Spring 5-31-2021

The Kati Module System: Modular Design for Delivering Character Focused Dialogue in Games

Stephen J. Marcel
University of New Orleans, sjmarce2@uno.edu

The Kati Module System: Modular Design for Delivering Character Focused Dialogue in Games

A Thesis

Submitted to the Graduate Faculty of the
University of New Orleans
in partial fulfillment of the
requirements for the degree of

Master of Science
in
Computer Science

by

Stephen Marcel

B.F.A. Nicholls State University, 2011

May, 2021

Table of Contents

Figure List iv

Abstract vi

1 Introduction 1

1.1 Motivation 2

1.2 Objective 2

1.3 Related Work 3

2 System Design 5

2.1 Kati Module System 5

2.2 Responsibility of the Game 6

2.3 Module Hub 6

2.3.1 Module Hub Components 7

2.3.2 Storyline Nodes 8

2.3.3 Location and Character Collections 10

2.3.4 Character and Module Accessibility Relationship 12

2.3.5 Module Decision Mechanics 13

2.3.6 Preparing for the Module 13

2.3.7 Character Data 14

2.3.8 Dialogue Package 14

2.3.9 History 15

2.4 Module 17

2.4.1 Dialogue Topics and Types 17

2.4.2 Structure of a Module 18

2.4.3 Controller 18

2.5 Parser 19

2.5.1 Branch Decisions 21

2.5.2 Required Rules 22

2.5.3 Case for No Available Dialogue 24

2.5.4 Leads To Rules 25
2.5.5 Response Type Dialogue 26
2.5.6 Dialogue Chaining 27
2.6 Dialogue String Parser 28
3 Results and Discussion 29
3.1 Experiment Setup 29
3.2 Job Interview Game 30
3.2.1 Job Interview Results 33
3.3 Text Adventure Game 34
3.3.1 The Eleven Townsfolk 35
3.3.2 Adding Rule Complexity 41
3.3.3 Changing Override Playstyle 42
3.3.4 Reviewing Results 46
3.4 Limitations 47
3.5 Conclusion 48
4 References 49
Vita 51

List of Figures

2.1 Module's Structure Overview 5
2.2 Module and Module Hub Relationship 7
2.3 Storyline Node 9
2.4 Storyline Node Structure 10
2.5 Location Object Scheme 1 11
2.6 Location Object Scheme 2 12
2.7 Character Module Relationship 13
2.8 Dialogue Package 15
2.9 History Structure 16
2.10 History Structure Example 17
2.11 Topic and Type Keys 19
2.12 Topic Types 19
2.13 Module Organization 20
2.14 Potential Dialogue Rules 23
2.15 Branch Tone Dialogue Chain 26
2.16 Sample Delivery Content 29
2.17 Rendered Sample Delivery Content 29
3.1 Job Interview Negative Response A 31
3.2 Job Interview Positive Reaction 32
3.3 Job Interview Negative Response B 32
3.4 Job Interview Negative Reaction 32
3.5 Job Interview Game Over Screen 33
3.6 Job Interview Survey Response Results 34
3.7 Albrecht Positive Relationship 37
3.8 Christina Positive Relationship 37
3.9 Dan Positive Relationship 37
3.10 Teta Positive Relationship 37

3.11 Final Positive Relationship Values 37
3.12 Sample Positive Dialogue 38
3.13 Albrecht Negative Relationship 38
3.14 Christina Negative Relationship 38
3.15 Dan Negative Relationship 39
3.16 Teta Negative Relationship 39
3.17 Final Negative Relationship Values 39
3.18 Sample Negative Dialogue 39
3.19 Albrecht Romantic Relationship 40
3.20 Christina Romantic Relationship 40
3.21 Dan Romantic Relationship 40
3.22 Teta Romantic Relationship 40
3.23 Final Romantic Relationship Values 40
3.24 Sample Romantic Dialogue 41
3.25 Sorceress Possible Dialogue Options 42
3.26 Player Dialogue Approaches 43
3.27 Generic Playstyle Approach Relationship Values 44
3.28 Positive Playstyle Approach Relationship Values 44
3.29 Romantic Playstyle Approach Relationship Values 44
3.30 Mean Playstyle Approach Relationship Values 44
3.31 Strategic Playstyle Approach Relationship Values 44
3.32 Final Relationship Values with Sorceress 45
3.33 Most Used Discussion Topics 45
3.34 Overall Discussion Tones 45
3.35 Sample Dialogue Strategic Playstyle 45
3.36 Sample Dialogue Friendly Playstyle 46
3.37 Sample Dialogue Rude Playstyle 46

Abstract

The Kati Module System is an interconnected set of programming modules intended to facilitate dynamic text authoring for interactive experiences (for example, games). It is a long-standing goal for interactive experiences to dynamically adapt their textual output based on the user or player's choices and predilections, but to account for this vast possibility space requires an amount of authoring that is frequently untenable, especially for small studios. Advances in machine learning have produced incredible progress in the field of Natural Language Generation (NLG). Though this produces impressive surface level text, it does so without an internal representation that can be reasoned over previous game states, resulting in output with deep local coherence and low global coherence. Kati attempts to provide the best of both worlds by allowing authors to author configurable text snippets. Kati dynamically rearranges and chooses dialogue phrases based on game state, allowing for high degrees of authorial control, global coherence, and dynamic adaptability to player choice.

Keywords: Game Development, Dialogue Delivery, Modular Design, Character Relationships

1. Introduction

The art of storytelling is a powerful medium that can take on many forms. Whether it be oral, digital or written, stories have been a part of human societies for millenniums. Across traditional narratives, a consistent underlying story structure emerged that revealed three primary processes: staging, plot progression, and cognitive tension (Boyd et al., 2020). In this paper, the author explores the Kati Module System, which is not concerned with developing a story but rather creating user defined character relationships for a more robust narrative experience. This is accomplished by a series of social interactions.

Through character social interactions, many stories are told with the relationship between said characters dictating how the narrative comes to fruition. Social interactions made with characters with no long standing history can undermine the sense of the player's impact on the story and damage the overall believability. Believability can be defined as the numerous elements that allow a character to achieve the illusion of life, including but not limited to personality, emotion, intentionality, physiology, and physiological movement (Riedl & Young, 2005).

Kati Module System (Kati) is a computational model designed for digital games to manage a character relationship state with the other story characters throughout the game's narrative. The goal is to maintain believability on a per conversation basis as an entity in a larger story. It uses a series of modules that define dialogue processing protocols for any number of topics. Each module deals with a specific subject or action that a player's character (PC) or non-player character (NPC) can encounter throughout the game through social state (Cowley et al., 2008).

Social interactions using Kati Module System provide opportunities to affect the state of an NPC directly. The Module Hub component determines an appropriate module based on the state of the game and the characters involved in the conversation. The module determines a "good fit" dialogue phrase by analyzing the game and character data and returns the selected phrase to the game. Each conversation is built upon multiple module calls chaining a series of dialogue pieces. Each conversation allows for player derived impact causing the next exchange with the system to change paths. These interactions use a cause-and-effect type structure. The player does some 'action' linked to Kati, and the effect of the 'action' changes the relationship status of the characters with dialogue encounters following a turn-based scheme. Turn-taking is a basic dialogue control element that concerns how we distribute and organize the moves in a conversation (Brusk & Björk, 2009). Kati allows for an alternate method of character relationship control that can alter the state of a narrative by delivering dialogue

derived from the socially build relationships of game characters. This can lead to the reuse of dialogue and more flavorful in-game character conversations.

1.1 Motivation

The motivation for creating the Kati Module System stems from RPG style games with open-ended concepts, particularly the hit indie game “Stardew Valley” (Barone, 2016). Though not unique to “Stardew Valley”, the game’s mechanics for building relationships with individual characters was the base for Kati’s creation. The idea of relationship-building between game characters based solely on the player’s in-game pursuits and actions is the Kati Module System’s crux. The goal is to build off of games like “Stardew Valley” and expand the depth in which character’s relationships intertwine and birth a form of a story by the player interactions and the game world their character exists in. The author designed and constructed the Kati Module System to attempt to achieve this goal. The hope is to establish a level of player immersion and flow (Mirvis, 1991) by eliciting a since of emotions through the conversations of player character (PC) and non-player characters (NPC) (Oudah et al., 2018). This is accomplished by means of appropriate dialogue delivery to a conversation that matches a situation or relationship state that two characters share.

1.2 Objective

The Kati Module System’s core objective is to provide a way to deliver authored dialogue content to a game directly derived from the player’s interactions. The aim is to maintain a sense of believability throughout and supply relationship driven narratives between game characters that are potentially unique each playthrough. It incorporates a modular design offering encapsulation of components, where each component can work as a black box. A module needs certain data elements in and returns dialogue data out. These data elements consist of the game’s current state like time, setting, or trigger event, the relationship status between the active participants, the relationship statuses between other characters or factions, and each characters’ attributes. This data allows the system to produce chains of dialogue using a series of modules. Each module’s specifics are unique to itself and have no bearing on the rest of the system but are combined to create a unique player experience through dialogue. In this paper, we will focus on the individual module and how it functions in the

system. We will dive into the technical structure of the generic module design and all decision protocols used thereby.

1.3 Related Work

Games using interactive narratives through social development are not a new phenomenon. The Sims Franchise (Wright, n.d.) is an example of a commercially successful, interactive game where much of the gameplay centers around social interactions with other NPCs. The player controls several sims, of interacts with other NPC sims, and can establish various relationships. Each sim is simply a virtually simulated person containing skills, relationships and the capacity to gain employment. The Sims offers a sense of autonomy that yields the illusion that the NPCs in the world live their own lives and make their own choices. The Sims uses a highly visual and auditory array of elements that inform the player of the relationships, moods, statuses, desires, and needs of the characters. These elements convey to the player the state of the individual relationships. The Kati Module System (Kati) uses the same core ideas of having undirected relationships between characters and establishing and presenting them differently. Juxtaposed with the Sims visual and auditory relationship indicators, Kati relies on a human-readable dialogue between character interactions to convey a social exchange.

The LabLabLab's game design research headed by Jonathan Lessard, Associate Professor at Concordia University in Montreal, has created a series of player interactive dialogue-driven games. They have produced the games "A Tough Sell," "Sim Prophet," and "Sim Hamlet" with the primary goal of utilizing interactive conversations as a game in itself. Each of these projects focused on gameplay, where a player can accomplish a task through conversational moves. The LabLabLabs described conversational moves as an utterance or series of utterances intending to change the conversation state (Lessard, 2016). The gameplay is reminiscent of a chat room where a player would directly type phrases to complete a task.

The three games explore Natural Language-driven gameplay using the Chat Script chatbot engine as the game's primary dialogue engine. Players can respond to situations in their own words, and Chat Script will process the input and respond. The Chat Script differs from the Kati Module System in that Chat Script relies on Natural Language Processing (NLP), where Kati manages pre-written dialogue based on character relationships and the current game state. Though Chat Script allows for much more freedom of user input, the dialogue cohesion is limited to the local dialogue (Lessard, 2016). In contrast, Kati does not process user text input, restricting freedom, but allows for a more cohesive global narrative by maintaining the relationship state of characters and providing dialogue to fit each given

state throughout the entire game. Because of Chat Script Engine's underlying nature, it can have a leaky fictional coherence with miscommunications and amnesia concerning the conversation state. Both designs have strengths and weaknesses, and their usefulness depends upon the application being produced.

The Kati Module System is a system that focuses on the delivery of appropriate dialogue for a situation based on a series of character and game factors, where dialogue coordinates information and involves building social cohesion (Allwood, 1992) and controlling and restricting the interaction (Allen et al., 2001). The Kati Module System differs from other systems, such as the GRIOT system, which uses user input to algorithmically generate a series of computational output narratives (Harrell, 2006) and the ELIZA system, which searches user input for keywords. ELIZA generates a response using a rule associated with the keyword (Weizenbaum, 1983), in that user input is derived from pre-written dialogue responses and other decisions made through interactions. The Kati Module System is concerned only with dialogue delivery with consideration for past and future conversations, where GRIOT and ELIZA focus more with NLP responding to the current state of the conversation. NLP uses natural language understanding (NLU) (Mateas & Stern, 2004; Sabah, 2011) on input data and natural language generation (NLG) (Koller & Petrick, 2011) on output.

