BRANCHING DIALOGUE SYSTEM DEVELOPMENT: HARNESSING THE POWER AND COMPLEXITY OF VERBAL COMMUNICATION IN DIGITAL GAMES AND SIMULATIONS

by

Brad Hennigan

THESIS

Presented to the Faculty of

The University of Texas at Dallas

in Partial Fulfillment

of the Requirements

for the Degree of

MASTER OF FINE ARTS IN

ARTS AND TECHNOLOGY

THE UNIVERSITY OF TEXAS AT DALLAS

August 2012

ACKNOWLEDGMENTS

Many thanks to Dr. Marjorie Zielke, for taking a chance on a purely artistic type who knew little about the gaming world three years ago; to my colleague and friend Professor Thomas Riccio, for constant inspiration and support; to my colleague and friend Gary Hardee for the many long brainstorming sessions; and to Dr. Thomas Linehan and the faculty and staff of the UTD Arts and Technology Department.

This work is dedicated to my mother, Mary Elizabeth Starcher, who has never stopped believing in me.

August 2012

BRANCHING DIALOGUE SYSTEM DEVELOPMENT:

HARNESSING THE POWER AND COMPLEXITY OF VERBAL COMMUNICATION

IN DIGITAL GAMES AND SIMULATIONS

Publication No._____

Brad Hennigan, MFA
The University of Texas at Dallas, 2012

Supervising Professor: Dr. Thomas Linehan

ABSTRACT:

As computational capability continues its rise in accordance with Moore's Law, the tools available to designers of digital games have become more robust, allowing high fidelity graphics and sound to become common, and resulting in a market saturated with kinetic-based games. However, consumers, and more recently educators, are eschewing such games for more complex and immersive stories, the creation of which has proven a more difficult mountain for designers to climb. A central reason is that story-immersive games rely on dialogue between the player character and non-player characters (NPCs). "Good" dialogue requires a high degree of *dynamic immersion*, which takes into account where the player has been, with whom she has spoken, and how her conversations affect the narrative of the game. The research documented herein examines elements of human communication, and how these elements are replicated in video games and simulations. Also presented is a methodology for writing dynamically immersive branching dialogue, as well as current research into proposals for improving upon that methodology.

TABLE OF CONTENTS

LIST OF FIGURES

CHAPTER ONE

INTRODUCTION

Ray Kurzweil in 1990 put forth an earnest proposition that "given sufficient capacity and the right techniques, our machines may ultimately be able to replicate human intelligence" ("Intelligent Machines" 13). Giving credence to his propositions would be wise: in 1999 Kurzweil stated that by 2009, "most purchases of books, musical 'albums,' videos, games and other forms of software do not involve any physical object, so new business models for distributing these forms of information have emerged" ("Spiritual Machines" 195). Considering the iPod®, iPhone®, iPad® and iTunes®, the late Steve Jobs and his cohorts at Apple Computer must have been Kurzweil fans. Many other researchers have also taken up Kurzweil's proposition to replicate human intelligence, directing their research toward quantifying human experiential phenomena in hopes of breaking them down into data more easily understood and reproduced by computer science.

One aspect of the human experience that has come under research scrutiny is human communication and the challenges inherent in replicating the exceedingly nuanced elements of nonverbal communication (gestures, facial expressions, body language), and verbal communication (language, syntax, semantics, contextuality). This work seeks to examine some of the relevant research into the human communication aspect of replicating human intelligence, specifically nonverbal communication modalities and natural language

processing; as well as burgeoning research in the area of narrative, or storytelling, and its emergence in serious digital games and simulations. Also presented is this author's research into branching conversation and dialogue systems designed to facilitate sustained player agency during game play through consistent maintenance of *dynamic immersion*.

Digitizing Human Communication

Human communication is highly complex. Through verbal and non-verbal means, humans are able to achieve a level of clarity in communication that sets them apart from all other animals on earth. Nonverbal communication, also known as body language, encompasses everything other than written or spoken language: gestures, facial expressions, physical response to stimuli, and more make up this most ancient means of human interaction. Body language is ingrained, a part of human nature and the basis of human communication (Furnham and Petrova 1).

For all of its complexity and nuance, nonverbal communication cannot compare to the richness, flexibility, specificity, and intricacy of the spoken language. And in fact verbal and nonverbal communication, seemingly disparate methods of conveying meaning, work intimately together to form the whole of human communication, to the extent that attempts to distinguish them are not usually successful (Knapp and Hall 11). A rift, however, has occurred when it comes to digital games and simulations: graphical representations of the physical world, and its occupants and their physical behavior, have proven easier to achieve, and indeed have become robust and ubiquitous; while representations of human face-to-face interactions that rely on narrative elements of spoken language, memory, and context have

proven a greater challenge in the quest to replicate human intelligence. Chapter Two further discusses nonverbal and verbal communication, as well as graphical and face-to-face digital representations, and their place in digital games and simulations.

Kurzweil's own decades-long research into natural language processing has yielded somewhat impressive results: the first commercially marketed large vocabulary speech recognition system in 1987; the first speech recognition dictation system for Windows in 1994; and the first "host / hostess" avatar on the web, called Ramona, to combine a lifelike photo realistic, moving and speaking facial image with a conversational engine in 2001 (Kurzweiltech.com). Yet these systems are limited in their application, as Tina Klüwer has observed: "chatbots" such as Ramona do not actually understand user input, and thus must rely on huge amounts of data to generate a response. By contrast, spoken dialogue systems use complex parsing of input into syntactic and semantic data in order to "understand" the input, which is then responded to accordingly (11). Research into natural language processing and spoken dialogue systems has made great strides recently, and is discussed in Chapter Three.

Human Communication in Digital Games and Simulations

These issues of semantic and contextual understanding of user input are only part of the picture in regards to replicating human communication in digital games and simulations. As discussed earlier, because high fidelity graphical representations of a game world and its occupants were more readily achieved, and because contextually and semantically relevant stories and conversations were more difficult to master, the first two decades of content

produced by the video game market put aside dialogue and story in favor of purely kinetic experiences. DOOM, released by id Software in 1993, is a first-person shooter game that takes its simple objective from early arcade games such as *Space Invaders*: shoot everything and anything before it kills you (Mäyrä 104). DOOM and its many offshoots rely solely on kinetic movement through the game world, with little if any reliance on plot, story, or dialogue. Such games proved wildly successful for their producers: Gamespy ranked the original DOOM number one in its top fifty list of games in 2004, declaring it "nothing short of a revolution" (Gamespy.com). This fascination with first-person shooter games delayed any market-driven impetus behind investment in the development of branching narrative and dialogue systems.

The need for such systems was recognized, however, in a section of the gaming world known as serious games, a term coined by Clark Abt for his 1970 book *Serious Games* (Abt). While Abt's use of the term referred to board and card games, current serious games refer to a digital game subset that has a non-entertainment purpose, which are being developed in many fields including healthcare, education, the workplace, and military (Cannon-Bowers and Bowers). Since serious games often attempt to educate in the arena of social interaction, the lack of systems for recreating key aspects of social interaction, narrative and conversation, has posed a unique problem for designers. The need for, and development of, narrative and conversation systems in serious games is discussed in Chapter Four.

Real World Application: The First Person Cultural Trainer

Currently in its fourth year of development and sponsored by the TRADOC G2 Intelligence Support Activity (TRISA), the University of the Texas at Dallas Institute for Interactive Arts and Engineering's First Person Cultural Trainer (FPCT) relies heavily upon narrative and conversation systems to achieve its training objective (Zielke, "FPCT4+"). Such systems are most often designed by computer scientists, or programmers, with little if any input from non-programmer subject matter experts (SMEs) with experience in the analogue production of the designated digital system's output. While the expertise of the computer scientist in designing digital systems is obviously required, the lack of SME input can result in systems that are understood and usable by computer science experts, but are more difficult for analogue SMEs to master.

Such was the case with FPCT's proprietary branching conversation system, Conscript™, which proved difficult to master for the writers tasked with producing the trainer's branching dialogue. While Conscript™ ultimately proved to be a robust tool for writing branching dialogue, it did not provide tools for creating and mapping the branching narrative, or story, of the training simulation. Since a satisfactory methodology specific to the creation and mapping of branching narrative and dialogue was not readily available at the time, this author created one. The results of this research, as well as a discussion of the problems inherent in narrative and branching dialogue systems designed without SME input, are outlined in Chapter Five.

Future Work

Research outcomes in the writing of narrative and branching dialogue in FPCT prompted further inquiry into the possibility of refining the methodology into an automated digital system that takes into account the reality that most SMEs in the writing field do not have programming expertise. Currently in early design phase, Visual Conscript™ aims to be a tool for branching narrative design, conversation tracking, and dialogue authoring (Zielke "FPCT4+"), with a research goal of allowing SMEs without programming knowledge the ability to simply create the story, the characters, and the dialogue, while Visual Conscript™ creates and codes the conditional logic needed for proper game play in the background. Chapter Six includes a discussion of Visual Conscript™, as well as concluding remarks.

CHAPTER TWO

HUMAN COMMUNICATION AND ITS REPLICATION IN DIGITAL GAMES AND SIMULATIONS

A 2010 release by Rock Star Games called *L.A. Noire* took the representation of human communication in video games to a new level. Developed by Team Bondi, the game incorporates the expected high fidelity in its graphical representation of 1947 Los Angeles, but also uses "groundbreaking new animation technology that captures every nuance of an actor's facial performance in astonishing detail" (L.A. Noire). During game play, the minute detail available through this animation technique allows the player, acting as a detective, to determine if the person she is interviewing is lying or telling the truth, the proper determination of which either wins the game, or prompts the player to try again. The animation technique used is an extension of motion capture, thus the facial expressions were originated by professional actors. The actor's voices were also recorded for the game's dialogue, with the end result being a remarkable representation of both verbal and non-verbal human communication in a digital game. And according to Gamespot editor Carolyn Petit, the new approach works: "the richness of these details in L.A. Noire makes rummaging around grisly crime scenes and perusing the personal effects of victims a compelling process" (Gamespot).

Figure 1. Facial expressions in *L.A. Noire* tap new heights in fidelity (Team Bondi).

Conversational Agents and Applied Linguistics

Commercial successes such as *L.A. Noire* were made possible by decades of research into the digital recreation of human verbal and nonverbal communication. Of particular interest in the field is the work of Dr. Justine Cassell, currently Director of the Human-Computer Interaction Institute at Carnegie Mellon University. Cassell's work is given focus herein because of her unique background in exploring human communication in all its forms, and her choice to apply that knowledge to the replication of human communication in the digital realm. A dual PhD in Psychology and Linguistics, Cassell's desire is not only to create Kurzweil's intelligent machine, but to create "a machine that *evokes* humanness in us - a machine that acts human enough that we respond to it as we respond to another human" ("Body Language" 365).

One of Cassell's early appointments was as Director of MIT Media Lab's Gesture and Narrative Language Group, which sought to create avatars that "can be designed with psychosocial competencies, based on a deep understanding of human linguistic, cognitive, and social abilities (GNL)." Under her tutelage, the group made field-altering strides in algorithm-based models for bodily and facial nonverbal communication in conjunction with verbal communication, as applied to and generated for what Cassell terms "embodied conversational agents" (Embodied 71). Cassell and colleagues Timothy Bickmore, Hannes Vilhjálmsson, Mark Billinghurst, Lee Campbell, Joey Chang, and Hao Yan started with a bang, proposing in 1999 the computational and architectural requirements for systems which support real-time multimodal interaction with an embodied conversational character (Cassell, "Architecture"). The group was motivated by their understanding that "human face-to-face conversation is a complex phenomenon involving understanding and synthesis across multiple modalities and time scales", and that "speech, intonation, gaze, and head movements function not just in parallel, but interdependently" (Cassell, "Architecture" 1). To create a computer character which could participate in face-to-face conversation, Cassell's team identified these architectural controls:

> *Multi-Modal Input and Output*: the system must replicate human sending and receiving of multi-modal information.
>
> *Real-time*: the system must reflect the parallel functionality of human interaction by allowing different processes to function simultaneously, but at different timescales.

> *Understanding and Synthesis of Propositional and Interactional Information*: dealing with propositional information requires both a static domain knowledge base and a dynamic discourse knowledge base, while presenting propositional information requires modules for planning and managing output.
>
> *Conversational Function Model*: a model which explicitly represents conversational functions across all modules, resulting in a symmetrical input/output architecture.
>
> *Modularity and Extensibility*: the system must expect revisions as the research into face-to-face communication yields new information. (Cassell, "Architecture" 3)

Cassell and her team implemented their architecture in the embodied conversational agent Rea (Real Estate Agent), "a computer generated humanoid that has a fully articulated graphical body, can sense the user passively through cameras and audio input, and is capable of speech with intonation, facial display, and gestural output" ("Architecture" 6). From her experiences with Rea, Cassell concluded that additional research into "conversational competencies" ("Architecture" 8) was warranted.

