainers often omitted the concept of gameplay since neither group was trained in this aesthetic. Further, traditional story writers such as ours are not trained in interactive media, and it was difficult for them to envision a human player taking over the role of their central character. While they were skilled at bringing in tension,

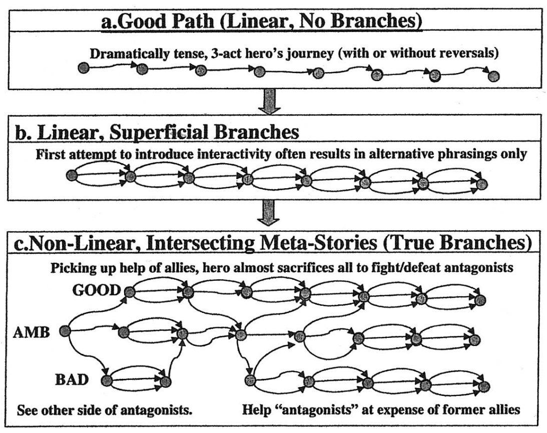

a.Good Path (Linear, No Branches)
Dramatically tense, 3-act hero's journey (with or without reversals)
b. Linear, Superficial Branches
First attempt to introduce interactivity often results in alternative phrasings only
c.Non-Linear, Intersecting Meta-Stories (True Branches)
Picking up help of allies, hero almost sacrifices all to fight/defeat antagonists
GOOD
AMB
BAD
See other side of antagonists.
Help "antagonists" at expense of former allies

Figure 5. *Typical evolution of a dialog graph as traditional writers are asked to introduce interactivity, nonlinear gameplay, and conflict aesthetics.*

climax, and drama, they tended to do so by placing the human on rails in a passive listening noninteractive role as in Figure 5a. Even stories with significant reversals are F5 invariably linear in their dialog graphs (e.g., Minority Report, Lord of the Rings, etc.). After pushing writers for interactivity, their stories begin to look like those of Figure 5b, where there is only superficial interactivity in the sense that the plot is linear and the player can choose which of about three possible phrasings to reply with for each situation. In this type of interactivity, the player has no sense of being in control, of being able to alter the outcomes, or of thinking there is a reason to play again. The Monkey Island series of dialog adventure games has pushed this approach fairly successfully, however. The alternative is a dialog graph like that of Figure 5c, where there are jumps to different storylines, truly different outcomes are possible, and proceeding down one of these paths eventually closes off options for returning to other outcomes. This corresponds to games like the successful Black & White or Grand Theft Auto, where players may choose to explore "good fun" or "bad fun," but not both in the same life.

Another way to think of the ideas of Figure 5 is that there are several central aesthetics and mechanics that should emerge from various deepenings of the game design. We mention these briefly here. Silverman, Holmes, et al. (2003a) discusses more fully. For one thing, the game designer must try to preserve the overall entertainment score (ENT) which is a function of the (F) or fun quotient that derives heavily from four contributing sub-processes, at a minimum, including: (R) rules of the game that are satisfying such as how to move around, interact, fight, persuade, and so on; (C) control remains in the hands of the player in the sense of creating his own story and selecting tactics to deal with dilemmas along the way; (O) fairness of outcomes along the way and in the end (did player's choices have believable consequences?)—social contract between designer and player; and (AT) accumulate-threaten aesthetic or reward-punishment—the game provides the player with an opportunity to collect things, threatens the player with their loss (scare/thrill), allows the player to try and protect them, and so on. These four mechanics are central to almost any game, and they must be appealing if the game is to be fun for the player. However, they also must be tuned to the class of player (demographics) targeted for the specific game.