Comme il Faut (CiF) (J. McCoy et al., 2010), revitalized as Ensemble (Samuel et al., 2015), is a social dynamics engine used for the game "Prom Week" (Josh McCoy et al., 2011). CiF provides a rich social climate using a social exchange structure powered by a malleable set of rules. Game characters are enriched by relational situations and interconnected histories defined by experiences, social forces, and current circumstances. Characters undergo social exchanges that contain an intent or intended change to the social state. The volition of each character to accept or reject a social exchange depends on an authored ruling system. By using a series of rules, character relationships can be established in a non-linear way.

CiF, in itself, is not playable. Characters commence a social exchange, and each character accepts or rejects the social exchange based on a series of conditions. These conditions consider the histories between characters and the rule structures that apply to weigh the social exchange's acceptance or rejection.

Kati takes much inspiration from the design and functionality of CiF. It, too, is an extension that games can use and is not playable on its own. Kati uses a form of the rules system and is built around

game character traits and relationships. Though many aspects of the two systems are similar, the two differ in their core concerns. CiF is more concerned with what actions characters will do in social exchanges, such as dating a character or becoming friends with them. In comparison, Kati does not execute an action such as causing character “A” and character “B” to become romantically involved. Instead, it provides a means to deliver appropriate dialogue for character “A” and character “B” for a situation with consideration for their relationship status of “is dating.”

2 System Design

2.1 Kati Module System

It is essential to understand how each systems’ elements interact with each other and their core responsibilities to access the Kati Module System’s full potential. The system comprises three main components: The Module Hub, the modules, and a dialogue collection. Firstly, the Module Hub is responsible for selecting which broad topic or module will be discussed. There can be any number of modules available to the Hub. Each module is a container for dialogue that conforms to a specific subject. Each module’s responsibility is to decide on the final dialogue string to be returned based on various information (we will discuss this in detail). Lastly, the raw dialogue is needed to feed the modules. This raw dialogue is stored in a series of JSON files to serve as a database for these strings. Each of these elements works together in conjunction with an outside entity, the game. The game relays information to the Module Hub, which then provides information to each module in turn.

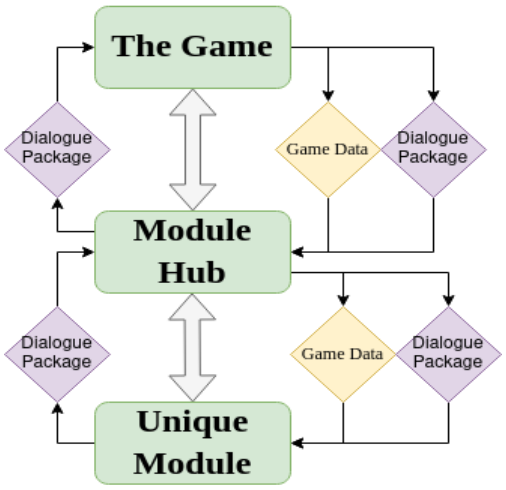


Figure 2.1: Overview structure of the Module System in relation to the game

2.2 Responsibility of the Game

The game is an essential element that the system is dependent on, as Kati is not playable in itself. Instead, the system uses vital information provided by the game before devising any dialogue scheme. Elements such as setting, time, personal character attributes, and social character attributes are imperative for producing meaningful dialogue. Without a regular update of the game's current state, it would be impossible for dialogue evolution to take place. Game elements about player location, actions that can occur, and available characters must be defined and reflected in the raw dialogue and rules.

The game state includes all of the predefined social character attributes and personal character attributes. Kati has a series of stats that ultimately decide the conversation tone. These stats can be managed through the game and through the system itself. Actions, game events, and player-defined dialogue decisions can affect these stats.

There are three primary types of parameters that Kati requires from the game. The first is game data, such as time, setting, and event trigger elements, like the player's Character (PC) has reached level 50 or completed a quest. This game data directly links to what the player is doing at any given time. The second parameter required is each characters' personal data. Kati utilizes two forms of character attributes. The first is a character's personality traits. Personality traits represent how that character behaves and their personal views, interests, desires, and taboos. The third required parameter is the social character traits. Social character traits function much like personality traits but express relationships for other characters. A social trait would be a directed relationship like "is dating" or the romance level two characters share. This romance level is a numerical value that gauges a quantifiable representation of the character's romantic relationship. Social character traits have a strong influence on the overall tone of the dialogue. If a character has a social relationship with another character, their speech will reflect the relationship. How this could affect dialogue can be seen by how a character reacts to something another character says.

2.3 Module Hub

Inquiry into the state of the game is the first step to discerning appropriate dialogue. This responsibility falls on the Module Hub. It is tasked with ensuring the game and character data is current

and accurate. The Module Hub decides on the best module to return as a conversation to the game by utilizing game state data. The Module Hub houses reference to all of the modules, where each module represents a potential topic of discussion. The subject defined in the chosen module is called by the Module Hub and passed the relevant information derived from the current game and character state to that module. Once a module is selected and called, the module will return a Dialogue Package matching the game and character criteria. The Dialogue Package contains the chosen dialogue string and additional metadata to instruct the Module Hub and the game itself.

The Module hub first reads signals from the game. These signals include when to fetch a dialogue phrase and signal appropriate types of modules to the game's current state. Once the game signals are accounted for, the personal character data and the social character data are considered. The modules are ranked based on the character data, and one is chosen based on a weighted probability. The social character data works similarly to personal data. Both the personal and social relationship data, along with the game data, is passed to the chosen module.

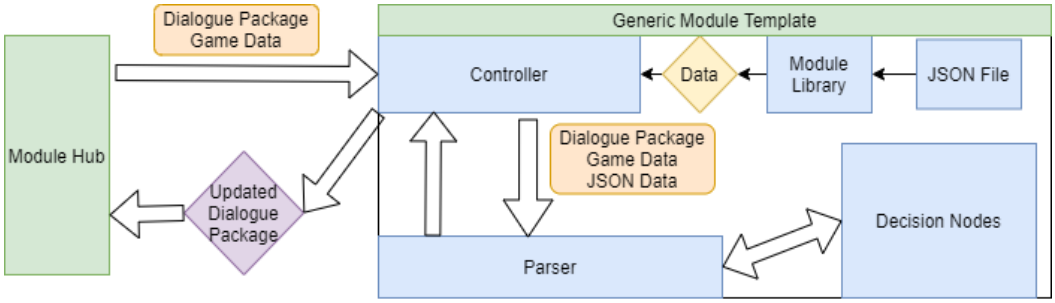


Figure 2.2: Structure of the module in relation to the Module Hub

2.3.1 Module Hub Components

Inquiry into the state of the game is the first step to discerning appropriate dialogue. This responsibility falls on the Module Hub. It is tasked with ensuring the game and character data are current and accurate. This is done by updating pertaintent game and character information on each iteration through the system. The Module Hub decides on the best module to return as a conversation to the game by utilizing game state data. The Module Hub houses reference to all of the modules, where each module represents a potential topic of discussion. The subject defined in the chosen module is called by the Module Hub and passed the relevant information derived from the current game and character state to that module. Once a module is selected and called, the module will return a Dialogue Package matching the game and character criteria. The Dialogue Package contains the chosen dialogue

string and additional metadata to instruct the Module Hub and the game itself. Concerning the Module Hub, these instructions may inform which module, topic, and even which dialogue value to choose next. The Dialogue Package may signal the game informing it which portrait card to supply on the game UI or other information like signaling that a quest conversation has been completed and the door to the quest can now be unlocked.

The Module hub first reads data from the game. This data are established by the game developers and relate to author defined rules provided to each module and house information like location and setting updates or game event elements like “town is under attack.” This data includes when to fetch a dialogue phrase and signal appropriate types of modules to the game’s current state. Once the game signals are accounted for, the personal character data and the social character data are considered. The modules are ranked based on the character data, and one is chosen based on a weighted probability. The social character data works similarly to personal data. The personal and social relationship data, along with the game data, are passed to the chosen module.

The Module Hub consists of a group of components that work together to decide which module to select and prepare data for the selected module to make an informed choice. These components are the Storyline node, location collection, character collections, history data, and character data. Each of these components will be briefly discussed in the next few pages.

2.3.2 Storyline Nodes

The Module Hub’s default structure is designed to handle the separation of significant story milestones or plot points into Storyline Nodes. Each Storyline Node encapsulates a reference to all characters, locations, and modules associated with a particular ‘Chapter’ in the story. To better illustrate this, consider a narrative about a country at war that contains three major plot points. The first point occurs before the war, the second, during the war, and the last, after the war has ended. These storyline elements fit together to make a whole, but each can have unique attributes that only fit their respective plot point. The dialogue’s overall mood and tone can differ depending on which ‘Chapter’ the narrative is in. Also, the possible locations and characters involved can change depending on which plot point is currently underway. Each of these significant points in the story can be translated into a Storyline Node. The Storyline Node idea is only to include elements of the game that currently exist in

the story's context at specific points. Figure 2.4 shows a simplified diagram of a Collection of Storyline Nodes.

Storyline Object	
Node 1	1st part of the game's Story
Node 2	2nd part of the game's Story
⋮	⋮
Node N	Nth part of the game's Story

Figure 2.3: Storyline Node Containing 'N' Chapters

Though it might be necessary for a game to include many significant shifts, whether it be the main character's death, setting alterations, or any other storytelling device, not all games will require this feature. In many instances, the location, characters, and overall mood will stay the same throughout the game's duration. The Module Hub only requires one Storyline Node to be created with the option to make any number of additional Storyline Nodes if needed. A Storyline Node translates vital plot points by tracking all locations or setting elements in a specific plot point and every character available during this 'Chapter'. The Module Hub maintains a reference to a collection of these nodes and can set any Storyline Node as 'active' guided by signals from the parent game. See figure 2.4 for the breakdown of the Module Hub's collection of Storyline Nodes. The active Storyline Node will provide the game with access to all of the modules relevant to the particular state of the overall story progression.

Storyline Nodes can be arranged in any order. From the perspective of Kati, a Storyline Node is a data structure that holds a series of references of all locations and characters of a particular chapter. The Storyline Node structures use string values to represent the 'Chapter' or Storyline segment. Each act points to a collection of string values that represent locations that are available in this act. Each location string holds a reference to a character that can be found there. Using the war example, the Storyline Node, "before the war," may have a location "town item shop," and in this location, characters "Shopkeep" and "Mr. Muscles" can be found here. The character references link to character objects which hold their specific data. On the second Storyline Node, "during the war," the location "item shop" may still be available, but only the character "Shopkeep" can be found here, "Mr. Muscles" joined the army and is off to war.

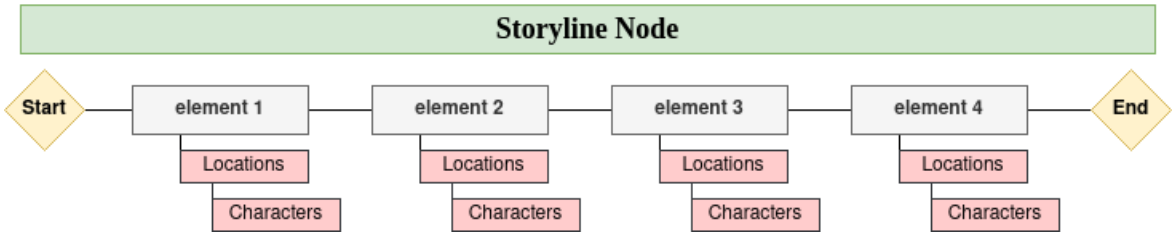


Figure 2.4: Structure diagram of a collection of Storyline Nodes

2.3.3 Location and Character Collections

Location collections are just collections of locations represented by a key string value. These keys represent locations in the game and give a reference to location-based rulings in the modules. Each StoryLine node contains a collection of locations, and each location contains a collection of characters in that location. The default design of the Module Hub assumes that all locations are constant to each Storyline Node. For instance, if a specific Storyline Node has a town location, forest location, and a mountain location, these locations will never be removed or changed, and a new location cannot be added. However, additional features overriding this default behavior can easily be implemented by the game developers. The Module Hub requires the current location of the dialogue sequence when a dialogue interaction is triggered.

Locations specified in a Storyline Node can be as general or as specific as the game developers require. For instance, locations can be as broad as a region in the game world or as specific as someone's bedroom in a particular house in a town located in a specific spot in a region. The more specific a location is, the larger the potential overhead is for managing all locations.