Cassell and Bickmore, with colleagues Yukiko Nakano, Candace Sidner, and Charles Rich continued their research by seeking empirical support for the correlation of nonverbal behaviors such as hand gestures, head nods, eye gaze, and posture shifts with "the underlying conversational structure and information structure of discourse," the understanding of which "enables improvements in the naturalness of embodied dialogue systems, such as embodied

conversational agents, as well as contributing to algorithms for recognizing discourse structure in speech-understanding systems" (Cassell, "Cues" 1). Their conclusions found a clear relationship between nonverbal behavior and discourse state, and yielded the type of quantifiable data around which a digital system for replication could be built.

Cassell's growing body of work was further supplemented by research into small talk and conversational storytelling, which allows speakers to achieve goals such as "developing a relationship (e.g., befriending, earning trust), establishing their reputations or expertise, or persuading their listeners to take some course of action" (Bickmore, 1); as well as external manifestations of trustworthiness, which Cassell views as "the loom on which is woven the social fabric of society" ("Trustworthiness" 50). These aspects of human communication were deemed important because of the effect they may have on an embodied conversational agent's interactions with humans. Successful replication could "lead the users of technology to judge the technology as more reliable, competent and knowledgeable – to trust the technology more" (Cassell "Trustworthiness" 50). These conversational competencies were incorporated into Rea.

Procedural Generation: BEAT

As stated earlier, the successful video game *L.A. Noire* chose to use an advanced form of motion capture to animate its characters. Cassell and her colleagues identified such techniques as expensive and time consuming, and set out to create a system for the procedural generation of the animations of verbal and nonverbal communication modalities their research had deemed mandatory to replicating human communication as a whole. The

resulting collaboration with researchers Bickmore and Vilhjálmsson yielded the Behavior Expression Animation Toolkit, or BEAT, which "allows animators to input typed text that they wish to be spoken by an animated human figure, and to obtain as output appropriate and synchronized nonverbal behaviors and synthesized speech in a form that can be sent to a number of different animation systems" (Cassell, "BEAT" 1).

BEAT seeks to not only identify and animate appropriate gestures, facial displays, eye gaze, head movements, and body postures in accordance with textual input, but to do so in a properly orchestrated manner, creating a "tight synchrony among communicative modalities," without which "satisfaction and trust in the outcome of a conversation is diminished" (Cassell, "BEAT" 2). BEAT was built in Java, with modularity and extensibility in mind, and to that end uses XML as its primary data structure. Within its operating structure, "processing is decomposed into modules which operate as XML transducers; each taking an XML object tree as input and producing a modified XML tree as output" (Cassell, "BEAT" 3).

In an elaborate input to output pipeline, the process begins with a language tagger, which processes the textual input and tags it with data from object and action knowledge bases; this information generates an XML tree which is passed on to a behavior generator, which applies generator rules to annotate the tree with appropriate behaviors; and the modified tree is passed on to a behavior scheduling module that compiles the XML tree into an action plan ready for execution by an animation engine. The knowledge bases, XML

transducers, behavior generators, and filters in BEAT allow animators or other subject matter experts to further define its input to output pipeline (“BEAT”).

While robust and extensible, BEAT has some shortcomings, most notably in its linguistic analysis functionality. Cassell points out that “computational linguistics is still an imperfect science, and BEAT's output behaviors can only be as perfect as the linguistic results they are based upon” (“BEAT” 9). In addition, the design choice to support extensibility causes run-time performance that might not be appropriate for real-time applications. Despite some flaws, BEAT is an impressive attempt at procedural animation generation of nonverbal communication modalities in synchronization with verbal communication modalities in embodied conversational agents, and moves research further toward Cassell’s goal of imbuing these agents with “precisely-described and well-motivated characteristics of human conversation” (“Embodied” 72).

Figure 2. The BEAT avatar before and after textual input (Cassell, “BEAT” 8)

As previously stated, Cassell’s work has had a distinct influence on the field of replicating human communication in embodied conversational agents. It would be remiss, however, to conclude this chapter without acknowledging some of the research that may

have inspired her. In imagining and building the Rea architecture, Microsoft Research's Persona Project (Ball), and Beskow and McGlashan's "Olga" (Beskow) were referenced. For determining systems to integrate conversational competencies into the Rea architecture, the works of Elisabeth André, Thomas Rist, and Jochen Müller (André, et al "AI") (André, et al "Integrating") were referenced. Cassell's BEAT system was influenced by Perlin and Goldberg's "Improv" (Perlin) and Becheiraz and Thalmann's system for animating agents based on emotion (Becheiraz).

CHAPTER THREE

ADVANCEMENTS IN NATURAL LANGUAGE PROCESSING

Ken Jennings had won more consecutive *Jeopardy!* games than any human in the history of the iconic answer-and-question television quiz competition, a total of seventy-four in a row. Yet on February 16, 2011, after three grueling days of answers and questions, Mr. Jennings wrote on his digital answer pad, in addition to his correct Final Jeopardy question, "I, for one, welcome our new computer overlords." He was referencing the winner of this particular tournament: IBM's supercomputer Watson, which had handed Mr. Jennings and another top *Jeopardy!* winner, Brad Rutter, a sound defeat (Markoff).

Figure 3. The final score: IBM's supercomputer Watson wins a 3 day *Jeopardy!* tournament (Markoff)

Three years earlier, researchers at IBM had set out to establish a research challenge to rival its success with Deep Blue, the IBM supercomputer that defeated Russian chess master Gary Kasparov in a six-game match in May of 1997 (Pandolfini). Part of the impetus behind Watson was IBM's recognition that current natural language-based question-answering, or QA, technology, such as that used by Google, yields results based on popularity and relevance to only a few keywords within the query. IBM researchers determined that enterprise systems desired a deeper analyzing of *all* relevant data in hopes of yielding a more precise answer that could be readily justified. Moreover, these researchers found the problem "attractive as it is one of the most challenging in the realm of computer science and artificial intelligence, requiring a synthesis of information retrieval, natural language processing, knowledge representation and reasoning, machine learning, and computer-human interfaces" (Ferrucci, et al 60).

It was a challenge that had yielded a half-century of only promising results. Prior to the mid-1960s, language data processing was mired in the paradigm that words are the units of meaning, and thus language data processing could be adequately achieved through a Boolean combination of words, a paradigm that repeatedly proved inadequate (Simmons). In 1965, Noam Chomsky changed the paradigm by breaking down linguistic analysis into syntactic, semantic, and phonological components:

> The syntactic component consists of a base and a transformational component. The base, in turn, consists of a categorial subcomponent and a lexicon. The base generates deep structures. A deep structure enters the semantic

> component and receives a semantic interpretation; it is mapped by the transformational rules into a surface structure, which is then given a phonetic interpretation by the rules of the phonological component (141).

Despite the seeming clarity with which Chomsky's breakdown asserts that these components are interdependent, work based upon this new paradigm tended to concentrate either on the syntactic component, or the semantic component. To some, the division of syntax and semantics raised serious issues, as it was clear that the process of comprehension used both, and could not exist if both were not present. Natural language processing researcher Christopher Riesbeck noted in 1975 that "false problems arise when the comprehension process itself is sectioned off into weakly communicating sub-processes, one of which does syntactic analysis, and the other of which does semantic" (11). Dissatisfaction with this subdivision was largely ignored, however, mainly because syntactic analysis lent itself more readily to computational methods, and thus attracted more researchers (Riesbeck).

Syntax or Semantics: The Road More Traveled

Kurzweil's work in natural language processing was heavily skewed toward the syntactic analysis side, concentrating almost exclusively on speech recognition systems. These systems have been segregated into a natural language processing subset known as Automatic Speech Recognition, or ASR, an area which has enjoyed many significant advances over the last two decades (Baker, et al). ASR technology can be found in a multitude of commercial applications, mainly in the customer service-via-telephone arena. Recently, Apple Computer's I-Phone 4S introduced Siri, an Intelligent Personal Assistant

that can "understand" its user's questions and give relevant answers. Siri does not, however, truly semantically understand user input. It operates within a limited framework of human informational needs, such as dining, sporting events, movies and entertainment, weather, travel, etc., and uses ASR technology to determine the user's specific need. What does set Siri apart is its use of databases and web services to present relevant responses to user input (Staska).

While Siri is an important and impressive step forward in expanding the usability of ASR technology, it does not tackle the question of semantic, contextual understanding of user input. As Kurzweil recently stated, "humans…have been unique in our ability to think in a hierarchical fashion, to understand the elaborate nested structures in language, to put symbols together to form an idea, and then to use a symbol for that idea in yet another such structure…, but computers have still not shown an ability to deal with the subtlety and complexity of language. That is, until now" (Kurzweil, "Significance"). The now to which Kurzweil refers is IBM's Watson.

DeepQA

The team of 20 IBM scientists behind Watson had a formidable task: to advance and incorporate QA technologies such as parsing, question classification, question decomposition, automatic source acquisition and evaluation, entity and relation detection, logical form generation, and knowledge representation and reasoning into a single entity; and to present that entity within the framework of a nationally syndicated game show as one of three contestants requiring confidence and precision in answering complex natural language

questions over a broad domain of topics, all within an average answer time of three seconds per question (Ferrucci, et al). Understandably, they dubbed the project DeepQA, a "massively parallel probabilistic evidence-based architecture…[using] more than 100 different techniques for analyzing natural language, identifying sources, finding and generating hypotheses, finding and scoring evidence, and merging and ranking hypotheses" (Ferrucci, et al 68). More importantly, the techniques combine to bring their strengths to bear for improvement in accuracy, confidence, and speed.

DeepQA runs on more than 2,500 core processors, necessitated by its massive parallelism. It was developed using the Unstructured Information Management Architecture, or UIMA (Ferrucci and Lally) and uses asynchronous messaging via UIMA-AS to communicate among the processors (Ferrucci, et al). The DeepQA architecture has 4 overarching principles:

> *Massive parallelism*: Exploit massive parallelism in the consideration of multiple interpretations and hypotheses.
>
> *Many experts:* Facilitate the integration, application, and contextual evaluation of a wide range of loosely coupled probabilistic question and content analytics.
>
> *Pervasive confidence estimation:* No component commits to an answer; all components produce features and associated confidences, scoring different question and content interpretations. An underlying confidence-processing substrate learns how to stack and combine the scores.

Integrate shallow and deep knowledge: Balance the use of strict semantics and shallow semantics, leveraging many loosely formed ontologies (Ferrucci, et al 68).

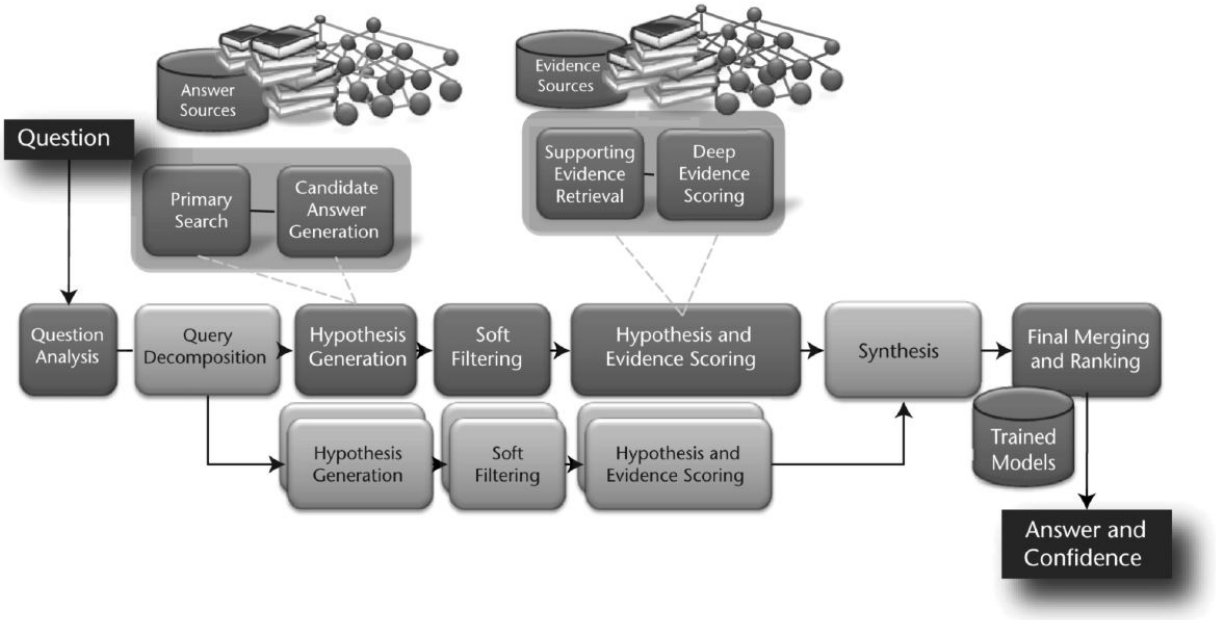


Figure 4. IBM's DeepQA architecture. (Ferrucci, et al 69)

Watson's achievement against its human opponents, both of whom are proven aces of the game, and both of whom he beat by an average 3.5 to 1 score, is at minimum impressive, and more accurately remarkable considering that such semantic and contextual analysis in natural language processing had been considered out of reach a decade ago. A fair question at this juncture might be: what does Watson and its DeepQA architecture have to do with digital games and simulations? The answer lies first in acknowledging that Watson and DeepQA are significant advancements in natural language processing; as Kurzweil observed,

"it certainly demonstrates the rapid progress being made on human language understanding" (Kurzweil, "Significance").

