Metalinear Story Agents - an Exploration in Construction and Delivery Interface

By

Yu Chen

Submitted to the Department of Electrical Engineering and Computer Science in Partial Fulfillment of the Requirements for the Degree of Master of Engineering in Electrical Engineering and Computer Science

at the Massachusetts Institute of Technology

May 23, 2001

Author____

Department of Electrical Engineering and Computer Science
May 23, 2001

Certified
by____

Glorianna Davenport
Thesis Supervisor

Accepted
by____

Arthur C. Smith
Chairman, Department Committee on Graduate Theses

Metalinear Story Agents - an Exploration in Construction and Delivery Interface

by

Yu Chen

Submitted to the
Department of Electrical Engineering and Computer Science

May 23, 2001

In Partial Fulfillment of the Requirements for the Degree of
Master of Engineering in Electrical Engineering and Computer Science

Abstract

Our concepts of story and storytelling have popularly been represented through linear mediums such as books and films. Only in traditions such as theater and oral storytelling, where an author has direct, instant feedback from the audience, has story been able to achieve a more malleable form. The present is pivotal because the evolution of interactive media technology has created new forms of digital expression which enable more reactive, computational ways of constructing, arranging, and presenting stories. Into this context, Kevin Brooks' 1999 Ph.D. work created Agent Stories, a metalinear narrative authoring software tool which employs a software agent-driven engine to produce one or many possible linear accounts, thus creating multi-linear stories which are especially well suited for our new interactive mediums. Java has further extended this system onto the widely distributed environment of the Internet, and the possibilities for collaboration open new dimensions to the art of computational story writing and storytelling.

Thesis Supervisor : Glorianna Davenport
Title : Principal Research Associate, Director of Interactive Cinema

Table of Contents

List of Figures

1 Background

1.1 Evolution of Cinema

The craft of storytelling has a long and continuous history, from older forms of oral narratives and written texts to newer media styles such as movies and television. At each step along the way, distinct milestones exist at points where new technological inventions such as the Guttenberg's movable type or motion pictures have jolted the established model of narration and fostered a branching in the ways that information has been brought to its audience.

There exists an ever evolving process where a structure or mode of thinking employed for telling stories has been established in its general form, then the advancement of technology creates an entirely new medium of expression, wherefore causing a period of experimentation and contemplation as artists explore this new system of composition. Janet Murray categorizes the present as "the incunabular days of the narrative computer" (p29), having once again reached this threshold when our assumptions of the form and style of storytelling will again be jostled by the powerful capabilities of the computer.

Cinema is standing exactly at this point right now, and the phrase "Interactive Cinema" seeks to capture that sense of evolution, when this cinematic medium will be pushed by the presence and involvement of the computer to forms that was unrealizable before.

> Interactive Cinema reflects the longing of cinema to become something new, something more complex and personal, as if in conversation with an audience (http://ic.www.media.mit.edu)

Cinema was born from the invention of Edison's Kinetoscope in 1891, and has endured a series of changes, from the creation of synchronous sound leading to The Jazz Singer in 1927 to the perfection of the color developing process embodied by Technicolor in 1932. The "talkie" and the subsequent appearance of color amazed its audiences and simulated reality to greater extents and increasing the immersive nature of the movie watching experience.

For a time, the Hollywood system of filming had established most people's perception of how movies are shot and how they should be shot. Everyone knew what to expect when they went to the theaters, and what to expect when they stayed home to watch TV. The way of filming also created an expectation of narrative styles and topics. The next evolution in cinema occurred when technology reduced the size and weight of older clumsy cameras, in addition to improving sound recording technology, enabling filmmakers to follow their action and take the space of narrative out of a studio.

The documentary stepped away from the generally accepted structure of fictional and personal narrative, establishing more fluidity and freedom in camera work, as well as taking a film to a heightened level of intimacy with its subject that was never before possible. This new mode of production created a new narrative space that is not only more personal and more flexible, but also prefigured the trend of making film more available to the audience, so that the audience can take a more active role in the experience of watching as well as making movies.

Accessibility of the narrative experience differs in its three available forms. The broadcast medium, such as radio, is limited to just one way communication, where the audience has no way of directly affecting the information being transmitted. The television/VCR format gives an end user access to materials and the power to record and manipulate what they see through the ease of reediting and transference. This tool provides the user with the ability to actively take a role in manipulating their own experience, though others may argue that the television format creates a much more socially isolated situation because it more people tend to watch TV alone at home, in a very unparticipatory fashion.

The third medium of storytelling is enabled by the computer and the Internet. A distributed story format over the web increases individual access to information as well as interactivity with an inherently nonlinear presentation. The structure of the web is based on keyword association, an idea first articulated in 1945 by Vannevar Bush in his article "As We May Think". His idea of the Memex machine is based on a system of associational indexing, which lead to the notion of hypertext, a format with many entry points and branches but no clear ending, which is quite antithetical of the linear story format embodied by linear mediums such as books or films.

Computers have also changed the presentational format of multimedia elements. Audio and video have become malleable domains where they are not only integral contents of the presentation but also act as expressive, dynamic pieces of information. They don't necessarily have to exist as part of a coherent linear structure, but can instead stand alone as customizable elements within a multiform story.

1.2 Past Research

One of the first nonlinear projects was the Aspen Movie Map, produced by the Architecture Machine Group in the late 70's. The streets in the town of Aspen, Colorado, as well as a few other miscellaneous scenes were filmed and printed onto optical videodisks. CAVS Videodiscs were an optical storage medium that gave a computational system nonlinear access to 54000 still frames or 30 minutes of video. A user can navigate the streets of Aspen using the touch screen or a joystick. The continuous sensation of navigation was intuitive and immersive, and illustrated a practically seamless presentation.

In the Interactive Cinema group, the New Orleans Interactive project was the first large scale documentary in which scenes were addressed and presented in sequence based on user query and a story model. A documentary shot by Glorianna Davenport and Richard Leacock, *New Orleans: A City in Transition* 1982-1986 portrays the evolution of the city before, during and after 1984 World Exposition. Like Aspen, this complex, multi-view-point documentary used multiple CAV videodiscs – six in all – as a storage and access medium. The project began to define computer aided content presentation as an important research area; for many years the research explored the merits of various approaches to content annotation; this annotation must provide the machine with sufficient information about each clip to be able to coherently arrange them as a narrative. This exploration was compatible with the idea of producing content over time and by 1995 had matured into the idea of "evolving documentary" as a new content form.

The storytelling system built for New Orleans was able to select and order sequences based on active key words and simple heuristics (such as select a next sequences from scenes that start with a higher frame number and or disc number). The system did not attempt to provide a system for selecting shots and creating sequences. This work, that almost by definition, required an emphasis on a more traditional approach which used close up, medium and wide shots, as well as variations in POV.

In 1993, Ryan Evans MS94 developed an annotation and filter system called LogBoy and FilterGirl as part of his thesis work. This set of complementary tools was created for authoring multivariant stories. LogBoy provides graphical annotation of content clips, while FilterGirl is a selection algorithm that acts as a filter to create a multithreaded narrative progression using the graphical annotation provided by LogBoy. The project illustrates the use of a narrative engine, essentially a software editor, to create scenes. This system was problematic in that the database needed to be richly populated in order for the filter mechanism to work reliably.

Gilberte Houbart's 94 masters thesis called "It was a Knowledge War" represented a movie system that supports multiple points of view about a central topic, specifiable by the user. In this case, the Gulf War was a case where the main source of information for journalists was the government itself, so the different journalists interviewed for this project have interesting perspectives on the same story. This system explores machine knowledge of media content as well as organization of a content database, but it was not very responsive to user interaction because it provided only very limited directional variables for the playback of content.

The next step in evolving documentary construction was taken in Michael Murtaugh's 96 thesis on Automatist Storytelling Systems. He created two versions called Contour and Dexter, two similar narrative engines. Contour, written in C++, used a spreading activation approach to selection. By accumulating and degrading weightings over time, the system picked the next most likely clip. This strategy insured that the next selection would either be thematically like the last or purposefully different depending on whether the weightings were being applied as positive or negative weights. Designed in 1995 to run on the WWW, Dexter emulated the links and trajectories that could occur using a spreading activation approach. Implemented in Java, Dexter could precalculate the effect that adding any particular clip would have on the graphical user interface and render the required graphical elements (hence the name “evolving documentary”).

Boston: Renewed Vistas, a content application, informed the early designs. In the video material, we discover multifaceted communities and perspectives as people around the Boston cope with the enormous Big Dig project that is rearranging the layout of the city. Each clip in the database is associated with metadata of possibly many keywords. Many different clips share the same keywords and a keyword is associated with multiple clips. The decentralized character of the Spreading Activation Network allows the system to assign each clip a certain activation energy, and either raises or decreases that energy level depending on the user's viewing history. When one clip is selected to play, all other clips check if they are related to the currently activated clip, and adjust their energy level depending on their relation to that clip. The higher an activation energy is, the more likely a clip will be the next selected for playout.

In this project as it runs in Contour, the state of the visual representation informs the user as it reflects the database. Each clip is sized and placed according to its activation level, with more related clips receiving more prominent display. The interface consists of a collage of frames, each representing an available clip, and also an outer border consisting of all available keywords. The user can select either a video or up to multiple keywords as parameters for the engine. Not only does this system provide a continuity of story and a visually fluid presentation, it also allows for increased audience participation in the story, allowing the audience to join in the creation of what they are viewing and not remain mere receivers of information.

The web based incarnation of this system, Dexter, was used to create the applet and webpage "Jerome B. Wiesner: A Random Walk through the 20th Century". Though using the same engine, this version was less visually dynamic because it employed a grid-like Java moviemap that only possessed four possible activation states, and a separate window for content presentation, whereas Boston integrated the content into their visual interface. In the first attempt to widely distribute this form of narrative on the web, many tradeoffs were made to facilitate content delivery but compromised its original design.

As the notion of an evolving documentary met the abilities of the computer and reached a large distributed audience space, MS96 student Lee Morgenroth created the "Lurker" system which can be categorized as a "thinkie". His assumption of audience interaction within a large information space was that they needed to be given headspace to adjust to this new interactive environment. As a result, the main point of online interaction shifts from merely navigating a gamespace to establishing a network or cooperation among its participants in which they establish relationships and form a community to work together to solve presented to the participants. This system changed they perception of storytelling in a widely participatory environment, by trying to fully engage an active segment of the audience.