As already mentioned, to try and encourage nonlinear dialog and story graphs among our writers, the lead author started the Heart Sense project with a paper-based version. This involved five notecards, one each for Act 1 and Act 3, and three more for the various quests or scenes of Act 2. The front of each notecard involved options for playing the hero as a "good guy" who saves all heart victims, rescues the kidnappee, helps and is accepted by the townfolk, and then as a result of all this wins the romantic affection of the leading lady and a job as the apprentice to the town doctor. The flip side of the cards is for driving your avatar into the anti-hero role—that of ignoring the heart attack victims in the effort to do things needed to win the respect of and a job in the organization of the antagonists. The player gains a major role in the antagonists' organization, but at the expense that the townfolk disdain you, the romantic interest shuns you, and the kidnappee believes you abetted his abductors. Conflict involves true tradeoffs—something must be sacrificed for gains in another dimension. This is entertaining and it gives the player real choices to make in the drama, not just phrasings of verbiage. With this as background the authors of

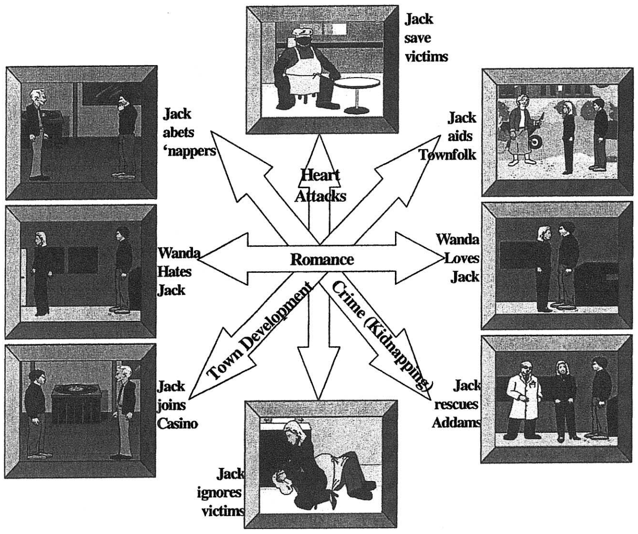


Figure 6. *Overview of the conflicts used to introduce gameplay into the script.*

this paper worked with the writers to try and get them to shift their approach to a nonlinear script writing effort.

F6 Figure 6 shows some of the conflicts that were adopted and integrated by the end of this process. As shown there, four major conflicting sets of goals were woven into the three act play. The result is a dialog graph that looks like Figure 5c, and a large number of true decisions for the player to have to make. There are three completely different endings, and one can only reach a given ending by playing toward that goal consistently. As mentioned above, these conflicts involving choosing either (1) to side with the protagonists, victims, and townfolk and thereby winning a lot of prizes including a job offer from the rescued kidnapee, a romance with Wanda, and the admiration of the townfolk; or (2) to side with the antagonists on the left side of Figure 6 in an effort to help their cause and thereby to gain a nice position in their organization and enjoy the "bad fun" of frustrating the plans of the townfolk. The left side of the figure also shows that one loses Wanda's affection in the process.

After the writers and training developers worked with these ideas for a while, it became apparent that parallel story outcomes needed substance if they were to intrigue the player and offer tempting alternate goals to strive for. Players need a reason to be "bad." That is, the player needs an alternate but legitimate set of goals to try and achieve—another story that might make as much sense and be as much fun as the "good" path (helping to heal everyone in town). This had to be a legitimate alternate story and set of goals such that the player could only pursue it adequately by being brusque with the various health victims. In the end we found three strong storylines, and got the writers and training content specialists to go along with them so the players could have a sense of controlling the outcome. Also, in each storyline there were contagonists modeling proper cues and providing feedback, rewards (things to collect, such as career options, family relationships, etc.) and antagonists meting out punishments or threatening things that might be lost. For each of these three main storylines, the lead author tasked the writers to create several dialog strategies as for the "good" storyline alone.

