Toward a Methodology for Writing Dynamically Immersive Branching Dialogue in Digital Games and Simulations

Brad Hennigan
The University of Texas at Dallas, Arts and Technology

Abstract. Game designers are masters of kinetic immersion, as evidenced by the popularity of first person shooter games such as *Halo* and *Doom,* which has resulted 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 (PC) and non-player characters (NPCs). Players have come to expect high fidelity in the graphical representation of the game world, and they expect the same fidelity in PC to NPC dialogue. Thus "good" dialogue requires a high degree of *dynamic immersion*, which takes into account where the PC has been, with whom he has spoken, and how his conversations affect the narrative of the game. This paper examines the state of the art in narrative creation and writing for video games and simulations, and proposes a methodology for writing dynamically immersive branching dialogue. Conclusions reached are based upon the author's research and development as head writer for the First Person Cultural Trainer (FPCT), UTD Arts and Technology's multi-award winning training simulation.

Introduction

Narrative in games and simulations is integral to player immersion and positive learning outcomes. (Fish, 2005) (Waraich, 2004) (Marchiori, 2011) (McQuiggan, 2008) Yet inconsistencies within narrative and dialogue in games and simulations are all too common. As researchers Janis 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." (Bowers, 2010)

Narrative rests at the heart of FPCT gameplay; its virtual environments and NPCs have a meta-narrative that provides the context that gives each NPC and Key NPC a storyline motivation. The process for developing this meta-narrative in FPCT, for example, involves analyzing and understanding the region's geography, history, language, folklore, social networks, and tribal and authoritarian structures. Once established, the meta-narrative provides the basis for dialogue between the player character (PC) and virtual characters (NPCs). (Zielke, 2012) This paper reviews the state-of-the art in dynamic narrative and dialogue generation, and describes

FPCT's current method of writing dynamically immersive branching dialogue. Future work includes a proposed hybrid approach to story generation, which is the development of Visual Conscript™, a graphically intuitive, non-technical authoring tool that would combine the elements of narrative, character, and dialogue generation.

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 (SMEs). This dilemma has spawned new 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, 2012)

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, 1992) and the Façade system (Mateas and Stern, 2003) 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. (Mateas and Stern, 2003) Similarly, DEFACTO (Sgouros, 1997), IDtension (Szilas, 2003) and Mimesis (Young, Riedl, et. al, 2004), though scalable, require significant content creation and ordering for longer narratives. Another system, INTALE (Riedl and Stern, 2006), 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, 2006)

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). (Barber and Kudenko, 2009) 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.

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, 2006)

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 & Mateas, 2008) 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 1. 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, 2008) 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, 2011)

Dynamically Immersive Branching Dialogue

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." (Aylett, 2006). 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, 2012)