2 Kevin Brooks' Agent Stories

2.1 Intentions

In his '99 Ph.D. thesis, Kevin Brooks detailed a number of characteristics which he viewed were feasible given our present ability to combine computation and media presentations in the context of interactive environments such as the internet or interactive TV. He saw that the combination of these new possibilities created an imminent form of narrative structure that he then dubbed "metalinear narratives". Even given modern interactive sources of entertainment such as CD-ROM stories such as "Myst" or videogames, he felt that they were potentially promising, but still remained relatively static because all plot paths and branching nodes were fixed or limited by methods such as branching algorithms and knowledge based AI. Those new forms of interactive entertainment still lacked the malleable and audience-customizable structures provided by oral storytelling traditions and the theater, where the author or actors have direct, instant feedback from the audience and may modify the story presentation on the spot.

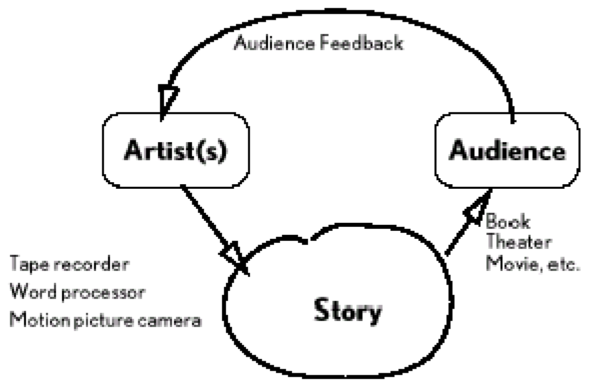


Figure 1 : Simple artist-story-audience structure, with feedback (Brooks 61)

In their place, Brooks proposed the “metalinear” form, consisting of a “collection of different linear stories from different points of view, with the aid of a story engine which sequences the story pieces” (Brooks 19). This new structure resides more satisfactorily in a more globally connected environment, where the multiple contexts exist simultaneously in overlapping digital spaces. Because linear stories typically cut out different points of view to preserve coherence of intention and theme, usually for a specific audience and in a fixed format, metalinear stories strive to break the old modes of thinking and include multiple first person points of view and granules of story stored in a web-like interconnected structure for increased variability.

Because this new form of narrative structure is different from those employed in more traditional storytelling mediums, no established authoring method or environment exists either. Ideally, the system would suggest interactivity to multilinear stories. To facilitate ease of use and overall comprehension, it would integrate both thinking space and writing space in the same area. Metalinear stories themselves delegate the determinism of narrative sequence to both the author and the audience characteristics, which means that the lines between creation and presentation are no longer clearly defined. In that sense, the authoring environment reflects those sensibilities as a constant reminder to the writers in their process.

2.2 The System

2.2.1 Theory

As part of the research on metalinear stories, Brooks created an authoring tool called Agent Stories. He followed the precept that a tool is “only as intelligent as the user is with the tool in their hands. Therefore, the tool’s empowerment of the user is of great importance" (Brooks 92). To create an adaptable tool infused with understanding of the writer's intentions, the engine should be able to knowledgeably weave together an array of character accounts and viewpoints from the manner with which the writer linked story granules.

As part of Brooks' thesis argument states, old habits of story writing are hard to break. Authors who have been accustomed to old habits of creating linear narratives would find it difficult to conceive of vibrant and coherent plots which sufficiently fit into multilinear frameworks. Given the belief that a story can contain multiple parallel streams of action, multiple points of view, many relations between the story elements, and not always the same outcome, a storyteller needs to juggle many variables in mind, and also try to appropriate the structure and content of the narrative to a specific audience. This significant task can become unmanageable even for writers accustomed to this environment. The writer has become an architect in need of a sophisticated tool with which she can comfortably navigate through a potential maze of nodes and connections.

As a logical result, the computer would be able to provide assistance to the writer in their creative process by meaningfully keeping track of story elements and also giving the author creative feedback, as well as keeping track of the author's intentions so as to better structure and personalize responses to and presentational formats for the author's preferences. The computer can fill the role of an intelligent tool that is dynamically adaptive and highly personalizable to the user.

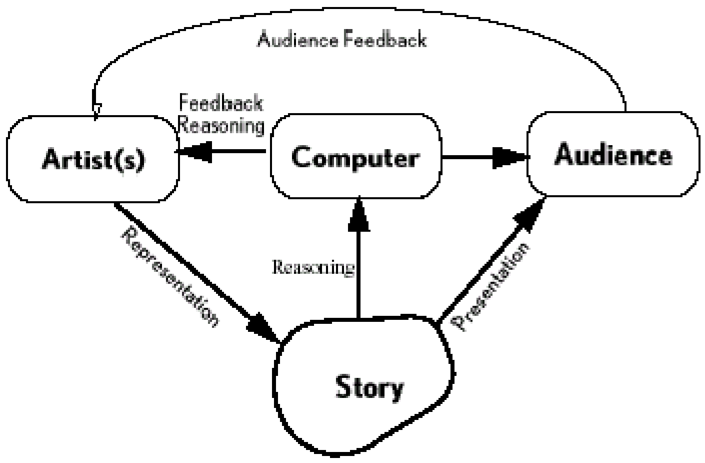


Figure 2 : Computer Assistance in the simple artist-story-audience structure. (Brooks 63)

Agent Stories was created in the wake of this new predicament. As the writer's task multiplies in difficulty and complexity, the writer's tool correspondingly increases in intelligence and autonomy in response to this need. Agent Stories is a software tool featuring multiple environments that aid in the various facets of the story writing process. It not only records the contents of story elements, but also stores the states and characteristics of those elements, as well as writer's intentions for organizing them along a certain framework structure. Given a set of clips and a structure of arrangement, a story engine will be able to select a final linear version of the story with the help of a story agent. The writer can specify desired characteristics to all parts of the system, and as a result can variably control and customize the results.

Software agents are integral in driving the story engine to create variable and personalizable story suggestions. Agent behaviors can be categorized into two main approaches, the more traditional knowledge-based rules of action, and behavior-based rules of action, which have been well contrasted by professor Pattie Maes of the Media

Lab. When applied to the context of narrative structure, a knowledge-based approach essentially is not concerned with adapting over time to developmental aspects of a particular problem, but instead can become very specialized in solving one problem at a time, given that the problem remains unchanged. On the other hand, a behavior-based approach espouses integration of multiple rules of action, and such an agent would be more situated in their environment, and thus can more effectively detect the behaviors of the system and not just its knowledge.

> Metalinear narrative research employs BBAI through the use of software agents. The software agents are less brittle and more adaptive to a dynamic narrative representation environment than a KBAI approach would be. Using software agents, story domains can grow or change, while the agent remains the same. (Brooks 67)

According to the principles of these software agents, the story engine of Agent Stories resides as an integral part of the whole system, and adapts itself to different story contents and structures.

Agent Stories' narrative engine is never explicitly represented in a particular environment. Even though the roles of engine and agent are inextricably linked, they nevertheless reside in different spaces within the system. Agents are visible entities, and have been accorded their own scripting environment. Engines will only run when called, and even then within the background. Brooks elaborates the notion of a story engine's role and characteristics.

> [T]he term story engine is used to describe a set of software algorithms designed to make decisions regarding how a computer-based story should proceed. That is, the story engine decides what's next in the story, embodying some of a human author's reasoning for doing the same task. Story engines are construction engines, deciding the sequence of each small detail, major event, and opposing or supporting position of the story. (72)

2.2.2 Structure

Agent Stories contains five environments presented on separate, exclusively visible screens, each of which represents a phase in the authoring process. The first is the Structural Environment, in which the structure of the narrative is sequenced using building blocks. These individual components are primitives called narrative classes, and there are seven : {Speaker Introduction, Character Introduction, Conflict, Negotiation, Diversion, Resolution, Ending}. The resulting sequence is called the Narrative Framework. The framework comprises the essential backbone of our story, and multiple stories may be created from a single collection of story elements by laying them in different order onto the framework.

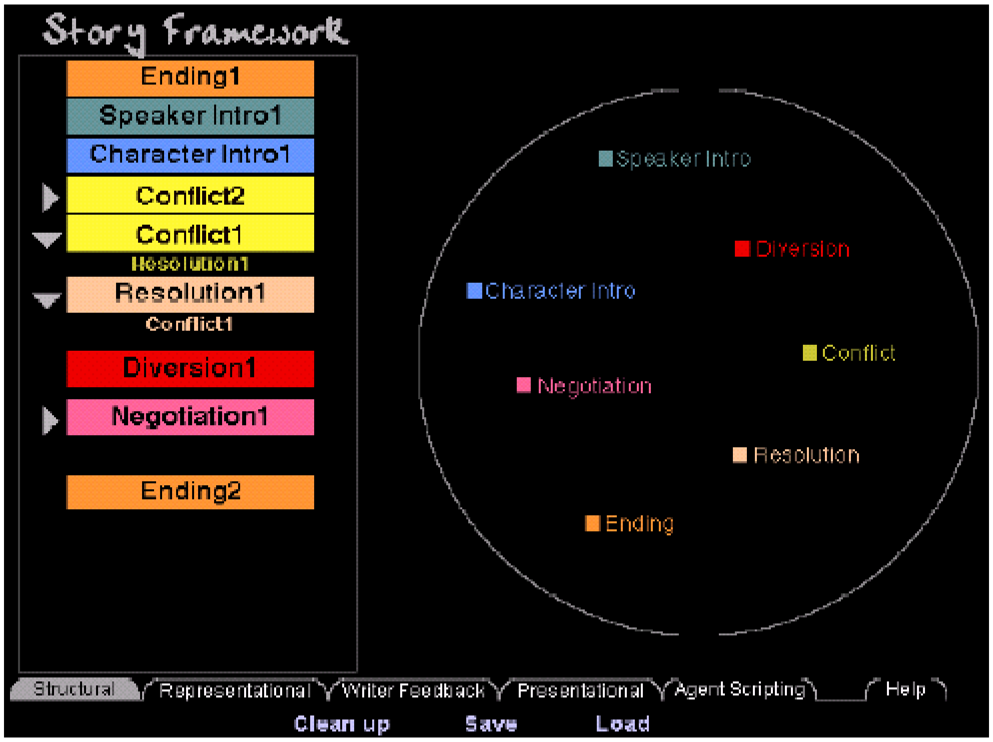


Figure 3 : A screen shot of the Structural Environment with sample story framework (Brooks 102)

Environment number two is the Representational Environment. Story granules, otherwise known as story clips, are contained here along with relationships between the clips. Each story clip, asides from being labeled with a narrative class, is also given a name, an associated point of view, text content, and a type. Clip relationships strongly affect the underlying shape of a story, which forms an interconnected web. These associations take on the form of links : {Supporting, Opposing, Conflict-Resolution, Factual Precede, Causal Precede, Must Include}. A web of linkages shapes the personality and potential of the linear stories to be constructed.