Second, one must look to the future of games and simulations to understand the impact of semantically and contextually accurate natural language processing: no longer will the player be restricted in dialogue choices to what the game's writers have deemed important to moving the game's plot forward, but rather the player will be able to simply talk to the game and its characters, and they will respond appropriately. Natural language processing opens up the narrative in games to the same potential level of high fidelity that their graphical representations have enjoyed for decades. It also potentially creates many unique challenges in the creation and writing of narrative and dialogue. Comfort can be taken in the knowledge that, though Watson was built to play *Jeopardy!*, it's creators designed it with extensibility in mind, and have already begun experimenting with "different business applications and additional exploratory challenge problems including medicine, enterprise search, and *gaming* [emphasis added]" (Ferrucci, et al 68).

CHAPTER FOUR

RECENT DEVELOPMENT IN NARRATIVE AND CONVERSATION SYSTEMS

It has been well established that narrative in games and simulations is essential to player immersion and positive learning outcomes (Fisch) (Marchiori) (McQuiggan) (Waraich). Yet inconsistencies within narrative and dialogue in games and simulations are all too common. As researchers Janis Cannon-Bowers and Clint Bowers note, game developers rarely release games with major, known graphics issues, but narrative problems are often overlooked and accepted. This acceptance is at odds with the goals of serious and educational games, for "if we are to subscribe to the notion of using serious games to better immerse, transport, motivate, and educate the player within a certain educational domain, then we must also consider the potential impact of including a story with incomplete or incongruent information within a specific game environment" (27).

To understand the need for narrative and conversation systems in serious games, one must first attempt to define a serious game. Clark Abt begins by defining a game itself: "reduced to its formal essence, a game is an activity among two or more independent decision-makers seeking to achieve their objectives in some limiting context. A more conventional definition would say that a game is a context with rules among adversaries trying to win objectives" (Abt, in Cannon-Bowers and Bowers xiv). In the case of digital

games, one of the two independent decision-makers can be the game itself, or a digital "player" created by the game. Add to this mix a training or educational outcome, and a basis in something other than pure entertainment, and a serious game is born. Because serious games can possess both entertainment value and the possibility of lessons learned, gamers, serious educators, and even parents are embracing their potential (Cannon-Bowers and Bowers 290). Unlike kinetic-centered commercial games, many training and educational serious games center around social interaction, and by nature require human verbal and nonverbal communication (narrative and dialogue) replication as a game mechanic. These in turn generally require the skills of a writer.

Writing for games can be tricky, particularly if the game world is "open," meaning the player is free to roam anywhere within the boundaries of the world and speak to whomever she chooses. Narrative game designer Mary DeMarle describes the potential for the writer's conundrum in such a world:

> A writer walks into a design meeting at We Make Fantasy Games, Inc., an independent game developer….She starts her pitch. She's barely five minutes into her description of how the hero stumbles upon a nondescript cabin in the woods and goes inside…, when the lead game designer interrupts her.
>
> "Hold on. What if the player doesn't go in the cabin?"
>
> "Sorry?"
>
> "I asked, what if the player doesn't go in the cabin?"
>
> "He has to go in the cabin."

> "Well, what if he doesn't? What if he decides to check out the next town first? Or what if he decides to blow the cabin up?"
>
> "…Why would he do that?"
>
> "Because he can." (71)

Clearly DeMarle's upstart writer had not thought about the myriad of possibilities available to the player at any given moment in an open-world game, nor had she begun to imagine how to accommodate those possibilities. Her example illustrates the need for tools to develop and map branching narrative, or story; and tools for mapping and writing the branching dialogue that accompanies and feeds such a narrative.

The Dynamic Automated Story Engine

Culture and populace based games such as FPCT require character- and conversation-based game play, yet the complexity of both story creation and branching conversation development can be challenging for subject matter experts. This dilemma has spawned research aimed at producing tools to bring SME knowledge to virtual environments quickly, reducing reliance on external help for story generation or computer programming. To date, research has concentrated on two areas: automatic interactive story generation and author-goal based interactive story generation (Zielke, "FPCT 4+").

Automatic Interactive Story Generation

Automatic interactive story generators attempt to eliminate the author altogether and create stories by breaking narrative into its atomic parts, then reconstructing the parts into new narratives. Currently, several automatic story generators exist, with the most common

issue being the difficulty in generating narratives that maintain plot integrity and hold user interest over a long period of time, both of which are integral to interactive narratives in open-world environments.

The Oz Project (Bates) and the Façade system (Mateas and Stern) use heavily pre-defined plot graphs requiring significant programmer input to achieve even short narratives. Using Facade, in particular, it took two man-years to create one fifteen minute experience. Similarly, DEFACTO (Sgouros), IDtension (Szilas) and Mimesis (Young, et al), though scalable, require significant content creation and ordering for longer narratives. Another system, INTALE (Riedl and Stern), requires all of the narrative's endings to be pre-defined, and thus is reliant on a large amount of pre-programming. Researcher Ruth Aylett's project, FearNot!, is also based upon ordering and predefinition to achieve its goal of utilizing automated interactive narrative generation to teach students how to deal with bullying (Aylett).

Researchers Heather Barber and Daniel Kudenko have attempted to tackle the issue of continuous, engaging interactive narrative generation with their project GADIN (generator of adaptive dilemma-based interactive narratives). Though GADIN also requires predefinition, once the generator is running it continuously generates narrative through offering the user dilemmas, the response to which determines the next direction of the narrative. However, the amount of predefinition required in the GADIN system is in direct correlation with the number of characters in the narrative. The higher the number of characters, as might be the

case in a large open-world narrative, the more predefinition is required, to the extent that it may be considered prohibitive (Barber and Kudenko).

Author-goal Based Interactive Story Generation

Author-goal based interactive generation embraces the input of the author, while automating some aspects of the creation of a branching narrative. Researchers at the Georgia Institute of Technology created the declarative optimization-based drama management (DODM) system, which guides the player through the game world by projecting possible author-created future stories and reconfiguring the world based on those projections. However, "the problem that immediately arises is that actually performing a complete search over all possible future combinations of DM actions and plot points is computationally infeasible because the search space's size grows exponentially with the story's size" (Nelson 37).

One of the Georgia Tech researchers, Michael Mateas, in corroboration with James Skorupski of UC Santa Cruz, has also concentrated on creating author-goal based generators that require less technical expertise, allowing non-technical writers access to the world of branching narratives. Mateas and Skorupski set about creating an author-goal based story generation tool that eschews the typical need for "technical expertise in computational models of story planning and structure, as well as the knowledge to formulate compelling plot arcs, rich dialogue, character conflicts and other story elements" (Skorupski and Mateas 1). Their resulting software is called Wide Ruled.

Wide Ruled is based on the UNIVERSE model of hierarchical structure and multiple paths to goal accomplishment. It provides a means for SMEs to create the elements of a story, such as goals, characters, and plot fragments, and the program determines the best story progression according to the variables the SME chooses and inputs as illustrated in Figure 3. Within a Wide Ruled story, Author Goals are the primary unit of story planning, with optional parameter variable inputs. Each of these relates to one or more Plot Fragments that describe a set of actions that fulfill its parent author goal (Skorupski & Mateas). Story generation begins with an initial author goal, which randomly selects among all executable plot fragments with valid preconditions, and then sequentially executes all of its contained story actions to successfully complete a story (Zielke, "Tools & Techniques").

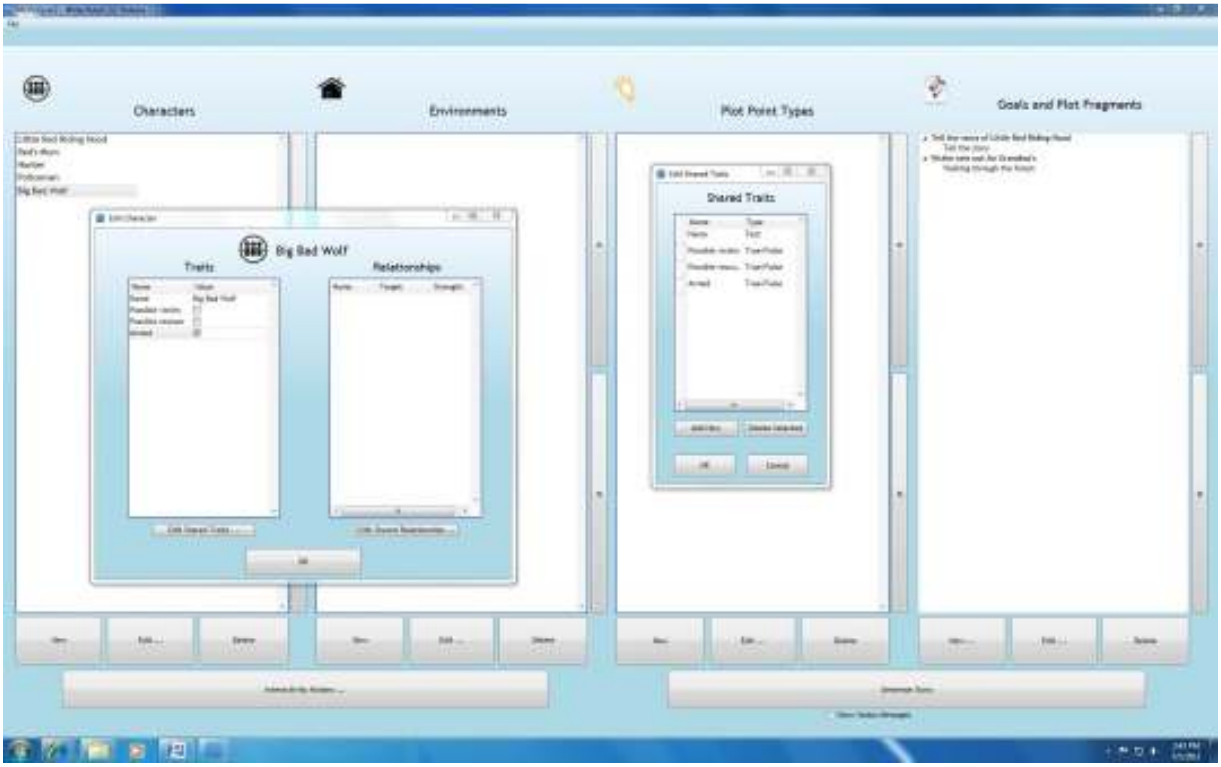


Figure 5. The Wide Ruled editor interface, allowing author-goal based story planning through SME chosen characters, plot fragments and variables. (Skorupski)

Branching Dialogue Systems

Ruth Aylett and her colleagues have established that "if characters are to interact intelligently and meaningfully among themselves, their different potential relationships with each other must be thought through and their place in the world must be clearly established" (43). To achieve this level of meaningful interaction, FPCT utilizes a branching complexity within its narratives and conversations. FPCT terms this complexity *dynamic immersion*, which plans for all potential PC movement and conversations throughout the game world. Dynamic immersion accommodates player agency, allowing for realistic, non-linear game play which simulates the feel of decision-making and self-determination, and provides the opportunity for distinct and unique teaching moments. (Zielke et al, "FPCT4+")

FPCT conducted extensive research into a viable methodology for writing and coding dynamically immersive, complex, non-linear branching dialogue. The need for such a methodology is evident; as Wizards of the Coast Senior Developer Frank T. Gilson states, "bad dialogue will jar the participant out of the experience. Great dialogue will engage and motivate. The challenge is to *prepare* dialogue to account for the interactivity of gameplay." (in Krawczyk & Novak 158) Preparation is the key: a game narrative utilizing ten characters and a three-choice branching system can have as many as three to the tenth power, or 59,049, unique conversation branches. While this total will not likely be reached due to dead-end or win-state paths, the writing of half this number could easily be required. FPCT concluded that the dialogue aspect of the complex branching narrative has not been researched to the

extent that story generation tools have, nor to the extent that nonverbal communication replication has. One software suite by Urban Brain Studios, Chat Mapper, is the exception.

Chat Mapper is a dialogue-based authoring tool that automatically organizes and links dialogue segments within larger conversations between a pre-determined player character and various NPCs created by the SME as illustrated in Figure 6. Chat Mapper allows branching narratives to be constructed from the standpoint of character and dialogue, and "reminds" the author of all character linkages previously determined. Once the story is authored, the SME (or a technical expert) must then script the variable and conditional logic that the narrative requires. Chat Mapper utilizes the Lua scripting language for programming the logic, as the software's designers determined that Lua is easily integrated into various gaming engines (Zielke et al, "Tools & Techniques").