Locations are a means to enforce rules about the Module Hub selecting a module or a module selecting dialogue. To illustrate locations and how they fit with rulings, consider three locations pointing to a progressively more specific game location, the first, "In town," the second, "Martha's House," and the last, "Martha's Bedroom." Each of these locations is located in the town, but each becomes increasingly more specific. Rules can be applied to each of these locations that alter the Module Hub's decision of which module is selected, and which dialogue phase wins. Since all three locations are essentially "in town," any modules that are not applicable for use when the location is "in town" are dismissed; likewise, any rule that specifies a specific location is ruled out unless they meet the requirement of the current location. If the player is speaking to someone in Martha's bedroom and there are three module or dialogue options that house three different requirements, all relating to the

location, the outcome is dependent mainly on rulings defined by the game developers. Suppose these three location requirements are “in town,” “Martha’s house,” and “Martha’s bedroom.” If this is the case, then all three options may be valid, or only the option containing the requirement “Martha’s bedroom” might be valid. This decision is up to the game developers and is established when location rules are created. All three of these locations can exist inside of the location collection. This level of control allows authors to create situations that are exclusive to specific locations. The designers might want to create rules that pertain exclusively to locations, such as conversations that can only happen in ‘Martha’s bedroom’ and not ‘in town’ even though ‘Martha’s bedroom’ exists ‘in town.’ Actions Again the design is dependent on the game developers.

With the active StoryNode and the location key, the next layer is the characters involved in the interaction. By default, the Kati Module System assumes that one of the characters in any dialogue interaction is the PC and the other is an NPC. Using this assumption that one character is the player, the Module Hub’s next entry is the name or key of the NPC that the PC is speaking with. The NPC in the interaction can be referenced, and from here, an individual module can be decided on.

Locations in the location collections can point to many characters, and multiple locations can point to the same characters. (See figure 2.5 and 2.6 for location and character collection relationship.) Characters can move from one location to another by altering a location’s character collection to match the game state. This alteration is a simple way to track which characters are in a specific location at any given time. Though the location collection feature exists in the Module System, it is not required. Alternative solutions created outside of Kati can be implemented to track character locations to better suit the game’s needs. It may be more comfortable or necessary for the game designers to track characters based on specific criteria such as graphics rendering, collision signals, or grid-based movements and pass the character’s reference in the interaction to the module system.

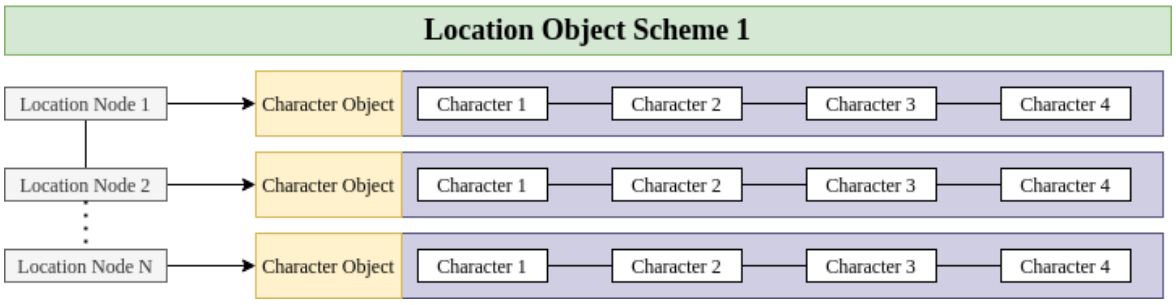


Figure 2.5: Location nodes point towards separate character reference lists

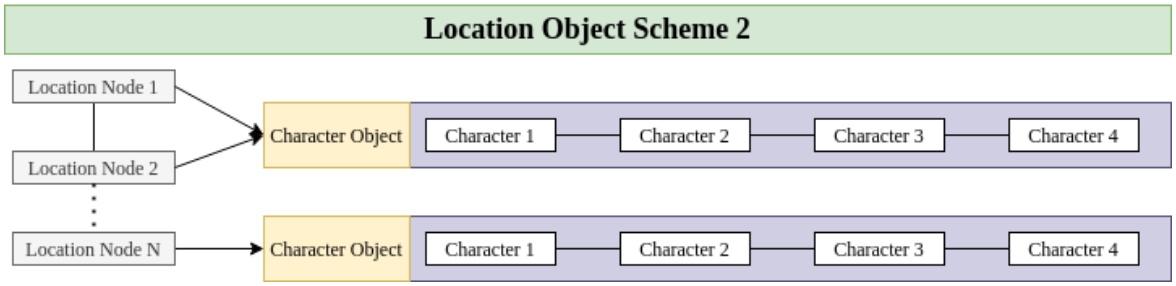


Figure 2.6: Location nodes pointing towards the same character reference lists

2.3.4 Characters and Modules Accessibility Relationships

A game can have many different dialogue modules that are used for a variety of situations. A module's relevance can be based upon many factors such as the Storyline Node, location and Setting elements, characters involved in the interaction, or the characters' relationships. From the Module Hub perspective, all modules are chosen by first providing a character involved in the dialogue interaction and then choosing a module that applies to the NPC in question. The Module Hub maintains a list of all modules that each character can access throughout a Storyline Node. Characters can have unique modules assigned to them as well as communal modules. Consider a game with a town guard, a blacksmith, and a town elder. All three of these characters may be in the same village and exist in the same Storyline Node. Since all three characters share this space in the game, it may be beneficial for each to share access to a general "around-town" module. This localized sharing allows for one module to service multiple characters. Even though these characters share the same space, they could have exclusive modules that cater to their specific profession. The guard might have exclusive access to a "Law" module, the blacksmith to a "barter" module, and the town elder to a "quest-giving" module.

Alternatively, each character can have exclusive modules designed for that specific character or all the modules shared by all characters. Consider the module “Small Talk,” which might house just general small talk. Many characters in a game might need this module, but the game designers may want to customize every module for specific individuals. The developers can produce different versions of the same small talk module like “Small Talk Tung” or “Small Talk Farah” for each character or only those that require it. Each new “Small Talk” version of the module is a new set of dialogue that is custom-tailored to “Tung” or “Farah.” Both versions are essentially unique instances of modules that share the broad topic “Small Talk.” The specifics are primarily up to the game designers and content

creators when creating the game. Figure 2.7 illustrate an exclusive only design and an exclusive and sharing design to the character to module accessibility.

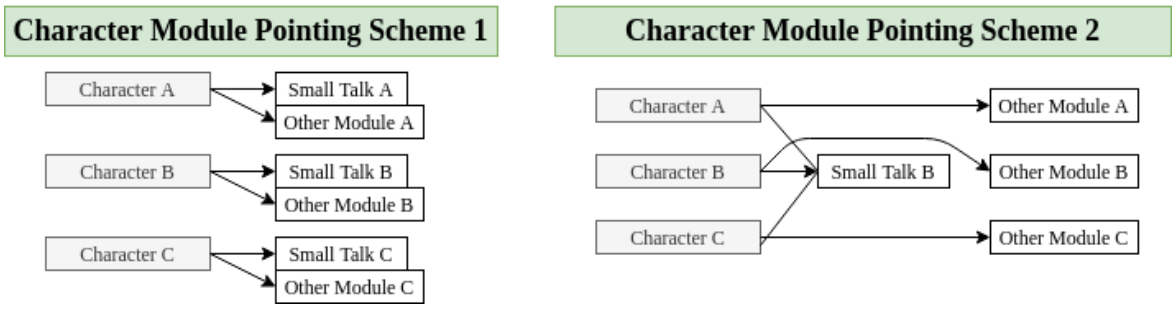


Figure 2.7: On the left, all characters have unique modules. On the right, all characters share module "Small Talk B" as well as having unique modules

2.3.5 Module Decision Mechanics

Using the active Storyline Node, the interaction location, and the interacting character reference, the Module Hub has access to all possible modules that can be used in the interaction. Because every game is different and will have unique modules and contingencies for selecting the module, The Module Hub decision structure contains only a single default protocol for choosing a module. It merely selects an available module to run randomly using a uniform distribution. Though this structure exists in the Module Hub by default, it is not meant to be used per se but overridden by functionality supplied by the game itself. This design was cemented during testing when all attempts to find a one-size-fits-all solution proved only to complicate and limit the game itself and ultimately unsuccessful out of the box. The Module Hub requires the game to define the rulings that narrow down and decide which module is selected. There is no way of knowing beforehand what each game will value as important or what qualifies as a means to select a module.

2.3.6 Preparing for the Module

After deciding which module to use, certain data elements in the Module Hub must be updated to prepare for that module's execution. Elements like the current state of the characters and any game-triggers needed to be reflected in the Module Hub for the chosen module to decide which dialogue option is best. The main form of communication between the Module System and the game takes the

form of game data objects and the Dialogue Package. The Dialogue Package is responsible for conveying instructions to both the game and the Module System. The Dialogue Package needs to be updated on each iteration of the system. Many decisions a module will make are dependent on the values in the Dialogue Package. The game data objects update all characters' states and elements like the previously mentioned location data. This update provides all of the character data needed by the module. Just like the Dialogue Package, each module requires an updated character state to make informed decisions.

2.3.7 Character Data

The Module Hub provides a place to store character attributes generated and altered by Kati and the game itself. The character attributes supported by the Module Hub by default are personal attributes, social attributes, and branch tone attributes. The personal attributes reflect the character's personality and way of thinking. For example, the attribute "selfish" would be a personal character attribute. Social attributes are directed attributes that a character feels towards another. Attributes like "enemy with" or "has a crush on" would be examples of social attributes where a character has a directed relationship status with another character. Both the personal and social attributes are not required and can be handled by the game entirely if so desired. The branch tone attributes are a type of social attribute that tracks eight relationship types crucial to choosing dialogue in every module. Unlike the social and personal attributes, branch tone attributes are required for the system to run correctly. Each of these three character elements will be discussed in greater detail later on in the Module section.

2.3.8 Dialogue Package

The Dialogue Package is paramount to the functionality of the Kati Module System. It holds signals from the game to the Module System, from the Module System to the game, and also signals that instruct different components of Kati. The Dialogue Package contains data that allow Kati to bypass rules, change modules, end conversations, force specific dialogue options to be picked or rejected, signal question, statement, and response type dialogue options, and much more. The Dialogue Package also provides the game with updates to the character's state and game state based on the module system's decisions. An example of a decision that can alter the game state is a response-type chosen by the player. The Dialogue Package will carry the consequences of the player's choice to the game. It is also responsible for supplying the dialogue selected by the module to the game. In short, the Dialogue

Package is the main form of communication between various components of the Kati Module System and the game that is using it.

The Dialogue package holds references that connect lines of dialogue phrases between iterations so that the system knows exactly where it left off. It does this by tracking the next module, module topic, module type like statement, a question or a response (See figure 2.12 for a list of the types). It also communicates whether a conversation is pieced together with a chain of dialogues. These chains require more than one iteration of Kati to complete and are useful for conveying long conversations about a specific topic. The Dialogue Package also contains all of the dialogue to be delivered to the game for each iteration of the system. This includes both the dialogue from the NPCs and the response data for the user to choose from. The Dialogue Package also tracks the Storyline node and current location for the current iteration of the system. It is a key element used by the History mechanisms to log current events.

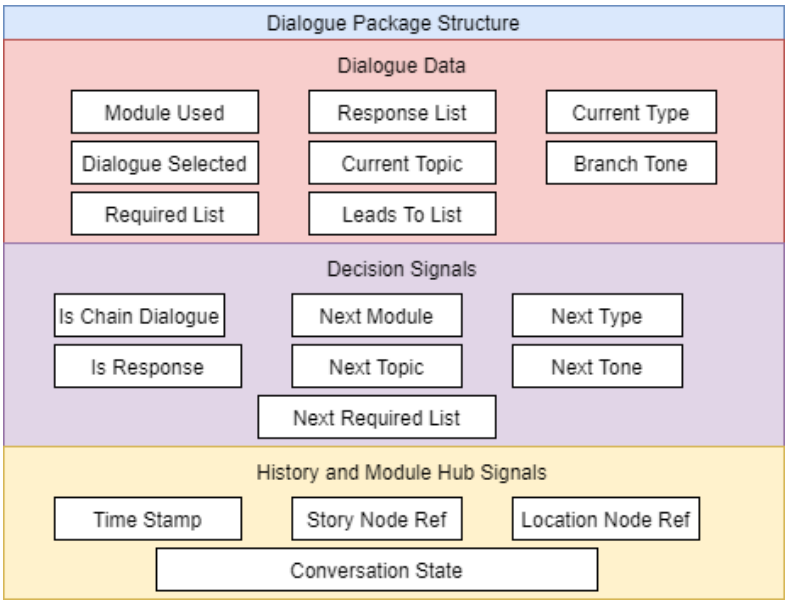


Figure 2.8: Dialogue Package contents categorized by Dialogue Data attributes, Decision Making Signals, and History and Module Hub Signals

2.3.9 History

There are two forms of history considered in Kati. The first is the long-term history. This type is a logging system that tracks all dialogue exchanges for each interaction throughout the game. The second type is the short-term history. The short-term history functions precisely like the long-term history except that elements contained in it are removed based on some duration defined by the game. The game can also decide on how this removal should take place. Since both the long-term and short-

term histories function like a queue, the default removal methods are dequeuing an element or emptying the entire short-term history bank. Figure 2.9 and 2.10 shows the history logging scheme relative to the player.