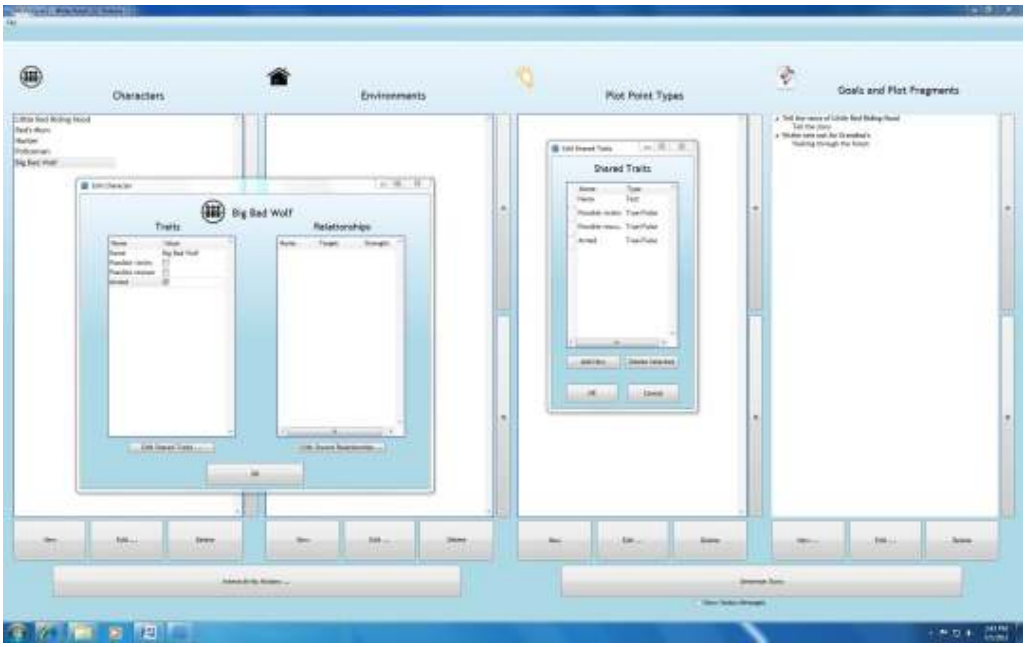


Figure 1. The Wide Ruled editor interface, allowing author-goal based story planning through SME chosen characters, plot fragments and variables.

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, 2006) 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, as noted earlier. 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 2. 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 through dialogue, the SME (or a technical expert) must then script the variable and condition logic that the narrative requires. Chat Mapper utilizes the Lua scripting language for programming the variable and condition logic, as the software's designers determined that Lua is easily integrated into various gaming engines. (Zielke, 2011)

Chat Mapper succeeds in assisting the author in keeping track of multiple 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, giving rise to the possibility of inconsistency in dynamically immersive branching dialogue throughout the narrative as a whole.

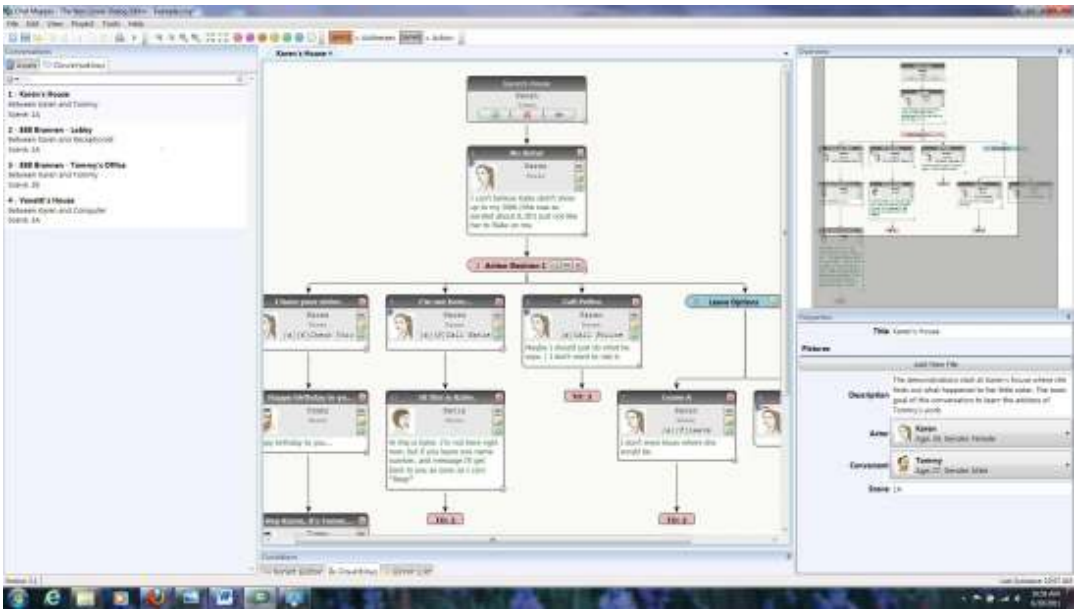


Figure 2. The editor interface for ChatMapper, which creates branching, interlinked conversations.