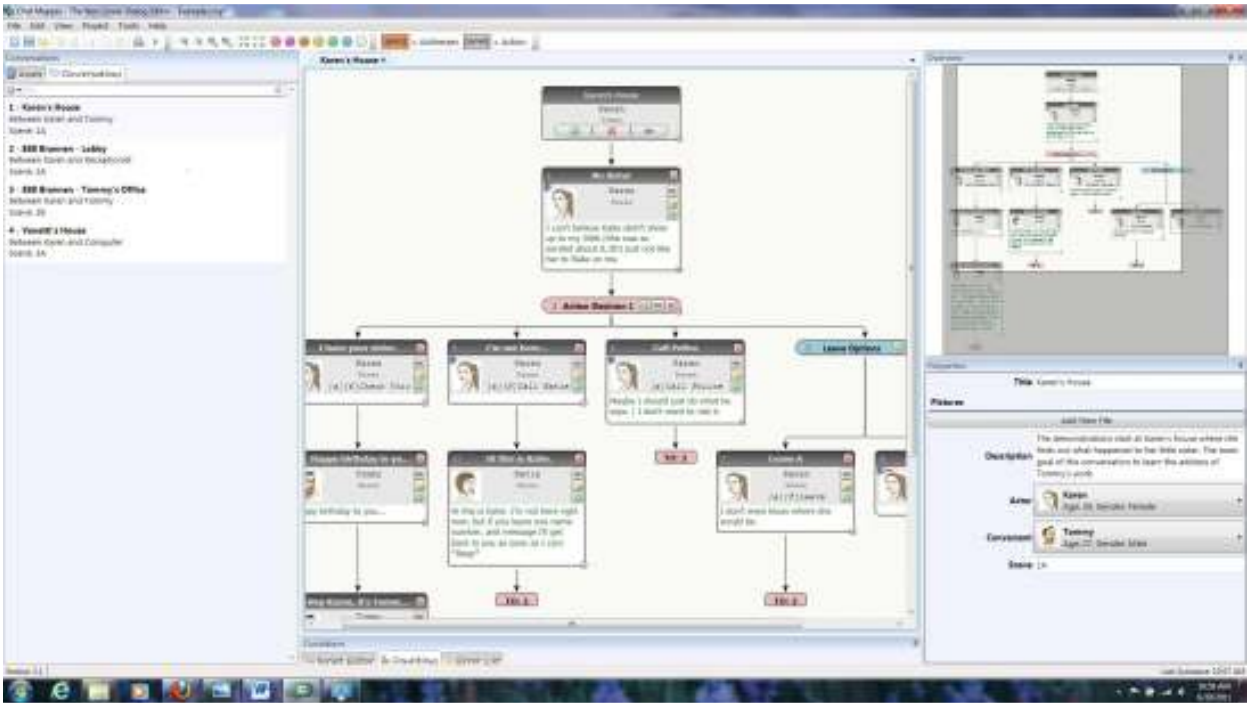


Figure 6. The Chat Mapper branching dialogue system user interface,

Chat Mapper succeeds in assisting the author in keeping track of multiple dialogue branches within a single conversation; the author can easily retrace her steps to verify that the dialogue remains contextual, and thus immersive. Chat Mapper does not assist the author in keeping track of multiple conversations within a large branching narrative. Its lack of a means to easily reference the player's potential pathways through the game gives rise to the possibility of inconsistency in dynamically immersive branching dialogue throughout the narrative as a whole. Chat Mapper also does not provide a mechanism for creating and/or tracking the branching story, or plot lines, that are the backbone of dynamically immersive branching dialogue.

CHAPTER FIVE

TOWARD A METHODOLOGY FOR WRITING DYNAMICALLY IMMERSIVE BRANCHING NARRATIVE AND DIALOGUE

A natural language processing system for games and simulations that allows the player to speak freely to the game's characters, asking whatever questions may come to mind, is of course the gaming holy grail. Like IBM's Watson, such a system may use data mining of web-based sources to determine the best possible response that fits within the parameters of the game's narrative. Until such a system is implemented, dialogue in games and simulations must be authored, which poses a constant challenge to the most widely accepted digital open-world gaming paradigm: that "the essential difference between traditional stories and game stories is putting authorship into the hands of the player" (Krawczyk and Novak 89).

In order to meet this challenge, the game's authors must create for the player the illusion of self-determination. Kevin Saunders, lead designer at Obsidian Entertainment, describes this illusion:

> For games with non-linear (or seemingly non-linear) stories, the primary difference is interactivity. The player is provided with choices of how to proceed. These options need to make sense and have to be entertaining and worthwhile. In traditional storytelling, the writer decides how the main

> character will act in each situation, while in games…the player must always feel that their decisions are driving the story. (in Krawczyk and Novak 89)

In games that have the objective of education or training in areas of social interaction such as FPCT, the choices given the player arise most frequently out of the branching dialogue system, the mishandling of which in turn gives rise to the potential for illusion-shattering conversational errors that remove the player from being immersed in the game.

The Role of Memory

Imagine a customer walking down an aisle at the local do-it-yourself store, when a cheerful young employee looks at her and smiles, welcomes her to the store and asks if she needs help finding something. She tells him what she needs, which he informs her is located at the back of the store, and off she goes. On her way back to the registers, she walks down the same aisle and finds the same cheerful employee. He looks up at her and smiles, and just as she is about to thank him for his help, he welcomes her to the store and asks if she needs help finding something. She is immediately disconcerted, that out-of-body feeling of déjà vu creeps through her, making her skin crawl. She asks herself in a brief moment of panic, "Am I going crazy?"

The experience described above is the exact opposite of dynamic immersion, because the interaction lacks the essential element of memory. Human communication can be viewed as intrapersonal, interpersonal, and public/socio-cultural interdependent systems (Mortensen). Memory is integral to these communication systems; they require the memory

components of retention and recall to properly function. Within these systems, memory serves three broad purposes:

1. It acts as a repository for the experiences, concepts, and words (the rhetorician's "available means of persuasion") which are the raw materials of speech inventions.
2. It acts, in connection with processes of thinking and reasoning, as a setting for linking experiences and concepts with words to produce oral expression, or the generation and transmission of a message.
3. Perhaps most importantly, it acts as a vehicle for interpreting and evaluating messages, and for determining how one should respond to them (King 9).

As discussed previously, designers and builders of digital games and simulations have made great strides in the replication of the physical world: environments, objects, and characters. But far less attention has been paid to creating conversations for those characters, dialogue that properly replicates the role of retention and recall, or memory, in the communication process. One of the most successful commercial games of 2011, *Skyrim*, chose to depart from purely kinetic game play with a well-constructed, character-driven branching narrative. However, the overlooking of narrative issues pointed out by Cannon-Bowers and Bowers is prevalent in the game, particularly in regards to conversations and dialogue. For example, a mere ten minutes into the game experience, the player encounters a master of torture deep in the dungeons of Helgen's Keep. If the player speaks to him, he is

told to leave the torturer alone, unless the player would like to join him in his work. It is clear the conversation is over, and the player moves on.

If the player then explores the rest of the torture chamber, which holds several important objects, and then speaks to the torture master again (perhaps in hopes of prying more information from him), not only does the torture master give no indication of remembering the player, but in fact repeats the exact same dialogue, with the exact same intonation, gestures, and facial expressions as the first encounter (Skyrim). Dynamic immersion is lost, and as Frank T. Gilson stated earlier, the bad dialogue jars the player out of the game's experience.

Building Dynamic Immersion Through Memory Replication

With the need for retention and recall in conversations clearly established, the question becomes how best to integrate memory into a branching dialogue system. To answer this question one must first determine the possible situations that must be replicated within an open-world game in which memory serves a function. Keeping in mind that the player is free to move about and talk to anyone she chooses in such a world, planning for as many, if not all possible interactions becomes mandatory. Additionally, each function may have several layers of depth, and if only a few layers are accommodated, a more severe loss of immersion may occur when the player encounters a conversation error due to her previously established expectation of contextuality.

The following memory functions have been identified during the writing of FPCT:

- *The NPC knows the player*: At the beginning of each interaction with a NPC, the system must determine whether the NPC has already spoken to the player. This memory function can be highly complicated and many layers deep.
    - If the player is known, did the NPC send the player on an errand (to speak with another NPC, for example)? If so, was the errand completed?
    - If the player is known, how many times has the player returned to speak to the NPC? The dialogue should contextually recognize each repetition.
    - If the player is known, was information relevant to plot movement imparted to the player? If so, has the player acted on the information?
    - If the player is not known, is it possible the NPC has "heard" of the player's presence in the game world?
- *The NPC's emotional state*: The system must be able to alter the NPC's emotional state according to the dialogue. It must also keep track of that state if the player leaves and returns to the PC. For instance, if the player upsets the NPC before exiting the conversation, then returns later, is the NPC still upset with the player? Did "news" of something the player did improve or degrade the mood of the NPC?
- *Player actions during game play*: The system must keep track of what actions the player has executed during game play.

- Is the NPC aware of the player's actions? If so, contextual dialogue that acknowledges the actions and their influence on the game's narrative must occur.

To accommodate the need for dynamically immersive dialogue that allows for sensitivity to memory in FPCT, a branching dialogue system was developed in the first years of the project called Conscript™, which is detailed in the next section.

Writing Branching Dialogue for FPCT: Conscript™

Conscript™ is a proprietary branching conversation system created for FPCT, but designed as a stand-alone program with the capability of interfacing with multiple game development engines. Written in the C++ programming language, it is a linear logic system, meaning it scans the coding from top to bottom and left to right until it locates an executable command. It is also single-space indention dependent, meaning the coding must be precisely written with any "nested" elements indented uniformly. Conscript™ is text-based, with no graphical user interface. While these attributes are part of what allows the program to easily interface with virtually any game development engine, they also create a burdensome environment for writing the conditional logic required of dynamic immersion, as well as being susceptible to frequent indention-related issues.

To achieve dynamic immersion during game play, the conversation system must be able to track with whom the PC has spoken and what information she has received. Armed with this knowledge, the system then routes the PC to contextually relevant dialogue when she chooses to interact with any NPC. To this end, Conscript™ uses conditional logic to

check variables set within conversations during game play. Variables can be named anything, and are not limited in their value potential. The variables that have been set throughout the game's conversations are then checked in conditional logic "trees" that begin each character's conversation file. The following is a brief sample of a conditional logic tree in Conscript™:

```
condition pc["n5_knows"]
 =1
  condition npc["n5_ret"]
   =1
    goto topic "n5_ret-2"
   =else
    condition pc["seeks_n5"]
     =1
      goto topic "n5fn1_id"
     =2
      goto topic "n5fn3_id"
     =else
      goto topic "n5_ret"
 =else
  "…."
```

In this sample, the system first checks if NPC 5 knows the player. If that condition resolves to 1 (made so by the variable ["n5_knows"] set to a value of 1), the system then checks if the variable ["n5_ret"] has been set to a value of 1; if so, the player is routed to the conversation "n5_ret-2"; if not (or "else"), the system checks the variable ["seeks_n5"]. If that variable has been set to a value of 1, the player is routed to the conversation "n5fn1_id"; if it is set to a value of 2, the player is routed to the conversation "n5fn3_id"; and if set to neither, or "else", the player is routed to the conversation "n5_ret." Each of these conversations would

contextually verify that the player knows N5, since each is "nested" within the original condition that checked the "knows" variable. The final "else" in the tree also refers back to the original condition checking whether NPC 5 knows the player; if that condition does not resolve to 1, but to "else," the next set of conditions (not shown) would be checked, and would route the player to conversations wherein the player does *not* know N5.

Any variable checked by a condition statement must be resolved, which is the purpose of "else." A condition check of any variable can be nested in the resolution of a previous condition check, and is dependent in Conscript™ upon correct indentation levels. In the sample above, the conditions ["n5_ret"] and ["seeks_n5"] are nested in the first condition ["n5_knows"], and will only be checked if ["n5_knows"] resolves to a value of 1.

Standardizing Conscript™

Prior to this researcher's involvement in FPCT, Conscript™ was used with no naming conventions; variables, conditions, and conversation topics were named at the current writer's discretion. The writers tended to assign variables when needed and name them at that moment, resulting in multiple variables set to a value of 1. This proved to be a highly inefficient means of utilizing the capabilities of the program, and creating naming conventions became a first order priority, the process of which revealed ways to streamline variables and thus reduce the conditional logic requirements during game play. Naming conventions were applied to all aspects of writing in Conscript™, the results of which can be found in Appendix A.

To exemplify the need for such conventions, imagine during game play the PC has just finished a conversation with NPC John, who told her to speak with NPC Bob. In the previous coding style, a variable was set at the end of the conversation with John signaling that the PC had spoken to him and was told to speak to Bob:

```
set pc[“john_ok”] 1
```

In the conditional logic of Bob’s conversation file, the variable [“john_ok”] is checked. If the value resolves to 1, the player is routed to a conversation contextually relevant to the fact that the PC has spoken to John, and that John sent the PC to Bob.

However, during game play, other NPCs have also told the PC to speak to Bob, and each has been assigned a variable:

```
set pc[“jane_ok”] 1
set pc[“rick_ok”] 1
set pc[“paul_ok”] 1
set pc[“eric_ok”] 1
```

Now in Bob’s conversation file, the conditional logic tree must include checks for each of these variables, with the resulting coding:

```
condition pc[“jane_ok”]
  =1
   goto topic “jane”
condition pc[“rick_ok”]
  =1
   goto topic “rick”
condition pc[“paul_ok”]
  =1
   goto topic “paul”
condition pc[“eric_ok”]
  =1
```

```
  goto topic “eric”
condition pc[“john_ok”]
 =1
  goto topic “john”
```

Such a coding style was used in the third year of FPCT, which was a game scenario with eight Key, or conversational, NPCs, and resulted in conditional logic trees that were complex and long, as each variable relevant to that character was checked for a value of 1. The sheer volume of conditions to be checked in some conversation files caused the game to perform poorly on older computers. Appendix B is a conditional logic tree from FPCT year three, in its entirety.