Both the long-term and short-term history can be used for selecting dialogue in a module. Possible rules such as “only play once” or “repeat dialogue three times only” can take advantage of the long-term history’s inputs. If an entry exists in the long-term history that matches the module selection dialogue in question and has the “play only once” rule, then this line is no longer available to be selected for the entirety of the game. Rules like “don’t repeat too frequently” can take advantage of the short-term history by checking to see if the dialogue is contained in the short-term history and if it is, then it is no longer a candidate for selection. Since all elements of the short-term history have a limited duration, dialogue repetition can be spaced.

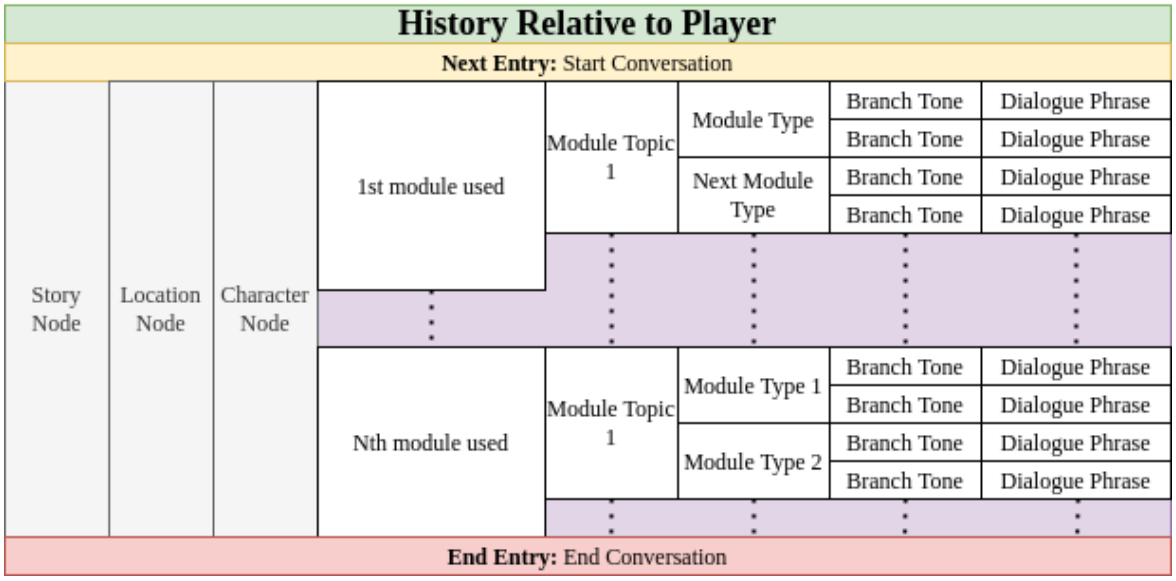


Figure 2.9: Overview structure of a History Entry

History Example using Dialogue Text							
Start Conversation							
Only Story Node	Town Center	Sorceress	Around Town	History Lesson	Statement	Friend	Did you know that there is a magic in the forest that breeds vicious plants and monstrous insects.
					Statement	Friend	The common folk around here won't tell you that. They don't like to talk about the specifics.
					Statement	Friend	People believe that speaking evil draws evil to you and no body wants their crops to murder them.
					Question	Neutral	What do you think?
		Player	Around Town	History Lesson	Response	Positive	I would agree. Messing around with evil won't do anyone any good.
		Sorceress	Around Town	History Lesson	Statement	Friend	There is some truth to that.
End Conversation							

Figure 2.10: Overview structure of a History Entry using example data

2.4 Module

We have seen the importance of the game and how it interacts with the Module Hub. We have also discussed the Module Hub and how it links locations and characters to major plot points, the next step is to discuss the power house of the system, the modules. A game can have any number of modules, with each housing a single unique broad topic for conversation. Each module topic is up to the discretion of the content author. Modules are chosen based on labels that correspond to some action that is happening in the game. Each module can contain many subtopics that are branches of the larger topic. Each of these subtopics can have several verbal tones, such as romantic, friendly, and hateful. Using the data provided by the game and module Hub, a module will choose a subtopic and a verbal tone to finally access possible dialogue phrases to be returned to the game. At this stage, a series of rules and protocols are followed to narrow down a “good fit” dialogue phrase that best matches the game’s current state. We will discuss the protocol to determine a “good fit” dialogue phrase in the context of the current game state later on.

2.4.1 Dialogue Topic Types

Subtopics in each module are categorized as either a statement, questions, or responses. Each of these categories provides different styles of conversational movement (Lessard, 2016). Statements are dialogue collections that contain regular statements, generally, if not exclusively, conveyed by an NPC to the PC. Question type dialogue collections pose questions to the player and almost always require a response. Response types are simply an authored list of phrases that can be used in response to a question.

2.4.2 Structure of a Module

The Module Hub holds a reference to the modules' collection, and the modules contain all components related to broad topics of discussion. The individual modules make up the core of the system and house most of the decision-making functionality for picking a single dialogue phrase to be returned to the game as a response to some action. Each module is based on a generic template that contains various components. In the following sections, we will review each of these components in turn.

2.4.3 The Controller

The Controller is the central hub of a module. It is responsible for communicating with every module element and delegates tasks to each in turn. It holds all information relating to the game data and character data. This data is passed in by the Module Hub and originates from the game itself. The Controller must decide on which subtopic to use from the library data and which topic type (See figures 2.11 and 2.12 for sample topics and type descriptions). The library data contains all of the authored content that pertains to the current module topic, and the topic type is type' statement', 'question,' or 'response.' The decision of the topic type combined with the sub-topic narrows down the applicable dialogue phrase set to be used.

The Controller uses the game's guidance, the Module Hub, and previous dialogue properties to decide on the sub-topic and topic type. The decision-making constraints resides in a container called the Dialogue Package. The Dialogue Package, as previously discussed, is a collection of instructions for various sections of Kati along with other elements. It contains direct instructions from the game, the Module Hub, and previous module data. The Controller checks the Dialogue Package's state looking for restrictions and weights to apply to the topic and type. The contents of the Dialogue Package are the primary vehicle for choosing the sub-topic and type. The Dialogue Package is used through the system. It contains any excluded subtopics present in the module. The Controller reads these exclusions as instructions on which topic to include as applicable for the current game state. The next consideration is the topic weights. By default, all topics have uniform weight, and each topic's probability of being chosen has a uniform distribution. The dialogue package can override this by assigning weights to individual topics, making them more likely to be selected as the final subtopic. After processing the Dialogue Package criteria, a topic is decided based on the currently prescribed probability.

With the topic chosen, the Controller ascertains the topic type. As previously mentioned, these types are either a ‘statement,’ a ‘question,’ or a ‘response.’ Unlike the default uniform distribution of the topics, the types have a strong bias towards ‘statements’ and zero percent default chance for the type to be a response. Just as with the topic decision, the dialogue package can override these weights and mandate a particular structure type must be used. This override is the only way for the response type to be triggered. With both the topic and type decided upon, the Controller calls upon the Parser.

Module: Fighting Words

Subtopics	types	Keys
Money Argument	Statement	money_argument_statement
	Question	money_argument_question
	Response	money_argument_response
Work Argument	Statement	work_argument_statement
	Question	work_argument_question
	Response	work_argument_response

Figure 2.11: Example list of data keys used to access dialogue options

Topic Types

Statement	General dialogue statement usually given by NPC
Question	General dialogue question usually given by NPC
Response	Response generally given by the player

Figure 2.12: List of topic types

2.5 The Parser

The Controller mainly handles the communication between module sections and the hub, with only a fraction of the final decision-making. In contrast, the Parser’s sole purpose is deciding a “good fit” dialogue segment to return to the game by parsing the authored content and rules. Much like the Controller, the Parser delegates tasks to subsections to narrow down the possibilities until only one dialogue phrase remains. The term “good fit” is most appropriate for the Parser’s method. It does not seek to find the best fit dialogue phrase but rather a “good fit.” The notion of a “best fit” dialogue phrase is subjective and is the case only when a single phrase is applicable for return. Finding a “good fit” dialogue phrase requires all dialogue phrases in the chosen subtopic and topic type to undergo a

series of checks. The first of these checks is to decide on the overall tonality of the dialogue. In music, the tonality of a piece is distinguished by the key in which it is played and the relationship between the notes and scales (Tonality, n.d.). Much like a song, Kati classifies the dialogue by its tone. Instead of a music key, the dialogue uses the characters' relationships to establish how a character will say something. These tones are relationship factors like characters' level of friendship or disgust with one another. They are known as the Branch Tones and are decided by the Branch Decision Node. The Parser calls on the Branch Decision Node to find the dialogue's proper tone based on the game's character data. We will look more into the Branch Decision protocol later on.

Module: Small Talk	
Subtopic Key:	greeting_statement
Branch Tone:	friend
Potential Dialogue	"Good morning."
Required Rule	game time equals morning

Figure 2.13: Data organization of an example module

Once a tone is established, then a small subset of dialogue phrases is available to sort through. Each dialogue phrase is associated with rule sets, allowing for the fine-tuning of when a specific dialogue phrase is applicable. These rules are decided during the initial authoring of the dialogue content and are optional. The Parser will call on a series of nodes to validate personal and social character rules, game rules, and dialogue weights applied to any given dialogue. These rules are listed in the library data and are established during the initial authoring of the dialogue content. We will see each of the decision rule nodes decide on a "good fit" dialogue phrase.

Once all decision nodes are called and a dialogue phrase is selected, it is packed into the Dialogue Package and returned to the Module Hub. The packaging of the dialogue includes the required rules and other data connected to each dialogue phrase. The Module Hub will submit it to the game, and the game will display it to the user interface.

2.5.1 Branch Decisions

Each conversation topic key (see figure 2.11 for topic key example) has subcategories nested within. These subcategories are dialogue tones. As previously mentioned, dialogue tones correspond to character relationship states such as a friendship and romance. These are referenced as the conversation tone branches. Each conversation tone branch, or branch for short, is an emotion type that contains different manners of speech for a single subject. The default tone branches are Neutral, Romance, Friendship, Professional, Respect, Affinity, Rivalry, Disgust, and Hate. They are roughly categorized into positive, neutral, and negative tones and are scalar. Each of these branches' house dialogue is modeled under the assumption that the characters speaking share this tonal relationship. If a character has no strong tonal feelings towards the other character they are engaged in conversation with, then the neutral tone is supplied.

The branch tone is decided by how the NPC and player view the other character in the conversation. The raw dialogue provided by the data library does not need to contain all tone branches for any given subject. Instead, each topic can have any number of tone branches. In some cases, individual branches may not make sense in the context. It is up to the dialogue author's discretion to decide what dialogue tones should exist in each subtopic in any given module and the bounds they encompass.

It is important to note that branch tone encompasses the longstanding core relationship view one character has for another. If a character has a higher "Romance" tone towards another character than a "Friend" tone, that character would be more inclined to say they love the other character than they are friends with the other character. These tones are built throughout gameplay and are a stable overview of how a character feels about another character. Nevertheless, they can be augmented by character traits and temporary statuses dictated by the game. Rulings for these must be established upon the creation of the game. An example of this temporary status might be a character with the status of "angry." Because of this angry status, they may get a temporary boost to the "Disgust" tone. If they are engaged in a conversation with their lover, the "Disgust" Branch may win out over any other branches, though their core view of the person they are conversing with does not reflect this outlook. After a duration, the status of "angry" is removed and the boost to "Disgust" Branch is removed. Much like many features in the Kati Module System, temporary status boosts are optional.