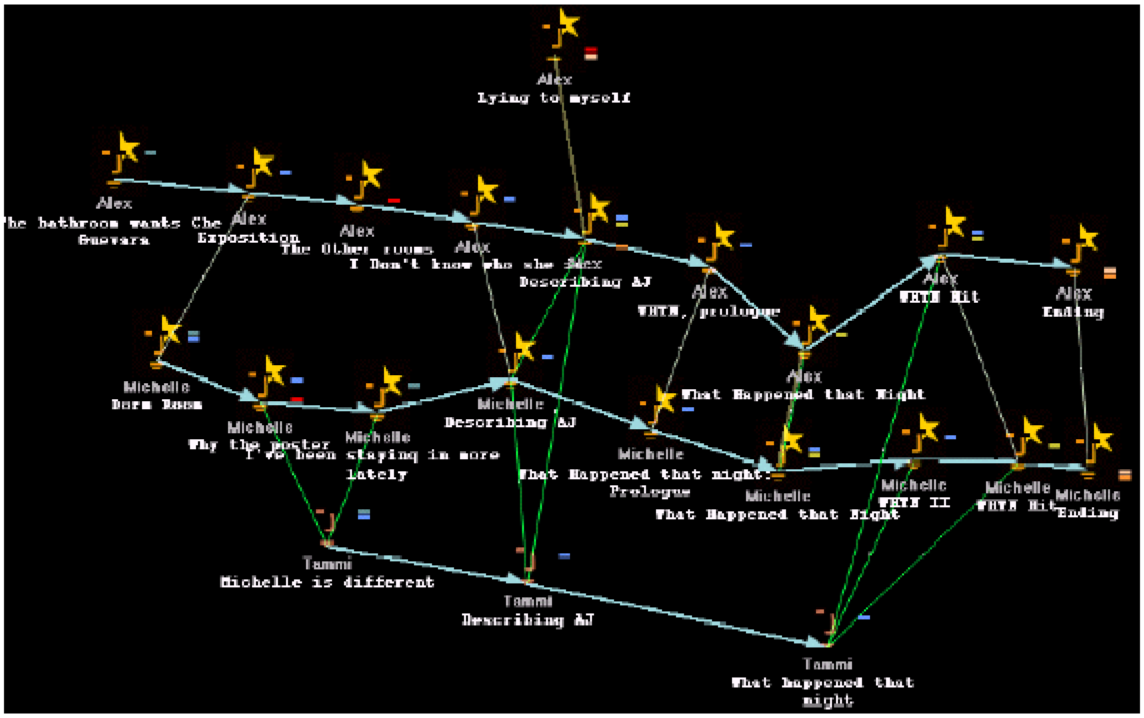
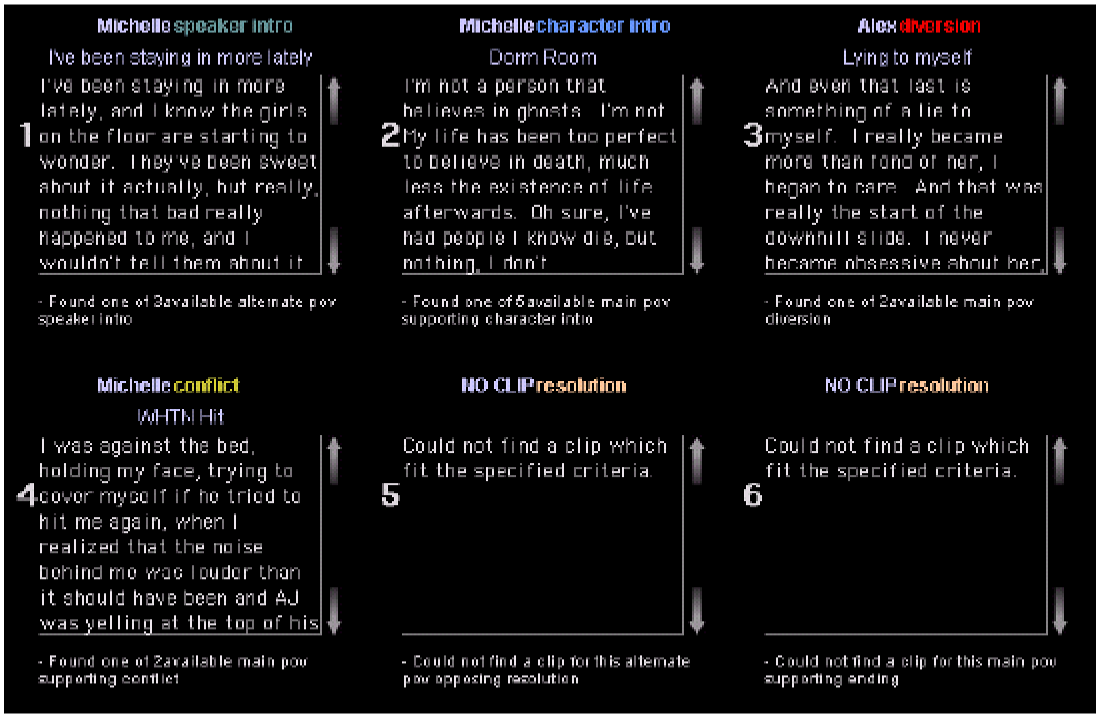


Figure 4 : Screen shot of the Representational Environment displaying a story (Brooks 109)

A Writer Feedback Environment provides the writer with a means for clearly understanding the state of the story and its structures of representation by displaying a sketchy outline for a linear story. A sketchy outline is the simplest story possible, given a story structure and a style of reasoning about the construction. In the WFE, a story agent combines the story framework of the Structural Environment with the story representation of the Representational Environment, and outputs a linear story with explanations of why each particular clip was chosen. This form of feedback gives the writer insight into how appropriate parts of the system are working together. Given that information, she can then further modify different the components of the system as she sees fit, to further experiment with the story or simply to fine tune smaller settings.


Michelle speaker intro

I've been staying in more lately

1 I've been staying in more lately, and I know the girls on the floor are starting to wonder. They've been sweet about it actually, but really, nothing that bad really happened to me, and I wouldn't tell them about it

- Found one of 3 available alternate pov speaker intro

Michelle character intro

Dorm Room

2 I'm not a person that believes in ghosts. I'm not. My life has been too perfect to believe in death, much less the existence of life afterwards. Oh sure, I've had people I know die, but nothing, I don't

- Found one of 5 available main pov supporting character intro

Alex diversion

Lying to myself

3 And even that last is something of a lie to myself. I really became more than fond of her, I began to care. And that was really the start of the downhill slide. I never became obsessive about her,

- Found one of 2 available main pov diversion

Michelle conflict

WHTN Hit

4 I was against the bed, holding my face, trying to cover myself if he tried to hit me again, when I realized that the noise behind me was louder than it should have been and AJ was yelling at the top of his

- Found one of 2 available main pov supporting conflict

NO CLIP resolution

5 Could not find a clip which fit the specified criteria.

- Could not find a clip for this alternate pov opposing resolution

NO CLIP resolution

6 Could not find a clip which fit the specified criteria.

- Could not find a clip for this main pov supporting ending

Figure 5 : Screen shot of Writer Feedback Environment displaying a story feedback. (Brooks 122)

Even though the previous version of Agent Stories was not yet implemented with video or audio media displaying capabilities, a Presentational Environment was nonetheless created with that crucial functionality in mind. Since the Agent Stories tool was originally envisioned to fulfill a need in the context of media-rich, interactive settings such as the internet and interactive TV, multimedia content was even envisioned as presentable in a dynamic, simultaneously active display area. Due to the limitations on time, those goals were put oh hold for later versions to implement.

The fifth and one of the most important environments is the Agent Scripting Environment, in which the writer can build behaviors for software agents, the entities which will run the narrative engine and output a linear story. The principles behind scripting of story agents are the most complex algorithms of the system because the behaviors of different agents drive the narrative engine with their particular settings, and the resulting story and clip selection will follow the guidelines of the narrative framework and be chosen by the narrative engine.

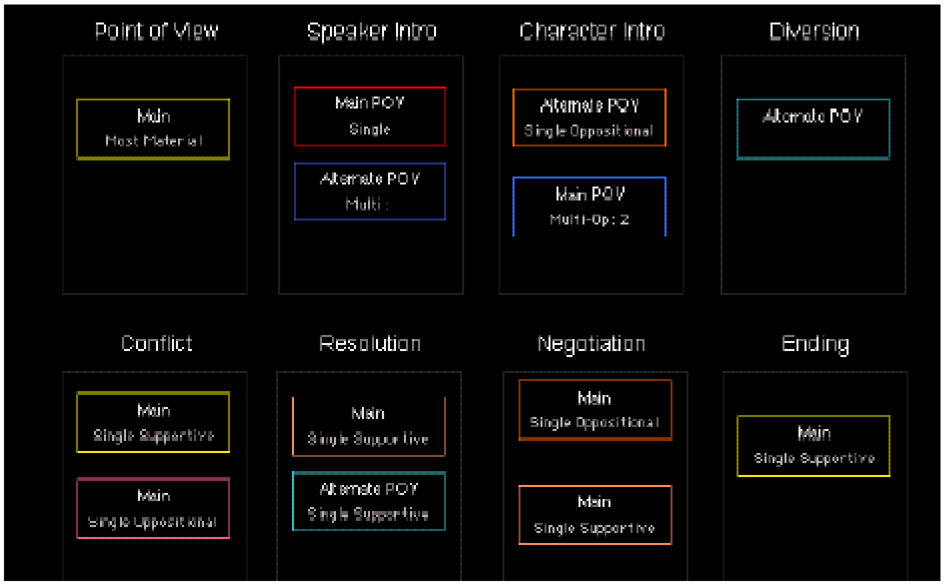


Figure 6 : Agent Scripting Environment. Each smaller box represents a method for choosing a clip for the primitive type box it belongs to. (Brooks 139)

A writer is given the option of determining agents’ internal parameters and therefore can give different agents their own behavior rules.

> Story agents select and sequence the story pieces, according to (a) a user specified abstract narrative structure, (b) the relationships between the story pieces, and (c) the unique parameter values of the story agents. (Brooks, 26)

Each agent can be accorded one main behavioral action and many more secondary actions for each abstract narrative class. Those rules would determine the story engine's actions when selecting story clips to fit into the narrative framework. The writer can also decide to create multiple agents to create one or more stories of different styles. Through control of these agent parameters, a writer can exert variable levels of control on the actions of the story engine.

For example, Brooks' system came included with five predefined story agents. The first agent is Bob, "a happy story agent, who doesn't like a lot of conflict between characters. He'll avoid using characters who don't play well with others" (124). This description means that when Bob is given the task of choosing clips, he'll give preference to those not in conflict with his Main POV. He'll avoid Oppositional links, where clips from a certain point of view will disagree with the situation described by the main character. "This is not to say that Bob avoids conflicts, both in the general sense and the Agent Stories sense. It is just that Bob would construct stories more of the man vs. nature type rather than the man vs. man type" (124).