To address this issue, it was decided to break the possible variables down into narrative events, and assign a single common variable to each event:

“seeks_NPC”	if set, PC has been told to talk to that NPC
“knows_NPC”	if set, PC knows that NPC
“event_ok”	if set, a particular narrative event has occurred

In the example with John, Bob and friends, each of the NPC conversations wherein the NPC told the PC to speak to Bob have the same variable:

```
set pc[“seeks_Bob”]
```

And because variables do not have a value limit, each NPC is set to a different value:

```
set pc[“seeks_Bob”] 1
set pc[“seeks_Bob”] 2
set pc[“seeks_Bob”] 3
```

1 represents the conversation with Jane, 2 represents the conversation with Rick, 3 represents Paul, and so on. Now the conditional logic tree in Bob's conversation file must only check the single common variable:

```
condition pc["seeks_Bob"]
 =1
  goto topic "jane"
 =2
  goto topic "rick"
 =3
  goto topic "paul"
 =4
  goto topic "eric"
 =5
  goto topic "john"
```

A significant issue arises from a basic operational aspect of Conscript™. Since the program reads coding from top to bottom, left to right, the first executable command it finds would cause the player to be routed to its corresponding conversation file. In the example above, what if the player has also spoken to Rick before she speaks to John, and Rick also told her to speak to Bob? Even though John was the latest NPC to tell the player to speak to Bob, *because Rick's variable is located ahead of John's variable in the conditional logic tree*, and it has been set to execute because Rick also told the player to speak to Bob, the player would be routed to a conversation contextually relevant to as if the player had just spoken to Rick rather than John. This routing error would reduce or eliminate the achievement of dynamic immersion.

Several workarounds are possible: the programmer could turn any given variable off, by setting it to a value of 0, once the player talks to Bob. This is effective, but also clogs the

system with more variables to set and check, which defeats the desire for greater efficiency in the conditional logic tree. To resolve the issue in FPCT, the conditional logic tree is arranged in the order that the player is most likely to encounter various NPCs according to the desired narrative outcome. This makes creating a conditional logic tree in FPCT like attempting to solve an intricate brain teaser, which by nature is time consuming. For obvious reasons (see DeMarle's example in Chapter 4), the FPCT workaround is a dangerous proposition in an open world game. Because FPCT is a training simulation there is less likelihood the player will stray too far from the narrative parameters; however, in a true open-world game, this issue with Conscript's™ basic operational parameters could cause serious problems.

For its relative simplicity, Conscript™ is a robust branching conversation system, able to handle most if not all requirements of a project such as FPCT. However it has significant flaws, including linear command execution and indentation-dependent coding, that prevent it from being used in large, multi-character open world games. Research is currently underway at UTD's Institute for Interactive Arts and Engineering to address these and other issues, and is discussed further in Chapter 6.

Branching Narrative and Dialogue Mapping

As head writer for FPCT, this author was tasked with creating a non-automated methodology for mapping dynamically immersive branching dialogue, utilizing a three option question/response system, with the desired outcome of producing a guide from which to easily reference the path a player has taken, and will take, through the course of game

play. Pre-planning of the player's path allows the writer to then create dialogue specifically contextual to the player's game play, as the author will know with whom the player has spoken, what information was exchanged, and how that information fits within the narrative.

Two versions of the methodology were created: methodology one approaches the task from a micro level, detailing each individual branch of a conversation. Methodology two approaches the task from a macro level, detailing each conversation as a whole. Both require the writer to create the narrative, or story; the characters the story requires; and a visual representation of the player's movement through the story.

Methodology One: The Multi-Pass System

1. Create a scenario master narrative.
2. Determine the Key NPCs required to complete the master narrative and assign each Key NPC an identifying number.
3. Storyboard the master narrative.
4. Create a visual Player-to-Key NPC interaction map detailing the player's potential movement through the game map and possible interactions with all Key NPCs during gameplay. Each interaction uses the three-dialogue-options architecture. (see Figure 7).
5. Once the interaction map is completed, the next stage for writers is to create a breakdown of Player-to-Key NPC conversations. This step is a multi-pass process for designing multiple conversations with many Key NPCs and keeping track of what has been said in dialogue, and is integral to designing non-linear narrative, and tracking

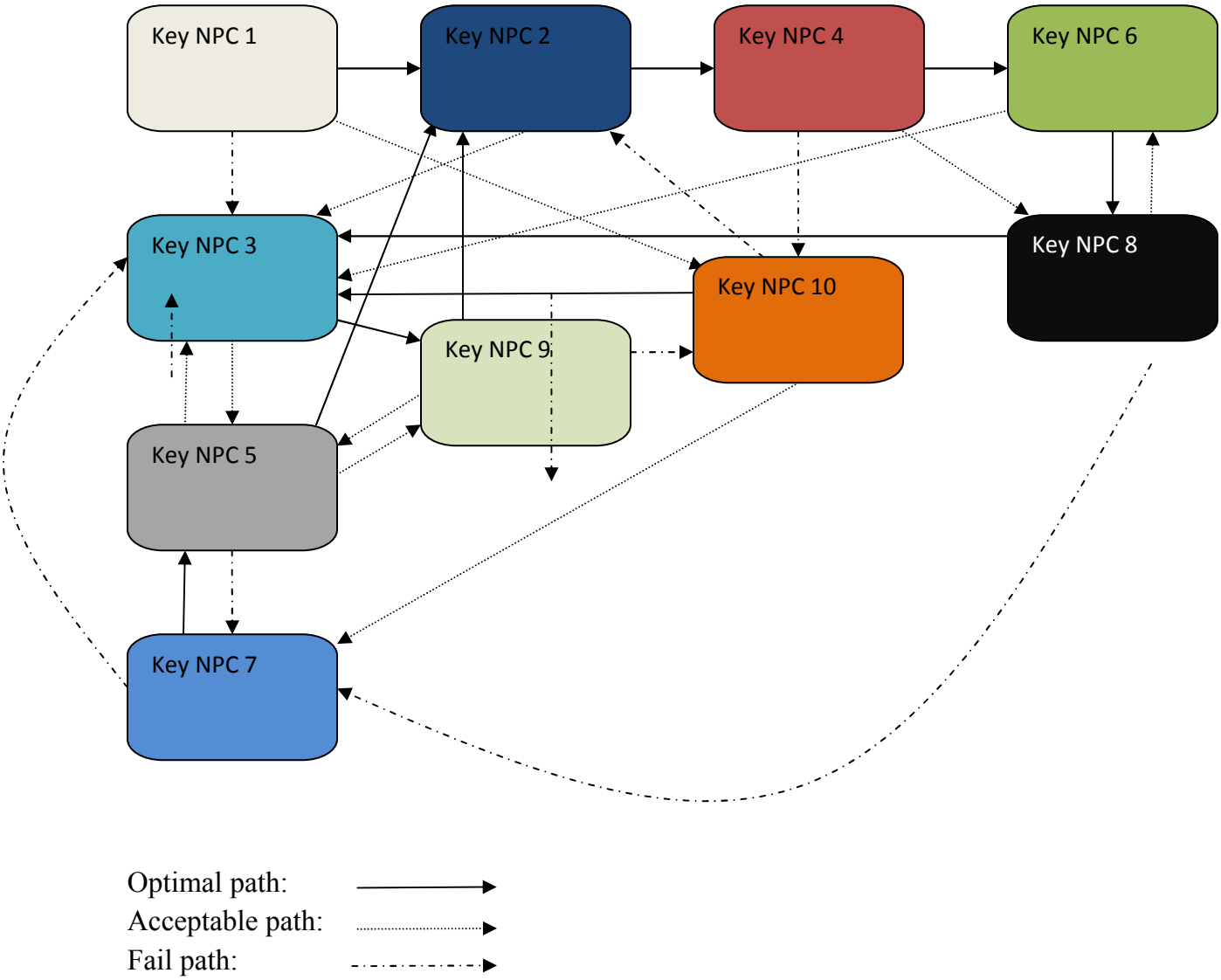


Figure 7. Player-to-Key NPC interaction map

player movement in order to create contextually relevant dialogue. Full dialogue is written later. The complicating factor in this step is the number of pathways generated by the number of response options given the player. Though FPCT script writers used the three-pathway architecture, any number of pathways could be developed. An

example of the process, based on the interaction map in Figure 7, is found in Appendix C. This example represents one potential player conversation path as it begins with Key NPC1. The process is constructed as follows:

a. Label conversation pathways alphabetically under each Key NPC. Example: Key NPC 1 [A], Key NPC 1 [B], etc. This allows the writers to keep track of the order in which a Key NPC is approached by the player so that dialogue reflects what the player knows, who the player has met and where the player has been. (Tip: the author may choose to include a brief description of the Player-to-Key NPC interaction during each conversation, which will help in keeping track of story elements that are important in return or future conversations that result from alternate paths.)

b. Within the conversation, detail player responses according to the number of desired branches. Appendix 1 outlines the three-branch game system with Optimum, Acceptable, and Fail responses. Each response level detail must include where the player is sent next.

c. Using the visual Player-to-Key NPC interaction map (Figure 3) and the Key NPC conversations that have already been determined, describe each new conversation by the pathway that the player has taken to reach this instance of conversation. Example: the player comes to talk to Key NPC 2 [B] (Path: Key NPC 1[A] Fail - Key NPC 3[A] Optimal - Key NPC 5[A] Optimal - Key NPC 2)

d. Determine paths and optional descriptions for conversations [A] for each Key NPC in the scenario. Then go back to the first Key NPC and repeat for conversations [B] in the same order. Repeat for conversations [C] and so on until breakdown is complete.

e. As the need for a new conversation with any Key NPC is identified through this process, insert the next sequential conversation letter and path under the appropriate Key NPC, but do not write the interaction description and where the player is sent next until that letter's pass. This allows the author to keep track of all future conversations without having to search back through the breakdown. Example: during the second (conversation [B]) pass with Key NPC 2, a path leading to conversation [D] with Key NPC 4 is generated. Insert a placeholder for [D] under Key NPC 4, but wait until pass four (conversation [D]) to fill in the response level details (see Appendix C).

f. For any change in the game play path, a new conversation letter must be assigned. This will result in some conversations that have the same response level details, but it is invaluable in determining the dialogue details and variables required for proper game play.

6. Once the breakdown is complete, the writing of dialogue can commence. Use the breakdown and its pathing information to help determine who the player has already spoken to, any change in emotion that should occur, and what should be said during each conversation as a result of the route the player has taken to that conversation.

Methodology Two: The Spreadsheet System

The second methodology follows the same basic principle of breaking down the player's potential movement through the game, but utilizes the organizational properties inherent in a spreadsheet to create a readable "map." If desired, response level details may be added via the spreadsheet's comment capability. The second methodology is outlined below; a portion of the spreadsheet created for FPCT Spiral 3 is presented in Appendix D.

1. Create a scenario master narrative.
2. Determine the Key NPCs required to complete the master narrative and assign each Key NPC an identifying number.
3. Storyboard the master narrative.
4. Create a visual Player-to-Key NPC interaction map detailing player's movement through Key NPCs during gameplay (see Figure 3).
5. Create a spreadsheet with the first Player to Key NPC interaction in the upper left cell (cell A1). This and all subsequent interactions detail the Key NPC number and the response type: 1 = Optimal, 2 = Acceptable, 3 = Fail. Thus, N1_1 denotes an interaction with Key NPC 1 resulting in the player making an Optimal response. Follow each interaction detail with the word "to."
    a. Cells A2 and A3 denote the Acceptable and Fail responses in the first interaction, and are written N1_2 and N1_3.

b. Cells B1, B2, and B3 indicate the Key NPC whom the player is told to see based on the dialogue path the player has chosen. (Example: if the player fails with N1, the player is put on Path 3 and is told to talk to N5.)

	A	B	C
1	N1_1 to	N2	
2	N1_2 to	N3	
3	N1_3 to	N5	

c. Moving left to right, column C is skipped for visual organization purposes. Cell D1 continues the path begun in Cell A1. Note that D1-D3 list only the Optimal responses from the previous section (N1_1). This is because the initial triad (rows 1, 2, and 3) deals with the path begun with the PC choosing the Optimal response with N1, and details only that path, from left to right, to its conclusion of either win or fail, which is presented in its entirety in Appendix 2. The Acceptable and Fail paths are detailed below.

	D	E	F	G
1	N1_1 to	N2_1 to	N4	
2	N1_1 to	N2_2 to	N3	
3	N1_1 to	N2_3 to	N5	

d. Moving left to right, column G is skipped, again for visual purposes. Cell H1 continues the path begun in A1; again, note that H1-H3 and I1-I3 list the initial Optimal responses of the previous sections, while J1-J3 list the new responses encountered with N4. This pattern is repeated, left to right, completing the

path begun in A1 until rows 1, 2, and 3 result in either a win or fail state (see Appendix 2):

	H	I	J	K	L
1	N1_1 to	N2_1 to	N4_1 to	N6	
2	N1_1 to	N2_1 to	N4_2 to	N3	
3	N1_1 to	N2_1 to	N4_3 to	N5	

e. Now, moving top to bottom, row 4 is skipped for visual and organizational purposes, and Columns A, B and C are skipped. Cell D5 continues the path begun in Cell A2. From here, the pattern described above is repeated, left to right, completing the path begun in A2 until rows 5, 6, and 7 result in either a win or fail state (see Appendix 2).