Not only does a branch tone not need to exist in the subtopic of a single Module, but it also does not need to exist in the game at all. The current version of the system allows for eight branch tones and

a neutral tone. The naming conventions used are currently permanent, but the authored content decides the mapping of dialogue to a tone. If the game requires a strong family bond between characters but does not allow for a romantic bond between characters, then the author can map strong family ties by the "romance" tone. Each tone in itself is just a means to map dialogue to a situation in the game. One exception to the freedom of branch exclusion is that each subtopic must contain the Neutral branch. It serves as a default branch when nothing else fits the conversation based on the current deciding conditions. The Neutral branch also captures the conversation between characters that are strangers or merely acquaintances.

The Branch Decision mechanism has three thresholds that mark the cut-off points for different ordering tiers. All attributes are compared to the highest threshold first, then the middle, and finally the lowest. After each threshold testing starting with the highest, each attribute is weighted based on where it sits on the priority list (how prominent it is in the characters' relationship). Each attribute is chosen one after another and added to a master list. This method allows higher priority tones to generally take precedence but not become predictable by always taking the lead. This same sequence happens for both the middle and lower thresholds. The master list will contain all attributes ordered from best fit to worse fit. Attributes that fall below the lowest threshold are subsequently dropped. Following the threshold comparisons, the master list adds the Neutral branch as the last element. This master list is the order of Branch Decisions in which the module will use to determine the final tone. These tone calculations are rendered on each iteration of the system as new dialogue is required.

On the chance that all attribute values fall below the lowest threshold, the master list will contain only the default tone of Neutral. When the master list contains only the Neutral Branch, it reveals that the initiator has no strong relationship with the responder. Each module must contain the Neutral branch for this very reason. If a branch tone does not exist in the data library, it is removed from the master list. Once the master list of ordered tones has been purged, it is returned to the Parser to move on to the next stage in decision making.

2.5.2 Required Rules

With the acceptance of the conversation tone branch, the Parser must narrow down a collection of dialogue phrases. Each of these segments has conditions and further instructions to pass to the game if chosen. The Parser must eliminate all dialogue segments with conditions that are not present in the

game's current state and participating characters. Dialogue required rules provide a means to force dialogue segments to only be available when individual states are achieved or are present. There are two major required rule types. Some rules deal with the game data, such as the game's time and setting, and other rules deal with character data such as characters' personalities and relationships with others.

The inclusion of required rules is solely up to the dialogue author. The benefits of required rule inclusion allow for more specific dialogue. Consider a greeting dialogue that falls under the neutral branch. A possible generic greeting might be "Hello." This greeting may not require any rules. This simple phrase is usable in any circumstance. Adding more complexity or specificity to the dialogue like, "Good morning." the need for a custom rule like, "time of day must be morning" may be required. Increasing the dialogue complexity further with the addition of "It sure is a lovely day." might require another rule, "weather is nice." These are three possible dialogue options that can exist in the data library. The author can use one or all of these greetings (See figure 2.14 for a mapping of these phrases to their potential rules). Each of these greetings provides a layer of conversation depth relative to rules that are valid for the current instance of the game. If the dialogue options include all three of these greetings and all of the rules are true, then the possible options for the greeting can be any of them, they all make sense in the current context. If the game state is morning and the weather is terrible, then the dialogue option that requires nice weather would be removed from the applicable phrases. It would not make sense to say the weather is pleasant during a tornado. The inclusion of dialogue requirements is a powerful way to allow for complicated and specific dialogue to exist in the game, but it comes at a cost. The more rules used, the need for a more extensive dialogue pool to cover a broader base of scenarios increases.

Greeting Dialogue Options	
Dialogue	"Hello"
Rules	<none>
Dialogue	"Good morning."
Rules	"time is morning"
Dialogue	"Good morning. It sure is a lovely day."
Rules	"time is morning" and "weather is nice"

Figure 2.14: Potential rules associated with different dialogue values

There are three main categories of rules that are considered in each module. There are game rules, personal character rules, and social character rules. Game rules deal primarily with the time,

setting, and events occurring in the game. Time rules include elements such as time of day, day of the week, month, and season. The setting rules deal with the location of the player in the game and elements such as the weather. The event rules deal with time events like when a store opens or if the annual carnival is in town and trigger events. Trigger events are game achievement events that are unlocked when the player reaches some goal.

Personal attributes are concerned with the traits that describe a character's personality. These traits include elements such as interests, physical features, and temporary statuses like injured or drunk. Personality attributes house traits that a character is born with and are immutable and traits that can be changed, acquired, or lost through interactions. The social rules deal with relationships that the initiator has with other characters. Much like the branch decisions, the social rule decision mechanism uses the initiator's perspective when applying social rules. The exclusive cultivation of character traits through interactions fuels the rule decisions. Both personal and social rule requirements contain Boolean and scalar type attributes.

The Parser calls a decision mechanism for each of the rule types, which remove dialogue phrases that contain rules that do not line up with the game's current state. With all rule restrictions enforced, the remaining dialogue segments can be chosen as the dialogue segment and returned to the Parser for the next step.

2.5.3 Case for No Available Dialogue

The Parser calls a number of mechanisms that ultimately return a single dialogue string to the game. There are instances where following the protocol for each of these decision mechanisms will lead to a dead end. These dead ends happen when the game and character conditions do not match any dialogue required rules, and Kati is rendered speechless. Multiple paths can be explored to combat this, looking for a viable dialogue phrase to return to the game. The Parser will first call the Branch Decision node, and the Branch Decision will return a list of all of the available branch tones ordered from most influential to least influential, where the least influential is the neutral tone. The Parser will try the most influential tone and check all rules and weights required for the game state and circumstances. If there is no dialogue phrase, the Parser will repeat the process with all of the tones until it reaches the least influential, neutral tone. The neutral tone is the default tone and is the only tone that is required. If

the neutral tone provides no results, nothing is returned to the game, and the conversation is terminated.

2.5.4 Leads To Rules

A requisite aspect of the module structure is deciding where to start and end a conversation. We have discussed the module system's method to determine what dialogue topic is chosen based on a specific set of criteria and the game's current state. Once a dialogue phrase has been selected, it is crucial to establish how this affects the game state and what dialogue comes next, if anything. The system has built-in mechanics to decide what to choose next, assuming that no one topic or type is pressing beyond its default weight such as a dialogue phrase that corresponds to an urgent rule like "the house is on fire." However, some dialogue phrases may lead the conversation in a specific direction. Each dialogue phrase contains a "required rules" list and a "Leads To" list.

The "Leads To" field supplies the module with information about which module, topic, and type to use for the system's next iteration. The "Leads To" elements override the default decision making mechanics and provide a way for the dialogue to take a more static structure and distribute a predefined dialogue chain. The "Leads To" list makes dialogue chains easier to produce by allowing Kati to point to the next dialogue phrase to be used directly, bypassing all decision-making elements.

This linking scheme is handy for creating interesting small chains of dialogue, allowing for greater complexity and specificity of complete thoughts. Using this technique in conjunction with tonality branches in a module can produce unique, complete thoughts that reflect how a character feels towards another character.

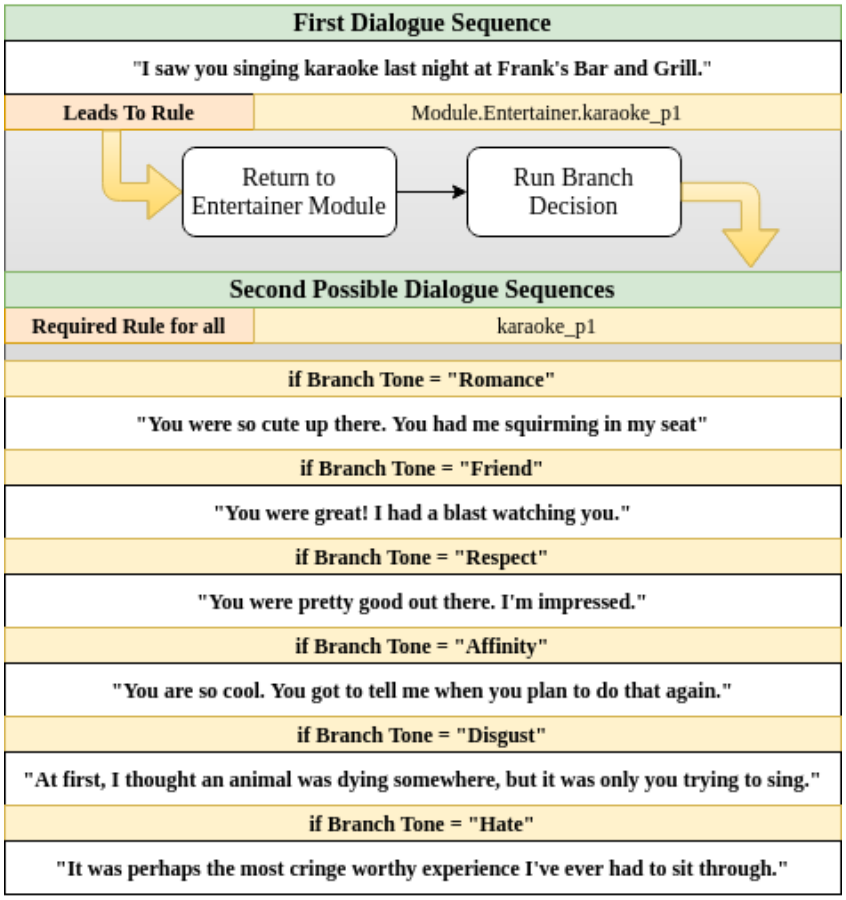


Figure 2.15: Micro dialogue chain sample using tonality as a guide to finishing the complete thought

Another example of directing a dialogue conversation using a “Lead To” element would be a dialogue type “Question” that might invoke a dialogue type “Response” from the player. The “Lead To” data would direct the module to run the next dialogue instance as a response from the current module. The response could be further narrowed by referring it to a limited number of possibilities to ensure the response makes sense with the asked question. Beyond navigating in the current module, the “Lead To” field could invoke elements from another module altogether.

2.5.5 Response Type Dialogue

The “lead to” option is a direct way to manipulate any conversation coherently. It is the only way to execute the third dialogue topic type, the response. The “Statement” and “Question” types are delivered by NPCs generally spoken to the player. A question will lead to a response, and each possible

response will have a requirement that connects it to the question. If the question has no particulars about the responses required, then any response can be used. In most cases, responses will have requirements connected to the question's "lead to" restrictions. Response requirements ensure responses make sense in the questions' contexts and prolongs the dialogue's believability.

Responses allow the player to interact with the game directly. Questions can take on many forms, thus allowing for significant variation in responses. An NPC might ask the player their opinion on something, ask if they will attend a function, or ask about the player's past experiences. The response can be as simple as a yes or no and as complex as a player choosing backstory. In the current phase of development of Kati, responses contain groups of possible replies with a range of a very positive outlook on the question to a very negative one. When a player is prompted to respond to an NPC, they are given a list of replies containing one positive value, one neutral value, one negative value, and one value that corresponds to the player's relationship with that NPC. The idea is to allow the player to forge their relationship with any given NPC but still gives an edge to the response type reflecting the current relationship with the PC.

2.5.6 Changing Character Attributes through "Leads To" Lists

Responses allow the player to manipulate the conversation state and directly alter the relationship between the player and the NPC speaking. Changing character relationships is twofold. It is the combined responsibility of the Kati Module System and the game. Kati is concerned with the Branch Decision tones and will use "Lead To" data to alter these scalar values. These tonality values are the only values that are directly tracked by Kati. Since the game provides the custom personal and social character elements defined in each game character, it is the game's responsibility to use the "Leads To" data to update these character attributes. Though the Kati Module System can easily use game defined character attributes, these are optional and are implemented at the game designer and the content author's discretion. The "Leads To" data is stored in the Dialogue Package and returned to the game. The game reads these entries and then updates them according to its protocol. Despite the Branch Tones being a child of the Module System, the game can access these values and build correlations between game derived character attributes and the Branch Tones.

2.6 Dialogue String Parser

A module could not function without the raw dialogue data. It contains topics with types, each containing several tone branches. The dialogue data itself exists in one of these tone branches and includes requirements for it to be chosen and “Lead To” elements dictating further actions needed. The module system translates all of this data to produce a stream of dialogue that keeps a reflection of the participating characters relationship.

One additional feature that the Kati Module System indirectly supports is key strings or tokens embedded in the dialogue itself. These grammar tokens are built-in instructions to cue the game to provide functionality or replace the token with a particular word or phrase. These cues can change the game’s user interface elements, such as changing a character’s portrait card in a textbox to match the dialogue, start a new textbox, or even trigger a game event. Figures 2.16 and 2.17 show an example of tokens embedded in a dialogue phrase and the game’s rendered outcome.