In general, the writers were uncomfortable in authoring their three story versions on anything other than a word processor, even though the game would be unimplementable in that format. Not wanting to destroy their creative process, we supported their effort, and then had a secretary move the dialog to the graph editing tool after the work was finalized. The authors and content experts then verified the results both via printouts and via play testing. The end results included about 100 pages of script, which translates into 346 state nodes, 480 edges, 691 dialog acts, and 1346 gesture commands invoking 461 unique gesture macros. Overall, this authoring effort required about one person-year broken down in round numbers as 500 person-hours from the dialog writers to author the script, 80 person-hours for the secretary to enter the script into the trees, 400 combined person-hours from the two graduate research assistants to add choreography (action macros) to the dialog graphs, and perhaps 6 person-months equivalent across all the faculty co-investigators (content developers, story critics, and play testers). Based on these results we have begun efforts to design more intuitive interfaces for directly eliciting nonlinear plots, conflict aesthetics, and other aspects of gameplay for interactive fiction and drama.

One successful aspect of our approach has been its robustness, or ability to provide support across diverse workers in our high turnover environment. That is, the development of Heart Sense relied heavily on student labor for writing, artwork, and some programming, and consequently we could not afford to use a system that required a long period of training. Furthermore, since these students tended to work at home or during scattered hours, there was little opportunity for direct communication outside of scheduled meetings. The clear separation between the various components of the workflow for this project, however, contributed substantially to the feasibility of creating a finished product. Thanks to the widespread support for XML in our programming environments, we were able to freely exchange data between components of our authoring package. The majority of the people who contributed to this project knew little about the pieces that were not their own. And in many cases, they were able to use tools they were already familiar with to produce their portion of the game's content.

5 Discussion of Results and Next Steps

Our research up to this point has revealed some surprising facts. First, there are no environments one can turn to for rapid authoring of pedagogically oriented interactive drama games. While games from other genres are beginning to arrive packaged with sophisticated editing tools, the educational gaming community generally is forced to create non-modifiable games on a per-subject, per-audience basis. There exists a growing number of tried and true guidelines for creating fun games. There exists a huge body of work on the subject of effective methods of education (e.g., Gibbons & Fairweather, 1998; Reigeluth, 1999). And narrative has its own effectivity metrics. But, at present, most games are designed from the start with entertainment as the primary goal, with any learning on the part of the player as a beneficial side-effect. Pedagogical games, on the other hand, begin with rigid learning objectives that must be satisfied, which place severe constraints on the design of the game. This tension has created a deep gap between the creators of educational games and the creators of entertainment games, and consequently little mutual benefit is generated from work in either community.

We believe that the solution to this problem lies in the creation of a system that provides the building blocks of interactive storytelling by implementing the inner workings of a variety of gaming devices as composable parts, with their actual arrangement and content determined by the educators. Dialog, character movement, puzzle manipulation, resource models, combat, and other mechanics would be weaved generically into a unified game engine, with the educators able to simply choose which ones suit the story, pedagogical goals of the game, and the needs of the target audience. While a game like Heart Sense is inherently dialog-oriented with its focus on persuasion and interpersonal relationships, our Black Hawk Down recreation in Unreal Tournament is a first person shooter training game with autonomous agents and emergent crowd behaviors (Silverman, O'Brien & Cornwell, 2003). This has caused us to think more broadly about the range of games we might have to create. Such a unified engine is becoming an increasingly realistic possibility, with many recent games beginning to blend elements from a variety of others, causing genres, and more importantly game engines, to converge. Given an environment such as this to work within, designers can harness the state of the art in the technical aspects of interactive storytelling while staying focused on content creation.

In terms of the three goals stated at the outset of this paper, there has been some forward progress, and with it has come the realization of new challenges yet to conquer.