	A	B	C	D	E	F	G
1	N1_1 to	N2		N1_1 to	N2_1 to	N4	
2	N1_2 to	N3		N1_1 to	N2_2 to	N3	
3	N1_3 to	N5		N1_1 to	N2_3 to	N5	
4							
5				N1_2 to	N3_1 to	N1	
6				N1_2 to	N3_2 to	N5	
7				N1_2 to	N3_3 to	N7	

f. Next, row 8 is skipped, and the left to right pattern begins with Cell D9, which continues the path begun in A3 to its conclusion, until rows 9, 10, and 11 result in either a win or fail state.

g. Stepping back, we see that so far the paths begun in the first section of rows 1, 2, and 3 (A1, A2, and A3) have resolved to their conclusions. However, in

doing so, new paths have been spawned at each new section's Optimal, Acceptable, and Fail response cells. These new paths are resolved in the same left to right, top to bottom pattern described above. In Appendix D, the new paths spawned along the path generated by A1 are color coded, with their matching left to right resolutions in the same color top to bottom.

Methodology Application

While these methodologies are tedious, they assure that all movement of the player is accounted for, and allow the writer to adjust dialogue accordingly by referencing the player's game play path, which is integral to dynamic immersion. The potential figures for the number of conversations and branches discussed in Chapter 4 were proven out when the methodology was applied to writing the latest full version of FPCT, which utilized eight characters, or Key NPCs. This resulted in 258 unique player to Key NPC conversations, and 2,263 unique conversation branches. Application of the methodology, including fully mapping all player-to-NPC interactions, and writing and coding the dialogue, took approximately 400 man hours and resulted in 32,000 lines of coded dialogue. The resulting scenario requires approximately 45 minutes of game play to complete.

CHAPTER SIX

FUTURE WORK IN DYNAMIC IMMERSION AND BRANCHING DIALOGUE SYSTEMS

Implementation of the FPCT methodology for planning, mapping, and writing dynamically immersive branching narrative and dialogue sparked a keen interest in the possibility of automating some of the processes involved. Several research outcomes support the need for such automation:

- *Volume of conversations was verified*: The number of unique conversation branches for a scenario with eight Key NPCs was estimated at $3^8 \div 2$, or $6561 \div 2 = 3280$. The actual number of unique conversation branches realized in FPCT year three was 2,263. Note that one of the Key NPCs was a "help agent" that had direct, rather than branching, conversations with the player, which accounts in part for the reduction in unique conversation branches.
- *Resource requirements of methodology*: 400 man hours to map, write, and code 45 minutes worth of game play in FPCT is highly inefficient. In comparison, *Skyrim's* main quest takes approximately thirty hours of game play to complete (Hill). If the game were written and coded using the FPCT methodology, it would have taken 16,000 man hours, or a team of three writer/coders almost two and a half years to complete. *Skyrim's* producers

estimate an additional 200 hours of side quests are available in the game…which would increase the three man team's commitment to almost twenty years.

- *Inefficiency of Conscript*™: Puzzling out the conditional logic trees, increased bug testing due to frequent indentation errors, and manually writing all code and dialogue contribute to Conscript's™ inefficiency.

A new version of Conscript™ aimed at process automation and increased efficiency is currently in the initial design phase, and is outlined in the following section.

Visual Conscript™

The design team for Visual Conscript™ has taken into account research in the fields of story generation and branching dialogue tracking, and proposes a hybrid design that incorporates the most successful attributes of a disparate array of narrative generation techniques, including elements of automatic story generation and author-goal based story generation, in an amalgam that seeks to reduce the reliance on technical experts (Zielke "FPCT4+"). Visual Conscript™ will also seek to eliminate the linear logic approach employed by Conscript™, opting instead for a system that does not rely on specific indentation or top down, left right command execution. Possible programming languages include an object-oriented language such as Action Script or Java Script, or an associative array language such as YAML.

The resulting Visual Conscript™ Editor will serve as a "tool for branching narrative design, conversation tracking, and dialogue authoring…[and] would make scenario writing

more efficient by generating the necessary programming code in the background as the author creates the narrative" (Zielke "FPCT4+" 8). Visual Conscript™ will strive to eliminate the necessity for a narrative and dialogue writer to have knowledge and/or experience in computer programming code. Figure 8 illustrates the potential for such a combination.

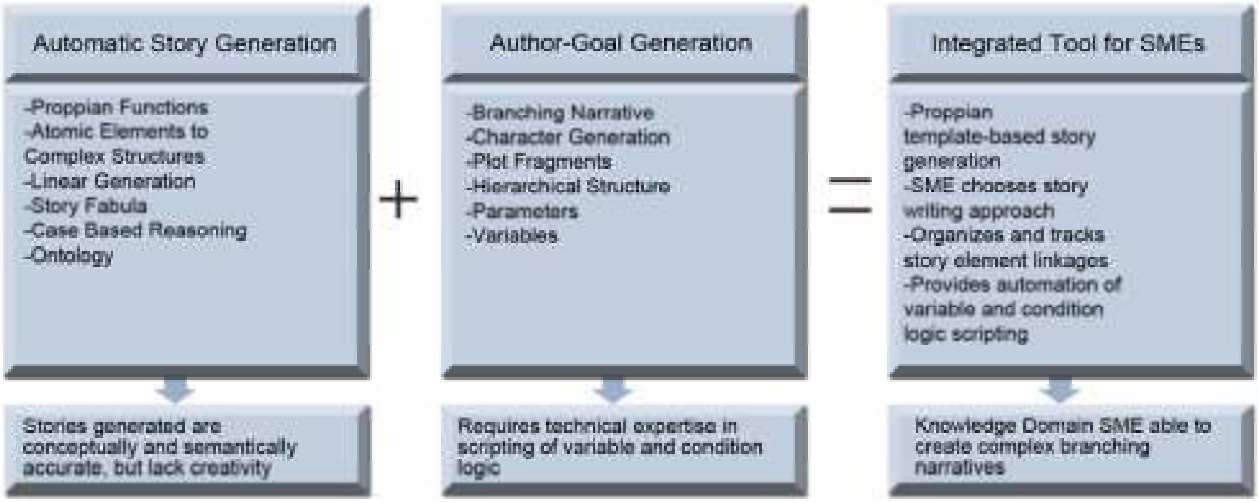


Figure 8. Visual Conscript™: an integrated tool that would reduce or eliminate an SME's reliance on writing and technical experts (Zielke "FPCT4+" 8)

Research into tools to reduce reliance on technical expertise in game authoring is not without precedent. Mateas and Skorupski's WideRuled, discussed in Chapter 4, was a solid attempt at automatic narrative generation based on author or SME input; and Eugenio Marchiori and his colleagues created a DSVL, or domain-specific visual language, designed to assist educators in developing digital games as a teaching aid without requiring the educators have prior expertise in game design or programming. Marchiori's DSVL uses the "choose your own adventure" story model as its base, demonstrating how story automation and branching narrative can be mutually beneficial (Marchiori, et al).

Conclusion

Exploration into how best to implement dynamic immersion into digital games and simulations is comparatively in its infancy, having only been seriously researched for some fifteen years. However, research into the ancillary areas that support the design and implementation of such systems is plentiful: human verbal and nonverbal communication systems, natural language processing systems, and automated narrative systems have been well researched, and serve to feed current exploration into the design of narrative and dialogue systems that more closely replicate human social interactions. The work documented herein is the culmination of several years of diligence in addressing the increasing need for dynamically immersive narrative and conversation systems in digital games and simulations, resulting in valid conclusions supporting a continued commitment to increased automation of the technical aspects of generating narrative and conversation, while acknowledging and embracing the creative input of the human writer.

The future of games and simulations, both commercial and educational, relies on perseverance in the ongoing effort to create life-like worlds, populated with virtual humans that replicate the human experience as closely as possible. Dynamically immersive narrative and dialogue systems that harness the complexity and power of human communication are an integral aspect of such replication, and must be highly regarded when considering continued research commitments.

Works Cited

Abt, Clark. *Serious Games*. New York: The Viking Press, 1970. Print.

André, Elisabeth, et all. "Employing AI Methods to Control the Behavior of Animated Interface Agents." *Applied Artificial Intelligence*. 13 (1999):415-448. Print.

André, Elisabeth, et all. "Integrating Reactive and Scripted Behaviors in a Life-Like Presentation Agent." *Proceedings of Autonomous Agents '98*. Minneapolis/St. Paul: ACM Press, 1998. Print.

Aylett, Ruth, et al. "Unscripted Narrative for Affectively Driven Characters." *Computer Graphics and Applications.* 26.3 (2006):42-52. Print.

Baker, Janet, et al. "Research Developments and Directions in Speech Recognition and Understanding, Part 1." *IEEE Signal Processing Magazine*. 26.3 (2009):75-80. Web.

Ball, Gene, et al. "Lifelike Computer Characters: The Persona Project at Microsoft Research." *Software Agents*. Jeffrey Bradshaw, ed. Cambridge, MA: MIT Press, 1997. Print.

Barber, H., and D. Kedenko. "Generation of Adaptive Dilemma-based Interactive Narratives." *Computational Intelligence and AI in Games.* 1.4, (2009):309-26. Print.

Bates, J. "Virtual Reality, Art, and Entertainment. *Presence: Teleoperators and Virtual Environments.* 1.1 (1992):133–138. Print.

Becheiraz, Pascal, and Daniel Thalmann. "A Behavioral Animation System for Autonomous Actors Personified by Emotions." *Proceeding of the 1st Workshop on Embodied Conversational Characters.* 1998. Web.

Beskow, Jonas, and S. McGlashan. "Olga – A Conversational Agent with Gestures." *Proceedings of the IJCAI '97 Workshop on Animated Interface Agents, Nagoya, Japan.* San Francisco: Morgan-Kaufman, 1997. Print.

Bickmore, Timothy, and Justine Cassell. "Small Talk and Conversational Storytelling in Embodied Interface Agents." *Proceeding of the AAAI Fall Symposium on Narrative Intelligence.* Meno Park, CA: AAAI Press, 1999. Print.

Cannon-Bowers, Janis, and Clint Bowers, eds. *Serious Game Design and Development: Technologies for Training and Learning*. Hershey, PA: Information Science Reference, 2010. Print.

Cassell, Justine, et al. "Requirements for an Architecture for Embodied Conversational Characters." *10th Eurographics Workshop on Animation and Simulation, Milan, Italy*. Springer-Verlag, 1999. Print.

Cassell, Justine. "More Than Just Another Pretty Face: Embodied Conversational Interface Agents." *Communications of the ACM*. 43.4 (2000): 70-78. Print.

Cassell, Justine, and Timothy Bickmore. "External Manifestations of Trustworthiness in the Interface." *Communications of the ACM.* 43.12 (2000): 50-56. Print.

Cassell, Justine, et al. "Non-verbal Cues for Discourse Structure." *Proceedings of the 39th Annual Meeting on Association for Computational Linguistics. Toulouse, France.* New York: ACM, 2001. Print.

Cassell, Justine, et al. "BEAT: The Behavior Expression Animation Toolkit." *Proceedings of the 28th Annual Conference on Computer Graphics and Interactive Techniques*. New York: ACM, 2001. Print.

Cassell, Justine. "Body Language: Lessons from the Near-Human." *Genesis Redux: Essays in the History and Philosophy of Artificial Life.* Jessica Riskin, ed. Chicago: University of Chicago Press, 2010. Print.

Chomsky, Noam. *Aspects of the Theory of Syntax*. Cambridge, MA: MIT Press, 1965. Print.

DeMarle, Mary. "Nonlinear Game Narrative." *Game Writing: Narrative Skills for Videogames*. Chris Bateman, ed. Boston: Charles River Media, 2007. Print.

Ferrucci, David, and Adam Lally. "UIMA: An Architectural Approach to Unstructured Information Processing in the Corporate Research Environment." *Natural Language Engineering*. 10.3–4 (2004): 327–348. Print.

Ferrucci, David, et al. "Building Watson: An Overview of the DeepQA Project." *AI Magazine*. 31.3 (2010):59-79. Print.

Fisch, S.M. "Making Education Computer Games Educational." *Proceedings of the 2005 Conference on Interactive Design and Children, in Boulder, CO*. New York: ACM, 2005. Print.

Furnham, Adrian, and Evgeniva Petrova. *Body Language in Business: Decoding The Signals.* Palgrave Macmillan [UK], 2010. *eBook Collection (EBSCOhost).* Web. 22 July 2012.

Gamespot. "L.A. Noire Review." Carolyn Petit, ed. *Gamespot.com*. 16 May 2011. Web. 25 July 2012.

GameSpy.com. *GameSpy's Top 50 Games of All Time*. GameSpy Industries, 2004. Web. 24 July 2012.

GNL. "Gesture and Narrative Language Group." *MIT Media Lab*. 2003. Web. 24 July 2012.