When constructing a story structure and associated agents, the writer needs to keep in mind how the two components work together, so as to avoid creating a disparity between the clip choices and their selection method. A story structure that does not cater to the agent's preferences may result in very few selected clips, or stories which contain selections from only a narrow range of materials available to the agent.

The Agent Stories system combines the tasks of clip annotation and structural creation in the user manipulable environments. The computer also has a task in filtering and choosing content through work of a story engine, so as to dynamically create multivariant stories as well as provide meaningful feedback to the author, with the intention of further stimulating the author's creative process by presenting new story possibilities which may not have been initially evident.

Brooks further designed the user interface with the intention of creating a visually stimulating presentation that would be intuitive and logical to use, as well as inspire creativity, because the role of a tool should be to help the user in every possible aspect of its functions. With the evolution of Agent Stories into the Java language and Windows platform, the system’s engine capabilities were only somewhat compromised, but to a large extent, its user interface has faced a complete overhaul. As with the challenges of revising any large system, the new version of Agent Stories has been a large compromise in creating extensible support for new features while supporting original values.

3 Agent Stories for Java and WWW

3.1 Previous work

Reconstruction of Agent Stories in Java was begun immediately after the departure of Kevin Brooks. With the intention of enabling this tool to be distributed to a larger audience, Agent Stories was decided to be rebuilt in Java, a platform independent language in which compiled executables can be ported between different platforms. Java applets are also accessible over the internet, so an applet version of Agent Stories would potentially be able to be accessed from anywhere in the world. The future of Agent Stories is to achieve all functionalities originally envisioned by Brooks, as well as be able to reside in a community space as a tool for collaborative story writing and idea sharing.

Anthony Young-Garner implemented the first Java version of Agent Stories. As a UROP under Brooks, he had already built much of the original system in mTropolis, a graphical programming language and environment for the Macintosh. The Java version was originally written in a Netscape IFC, which employed different methods and packages and classes.

The subsequent version, written by Francisco Tanudjaja as his AUP project, was updated to Sun's Java v1.2 with Swing, and documented for reusability. Swing has been integrated into the Java API for ease of use in building generic user interface components such as lists and tables, and also provide pluggable "look & feel" features. The program was developed on the UNIX platform of MIT's Athena network and didn't completely run on windows. But it was successful in bringing the Agent Stories system another step closer to universal accessibility and easy of use.

3.2 Further developments

3.2.1 Immediate goals

At this stage of construction, many immediate possibilities were present for the next implementation. Java v1.3 has been released and has begun to be widely distributed and employed. For example, MIT’s Athena system and many browsers are now equipped to handle the new version by default. The update to v1.3 has also introduced compatibility errors because methods and principles of action have been changed.

The state of the original narrative engine was far more advanced and complex than the existing version in Java. Particularly, Agent Stories in Java was not able to express neither all types of link representation nor all the distinctions within the narrative framework. Each story representation was also not modularized, meaning that the writer was not able to interchange multiple agents for use with the same story setting.

Many features that the original system had intended to provide but, for the sake of time constraints, did not, were now also imminently possible in Java. The QuickTime for Java package provided by Apple can be seamlessly imported into Sun’s Java development environment, and can therefore provide video and audio import and playback capabilities. The Presentational Environment can now be fully realized because the technology for manipulating video is present and easy to use.

Agent Stories in its applet form was also existent at this point in a very basic format. It runs in Athena and loads in a text file. Many difficult implementation issues, such as the possible need for a Java servlet or a dedicated server to handle loading and saving of user settings and videos, were not yet addressed. Yet given the framework on a functioning applet, the system can now be extended to include those features.

3.2.2 Long term goals

A version of Agent Stories that satisfied all its immediate goals would be a fully functional and portable entity, ready for extensive usage as a writing tool. Having a well designed and smoothly functioning system in place will ease the transition into further expansions and revisions. Even unforeseen additions will be easily implemented as long as the system is built with good design concepts in mind.

Multiple research projects currently in progress in the Interactive Cinema Group explore notions of community and collaboration for story telling when they exist in remotely accessible, online meeting spaces. The Shareables architecture project provides the extensible architecture backend to support multiple personalizable applications that reside in public digital spaces. Since the Agent Stories system has progressed from a personal application dedicated to running on one computer to an applet available for access over the internet, further extension of this progression towards remote usage would be to integrate it with the Shareables architecture and possibly enable it to become a multi-user collaborative tool.

3.3 User Interface

As demonstrated by figures three through six, the logical placement, graphical layout, and even color of the original Agent Stories was extensively designed for efficiency, comprehension and aesthetics. Color selection for narrative relations were finalized after discussions with a color consultant, who chose color combinations based on their saturation, hue, warmth, and even their relative positions in color families.

Component shapes were also designed to be distinctive, yet fit in logically with its environment and related representations.

The incredible amount of attention devoted to details of interface graphics created a well organized set of display environments that were not only aesthetically pleasing and fun to use by themselves, but were also thematically structured and well integrated with each other. The fade-to-black transition between environments, though not designed for speed, was immediately intuitive to anyone who’s ever watched a movie. The most important quality to be credited to this interface is that it makes learning the tool such an engaging experience that the user would be prompted to play and experiment even when she does not have the explicit goal of writing a story.

As a tradeoff for beautiful user design, each individual environment was given exclusive screen property. The amount of allotted space allows added liberty in creating clear and understandable interfaces with room for additional creativity and increased complexity within each environment. However, switching between two or more environments enforces a delay in transitional time, which disrupts the writer’s thought process, as well as being a minor annoyance for users familiar with quick-response systems. On top of that, not being able to gain an overview of the entire story state can become a considerable problem as a story grows in size. One of the main reasons for the creation of Agent Stories was the hypothesis that as story size and interconnectedness expand, maintaining coherency in the writer’s mind will also increase in difficulty. Brooks has created an excellent compositional structure, replete with software agents, to counter this difficulty. Yet due to the tradeoffs made in user interface design, elements of the scalability obstacle remain unresolved.

Since the Java version of Agent Stories was written in Swing, a package which provides “pluggable look & feel”, the interface components were implemented in prepackaged Swing structures such as lists and tables, and their shapes and colors inevitably became conformed to the default look & feel of the Java API. Partially to resolve the environment overlay problem of the old version, and partially to increase easy and speed of implementation, the new interface was also designed to partition its screen space so that all five environments are now simultaneously visible.

As the old interface was lost in the revision, new problems arose in effective user interaction with the application, as well as the role of interface design in promoting user creativity and understanding of the framework and state representations. Since every extension to the program involved an evaluation of its role in retaining the integrity of the interface, comprehension of user interface became a uniting concern throughout the process of development. I will therefore introduce some theories of effective user interface design and concerns of interface tradeoffs in Agent Stories because they provide comprehensive and revealing looks at the integration and development issues encountered.

3.3.1 Theory of effective interface design

Due to limitations of implementing custom graphical interfaces in Java as well as existing time constraints, the revised Java version was given a brand new interface that did not reflect the same sensibilities and dedication to visual stimulation. As a result, any change to this interface needs to strongly reflect solid interface design principles for an information intensive structure.

First and foremost, we consider the statistics of our user base. Agent Stories as a story-writing tool will not be for the casual user because it involves a high initial learning curve. The assumption that most users would be computer literate and familiar with the basic ideas of narrative seems highly probable. Franzke noted that users who have used a variety of applications before encountering a new one “are therefore more likely to attempt learning new functionality by exploration rather than by reading manuals and tutorials” (422). He further insists that even for casual users, the interface should provide ample cues for usage. This revelation suggests that an interface would serve both expert and novice writers by providing excellent explorability and learnability.

Prevailing beliefs dictate that a graphical user interface needs to do more than just look good. For the purpose of explorability, layout of components should intuitively suggest the mode of interaction. Such information can be conveyed through natural and concise language usage and logical placement of components so the user doesn’t spend much time searching for every desired object. Franzke demonstrated through various experiments that user searches are guided by label-goal matches. In other words, if an element of the interface matches the expected target action, it will be easily differentiable from other elements and quickly recognized.

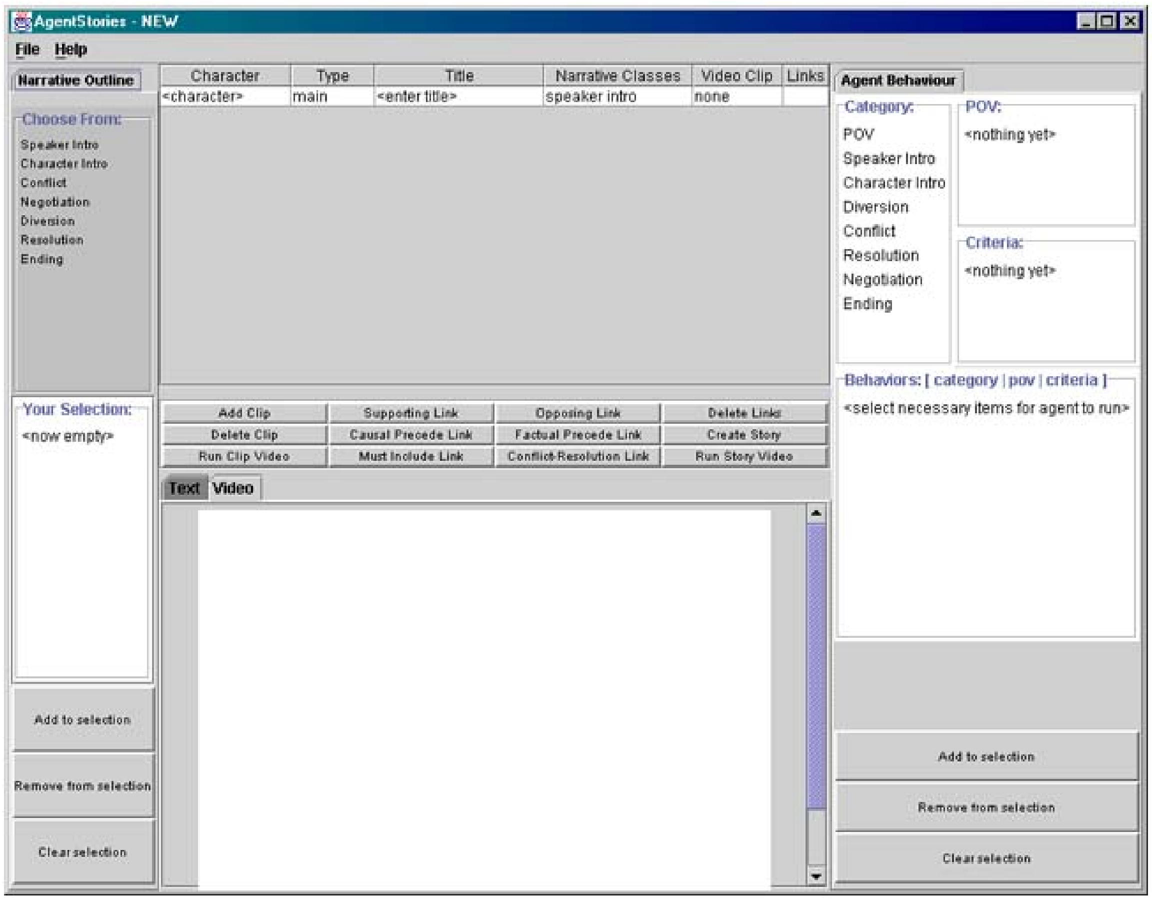
Further studies by Terwilliger and Polson reinforce these findings by categorizing user navigation into two groups – instruction following and task elaboration. In instruction following, users construct goals from representation of instructions, sustaining the principle that interface and instruction elements be given the same terms. The method of search called task elaboration illustrates that users also retain pre-existing representations in their minds, in which case such an interface would function more efficiently if it matched existing expectations instead of given instructions. Both studies dictate that an interface containing logical placements and labels would be simple to learn and interact with, and a user can retain a good overall comprehension of usage through basic visual cues.