5.1 Goal 1: Support the Authoring of Interactive Pedagogical Dramas

Thus far, our animated stories have been constructed initially in a word processor to permit creativity to flourish and then, once they are stable, they are converted to FSMs that can be executed by the game engine. We believed the FSM representation would provide an ideal middle ground between writers, programmers, and the game engine given their unambiguous and relatively easy to follow graphical representations, but our writers were in general uncomfortable authoring directly into FSMs. The FSM representation was a vital step, but it is probably not the final resting spot for our AESOP generator. We encountered two major difficulties: the writing effort before using the graph and the choreographic load once the dialog was inserted into the graph. In terms of the first of these, the creative effort required of the writers seems substantially greater than for writing linear stories, or even three linear stories at once. Our ideal system should require no more creative effort than a linear story, but should draw this baseline story out of the writer in such a way as to allow for many degrees of interactivity.

In the end, the FSM approach had no noticeable impact on reducing the load for the writers. They stayed away from it, and only after we locked in the script and dialog did we then convert the story to FSM form. Our future research is now aimed at finding tools that might actually ease the writers' burden. We are looking for a way to bridge the gap between author and machine that allows the author to describe the essence of the story in such a way that the machine can search for ways to provide interactivity. A simple example of a step in the right direction is to eliminate the unnecessary ordering constraints that an FSM imposes. For instance, in our current system, character A may ask B a question, and upon getting an answer ask C an unrelated question. While possible to allow the user to ask these questions in either order, state explosion becomes a problem quickly, leading to our writers basically not bothering about order anymore. By adopting a less strictly specified design, we can avoid this problem entirely. This would allow natural opportunities for interactivity, while simultaneously lending itself to adaptation from a linear script. Authors need only ask themselves which lines in their original story *must* come before which others, and the final ordering is deferred to the user at runtime.

Another shortcoming of the authoring approach we used was that, as mentioned above, it was nontrivial to find ways to make our story interactive. Testers complained that their choices were superficial in our early attempts at providing options, while the writers complained that they would be unable to have the story make sense if users were given more flexibility. It was not until we looked at the structure of the story at a high level that we were able to find a manageable set of ways in which it could play out differently. We believe the solution to this dilemma lies very early in the authoring process, before any dialog is written or scenes are envisioned. Authors begin the writing process with a vision of the conflicts the story will present. Each possible outcome of a given conflict provides a different direction in which the story can turn. We hypothesize that the types of choices people find meaningful will be precisely those that have impact on the outcome of these conflicts, and all other sources of interactivity can be considered superficial. Specifically, if we can elicit a high level description of the story from the author in terms of conflicts and their possible outcomes, we can turn over control of how these conflicts are resolved to the player. The gaming world is replete with conflict resolution mechanisms; at the heart of every game is a system for determining whether the player is winning or losing. It is also no coincidence that after particularly vigorous gaming sessions, players are often eager to share what happened with others.

Different types of gameplay may be used to resolve different conflicts. For example, if the story calls for the player to attempt a hostage rescue, the game can switch to a scenario constructed in a first person shooter environment. When this scenario finishes, the game can examine the state in which it ended and present the resulting conflict. In its simplest form this approach amounts to the level system used in virtually every game in existence, where if the player completes the level he goes on to the next, and if not, the game ends. Our challenge is to devise an authoring environment that allows for seamless transitions between levels, where the story reacts to whatever the player does.

One conflict resolution method that is noticeably lacking from the repertoire of the gaming industry is persuasion. Dialog-oriented games have notorious limitations that have led to their near extinction. Yet persuasion is a method of conflict resolution that can be just as critical to the advancement of a plot as combat. We are interested in exploring the possibility of borrowing aspects from combat systems created by the gaming community to model certain forms of verbal conflict. For example, where other combat models use such concepts as ballistics and material densities, this model could turn to the rhetoric model introduced by Aristotle. Beliefs, arguments, and counterarguments would be assigned ratings for ethos, the credibility of the source; pathos, the emotional content of the message; and logos, the logical content of the message. Under such a system, arguments "damage" beliefs, and the participant whose beliefs remain intact longest wins the argument. This creates a system that parallels the mathematical conflict resolution models typically associated with guns and targets in games. If written carefully, conflicts structured in this manner can play out differently each time a game is played. Furthermore, a system such as this allows for the same work on the part of writers to be used repeatedly in very different scenarios.