Hill, Owen. "E3 2011: Skyrim's Main Quest 30 Hours Long. Additional Content 'Two to Three Hundred' More." *PC Gamer*. 8 June, 2011. Web. 4 August 2012.

King, Corwin P. "A Functional Model of Memory in Communication." *Annual Meeting of the International Communication Association, New Orleans.* Education Resources Information Center:1974. Web. 30 July, 2012.

Knapp, Mark, and Judith Hall. *Nonverbal Communication in Human Interaction.* Belmont, CA:Thomson Wadsworth, 2006. Print.

Krawczyk, M., and J. Novak. *Game Development Essentials: Game Story & Character Development.* New York:Thomson Delmar Learning, 2006. Print.

Kurzweil, Ray. *The Age of Intelligent Machines.* Cambridge, MA: Massachusetts Institute of Technology, 1990. Print.

Kurzweil, Ray. *The Age of Spiritual Machines: When Computers Exceed Human Intelligence.* New York: Penguin Putnam, 1999. Print

Kurzweil, Ray. "The Significance of Watson." *Kurzweil Accelerating Intelligence*. Kurzweilai.net, 13 Feb. 2011. Web. 30 July 2012.

Kurzweiltech.com. *Ray Kurzweil: Curriculum Vitae.* Kurzweil Technologies, 2008. Web. 24 July 2012.

L.A. Noire. "L.A. Noire Information." *Rock Star Games.* Web. 25 July 2012.

Languagemonitor.com. *Number of Words in the English Language.* The Global Language Monitor, 23 July 2011. Web. 24 July 2012.

Marchiori, Eugenio J., et al. "A Visual Language for the Creation of Narrative Educational Games." *Journal of Visual Languages & Computing.* 22.6 (2011): 443-52. Print.

Markoff, John. "Computer Wins on 'Jeopardy!': Trivial, It's Not." *The New York Times* 16 Feb. 2011. *Nytimes.com*. Web. 27 July 2012.

Mateas, Michael, and A.Stern. (2003) "Façade: An Experiment in Building a Fully Realized Interactive Drama. *Proceedings of Game Developers Conference/Game Design Track, San Jose, CA*. Georgia Tech, 2003. Web.

Mäyrä, Frans. *An Introduction to Game Studies: Games in Culture*. Los Angeles: SAGE, 2008. Print.

McQuiggan, S.W., et al. "Story-based Learning: The Impact of Narrative on Learning Experiences and Outcomes." *Lecture Notes in Computer Science.* 50.91 (2008): 530 539. Print.

Mortensen, C. David. *Communication: The Study of Human Interaction.* New York: McGraw-Hill, 1972. Print.

Nelson, M. J., et al. "Declarative Optimization-Based Drama Management in Interactive Fiction." *Computer Graphics and Applications.* 26.3 (2006):32-41. Print.

Pandolfini, Bruce. *Kasparov and Deep Blue: The Historic Chess Match Between Man and Machine*. New York: Simon & Schuster, 1997. Print.

Perez-Marin, Diana, and Ismael Pascual-Nieto, eds. *Conversational Agents and Natural Language Interaction Techniques and Effective Practices.* Hershey, PA: Information Science Reference, 2011. Print.

Perlin, Ken, and Athomas Goldberg. "Improv: A System for Scripting Interactive Actors in Virtual Worlds." *Proceedings of SIGGRAPH '96, New Orleans, LA*. 1996.

Riedl, Mark, and Andrew Stern. "Believable Agents and Intelligent Story Adaptation for Interactive Storytelling." *Proceedings of the 3rd International Conference on Technologies for Interactive Digital Storytelling and Entertainment, Darmstadt, Germany*. USC: Institute for Creative Technologies, 2006. Web.

Riesbeck, Christopher. "Computational Understanding." *Proceedings of the 1975 Workshop on Theoretical Issues in Natural Language Processing*. Stroudsburg, PA: Association for Computational Linguistics, 1975. Web.

Sgouros, Nikita M. "Dynamic, User-centered Resolution in Interactive Stories." *Proceedings of the International Joint Conference on Artificial Intelligence, Nagoya, Japan*. National Technical University of Athens, 1997. Web.

Simmons, Robert F. "Natural Language Question-Answering Systems: 1969." *Communications of the ACM*. 13.1 (1970):15-30. Print.

Skorupski, James, and Michael Mateas. "Interactive Story Generation for Writers: Lessons Learned from the Wide Ruled Authoring Tool." *Proceedings of the 8th Digital Art and Culture Conference*. UC Santa Cruz, 2009. Web.

Smith, Ronnie, and Richard Hipp. *Spoken Natural Language Dialog Systems: A Practical Approach.* Cary, NC: Oxford University Press. Print.

Staska. "How Siri on I-Phone 4S Works and Why It's a Big Deal." *UnwiredView.com*. 12 Oct. 2011. Web. 30 July 2012.

Szilas, Nicolas. "IDtension: A Narrative Engine for Interactive Drama." *Proceedings of the 1st International Conference on Technologies for Interactive Digital Storytelling and Entertainment, Darmstadt, Germany*. 2003. Web.

Team Bondi. "Candy Edwards: Screenshot." *Rock Star Games.* Web. 25 July 2012.

Waraich, Atif. "Using Narrative as a Motivating Device to Teach Binary Arithmetic and Logic Gates." *Proceedings of Ninth Annual SIGCSE Conference on Innovation and Technology in Computer Science Education, in Leeds, United Kingdom.* New York: ACM, 2008. Print.

Young, R. Michael, et al. "An Architecture for Integrating Plan-based Behavior Generation with Interactive Game Environments." *Journal of Game Development.* 1.1 (2004): 51–70. Print.

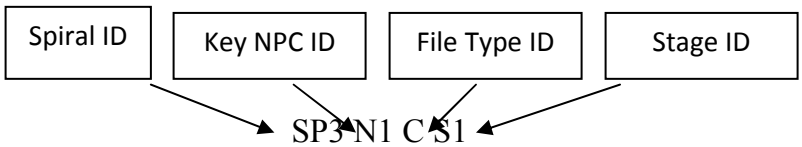
Zielke, Marjorie, et al. "Tools and Techniques for Managing Subject Matter Expert Input Virtual Simulations." *Interservice/Industry Training, Simulation, and Education Conference 2011 (submission)*. The University of Texas at Dallas: Institute for Interactive Engineering, 2011. Print.

Zielke, Marjorie, et al. White Paper. *FPCT 4+ Architecture: The Way Ahead*. The University of Texas at Dallas: Institute for Interactive Engineering, 2012. Print.

Appendix A
FPCT Conscript Naming Conventions: Spiral 3 and Forward

FILES:

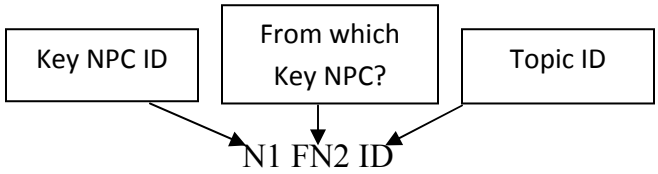
Conversation file:



Spiral 3, Key NPC #1, Conversation File, Stage #1
Written: sp3n1cs1

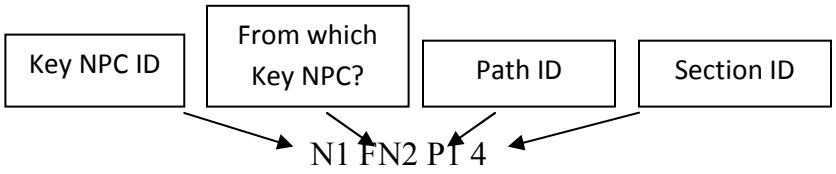
HIDDEN TOPICS:

Greet Topic:



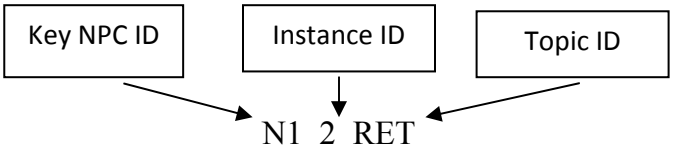
Key NPC #1, sent from N2, Greet Topic
Written: n1fn2_id Coding Samples: goto topic "n1fn2_id" hidden topic "n1fn2_id"

Path Topic:



Key NPC #1, sent from N2, Path #1, Section #4
Written: n1fn2_p1_4 Coding Samples: goto topic "n1fn2_p1_4" hidden topic "n1fn2_p1_4"

Return Topic:

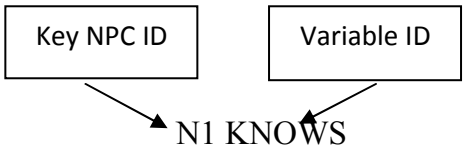


Key NPC #1, Instance #2, Return Topic

Written: n1-2_ret Coding Samples: goto topic "n1-2_ret" hidden topic "n1-2_ret"

VARIABLES:

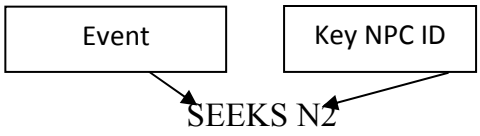
Knows Variable:



Variable denotes PC knows Key NPC #1

Written: n1_knows Coding Samples: set pc ["n1_knows"] 1 condition pc ["n1_knows"]

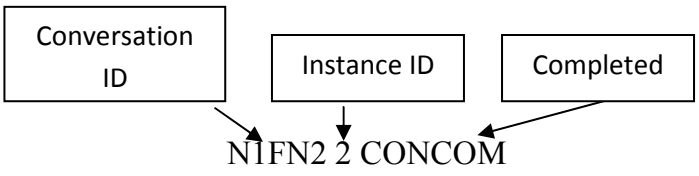
Narrative Event Variable:



Variable denotes PC has been sent to find N2

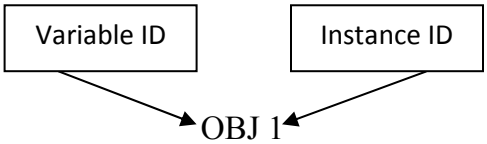
Written: seeks_n2 Coding Samples: set pc ["seeks_n2"] 1 condition pc ["seeks_n2"]

Conversation Complete Variable:



Variable denotes conversation with N1, sent from N2, Instance 2, Conversation Completed
Written: n1fn2_concom Coding Samples: set pc [“n1fn2_concom“] 1 condition pc [“n1fn2_concom”]

Objective Variable:



Denotes Objective Variable, Instance #1
Written: obj1 Coding Sample: set pc [“obj1”] 1

Appendix B
A complete conditional logic tree from FPCT, year three:

```
auto hidden topic "check_n4_knows"
 condition pc["n4fn8_ok"]
 =1
  condition pc["n4fn8_concom"]
   =1
    condition pc["n4-2_knows"]
     =1
      goto topic "n4-2_ret"
     =else
      goto topic "n4_ret"
   =else
    goto topic "n4fn8_p1_1"
 condition pc["n4fn7_ok"]
 =1
  condition pc["n4fn7_concom"]
   =1
    condition pc["n4-2_knows"]
     =1
      goto topic "n4-2_ret"
     =else
      goto topic "n4_ret"
   =else
    goto topic "n4fn7_p1_1"
 condition pc["n4fn5-4_ok"]
 =1
  condition pc["n4fn5-4_concom"]
   =1
    condition pc["n4-2_knows"]
     =1
      goto topic "n4-2_ret"
     =else
      goto topic "n4_ret"
   =else
    condition pc["n7_knows"]
```

```
    =1
     goto topic "n4fn5-5_p1_1"
    =else
     goto topic "n4fn5-4_p1_1"
condition pc["n4fn5_ok"]
 =1
  condition pc["n4fn5_concom"]
   =1
    condition pc["n4-2_knows"]
     =1
      goto topic "n4-2_ret"
     =else
      goto topic "n4_ret"
   =else
    condition pc["n3_knows"]
     =1
      condition pc["n7_knows"]
       =1
        goto topic "n4fn5-3_p1_1"
       =else
        goto topic "n4fn5-2_p1_1"
     =else
      goto topic "n4fn5_p1_1"
condition pc["n4fn3-4_ok"]
 =1
  condition pc["n4fn3-4_concom"]
   =1
    condition pc["n4-2_knows"]
     =1
      goto topic "n4-2_ret"
     =else
      goto topic "n4_ret"
   =else
    condition pc["n7_knows"]
     =1
      goto topic "n4fn3-5_p1_1"
     =else
```

```
      goto topic "n4fn3-4_p1_1"
condition pc["n4fn3_ok"]
 =1
  condition pc["n4fn3_concom"]
   =1
    condition pc["n4-2_knows"]
     =1
      goto topic "n4-2_ret"
     =else
      goto topic "n4_ret"
   =else
    condition pc["n5_knows"]
     =1
      condition pc["n7_knows"]
       =1
        goto topic "n4fn3-3_p1_1"
       =else
        goto topic "n4fn3-2_p1_1"
     =else
      goto topic "n4fn3_p1_1"
condition pc["n4fn2-8_ok"]
 =1
  condition pc["n4fn2-8_concom"]
   =1
    condition pc["n4-2_knows"]
     =1
      goto topic "n4-2_ret"
     =else
      goto topic "n4_ret"
   =else
    condition pc["n3_knows"]
     =1
      condition pc["n7_knows"]
       =1
        goto topic "n4fn2-10_id"
       =else
        goto topic "n4fn2-9_id"
```

```
    =else
     goto topic "n4fn2-8_id"
condition pc["n4fn2-5_ok"]
 =1
  condition pc["n4fn2-5_concom"]
   =1
    condition pc["n4-2_knows"]
     =1
      goto topic "n4-2_ret"
     =else
      goto topic "n4_ret"
   =else
    condition pc["n5_knows"]
     =1
      condition pc["n7_knows"]
       =1
        goto topic "n4fn2-7_id"
       =else
        goto topic "n4fn2-6_id"
     =else
      goto topic "n4fn2-5_id"
condition pc["n4fn2_ok"]
 =1
  condition pc["n4fn2_concom"]
   =1
    condition pc["n4-2_knows"]
     =1
      goto topic "n4-2_ret"
     =else
      goto topic "n4_ret"
   =else
    condition pc["n5_knows"]
     =1
      condition pc["n3_knows"]
       =1
        goto topic "n4fn2-4_id"
       =else
```

```
        goto topic "n4fn2-3_id"
      =else
       condition pc["n3_knows"]
        =1
         goto topic "n4fn2-2_id"
        =else
         goto topic "n4fn2_id"
condition pc["n4_knows"]
 =1
  condition pc["n4-2_knows"]
   =1
    goto topic "n4-2_ret"
   =else
    goto topic "n4_ret"
 =else
  goto topic "n4_id"
```