For the purposes of usability, an intuitive interfaces should also convey meaning or structure of concepts through information grouping or an easily navigable information hierarchy. An interface that reveals the logistics of underlying system structures will also increase user comprehension and therefore promote greater flexibility and aptitude of usage. To this extent, research has continued in issues of navigating information structures, especially faced with problems as these structures increase in size and complexity but the amount of interaction is still limited by screen space and time allotted. Furnas stated that assuming elements are organized in logical structures, these same elements should be placed next to their logical neighbors in information flow as well as workflow.

Especially in an interface where a large amount of information is conveyed, these principles of design are essential to ensure usability and comprehension. The current Agent Stories system is a large information structure where the current interface does not meet the design principles with which it was initially conceptualized. In the attempt for the new interface to satisfactorily meet original expectations, the edicts of learnability, explorability, and usability need to be thoughtfully integrated while acknowledging system limitations.

3.3.2 Interface overview

Upon first look at the new Agent Stories interface, major differences in look & feel and screen composition are immediately apparent. The default look & feel provided by Swing components are very similar to the Windows environment. Layout of system environments has also been compromised to fit into one encompassing screen space. Each environment fills a portion of the screen as they are always constantly visible. Due to the homogeneity of colors, shapes and fonts, the difficulties previously discussed for an information structure are clearly illustrated. The following figure displays Agent Stories in its current state.



Character	Type	Title	Narrative Classes	Video Clip	Links
<character>	main	<enter title>	speaker intro	none	

Figure 7 : Screen shot of Agent Stories in Java, in its new file state

On the left column, the Structural Environment displays a top window labeled “Choose From:” for the list of possible narrative classes from which the user can select. The lower window labeled “Your Selection:” is intended to house the user-selected narrative framework. Possible action functions are each illustrated by a clearly marked button that describes its intended purpose.

A table structure in the top center portion of the screen contains all functionalities of the Representational Environment. Each individual clip is now contained by a row in the table, and metadata associated with each clip now reside in distinct columns. As in the old version, each clip is associated with a principal character, the character/clip type (main, minor, dramatic, sonic), a title, and its narrative class. The video clip column contains the location of an associated piece of video. This functionality is only present in the most recent version. The “Links” column is the only one that doesn’t represent a field in the data structure. It was added on later in consideration of interface usability, which will be demonstrated later. As with the Structural Environment, possible actions are all represented by buttons, which exist in an array at the center of our screen.

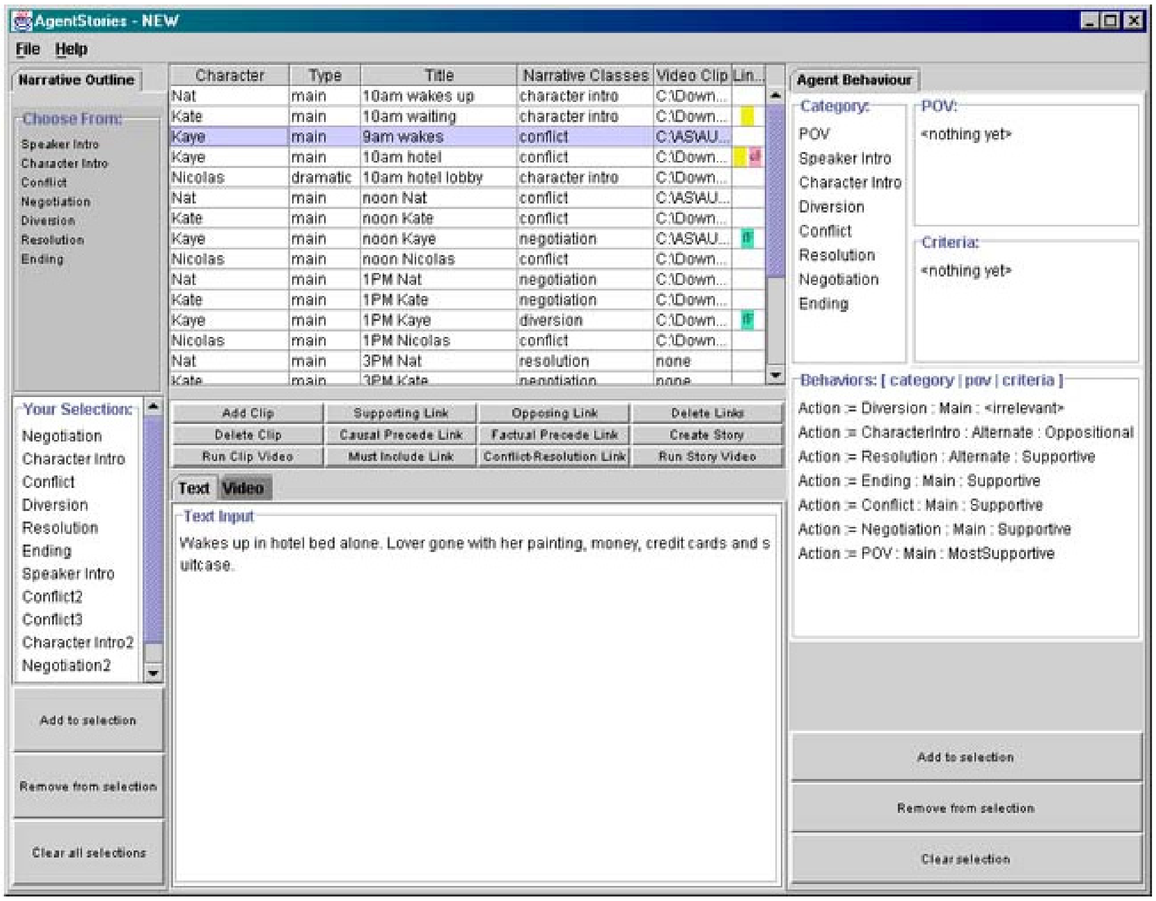
In the lower center portion we see a tabbed window housing two overlapping panels labeled “text” and “video”. The text window is a version of the Writer Feedback Environment, which displays text associated with each clip or a resulting story sequence generated by the narrative engine. As previously stated, the narrative engine is no longer able to give feedback concerning the choice of clip selection. The video window is able to display pieces of video associated with each clip, as well as generate a video sequence from an engine-selected story. As video capabilities have finally been incorporated into

Agent Stories, the Presentational Environment can in turn develop into a functioning and integral component of our system.

Finally, the Agent Scripting Environment resides in the right most column of our screen. Similar to the Structural Environment, the top window contains possible selections which the user can make. Finalized selections will appear in the lower window as a list, and all control actions are represented by an individual button at the very bottom of the column.

3.3.3 Design principles in practice

In keeping with the principles of learnability, explorability, and usability, the color coordination and component layout within each environment, and the relative positioning of the environments to each other and within the whole system all reflect an attempt to accommodate the user experience. This interface was created with an eye towards retaining the interface integrity of the original application, as well as attempting to further improve upon noticeable difficulties existent. To better illustrate the experience, figure 8 typifies the state of our system in the middle of a construction process. At this moment, the writer has input a significant amount of story clips and created a series of linkages between those clips. The narrative framework and story agent have also been extensively composited. This figure demonstrates the system capabilities when the user has gone beyond casual use or experimentation.



Character	Type	Title	Narrative Classes	Video Clip	Lin...
Nat	main	10am wakes up	character intro	C:\Down...	
Kate	main	10am waiting	character intro	C:\Down...	
Kaye	main	9am wakes	conflict	C:\AS\AU...	
Kaye	main	10am hotel	conflict	C:\Down...	
Nicolas	dramatic	10am hotel lobby	character intro	C:\Down...	
Nat	main	noon Nat	conflict	C:\AS\AU...	
Kate	main	noon Kate	conflict	C:\Down...	
Kaye	main	noon Kaye	negotiation	C:\AS\AU...	
Nicolas	main	noon Nicolas	conflict	C:\Down...	
Nat	main	1PM Nat	negotiation	C:\Down...	
Kate	main	1PM Kate	negotiation	C:\Down...	
Kaye	main	1PM Kaye	diversion	C:\Down...	
Nicolas	main	1PM Nicolas	conflict	C:\Down...	
Nat	main	3PM Nat	resolution	none	
Kate	main	3PM Kate	negotiation	none	

Wakes up in hotel bed alone. Lover gone with her painting, money, credit cards and suitcase.

Action := Diversion : Main : <irrelevant>
Action := CharacterIntro : Alternate : Oppositional
Action := Resolution : Alternate : Supportive
Action := Ending : Main : Supportive
Action := Conflict : Main : Supportive
Action := Negotiation : Main : Supportive
Action := POV : Main : MostSupportive

Figure 8 : Screen shot of Agent Stories after loading a story

Now that each environment is shown to be in active state, their full functionalities are better illustrated. The Structural Environment contains a long narrative framework that extends beyond the boundaries of our fixed-size window, where the choice was made to encapsulate this list in a scrollable panel. A potential tradeoff was to allow individual windows to resize as the size of its contents increase. However, the final decision was made in regards to usability, because if windows would resize and shift of their own accord depending on the amount of information it contains, the user will not be able to work with a constant configuration and achieve a comfort with the environment. Another frequent problem was confusion in differentiating between the two selection windows. As a response, I slightly darkened the “From” window so as to lead the eye to the list of user selected classes.