The second obstacle mentioned above concerned the choreographic workload. Adding in each hand movement and head nod is both tedious and hopefully unnecessary. What is interesting to observe, looking at the node counts, is that it was really only about twice as many behaviors (gesture commands) to fill in compared to the lines of dialog, but it took substantially longer than twice the amount of time to get done. While the dialog was laid out in sufficient detail that getting it into the editor was only a matter of data entry (2 weeks), for gestures, we had to think about what was appropriate when and how to adapt stage instructions that were not quite feasible. Getting the gestures right really requires working with the game itself to make sure things look like they are expected to look. An interesting question is whether there might be a way to short circuit this type of effort, particularly as the digital cast is reused from game to game, and as we gain more experience with types of gesture sequences that go with high level behaviors and conversations. Given the conflict system metaphor mentioned above, it seems highly likely that we could coordinate certain gestures with how the character's favored position is faring. We might also look at high level markup of the dialog, such as "Jack is concerned as he is saying this," with the system matching up a list of "concerned" gestures to find an animation to play. Certainly there are many such ideas that would be worth looking into.

5.2 Goal 2: Insulate Authors from Engine

Initially, we thought the scene-batch mode would be a useful approach for the HSG team, and that we would benefit from insulating the team from the engine. Indeed, except for its engine module, the AESOP generator is relatively independent of any given commercial implementation, and that did prove to be a benefit to our authors. One can safely author the game in the FSM trees and assume that the XML interface will convert the results into the syntax and instructions needed by the respective game engine. However, the engine wrapping side of our AESOP effort was not a small activity. It required about 0.5 person-year of the third author's time to wrap the Director engine, and another 0.3 person-year of a separate programmer's time to wrap Unreal Tournament. Very little was reused between those two efforts. A second issue is that the scene-batch mode of authoring is one of the obstacles mentioned above under the choreographic load. That is, at present, the authors must insert a gesture command (behavior) into the FSM tree and then play the game to see how it looks. To improve this batch process, at present we are eliminating the distinctions between the engine and the various editors. The ideal we are currently gravitating toward is that content developers and writers can directly manipulate sets and characters within the engine and edit positioning, gestures, dialog, and player choices in the context of each scene, beat, and dialog string. While there are a number of unsolved obstacles to doing this while preserving engine independence, this interpretive mode should be more gratifying to the developers, making the process less of a chore and more like a direct beat-manipulation interface where they can observe a portion they do not like or have not finished, back up, edit it, and replay the beat until they get it how they want it. However, there will still be the need for popup windows that are the batch mode viewers, since this method supports bigger picture viewing/manipulating of what is being authored for a scene, act, or story.

5.3 Goal 3: Reusable Cast

Another sizable challenge is the need for highly composable systems that allow interactive dramas and scenarios to be generated on demand and just-in-time for the purpose of story sharing. This is the "Holodeck" dream, which begs a flotilla of research and development priorities, only some of which have been addressed in this paper. We realize that we have only just begun to move down this path with our current cast of ASEOP characters, behaviors, and lessons learned.

Acknowledgments

We gratefully acknowledge the financial support of the National Heart Attack Alert Program of NIH, and the National Library of Medicine PO # 467-MZ-902060. Also appreciated is the help of the case design team members (Drs. Holmes, Kimmel, Green, Holmes, and Potok, as well as T. Banden, T. Oksman, K. O'Brien, and J. Cornwell), and numerous student helpers.

References

Campbell, J. (1973). *The hero with a thousand faces.* Princeton, NJ: Bollingen Series/Princeton University Press.

Decker, D. (2002). Anatomy of a screenplay. *First Annual Theory of Story Conference (Storycon).* Available at www.anatomyofascreenplay.com/book.htm.

Filipczak, B. (1997). Training gets Doomed. *Training, 34*(8): 24–31.