Appendix C
Sample of Methodology 1 – The Multi-pass System: pass 1 (placeholder conversations in red)

CONVERSATIONS WITH KEY NPC 1:

[A]: (Path: Start of Game)
Optimal: Key NPC 1 sends PC to Key NPC 2. (see convo [A] with Key NPC 2)
Acceptable: Key NPC 1 sends PC to Key NPC 10. (see convo [A] with Key NPC 10)
Fail: Key NPC 1 sends PC to Key NPC 3. (see convo [A] with Key NPC 3)

CONVERSATIONS WITH KEY NPC 2:

[A]: (Path: Key NPC 1[A] Optimal to Key NPC 2)
Optimal: Key NPC 2 sends PC to Key NPC 4. (see convo [A] with Key NPC 4)
Acceptable: Key NPC 2 sends PC to Key NPC 4. (see convo [B] with Key NPC 4)
Fail: Key NPC 2 sends PC to Key NPC 3. (see convo [B] with Key NPC 3)

[B]: (Path: Key NPC 1[A] Fail to Key NPC 3[A] Optimal to Key NPC 5[A] Optimal to Key NPC 2)
[C]: (Path: Key NPC 1[A] Fail - Key NPC 3[A] Acceptable - Key NPC 9[A] Optimal – Key NPC 2)
[D]: (Path: Key NPC 1[A] Acceptable - Key NPC 10[A] Optimal – Key NPC 2)

CONVERSATIONS WITH KEY NPC 3:

[A]: (Path: Key NPC 1[A] Fail - Key NPC 3)
Optimal: Key NPC 3 sends PC to Key NPC 5. (see convo [A] with Key NPC 5)
Acceptable: Key NPC 3 sends PC to Key NPC 9. (see convo [A] with Key NPC 9)
Fail: Key NPC 3 sends PC to Key NPC 9. (see convo [B] with Key NPC 9)

[B]: (Path: Key NPC 1[A] Optimal - Key NPC 2[A] Fail - Key NPC 3)
[C]: (Path: Key NPC 1[A] Optimal - Key NPC 2[A] Optimal - Key NPC 4[A] Acceptable - Key NPC 6[A] Fail - Key NPC 3)
[D]: (Path: Key NPC 1[A] Fail - Key NPC 3[A] Optimal - Key NPC 5[A] Fail - Key NPC 7[A] Optimal - Key NPC 3)
[E]: (Path: Key NPC 1[A] Fail - Key NPC 3[A] Optimal - Key NPC 5[A] Fail - Key NPC 7[A] Acceptable - Key NPC 3)
[F]: (Path: Key NPC 1[A] Optimal - Key NPC 2[A] Optimal - Key NPC 4[A] Optimal - Key NPC 8[A] Optimal - Key NPC 3)

[G]: (Path: Key NPC 1[A] Acceptable - Key NPC 10[A] Acceptable - Key NPC 3)

CONVERSATIONS WITH KEY NPC 4:

[A]: (Path: Key NPC 1[A] Optimal - Key NPC 2[A] Optimal - Key NPC 4)

Optimal: Key NPC 4 sends PC to Key NPC 8. (see convo [A] with Key NPC 8)
Acceptable: Key NPC 4 sends PC to Key NPC 6. (see convo [A] with Key NPC 6)
Fail: Key NPC 4 sends PC to Key NPC 10. (see convo [B] with Key NPC 10)

[B]: (Path: Key NPC 1[A] Optimal - Key NPC 2[A] Acceptable - Key NPC 4)

CONVERSATIONS WITH KEY NPC 5:

[A]: (Path: Key NPC 1[A] Fail - Key NPC 3[A] Optimal - Key NPC 5)

Optimal: Key NPC 5 sends PC to Key NPC 2. (see convo [B] with Key NPC 2)
Acceptable: Key NPC 5 sends PC to Key NPC 9. (see convo [B] with Key NPC 9)
Fail: Key NPC 5 sends PC to Key NPC 7. (see convo [A] with Key NPC 7)

[B]: (Path: Key NPC 1[A] Optimal - Key NPC 2[A] Optimal - Key NPC 4[A] Optimal - Key NPC 8[A] Acceptable - Key NPC 5)
[C]: (Path: Key NPC 1[A] Fail - Key NPC 3[A] Acceptable - Key NPC 9[A] Fail - Key NPC 5)

CONVERSATIONS WITH KEY NPC 6:

[A]: (Path: Key NPC 1[A] Optimal - Key NPC 2[A] Optimal - Key NPC 4[A] Acceptable - Key NPC 6)

Optimal: Key NPC 6 sends PC to Key NPC 8. (see convo [B] with Key NPC 8)
Acceptable: Key NPC 6 sends PC to Key NPC 8. (see convo [C] with Key NPC 8)
Fail: Key NPC 6 sends PC to Key NPC 3. (see convo [C] with Key NPC 3)

[B]: (Path: Key NPC 1[A] Optimal - Key NPC 2[A] Optimal - Key NPC 4[A] Optimal - Key NPC 8[A] Fail - Key NPC 6)

CONVERSATIONS WITH (KEY NPC 7 – DEAD END)

[A]: (Path: Key NPC 1[A] Fail - Key NPC 3[A] Optimal - Key NPC 5[A] Fail - Key NPC 7)

Optimal: Key NPC 7 sends PC to Key NPC 3. (see convo [D] with Key NPC 3)
Acceptable: Key NPC 7 sends PC to Key NPC 3. (see convo [E] with Key NPC 3)

Fail: (Key NPC 7) sends PC to find a nonexistent NPC in the market.

[B]: (Path: Key NPC 1[A] Acceptable - Key NPC 10[A] Fail - Key NPC 7)

CONVERSATIONS WITH KEY NPC 8:

[A]: (Path: Key NPC 1[A] Optimal - Key NPC 2[A] Optimal - Key NPC 4[A] Optimal - Key NPC 8)

Optimal: Key NPC 8 sends PC to Key NPC 3. (see convo [F] with Key NPC 3)
Acceptable: Key NPC 8 sends PC to Key NPC 5. (see convo [B] with Key NPC 5)
Fail: Key NPC 8 sends PC to Key NPC 6. (see convo [B] with Key NPC 6)

[B]: (Path: Key NPC 1[A] Optimal - Key NPC 2[A] Optimal - Key NPC 4[A] Acceptable - Key NPC 6[A] Optimal - Key NPC 8)
[C]: (Path: Key NPC 1[A] Optimal - Key NPC 2[A] Optimal - Key NPC 4[A] Acceptable - Key NPC 6[A] Acceptable - Key NPC 8)

CONVERSATIONS WITH (KEY NPC 9 – MEDIATOR)

[A]: (Path: Key NPC 1[A] Fail - Key NPC 3[A] Acceptable - Key NPC 9)

Optimal: Key NPC 9 sends PC to Key NPC 2. (see convo [C] with Key NPC 2)
Acceptable: Key NPC 9 sends PC to Key NPC 10. (see convo [C] with Key NPC 10)
Fail: Key NPC 9 sends PC to Key NPC 5. (see convo [C] with Key NPC 5)

[B]: (Path: Key NPC 1[A] Fail - Key NPC 3[A] Optimal - Key NPC 5[A] Acceptable - Key NPC 9)

CONVERSATIONS WITH Key NPC 10:

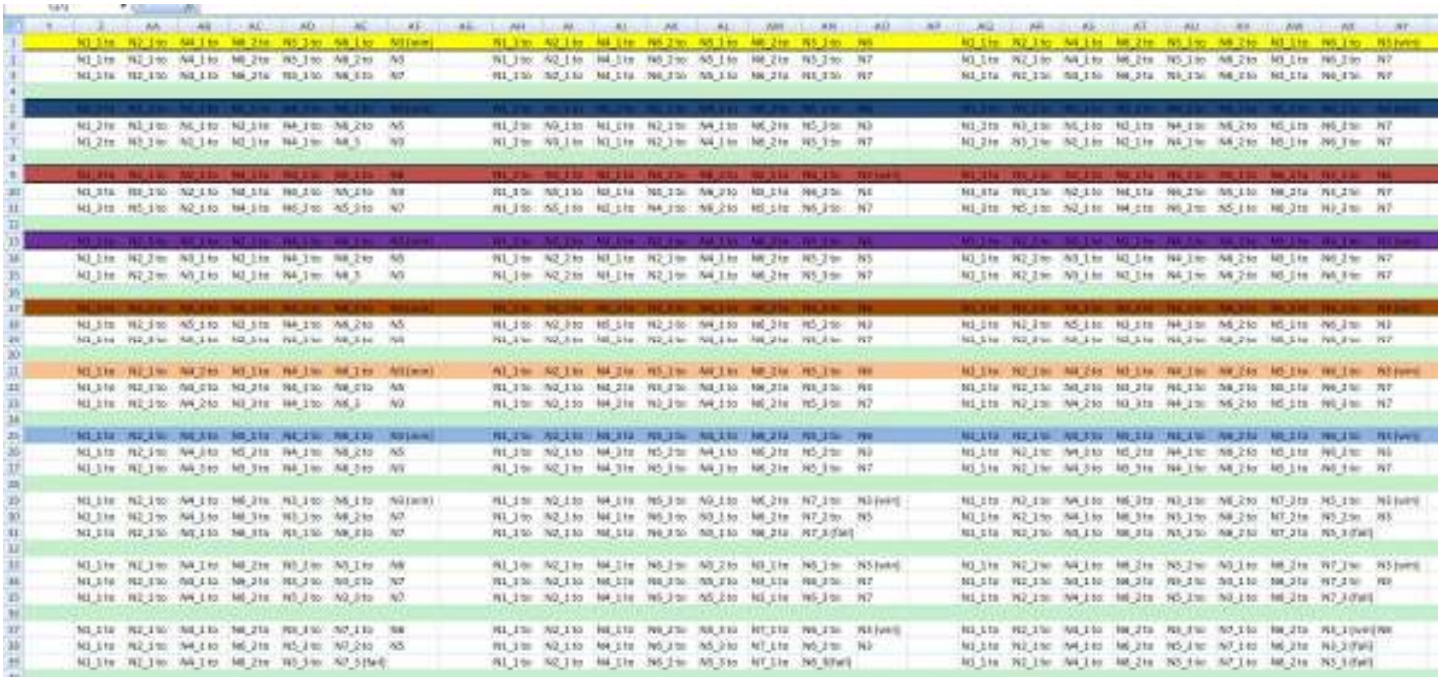
[A]: (Path: Key NPC 1[A] Acceptable - Key NPC 10)

Optimal: Key NPC 10 sends PC to Key NPC 2. (see convo [D] with Key NPC 2)
Acceptable: Key NPC 10 sends PC to Key NPC 3. (see convo [G] with Key NPC 3)
Fail: Key NPC 10 sends PC to Key NPC 7. (see convo [B] with Key NPC 7)

[B]: (Path: Key NPC 1[A] Optimal - Key NPC 2[A] Optimal - Key NPC 4[A] Fail - Key NPC 10)
[C]: (Path: Key NPC 1[A] Fail - Key NPC 3[A] Acceptable - Key NPC 9[A] Acceptable - Key NPC 10)

Appendix D

Sample of Methodology 2 – The Spreadsheet System



VITA

Brad Hennigan was born in Plano, Texas in 1964. After graduating from Plano East Senior High School in 1983, he attended Lon Morris College and The University of Houston, where he studied Fine Arts in Theater and Acting. He left university in 1986 to pursue a professional career in stage, television, and film. Brad spent 20 years working professionally in Dallas and Los Angeles as an actor, writer, director, and producer. He returned to his studies in 2006 at The University of Texas at Dallas, completing a Bachelor of Arts in Art and Performance in Fall 2008, and a Master of Fine Arts in Arts and Technology in Summer 2012.