The Representational Environment is also housed within a scrollable container for similar reasons. For the purpose of increasing usability as well as learnability, the “Links” column was added into the table as simple reflection of state, not data storage. As the user clicks on different clips in the table, color bars appear in the links column, each of which represents a link related to the selected clip. For example, if clip A and clip B are linked together by a supporting link, then whenever the user selects clip A, a color bar representing supportive links will appear next to clip B. In view of the original Representational Environment, where the overview presents a set of clips as well as the links that connect them, the “Links” column was added in as a substitute for the representation of narrative links. As noted by Barbara Barry during a session of feedback, the user can easily learn to gain an overall sense of the amount of and types of links present in the system. This ability is especially useful in the process of story writing, where the writer frequently scans the content in order to gain a sense of what exists and what more will be needed in the story.

Since the “text” panel displays output of a run of the narrative engine, it is the equivalent of the Writer Feedback Environment. At the same time, due to its proximity to the Representational Environment and the fact that this panel can easily display text associated with a clip, it also serves as the display for the Representational Environment. Even though this bifurcation of responsibilities does not follow the guidelines of the original version of Agent Stories, it represents a very intuitive combination of functionalities from the point of view of the user. All three notable goals of our interface design are satisfied since within a limited viewing area, we have one text display for all actions under this category. This fact is easy to learn and quickly memorizable. Since displaying text is such a common task in this system, even users who are learning through exploration will quickly discover this feature.

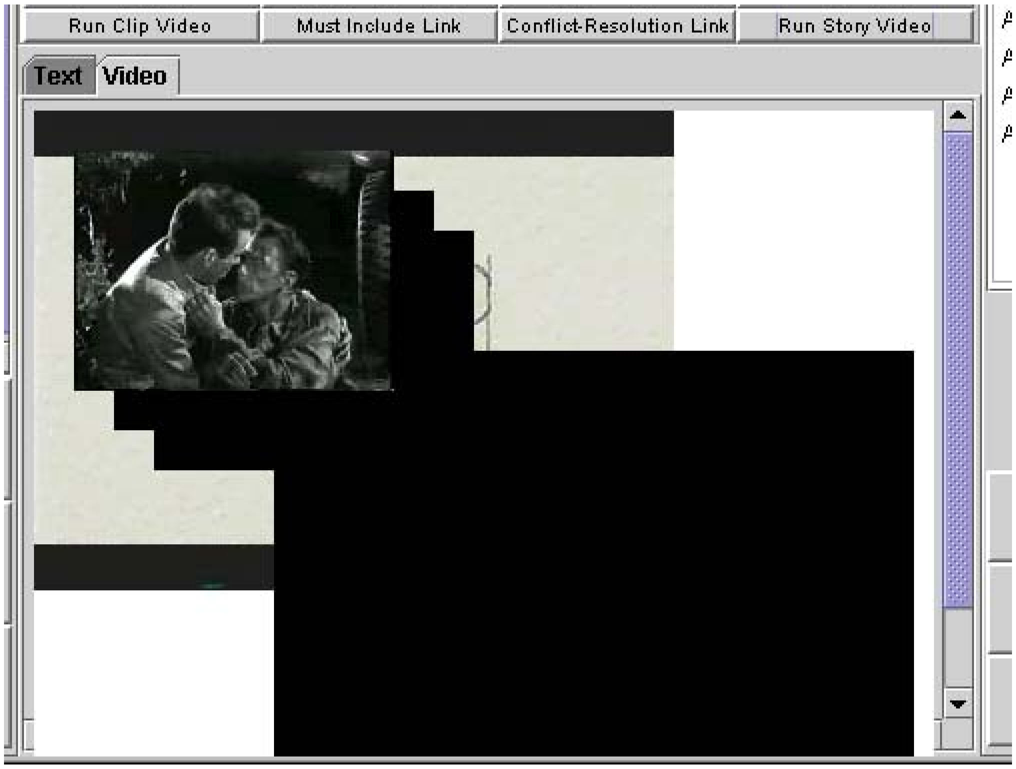


Figure 9 : Screen shot of Presentational Environment playing a video sequence

The current video display window has begun to implement selections of characteristics originally envisioned for the Presentational Environment. In striving for universality, all forms of drawable content supported by the QuickTime for Java package are also playable in this environment. Those supported formats include video files, such as QuickTime and AVI format, audio files, and even text files, for which the text will be displayed sequentially in a small display area. Currently, the drawable clips are staggered in position on the canvas, with the first one in the top left-hand corner and each subsequent clip placed in a lower position. As the sequence of clips play through, the current clip will be brought to the foreground so as to be completely viewable. Depending on the preferences, each sequence can play once or loop many times.

Looking ahead, many possible features can be added to this environment, which I feel is the most extensible in the system. Video playback does not need to feature static placement and strictly sequential ordering. Instead, a sequence can take full advantage of the computational abilities of the story engine and agents, and display a dynamically configurable layout that applies the conditions of engine decision and agent behavior into determining how a series of video and audio will play out. A most interesting topic of research would be to explore how the personality of a story agent will affect the editing of video sequences, particularly in the pacing, layout, movement and interaction between clips. This study is very possibly multidisciplinary and may involve tapping into the fields of AI as well as film theory.

The fifth and final environment, the Agent Scripting Environment, resides in the right most column. To parallel the principles exercised in the Structural Environment, the content windows remain on top, while action buttons are grouped on the bottom. Of the two selection windows, the top one contains the possible choices for agent behavior for which the user can configure, then the bottom one contains a list of selections that have been made. To increase the ease of learnability, parallel principles of design was employed for all possible environments. Even for the Representational Environment, all action buttons are placed below the component window.

Overall placement of each environment within the whole system was designed in an effort to maximize learnability and usability. Explicit decisions were made to overlap the text and video displays because logically, since the text was intended to serve as placeholders for the user to eventually shoot appropriate video clips, the displaying of one or the other can be mutually exclusive.

The order in which environment components were sequenced parallels the choice made by Kevin Brooks in his original tabbed layout. In relative proportion however, the Representational Environment, due to its large collection of dense informational structures and the fact that most of the time spent by the user will be in creating clip content, occupies the most space and resides in the center of the screen.

The decision that all environments would be visible at the same time has created the need for efficient space usage and interface design for usability. The tradeoff was made between user awareness of all states of the story and space constraints that limit the complexity of graphical design and amount of viewable information. Through research of interface design and feedback sessions with other graduate students, I've made various adjustments and additions in color, labeling, and component layout that hopefully has resulted in a much more efficient and aesthetic interface.

3.4 Illustrations of usage

In parallel with the development of various features of the application, I have also examined the feel and logistics of Agent Stories by entering narrative content into the system and testing its responses given different frameworks and agents. Various graduate students including myself attended a Narrative Workshop this past November, 2000 at the MLE in Dublin, which had been organized by Glorianna Davenport of Interactive Cinema. As an exploration of the possibility of creating multiple character-centric linear stories, and seeing how these characters, who are in the same city yet do not know each other, may cross paths and interact, or exert secondary influences in each others' lives.

The content generated at that brainstorming session revolves around four different characters, named Kate, Nat, Kaye and Nicholas. At the same time this morning, each one of them reaches a major crossroad in their lives, and faces a command decision. In the course of one day, they will pass through many of the same locations of this city and go about resolving their lives in crisis, sometimes meeting each other and being influenced. Each story clip is centered on one character at a certain time and location. When entered into the Agent Stories system and structured with various narrative links, these 22 individual events form a story space that evenly spans a small time and space.

The structure of these parallel lives would fit very well into a soap opera-like drama, so I frequently tested the system with agents that prefer to see a lot of alternate characters' point of views. Multiple runs of the system tended to generate story lines not radically different from one another, which reinforced the stability of story agents' selection algorithm. However, because agent knowledge is limited to story structure but not clip content, some sequences still lack a certain coherency. For example,

> **Nicolas**: Goes to hotel lobby to assemble revolutionary new shopping cart design. Builds it, locks herself to it, and heads off to test it.
> **Kate**: Meets likely lad and flirts outrageously. Why not? her husband doesn't care - today is the beginning of the rest of her life after all...
> **Kaye**: Wakes up in hotel bed alone. Lover gone with her painting, money, credit cards and suitcase.
> **Kaye**: Goes to bar for a quiet drink to gather thoughts.
> **Nicolas**: Goes to hospital for check up - sits in the waiting room for hours.
> **Kaye**: Goes down to hotel lobby to talk to concierge about what has happened. sets off for embassy.
> **Nat**: Wakes up with headache.
> **Kate**: Sitting in the bar getting smashed with bloke - who cares?!
> **Nicolas**: Emerges from hospital and is transfixed by wonderful singing. transported away from her troubles

This sequence gives a good illustration of the nonlinear preferences of story agents. Yet looking at the interaction between Kate and Nat, the audience would find it difficult to gain a solid comprehension of their story cohesion like the writer may have intended. Since these two characters meet each other only later in the afternoon and go get smashed in a bar together later at night, the agent was not constrained to similarly parallel their actions in the beginning of the day. As a result, Kate flirts with Nat on the second sequence, but Nat does not wake up in the morning with a headache until near the end of the story, right before he's getting roaringly drunk at a bar. Such a temporal sequencing may confuse the audience as to whether Nat woke up with a hangover this morning, or the next morning after carousing with Kate.

For better contrasting of the two characters, the agent should be able to exert greater foresight and better analysis of the story content. For example, one way to increase agent flexibility is to give them the ability to do a two passes on a story sequence. The first pass involves selecting clips to fit into the framework, and at the second pass the agent may either reorder clips or selectively rerun its engine on subsections. One problem that these behaviors may alleviate is the fact that agents often select a certain clip to fit a narrative framework, then be constrained by 'Precedes' links further down the line. Occasionally, a generated sequence may be very short because the agent begins by choosing a clip very near the end of the story, and can no longer access any related materials that had factually or causally preceded this initial clip.

Another caveat for the user to remember is that even though the Agent Stories systems is designed to initiate multilinear thinking, its linear narrative framework and the text implemented Presentational Environment still allows for linear sequencing. Agents can reorder story clips so as to suggest possibilities for multilinearity in the story's final incarnation, but a sequence of chosen clips, stringed together in accordance with the narrative framework, will still be presented one after the other. This result will be somewhat resolved when video capabilities has been fully integrated into the Presentational Environment. A generous display area will allow for multimedia versions of the story with simultaneous streams of video and audio material, which more closely resembles the original purpose of the system. A writer who is using this tool to generate ideas for a nonconventional mode of presentation, for example interactive story-telling over the internet or a physical art installation, need to regard generated story sequences with the reservation that a story will still need to fit its medium of delivery, not just satisfy the constraint of being 'non-' or 'multi-' linear.

Through the process of experimentation, I realized that our system functions best when it's fully populated. Given a set of story clips, and a proper narrative framework, the agent would only create a significant story if the story representation contains sufficient amount of links to satisfy the agent behaviors. On the other hand, a significant amount of narrative links would not necessarily result in a complex story sequence if the agent behaviors can not conform the given clip types to our narrative framework. This requirement creates a time barrier from generating the first batch of ideas to actually producing a likely story sequence. The starting user will experience difficulties not only in initially learning the system, but in generating enough material to fit the structure of system so that it can fully demonstrate its abilities to logically generate nonlinear narratives. If a user tries to run an agent on a small amount of clips, the resulting sequence may be even shorter and therefore confusing and unengaging.