The FPCT Methodology

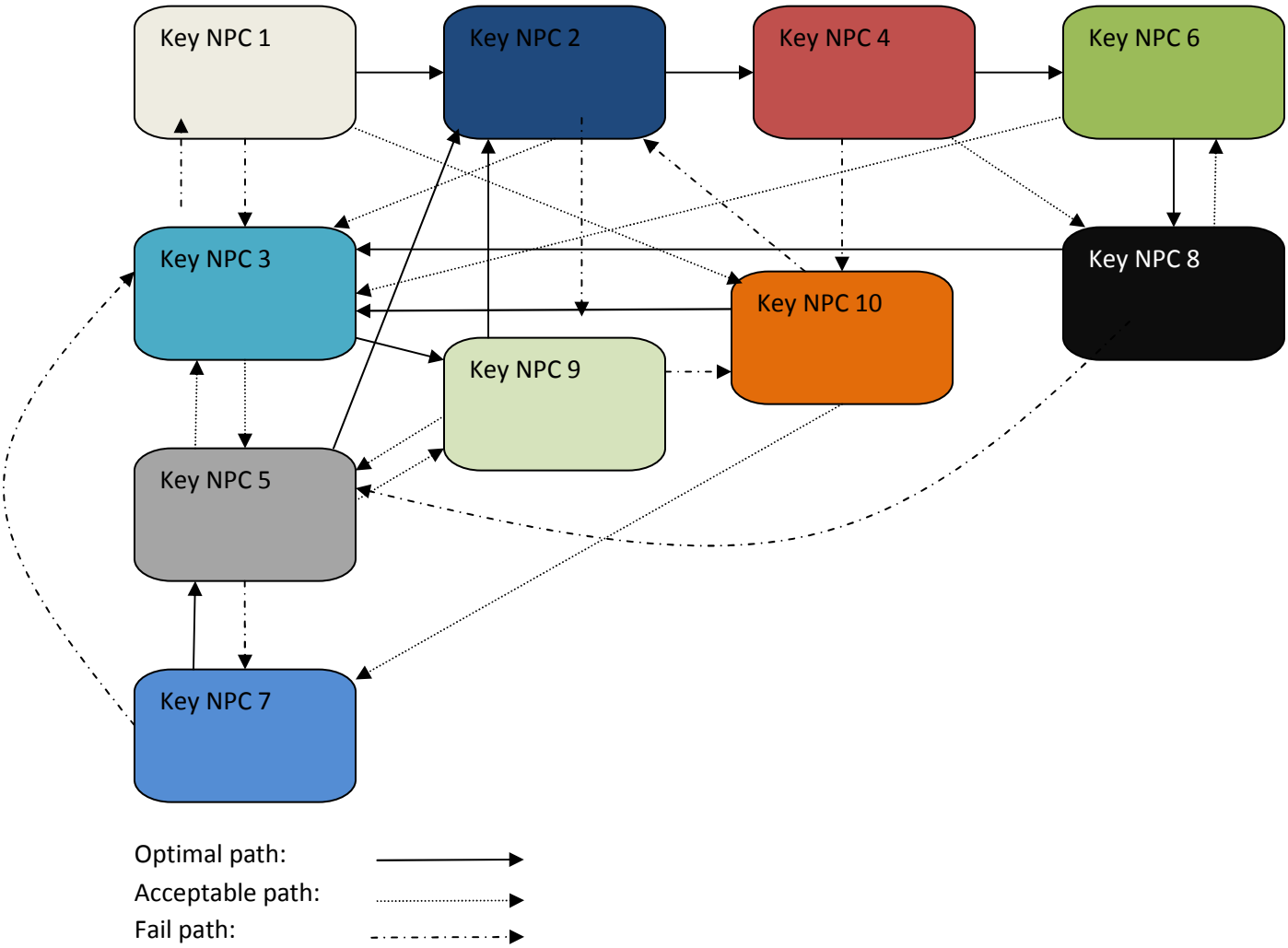
As head writer for FPCT, this author was tasked with creating a non-automated methodology for writing dynamically immersive branching dialogue, utilizing a three option question/response system, with the desired outcome of producing a guide from which to easily reference the path a PC has taken, and will take, through the course of game play. The knowledge of the PC's path allows the writer to then create dialogue specifically contextual to the PC's game play, as the author will know with whom the PC has spoken, what information was exchanged, and how that information fits within the narrative. Two methodologies were created: the first approaches the task from a micro level, detailing each individual conversation branch. The second approaches from a macro level, detailing each conversation as a whole.