Kati’s Dialogue String Parser works similarly to Kate Compton’s Tracery (Compton et al., 2015). Tracery is an open-source tool used to write text-generating grammars using JSON files with the principle of being simple to author. The Dialogue String Parser and Tracery share many similarities. The main semantic difference between the two in function is that Kati does not include recursive symbol translation, and Kati’s formal grammar symbols target text augmentation and game elements. Though the Dialogue String Parser allows for signals to alter non-text elements such as portrait expressions, UI elements and other such character emotion indicators, it is simple in comparison to middleware toolkits like The Virtual Character Behavior Toolkit (Yang et al., 2017) or the SARA project (Arellano et al., 2018) providing emotions through rendered characters.

Since Kati Module System doesn’t assume any information about the game’s plot or style, these embedded tokens are the responsibility of the content authors and game designers. In short, the game must translate these tokens. Kati’s functional goal is to deliver appropriate dialogue for a given situation in a narrative. The game designers and content authors can use this system by introducing custom signals to the game that correlate to certain conditions. The use of embedded tokens is just a handy way to utilize custom signals for particular situations

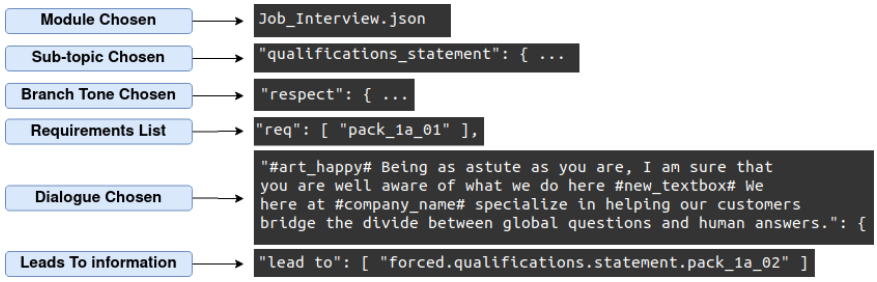


Figure 2.16: Authored content file structure derived from a sample game Job Interview

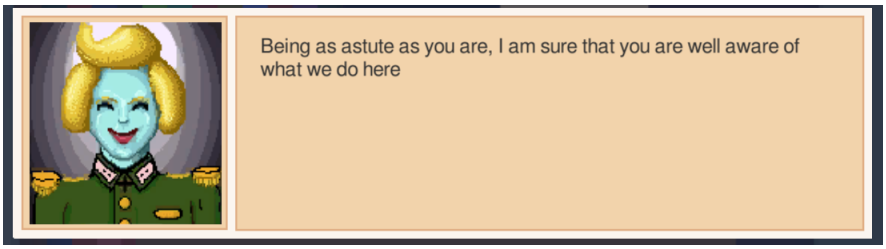


Figure 2.17: Dialogue choice from figure 2.16 rendered to the screen in the sample game Job Interview

3 Results and Discussion

3.1 Experiment Setup

The Kati Module System aims to deliver appropriate dialogue to an application that reflects its characters' relationship status. A simple proof of concept game was developed, "Job Interview," which tested the success of the generic module template and used a single module. Next, a text adventure game like integration test was constructed to tests the system as a whole. The "Job Interview" game's premise is that the player must gain employment as an intern at a software company. The gameplay entails the player engaging in a conversation with a hiring manager. The manager rattles on about the company and asks the player questions along the way. The player's responses directly affect how the manager perceives the player as a potential candidate for the job and provides dialogue mirroring the tone representing how the manager views the player as an intern candidate. The "Job Interview" game was developed while the Kati Module System's Module Hub was still under development. This fact led the "Job Interview" game to focus on the generic module template's success.

After completing the "Job Interview" game and a working version of the Module Hub was realized, an integration test resembling a text-based adventure game was created to test the addition of the Module Hub and the overall functionality of Kati as a whole unit. Both the "Job Interview" game and the integration test text adventure game monitored the relationship between the player and the

conversing NPC and delivered dialogue accordingly. Despite their similarities, the two tests are different, both in execution and the results' nature. This section will look at the experiment and results for the "Job Interview" game, then the text-game integration test experiment and results, and compare and contrast the two along the way.

3.2 The Job Interview Game

The “Job Interview” game utilized a single module built directly from the generic module template. It took minimal effort to create it and adopts many of the generic module template’s rule decision nodes without alteration. The small number of altered elements from the template shows promise that the design is flexible. There was very little development time spent refactoring module components to fit the game. The entire development time to create the game was roughly two weeks, with the time spent customizing the specific Job Interview module taking a single day of the entire production time. The game used the Unity game engine for the user interface and incorporated simple pixel graphics. It was launched with minimal dialogue to test the robustness, quality, and clarity of dialogue with the few options.

The ‘Job Interview’ game uses four primary relationship values to make all decisions. The first three are friendship, respect, and disgust. These three relationship values are the ones that determine the dialogue decisions made throughout gameplay. All three of these values are represented on the screen in the lower left as User Interface (UI) bars. This UI allows the player to see how they are doing at any given time. The fourth, professional, is a hidden attribute value that determines the outcome of the game. The professional relationship is not directly tied to any of the three relationship attributes that control the dialogue. However, it tends to be more prevalent with dialogue selections that contain higher positive effects, friendship, and respect, than with the negative value, disgust. All relationship augmentations are derived through the PC selecting a response. Each response has a predetermined value that correlates to the hiring manager that the player is interacting with.

The “Job Interview” test game’s goal was to fully implement a generic module with a specific direction in mind. The game’s important implementation qualities were to use the PC and NPC relationship to select dialogue phrases for the interview and incorporate game conditions outside of Kati using the same relationship values. For this test, the game’s win condition was solely based on the impression made by the PC on the NPC in the form to the “Professional” relationship attribute. The test

game was required to provide a continuously streaming conversation that was not only on topic with a job interview but also made logical sense and correctly ordered. The game needed to demonstrate that the system can produce long-lasting conversations that fit together chronologically and have a sense of believability in the context of the current game state. In this context, the term 'believability' refers to the player's perception that the current dialogue being presented is not far-fetched and could be present in an actual job interview.

Figures 3.1 through 3.4 show various screenshots of the "Job Interview" game. The main game screen consists of a textbox that holds the hiring manager's portrait along with dialogue and responses provided by Kati. Figures 3.1, 3.2, 3.3 and 3.4 show a section to a play-through that deals with how ethical the player is in the workplace. The first two, figures 3.1 and 3.2, shows a play-through where the relationship state is predominantly friendly. This fact is displayed by the UI bar located on the left side under the textbox. The first panel shows the player choosing a not-so-good response, and the next panel shows how the hiring manager responds. The response is light-hearted and cheery considering the player's claim of lack of understanding of ethics. This positive response was chosen by Kati because of the relationship state when the player responded. After the player responds, the evidence of the player's actions becomes more apparent by the shift in the relationship status displayed by the UI bar. The Friendship relationship, denoted by the heart and red bar, drops, and the 'disgust' relationship, denoted by the mean face and orange bar, increase. The next decision made about dialogue selection will be built off of this latest change. Figures 3.3 and 3.4 show the similar discussion about ethics, but the predominant relationship in this play-through is one of disgust. The dialogue options are different, though the topic, work ethics, is the same. The player once again chooses the worst choice, and the weight of their actions is seen on the disgust bar as it maxes out and the hiring manager's retort.

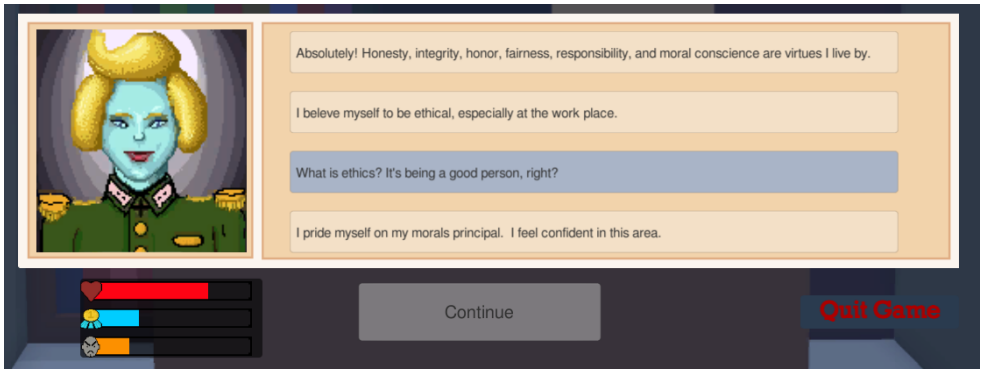


Figure 3.1: Player selects a negative response during a play through with a friendly relationship

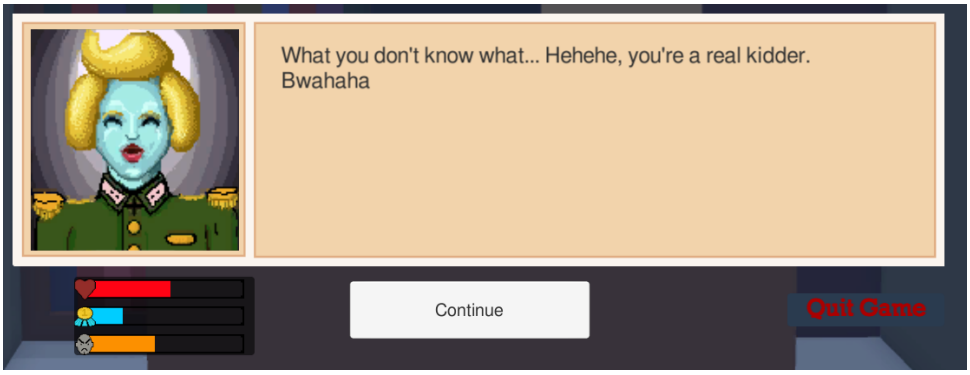


Figure 3.2: Hiring Manager reacts to the negative response in a friendly manner because of the established friendly relationship

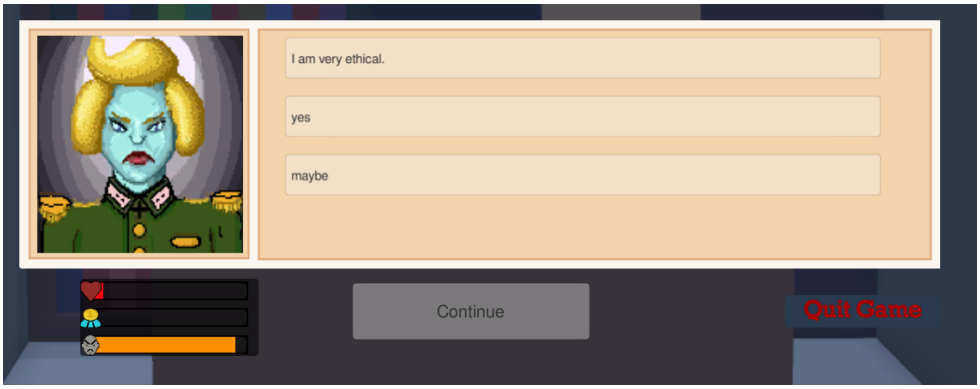


Figure 3.3: Player selects a negative response during a play through with a negative relationship

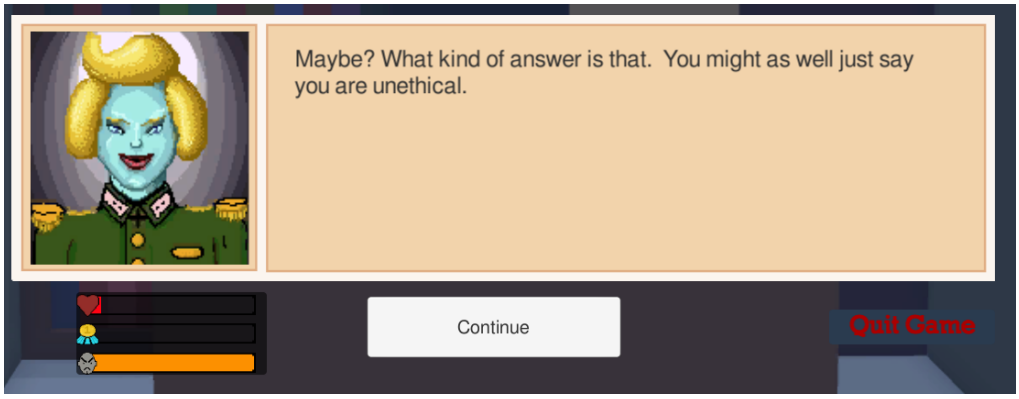


Figure 3.4: Hiring Manager reacts to the negative response in an unfriendly manner because of the established poor relationship