It is therefore not very likely to play with and generate a simple tale in our system. For example, many children's books are too short and logically structured to allow much multilinear manipulation. Even looking at innovative children's books such as the Dr. Seuss series, nonsensical words and characters, intended to break a child's conventional view of their world, are still sequenced carefully to provide a progression of ideas and teach the child about logic and cause and effect. These properties are partially the constraints of a linearly sequenced book format, and will be questioned and manipulated when stories are ported to interactive mediums that inspire a child's participation. Yet the basic constructs of a story form will remain, if the purpose of that form is to be educative in a logically sequential way.

The original Agent Stories was created by Kevin Brooks, for an audience of story writers not too unlike himself to create stories of a certain depth and complexity. As a result, this tool may not be the most suitable for initial writers who either have not had enough experience in constructing a large storyscape or writers who have in mind a certain story format that's not easily implementable by this system. However, Agent Stories is a flexible tool that still warrants experimentation by all writers, if not for serious usage in their work, but merely as a way of suggesting different methods of sequencing and editing to motivate new ways of thinking for the digital story-telling realm.

4 Technical specifications

4.1 Overview of subsystems

The code structure of Agent Stories has been organized so that the ‘as’ package contains code dealing with interface and connectivity functions, while the ‘agentstories’ package contains code that describes the underlying engine and agent behaviors. The ‘as’ package also the task of saving and loading a user’s story configuration, whether it be to the local system or remotely through a dedicated server.

When the system is running, all information collected through the interface will be stored and handled by classes in ‘as’. Once the user decides to run the engine, those pieces of information will be distributed to the engine, which parses all settings and then runs the selection algorithm of its particular agent. A resulting narrative is now stored in the system, and can be displayed as both a text and a video sequence.

4.2 Details of subsystems

The system diagrams provided to illustrate overviews of structural and procedural dependencies are not strict dependency diagrams. I felt that due to the size the program, a completely comprehensive MDD would be too complex to be of any help in understanding the behaviors of the system. One of the greatest difficulties that I encountered in implementing this project was the amount of time that was required to gain an understanding of the system. Due to the lack of clear and concise documentation, learning the details of the code occupied a significant portion of the project timeline.

In the following description of each subsystem, I will provide explanations of the subsystems' structural and procedural organization, as well as notable peculiarities in the methods of implementation and the integration of Java itself.

4.2.1 The 'as' package

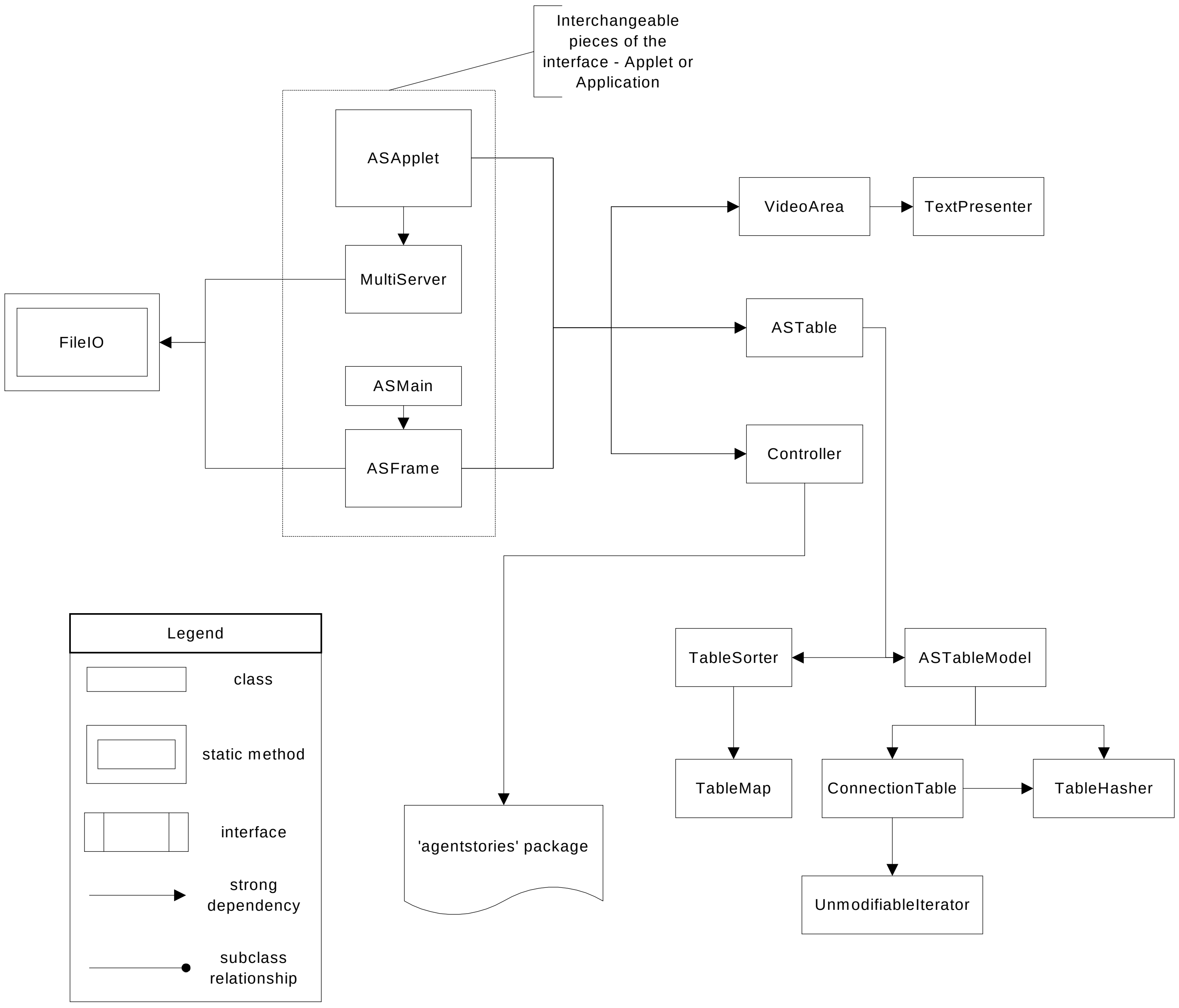


Figure 10 : Module Dependency Diagram of 'as' package

All essential and peripheral interface functionalities are contained in this class package. Because applications running on the local computer does not experience the same security restrictions as an applet, the system is fully implemented as an application, runnable from the file ASMain.java. An interchangeable applet interface has also been included in the package, and all features not related to the access of files function exactly the same way as in the application.

Applets in Java are not granted the permission to access the local hard drive, only publicly available files located at URL addresses are viewable. Therefore, to enable loading and saving, the ASApplet layer has been created to communicate all access requests from the MultiServer. When implementing the remote calls from applet to server, a choice was available as to at which stage of the request should static functions be called from the FileIO class, and how should the process be delegated between the applet and server.

One available option was for the applet to work very similarly to the application, and call functions of FileIO when a request is processed to load or save a file. FileIO would then pass all request parameters to the server and send responses back to the applet, therefore handling all parsing of information from server and preserving design uniformity of the interchangeable interface components.

The second and implemented option called for the applet to handle all communications with the server directly, which requires the server to process all file parameters and call FileIO. With this technique, establishment of contact with a server is confirmed at the initiation of the applet, and not after a request has been sent to FileIO. As a tradeoff for modular design, this method preserved the format and function of

FileIO, as well as its theoretical location as residing on the local drive. In addition, one principle of programming servers deem that all significant amounts of processes should be handled the server side and not by the client, due to variability in client platforms and processing power. In accords with that technique, only filenames and Vectors containing story settings are passed between applet and server, thus allowing network communication early in a request process. Even though the two interface components no longer parallel each other in terms of functional design, this technique retains a more robust concept of server client communication and allows greater user understanding of the communications process.

Video display capabilities are implemented through the QuickTime for Java package provided by Apple. To preserve functional modularity, all parsing of story and states are still controlled by the same classes, only the data structure of the NarrativeFragment class was slightly expanded. All relevant information is then passed to the VideoArea class, which manipulates all properties of video sequencing and playback according to those parameters. Under this method, the relationships between engine decisions and video sequencing remain controlled by the Presentational Environment. This implementation allows for easy accessibility to all visual presentation characteristics. Because the effects of agent behavior on story playback is a potentially large and exciting future project, I feel that the best implementation for the Presentational Environment is for it to possess all control of display options.

The process of integrating QuickTime for Java with Sun’s Java v1.3 API was tricky and filled with idiosyncrasies. Hopefully later versions of the QuickTime for Java package will eliminate all those problems. Two notable problems of integration still remain in the system. Firstly, the video display area, QTCanvas from the QuickTime package, does not integrate well with Swing components. Even though the canvas is wrapped in a JScrollPane, it would roll out of the scrollable area and cover some action buttons belonging to the Representational Environment. Secondly, the video display encounters inconsistencies that aren't always replicable. Each sequence does not always reorder the currently playing clip to the foreground. All video sequences also do not clear properly, causing phantom audio streams to play alongside other video streams.

Many other details were added to the interface. The user can now save and load agents separately from the entire story setting. She can also save a particular story's text version into a file. The narrative framework in the Structural Environment now supports multiple classes and the drag and drop method of reordering elements to increase ease of usage. Video clips can be chosen by simply clicking on the video column of the desired clip, and a selection window will appear automatically for the user to designate a file either from the local file system or from a URL location. To enable displaying of links information in the Representational table, the data format in ASTableModel.java was expanded to support additional information. However, this field represents transient information and will not be saved.

Because most new research revolving around software agents' and video editing behaviors will occur in the interface domain, I've attempted to create robust and modular display components that are easily understandable and modifiable. This remains one of the most potentially exciting areas of expansion in this system.