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 PC-to-Key NPC interaction map detailing the PC'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 3).

Figure 3. PC-to-Key NPC interaction map



5. Once the interaction map is completed, the next stage for writers is to create a breakdown of PC-to-Key NPC conversations. This step is a multi-pass process for designing multiple conversations with many Key NPCs and keeping track of what's been said in dialogue, and is integral to designing non-linear narrative and tracking PC movement in order to create contextually pertinent 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 PC. 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 3, is found in Appendix 1. This example

represents one potential PC 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 PC so that dialogue reflects what the PC knows, who the PC has met and where the PC has been. (Tip: the author may choose to include a brief description of the PC-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 PC 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 PC is sent next.
c. Using the visual PC-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 PC has taken to reach this instance of conversation. Example: PC 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 PC 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 1).
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 PC 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 PC has taken to that conversation.

Methodology Two: The Spreadsheet System

The second methodology follows the same basic principle of breaking down the PC'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 2.

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 PC-to-Key NPC interaction map detailing PC's movement through Key NPCs during gameplay (see Figure 3).
5. Create a spreadsheet with the first PC 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 PC 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 PC is told to see based on the dialogue path the PC has chosen. (Example: if the PC fails with N1, PC 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 2, 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 PC is accounted for, and allow the writer to adjust dialogue accordingly by referencing the PC's game play path, which is integral to dynamic immersion. The potential figures for the number of conversations and branches discussed earlier were proven out when the methodology was applied to writing the latest version of FPCT, which utilized eight characters, or Key NPCs. This resulted in 258 unique PC to Key NPC conversations, and 2,263 unique conversation branches. Application of the methodology, including writing and coding the dialogue, took approximately 400 man hours and resulted in 32,000 lines of coded dialogue.

Future Work

FPCT's experience in creating and applying the methodology for writing and coding dynamically immersive branching narrative and dialogue led to the conclusion that automation of the processes involved is vital, in terms of both time management and cost efficiency. FPCT researchers posit that an amalgam of the elements found in automatic story generation, author-goal based story generation, and dynamically immersive branching dialogue writing could result in a robust knowledge domain SME tool, a dynamic automated story engine that would reduce the SME's reliance on writing and technical experts. (Zielke, 2012) The FPCT team is currently working to combine the elements of narrative, character, and dialogue generation into one graphically intuitive, non-technical authoring tool called Visual Conscript™.

Visual Conscript™ would serve as a tool for branching-narrative design, conversation tracking and dialogue authoring. The Visual Conscript Editor will help SMEs in the complex task of designing training scenarios and writing the contextually accurate branching dialogue for NPCs. The Visual Conscript™ Editor tool would make scenario writing more efficient by generating the necessary programming code in the background as the author creates the narrative. Currently, the authors code Conscript™ conversation files as they design narrative scenarios and write dialogue. The objective is to have the Visual Conscript Editor handle these functions. (Zielke, 2012)

Conclusion

FPCT's research concluded that the creation of a methodology for writing dynamically immersive dialogue was necessary to ensure constant and consistent PC dynamic immersion in game play. Through the process of creating and applying this methodology, it became evident that the detail-centric requirements of successfully completing the dialogue for even a relatively small narrative were time consuming, to the point of being considered prohibitive. However, the resulting dynamic immersion level achieved was highly desirable. Thus an automated approach has been proposed by the FPCT team in the form of Visual Conscript™.