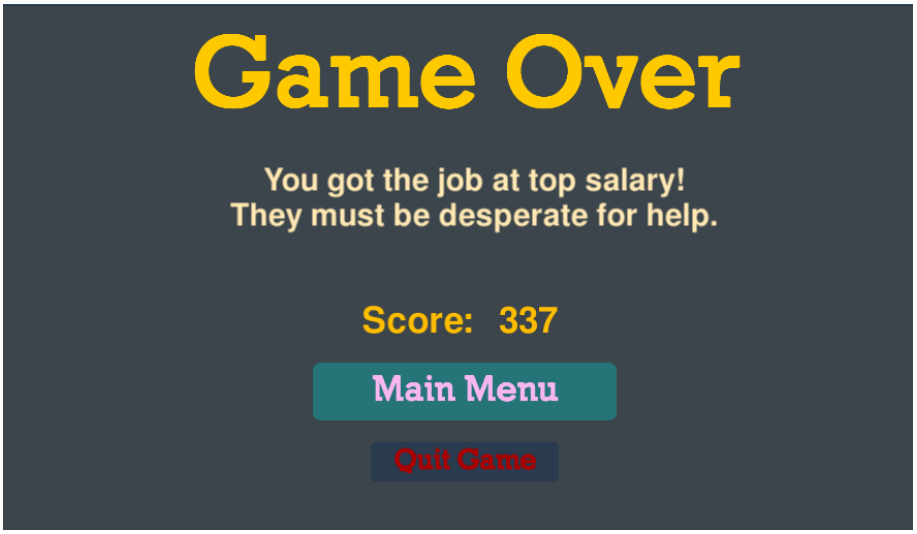


Figure 3.5: Game Over screen for the Job Interview game

To test the success of the Job Interview game, a group of introductory computer science students from the University of New Orleans was asked to test the game by playing it a few times then fill out a survey. The survey consisted of 10 questions about the following, did the game's dialogue make logical sense, did their choice affect the outcome of the game, did the dialogue differ each play-through, and was the dialogue presented believable in the context of the game. Unfortunately, only a small number of students participated, giving only 19 complete responses.

3.2.1 Job Interview Results

The overall responses given by the test game tended towards the positive. This fact gives credence that the Kati Module System functions as designed. The game scored the highest on the content making logical sense and the next to highest was the believability of the dialogue in the current context. The lowest score was on noticeable differences in the dialogue between play-throughs. (See Figure 3.6 for test results) This result is not unexpected and leads to aspects of the development that proved more difficult. The game used a minimal amount of dialogue and only used three-branch tonalities, respect, friendship, and disgust. Authoring dialogue and rule content used half of the total development time. This time allocation is a rather significant overhead for a minimal viable product. On the positive side, once the content was created, the system able to accurately select good fit dialogue that reflected the characters' relationship. The initial authoring cost is steep, but there is potential for

re-usability with repeated dialogue phrases pieced together by different characters using different branch tones under different game situations. Variation in the possible game and character states can create different player experiences, despite the recycling of dialogue bits.

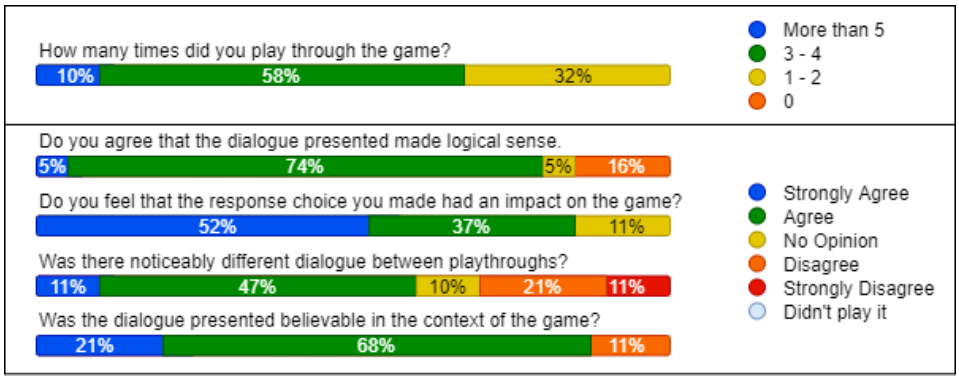


Figure 3.6: Survey response for Job Interview game

3.3 Text Adventure Integrated Test

The Job Interview game had promising results but did not test the system as a whole. With the completion of a functioning Module Hub, a new testing mechanism was needed to test the functionality of modules encapsulated in the Module Hub. A form of integrated testing that resembled a text adventure game was created to test the Module Hub and the relationship building through each interaction. The integrated test text adventure, or text game for short, is not a game in its own rights but has game elements that resemble a role-playing game's qualities. The text game consisted of three distinct locations, the Town of Biggs, the Gringor Forest, and the Mountains.

Twelve NPCs populated the environment, with eleven NPC sharing five modules and one NPC having a custom module much like the Job Interview game. Each of the eleven NPCs had two or more of the following modules as possible sources of conversation. The first and most universal is the 'Around Town' module. This module contained dialogue about town locations, people in town, history about the area, and greetings. The following two, 'Young Love' and 'Fighting Words,' contain relationship-driven phrases that pertained to dialogue that their name suggests. The next, ‘Forest Talk’, held dialogue designed for the dangerous forest that lay outside of town. The last module, ‘Questing’, was designated for dialogue involving quests and missions brought about during gameplay. Since the text game was not a full game, the ‘Questing’ module only contained dummy dialogue used for testing purposes. The twelfth NPC, the sorceress, contained a single large module that supplied all of the topics of the five modules that were shared by the other eleven NPCs but packaged into a single module. This module

alteration was done to experiment with alternate ways of producing similar effects but allowed for custom dialogue decision rulings for the eleven and a unique set for the one. The eleven had a community pool of dialogue, while the one had a custom tailored dialogue pool.

Besides the inclusion of multiple modules, the text game and the Job Interview game have many distinct differences. The Job Interview game contained a single extended conversation in which all of the topics of a single module were explored in a specific order. In contrast, the text game uses dozens of short dialogue phrases chosen based on the current game state and the relationship of the characters. The Job Interview game relied on the relationship to produce a narrative, but the conversation's overall direction was predetermined. Instead, the text game pooled a group of possible dialogue snippets based solely on the characters' relationship and other contributing factors like the weather, time of day, location, which NPC the player is speaking with, and the dialogue history. The text game produced more randomized conversations that are reminiscent of NPC characters of RPG-style games.

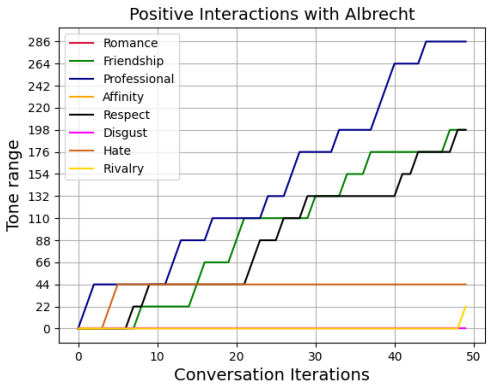
The text game integration test's goal was to test the system as a whole using the Module Hub to decide on the module and the modules to decide on the specific dialogue test using a variety of rule types. The text game used many of the prebuilt rule types (personal rules, social rules, branch attribute rules, game rules, and leads to rules) provided by Kati along with two custom rule sets, history rules, and location rules. Emphasis was placed on the system choosing a dialogue snippet that adheres to the game's rules and less on creating a single cohesive conversation. Just as important as following the rules is whether the module system's dialogue reflected the relationship established through interactions. Less emphasis was placed on NPCs' unique values due to the limited dialogue pool and the large variety of rules restricting the possible dialogue options.

3.3.1 The Eleven Townsfolk

The text game does not rely on “leads to” values to change the characters’ state relationships like the Job Interview game. Instead, the text game takes the stance that gameplay elements such as taking on quests or initiating conversations change the NPCs’ relationship states with the player. For ease of testing, a few simple overrides were incorporated into the game. These overrides forced all player interactions to take a friendly path, a mean path, or a romantic path. It is the equivalent to always picking the friendliest choice, the meanest choice, or the most romantic choice, depending on which override is enabled. The friendly path is labeled as being the ‘positive’ playstyle in that it directly affects

only the 'positive' relationship values of romance, friendship, professionalism, respect, and affinity. Likewise, the mean path is labeled as the 'negative' playstyle in that it directly affects the relationship values of disgust, hate, and rivalry. Lastly, the romantic playstyle targets the romance attribute if the character is allowed romance with the player. If not, then the playstyle defaults to the positive, friendly playstyle. These playstyles, positive, negative, and romantic, directly increase the relationship values they affect. The idea was to monitor the appropriateness of the dialogue outputs of the system compared to the current relationship state. The playstyles provided a controlling mechanism for specific relationship attributes alterations that should produce an expected result. This expected result being friendly dialogue, rude or harsh dialogue, and romantic dialogue, respectively.

The eleven NPCs that share the five communal modules are categorized into two main types. Each NPC could either have or not have a romantic relationship with the player. Those that could have romantic relationships also had access to the Young Love module and the tonal attributes' Romance' and 'Affinity', were utilized when interactions were invoked and barred access to the tonal attribute 'Respect.' Those that were not allowed romance with the player were denied the Young Love, the 'Romance' and 'Affinity' tone but allowed the 'Respect' tone. Figures 3.7 to 3.24 below follow four characters, Albrecht, Christina, Dan, and Teta, through the three playstyles overrides for fifty separate interactions. Albrecht and Dan are categorized as non-romantic, and Christina and Teta are categorized as romantic. Each character's tonal attributes are tracked, and sample dialogue snippets are captured for each of the overridden play styles. These figures demonstrate how the module system delivered dialogue that followed the established relationships by the game characters. This was done shuffling through all five modules and selecting the dialogue that was deemed appropriate for the situation based on the simple ruleset provided by the text game and JSON dialogue data.



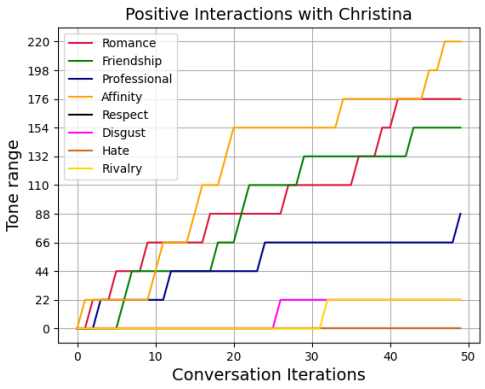
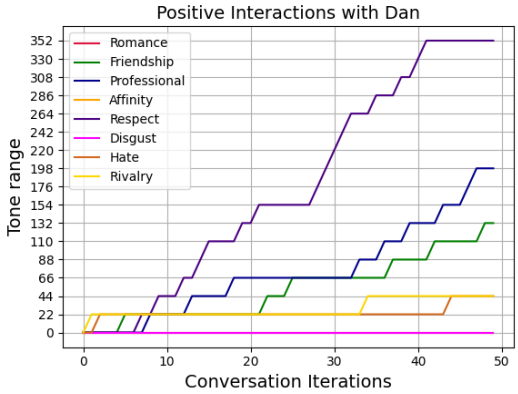


Figure 3.7 (Left): Relationship values with Albrecht using a positive playstyle

Figure 3.8 (Right): Relationship values with Christina using a positive playstyle



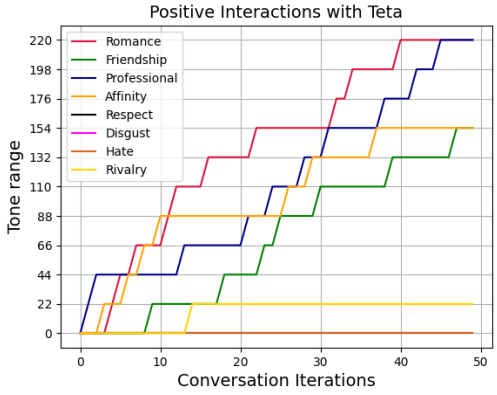


Figure 3.9 (Left): Relationship values with Dan using a positive playstyle

Figure 3.10 (Right): Relationship values with Teta using a positive playstyle

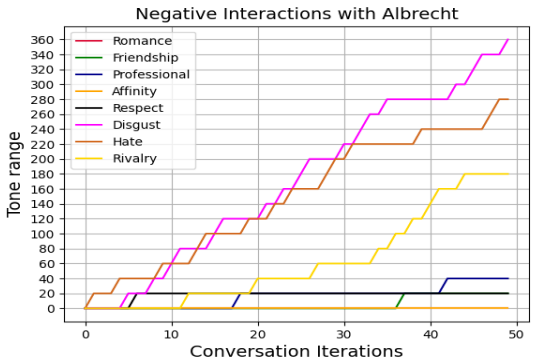
Final Scores after 50 Dialogue Iterations