4.2.2 The ‘agentstories’ package

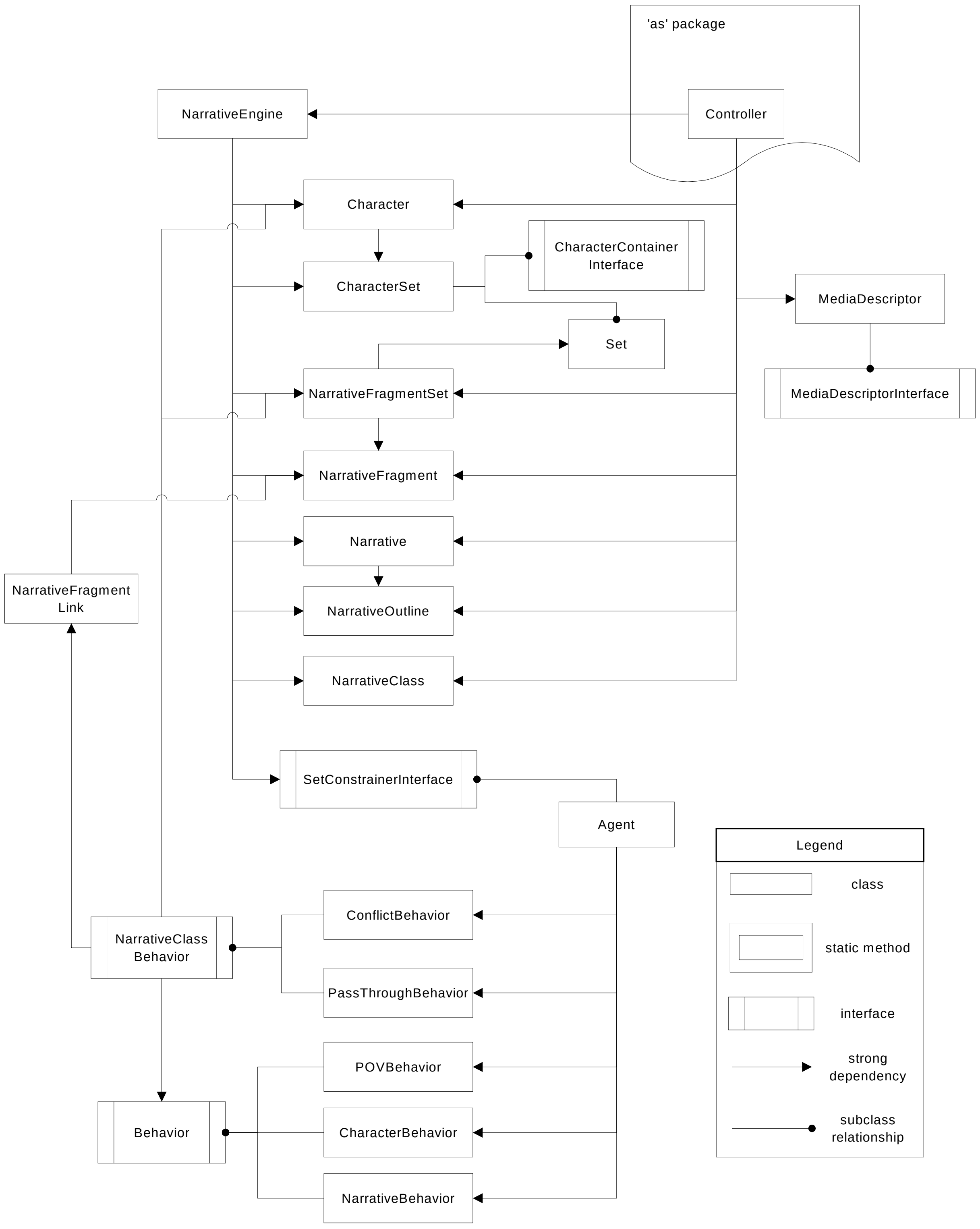


Figure 11 : Module Dependency Diagram of ‘agentstories’ package

The agentstories package contains all classes that detail the engine and agent behaviors. Even though the roles of narrative engine and story agent in selecting story sequence may overlap, their code structures are reasonably disjoint, as can be seen in the package MDD. When the user runs the engine to create a story, the Controller will parse all story information and structures and set the engine with those parameters. Then, the engine will receive the specifications of an agent and call the agent to create a selection of story clips.

Most of the amendments implemented in this class involved adding back engine functionalities which were available in the original, mTropolis version of Agent Stories, but were not yet ported to Java. In particular, the previous Java version of Agent Stories only contained the ability to make ‘supporting’ and ‘opposing’ links between clips. This connections are considered bi-directional and mutually exclusive. When a clip is chosen to support or oppose another clip, then the latter clip will automatically be assigned to support or oppose the first. A pair of clips can only retain either a supporting or an opposing relationship.

In this version, all of the original links are not available. Both Causal Precede links and Factual Precede links affects the sequence of the story by imposing an ordering of clips. This decision is made in the Agent.constrainSetOnNarrative method because this function limits the set of potential clips by looking at previous clips that have already been added to the story. At this point, I’ve also deleted the possibility of adding the same clip twice in one story, though it is very possible that such an arrangement may be reinserted at a later time.

The Must Include link is considered a second class link, which means that the link is not represented in the Structural Environment, much like the Precede links. If the engine cannot fulfill this requirement it will simply continue the algorithm. It will not fail in an algorithm if it does not satisfy this link. In the NarrativeEngine.getNarrative method, the engine first calls upon the agent to create a queue of available clips connected by Must Includes links. Whenever the engine had a choice of multiple clips from which to select, it would look in the Must Include queue to see if one of those clips are also in the queue. This action is called in the Agent.constrainSetOnRule method.

The Conflict/Resolution link is considered a first class link because Conflict and Resolution classes can be linked in the Structural Environment. As a sidenote, this functionality is not yet possible in the current code version, but was originally available. This option was not implemented because linking classes in the Structural Environment is a significant problem completely distinct from the issue of adding links between story clips in the Representational Environment. As is currently implemented, if the narrative framework contains conflict instances and then resolution classes, then upon reaching the resolution class, the engine will limit the fragment set to only resolutions clips which have been linked to the conflict clips already chosen. In the original implementation of Agent Stories, linking conflict and resolution in the narrative framework would further limit the selection of clips to be even more specific. If the user chooses not to link these classes then both the original and the current engine would run identically.

The data structures in the ‘agentstories’ package are very complex and interlocking. Making changes to clip information and trying to form conflict-resolution pairs in the narrative framework often requires amendments in multiple files. One helpful modification to the code would be to increase documentation and code modularity so that future changes to the engine or agent characteristics will be not necessitate understanding the entire engine selection process and a majority of the other files of the package.

4.3 Unfinished features and suggestions for future amendments

Asides from some unfinished features already mentioned in the previous sections, the current system continues to lack some original functionalities. My amendments mainly concerned adding link handling capabilities back into the engine. Much of the agent behaviors have remained neglected.

Main POV:
- Main, Minor, Dramatic, Sound
  - Most Material, Least Material, Most Oppositional, Least Oppositional, Most Supportive, Least Supportive, Random
- Specific
  - Character Name____

Speaker Intro
- Main POV, Alternative
  - Single
  - Multiple
    - How Many?___
- Specific
  - Character Name_____

Figure 12 : Agent behavior for two sample narrative classes

This figure gives a good sample of typical agent behaviors in the original system. For the Main POV, the agent can choose a random character. For all classes except Main POV, Diversion, and Ending, the agent can be given the “How Many?” option, meaning that in the Writer Feedback Environment, it will display multiple possible clips which can fit into a certain class in the narrative framework. Due to the limitations of a stripped down WFE, this option is not easily implementable at this stage. For the narrative classes Main POV, Speaker Intro, Character Intro, and Ending, the user can also specify a particular character’s name.

In addition to agent specifications, the current version is also missing the ability to concatenate an agent behavior from a number of sub-behaviors. Originally, the Boolean conjunctions AND, OR, and the prefix NOT can optionally be used between behaviors for added user control. The default conjunction of OR is assumed to exist between behaviors if the user neglected to specify any. Because we do not have the option of specifying conjunctions right now, the default OR is automatically built into the engine algorithm. To add this amendment to our system in the future, we will need to add these actions to the interface, which would mean that the display will become even more space sparse.

As noted from Francisco Tanudjaja's AUP report, the 'agentstories' package originally included many more specific agent behavior specifications : CharacterIntro, CharacterIntroBehavior, Conflict, Diversion, DiversionBehavior, Ending, EndingBehavior, Negotiation, NegotiationBehavior, Resolution, ResolutionBehavior, SpeakerIntroBehavior. This files can be reintegrated into the code if in the future we decide to increase the flexibility of the engine algorithm.

Another way of augmenting engine algorithm is by increasing its efficiency. "Most of the loop operations are in order O(N), with order $O(N^2)$ being maximum. The code is not tailored for most efficiency, but rather to cover a broad range of topics in a short time" (Tanudjaja 10).

5 Users Manual

Because the interface has been sized for efficiency and not transparent usage, I feel that providing a user manual is essential for knowing all features of this system, for the casual user as well as a long time writer.

- The applet has been hardcoded to request connection to a specific server. To run the applet, run the MultiServer.java file from a networked computer and remember its name. Then go into the ASApplet.java file's init() method and input the computer name by changing the line : s = new Socket("***", PORT);. Replace *** with desired computer name and recompile. Then you can run the applet. Note : applet works right now with appletviewer.exe, which is packaged with Sun's Java SDK. It does not yet work in a browser.
- To create a sequence, the user must first create a story by clicking on "Create Story". This will display the text sequence in the text display panel. Only then can the user view the video sequence of the created story by clicking on "Run Story Video".
- To link multiple clips, hold down the Ctrl key to select multiple clips then click the links button desired. For Precedes links, we can specify one clip to precede multiple clips. Select the first clip, then select other clips to follow the first by holding down Ctrl and clicking on as many others as desired. Then choose the desired Precedes link. This procedure will add a Precedes link from the first clip to each subsequent clip. No links will be added between subsequent clips.
- The narrative framework ("Your Selection:" panel) in the Structural Environment can now be reordered with a simple drag and drop methodology.

Bibliography

Brooks, Kevin Michael. *Metalinear Cinematic Narrative: Theory, Process, and Tool*. MIT Ph.D. Thesis, 1999.

Davenport, Glorianna. *Synergistic StoryScapes and Constructionist Cinematic Sharing*. IBM Systems Journal, Vol 39, Nos. 3 & 4, pp. 456-469, 2000.

Franzke, Marita. *Turning Research into Practice : Characteristics of Display-Based Interaction.* in Proc. CHI '95. Human Factors in Computing Systems, (1995), ACM, pp 421-428.

Furnas, George W. *Effective View Navigation*. CHI '97. Human Factors in Computing Systems, (1997), ACM.

Murray, Janet. Hamlet on the Holodeck. New York: The Free Press, 1997.

Tanudjaja, Francisco. (MIT Media Laboratory, Massachusetts Institute of Technology). *Agent Stories Part II*. AUP Report. Cambridge, MIT Media Lab. 2000 May 22. 22p.

Terwilliger, Robert B. and Polson, P. *Relationships Between Users' and Interfaces' Task Representations*. CHI '97. Human Factors in Computing Systems, (1997), ACM.

Tiongson, Phillip Rodrigo. *ActiveStories: Infusing author's intention with content to tell a computationally expressive story*. MIT MS Thesis, 1998.