References:

Aylett, R., et al. (2006) Unscripted narrative for affectively driven characters. *Computer Graphics and Applications, IEEE 26*(3), 42-52.

Barber, H., & Kudenko, D. (2009) Generation of adaptive dilemma-based interactive narratives." *Computational Intelligence and AI in Games, IEEE Transactions on 1*(4), 309-26.

Bates, J. (1992) Virtual reality, art, and entertainment. *Presence: J. Teleoperat. Virtual Environm 1*, 133–138.

Bowers-Canon, J., & Bowers, C. (Eds.). (2010) *Serious game design and development : Technologies for training and learning*. Hershey, PA. Information Science Reference.

Fisch, S.M. (2005) Proceedings of the 2005 Conf. on Interact. Des. and Child. *Making education computer games "educational."* Boulder, CO.

Krawczyk, M., & Novak, J. (2006) ***Game development essentials: Game story & character development.*** New Yok. Thomson Delmar Learning.

Marchiori, Eugenio J., et al. (2011) A visual language for the creation of narrative educational games. *Journal of Visual Languages & Computing 22*(6), 443-52.

Mateas, M., & Stern, A. (2003) Proceedings of Game Developers Conf./Game Design Track. *Façade: An experiment in building a fully realized interactive drama*.

McQuiggan, S.W., Rowe, J.P., Lee, S., & Lester, J.C. (2008) Story-based learning: the impact of narrative on learning experiences and outcomes. *Lect. Notes Comput. Sci., 5091*, 530-539.

Nelson, M. J., et al. (2006) Declarative Optimization-Based Drama Management in Interactive Fiction. *Computer Graphics and Applications, IEEE 26* (3), 32-41.

Riedl, M., & Stern, A. (2006) Proc. 3rd Int. Conf. Technol. Interactive Digit. Storytelling Entertain. *Believable agents and intelligent story adaptation for interactive storytelling*. Darmstadt, Germany.

Sgouros, N.M. (1997) Proceedings of Int. Joint Conf. Artif. Intell. *Dynamic, user-centered resolution in interactive stories*, *2*, 990–995.

Skorupski, J., and Mateas, M. (2008) *Interactive Story Generation for Writers: Lessons Learned from the Wide Ruled Authoring Tool.* Retrieved from http://games.soe.ucsc.edu/sites/default/files/Skorupski_DAC09_WideRuledLessonsLearned.pdf

Szilas, N. (2003) Proc. 1st Int. Conf. Technol. Interactive Digit. Storytelling Entertain*. IDtension: A narrative engine for interactive drama*. Darmstadt, Germany.

Waraich, A. (2004) Proceedings of Ninth annual SIGCSE Conf. on Innov. and Technol. in Comput. Sci. Educ. *Using narrative as a motivating device to teach binary arithmetic and logic gates*. Leeds, United Kingdom.

Young, R.M., Riedl, M.O., Branly, M., Jhala, A., Martin, R.J., & Saretto, C.J. (2004) An architecture for integrating plan-based behavior generation with interactive game environments. *Journal of Game Develop. 1*, (1), 51–70.

Zielke, M., Hardee, G., Hennigan, B., & Leflore, J. (2011) Interservice/Industry Training, Simulation, and Education Conference 2011 (submission) *Tools and techniques for managing subject matter expert input into virtual simulations.*

Zielke, M., Donahue, J., Hardee, G., Hennigan, B., Kaiser, M., Keown, B., & Laughlin, J. (2012) White Paper. *FPCT 4+ architecture: The way ahead*.

Appendix 1

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 2

Sample of Methodology 2 – The Spreadsheet System