Gibbons A. S., & Fairweather, P. G. (1998). *Computer based instruction.* Englewood Cliffs, NJ: Educational Technology.

Green, G. S., & Bavelier, D. (2003). Action video game modifies visual selective attention. *Nature, 423,* 534–537.

Green, M., & Brock, T. C. (2000). The role of transportation in the persuasiveness of public narratives. *Journal of Personality and Social Psychology, 79*(5), 701–721.

Hayes-Roth, B., & Rousseau, D. (1997). *Improvisational synthetic actors with flexible personalities* (Report KSL-97-10). Stanford, CA: Stanford Knowledge Systems Laboratory.

Hopson, J. (2001). Behavioral game design. *Gamasutra.* Available at www.gamasutra.com.

Lasseter, J. (1987). Principles of animation applied to computer animation. *Computer Graphics, 21*(4), 35–44.

LeBlanc, M. (2002). Game design. Workshop presented at *Game Developers Conference.*

Marsella, S., Johnson, W. L., & LaBore, C. (2000). Interactive pedagogical drama. *Fourth International Conference on Autonomous Agents,* 301–308.

Mateas, M. (1997). *An Oz-centric review of interactive drama and believable agents.* (Technical Report CMU-CS-97-156). Pittsburgh, PA: Carnegie Mellon University.

Petty, R. E., & Cacioppo, J. T. (1986). *Communication and persuasion.* New York: Springer-Verlag.

Propp, V. (1968). *Morphology of the folktale.* Austin: University of Texas Press.

Reigeluth, C. M. (Ed.). (1999). *Instructional design theories and models.* Mahwah, NJ: Erlbaum.

Silverman, B. G., Holmes, J., Kimmel, S., Branes, C., Inns, D. Weaver, R., et al. (2001). Modeling emotion and behavior in animated personas to facilitate human behavior change: The case of the Heart Sense game. *Health Care Management Science, 4*(3), 213–228.

Silverman, B. G., Holmes, W., Green, M., Holmes, J. Kimmel, S., Mosley, J., et al. (2003). *Instruments for exploring the training and aesthetic dimensions of edutainment: Case of the Heart Sense Game.* (Technical Report 467-MZ-902060). Washington, DC: National Library of Medicine/NIH. Available at www.acasa.upenn.edu/heartsense/Instr-TechRpt.htm.

Silverman, B. G., Johns, M., Weaver, R. (2003). Satisfying the perceived need for free-play in pedagogically oriented interactive dramas. *The 16th IEEE International Conference on Computer Animation and Social* Agents (CASA 2003), 161–167.

Silverman, B. G., Johns, M., Weaver, R., O'Brien, K., Silverman, R., Cornwell, J. (2002). Using human behavior models to improve the realism of synthetic agents. *Cognitive Science Quarterly, 2*(3/4), 273–301.

Silverman, B. G., O'Brien, K., & Cornwell, J. (2003). *Scenario creation and runtime editing* [Videoclip of authoring scenarios in Unreal Tournament with AESOP, and with

PMFserv embedded as the mind of each gamebot.] Available at www.seas.upenn.edu/~barryg/unreal/demo6_small.avi.

Swartout, W., Hill, R., Gratch, J., Johnson, W. L., Kyriakakis, C., & LaBore, R. (2001). Toward the holodeck: Integrating graphics, sound, character and story. *The Fifth International Conference on Autonomous Agents,* 409–416.

Toth, J., Graham, N., Van Lent, M., Alinden, R., Silverman, B., Cornwell, J., et al. (2003). Leveraging gaming in DOD modeling and simulation: Integrating performance and behavior moderator functions into a general cognitive architecture of playing and non-playing characters. *12 the Conference on Behavior Representation in Modeling and Simulation (BRIMS, formerly CGF).*

AUTHOR QUERIES

AUTHOR PLEASE ANSWER ALL QUERIES **1**