	Romance	Friendship	Professional	Affinity	Respect	Disgust	Hate	Rivalry
Albrecht	0	198	286	0	198	0	44	22
Christina	176	154	88	220	0	22	0	22
Dan	0	132	198	0	352	0	44	44
Teta	220	154	220	154	0	0	0	22

Figure 3.11: List of all of the final relationships for Albrecht, Christina, Dan, and Teta from the positive play-through

Early Stage (iteration 5-12)	
Albrecht	What can I do for you?
Christina	Hello.
Dan	I like visiting the Town Center but it can give me a headache if I stay too long.
Teta	I love to hangout in the Town Center. It's where the action is.
Mid Stage (iteration 24-30)	
Albrecht	You know that there are faeries in the forest. They're prankster and often a bad omen.
Christina	Despite the danger that lurks around every corner, the forest is beutiful, don't you think?
Dan	Good day sir. It's always a pleasure.
Teta	The forest is scary but I think I could risk going with you.
Late Stage (iteration 42-48)	
Albrecht	Sometimes the going gets tough here, especially in the winter months, but I can tell you are strong. You'll make a fine addition to this town.
Christina	Are you coming to patrol the town border? We can always use an experienced adventurer like yourself
Dan	It's great to have another adventurer here. Every skill is useful. Perhaps you can pick up a new trade while you are here.
Teta	You know you're kind of cute when you make that face. You must be thinking about something.

Figure 3.12: Sample Dialogue for positive play-through



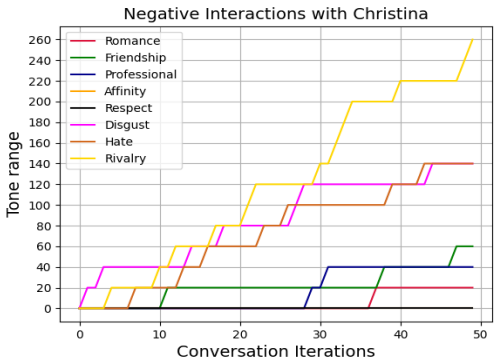
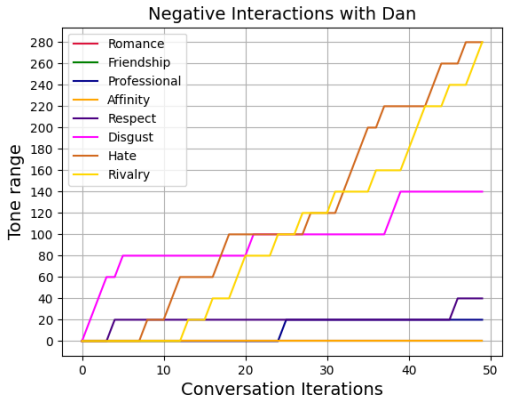


Figure 3.13 (Left): Relationship values with Albrecht using a negative playstyle

Figure 3.14 (Right): Relationship values with Christina using a negative playstyle



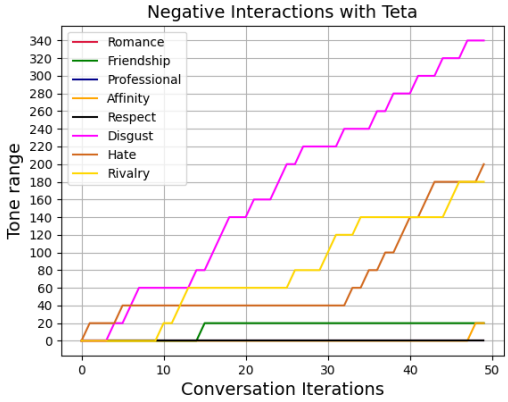


Figure 3.15 (Left): Relationship values with Dan using a negative playstyle

Figure 3.16 (Right): Relationship values with Teta using a negative playstyle

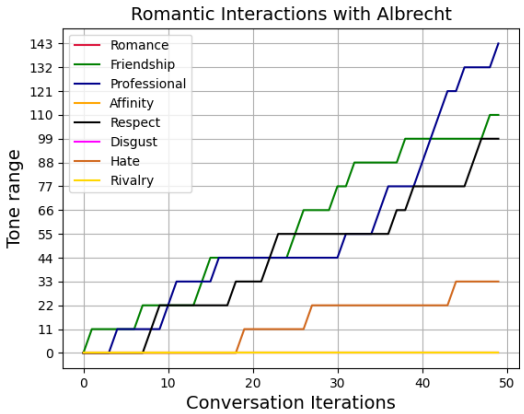
Final Scores after 50 Dialogue Iterations

	Romance	Friendship	Professional	Affinity	Respect	Disgust	Hate	Rivalry
Albrecht	0	20	40	0	20	340	280	180
Christina	5	20	60	40	0	140	140	260
Dan	0	0	20	0	40	140	280	280
Teta	0	20	0	20	0	340	200	180

Figure 3.17: List of all of the final relationships for Albrecht, Christina, Dan, and Teta from the negative play-through

Early Stage (iteration 5-12)	
Albrecht	The village of Biggs grows some of the best fruits and vegetables for miles around. It's almost like there is magic in the ground or something.
Christina	There is a lot of wild game in the forest, but you have to be a good shot.
Dan	Have you met Albrecht yet? He's the village's blacksmith.
Teta	I overheard someone talking about a cave in the mountains. I think they said something about crystals. I would love to see it some day.
Mid Stage (iteration 24-30)	
Albrecht	There are a lot of other people here to talk to.
Christina	It would be wise to stick to the trail when traveling through the forest.
Dan	Nice day isn't it?
Teta	You seem to be everywhere. Isn't it about time for you to leave this town.
Late Stage (iteration 42-48)	
Albrecht	The Forest can be a dangerous place if you're dense in the head. I would say it's very dangerous for you.
Christina	You're not impressing anyone by coming out here in the forest. You are just a fool that will end up dead.
Dan	I don't want to talk right now. I'm not in the mood.
Teta	Has anyone ever told you, you smell like a horse?

Figure 3.18: Sample Dialogue for negative play-through



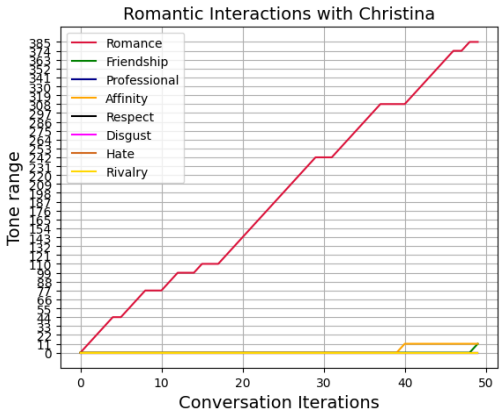
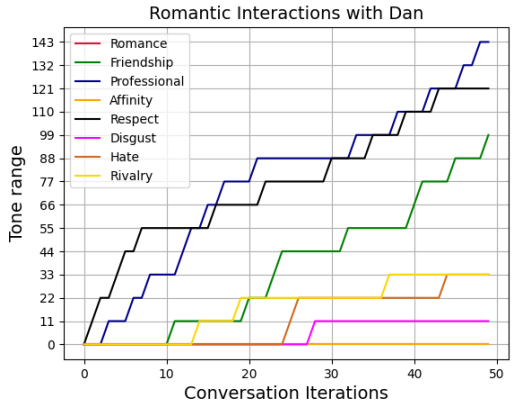


Figure 3.19 (Left): Relationship values with Albrecht using a romantic playstyle

Figure 3.20 (Right): Relationship values with Christina using a romantic playstyle



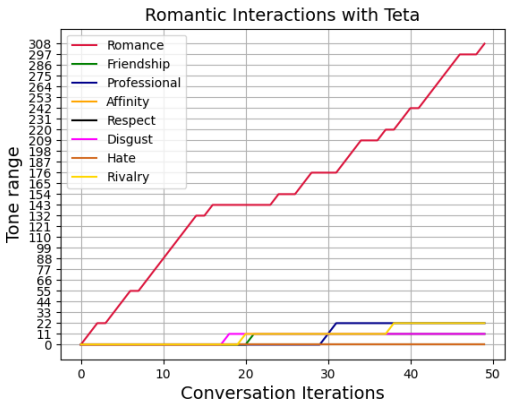


Figure 3.21 (Left): Relationship values with Dan using a romanitic playstyle

Figure 3.22 (Right): Relationship values with Teta using a romantic playstyle

Final Scores after 50 Dialogue Iterations

	Romance	Friendship	Professional	Affinity	Respect	Disgust	Hate	Rivalry
Albrecht	0	110	143	0	99	0	33	0
Christina	385	11	0	11	0	0	0	0
Dan	0	99	143	0	121	11	33	33
Teta	308	11	22	0	0	11	0	22

Figure 3.23: List of all of the final relationships for Albrecht, Christina, Dan, and Teta from the romantic play-through

Early Stage (iteration 5-12)	
Albrecht	Good day.
Christina	Did you know that there are mountains past the forest? They are loaded with coal and iron ore.
Dan	I'm Dan. I own the General Store here in Biggs.
Teta	Hello, My name is Teta. Nice to meet you.
Mid Stage (iteration 24-30)	
Albrecht	What do you think of the village?
Christina	topic: puppy_love romance test 1
Dan	We have a resident sorceress here in Biggs. Her name is Lerin. She's always so busy though. Not much time to chat.
Teta	Everything we need is close by. This is a nice place to live don't you think?
Late Stage (iteration 42-48)	
Albrecht	It's not to often we get visitors here at Biggs. It's nice to see a new face.
Christina	You know they say that if lovers stay out in the forest all night and watch the sunrise through the mountains, then they will never part. Maybe we should try it.
Dan	The Town Center is always busy. People come here to barter and socialize
Teta	I didn't know something was missing in my life until you came along. I'm so glad that we've met.

Figure 3.24: Sample Dialogue for romantic play-through

The eleven used a straightforward relationship-building scheme designed to see what dialogue would display and focused less on how each relationship was established. This simplistic relationship development protocol had only a few layers of dialogue output tones. Those outputs being positive, friendly tones, negative, mean tones, and romantic dialogue tone. Even with the further lack of conversational depth, the system shows promise. The module hub successfully followed the game-defined rules for module selections, and the dialogue, though limited, reflected the relationships of the characters.

3.3.2 Adding Rule Complexity

A series of simple rulings provided in the communal modules drove the eleven townsfolk, allowing only a few tiers of difference between the most positive dialogue and the most negative (friendliest to cruelest). Nevertheless, a difference in tone is evident and can be seen in figures 3.7 through 3.24. The conversations spun up by Kati followed the relationship patterns established throughout the 50 iterations of each playstyle undergone by the four test NPCs. The progression of each character's branch tone attributes is a product of simple procedures established by the game and falls back on whether a character can be romantic or not with the player. The joining of the module-defined ruleset and the game's simple attribute override protocol produce enough information for Kati to select dialogue phrases. Even though the simple ruleset and protocols produced results, the singled out NPC,

the sorceress, which contains one large module, was used to test the system with a more extensive ruleset. The sorceress has a richer set of authored rules provided with the dialogue options in the form of required rules and leads to rules. Also, the game's protocol on attribute augmentations based on gameplay increased in complexity compared to the communal eleven.

The use of richer rulesets and more complex attribute protocols were established to create a less predictable character without being random. In conjunction with these changes, the dialogue distribution based on branch tone is not uniform. This distribution scheme played with the idea that the sorceress would open up to the player in various degrees based on the relationship. The highest number of possible dialogue outputs belonged to the Romance branch, with the least dialogue options belonging to the Disgust and Hate branch. Figure 3.25 shows all of the possible dialogue outputs broken down by relationship tone and the module's topics.

Tones	Topics							Totals
	Greeting	Location	History	Townsfolk	Close	Quest	Magic Lesson	
Neutral	7	22	4	5	1	7	4	50
Romance	10	27	48	7	17	-	14	123
Friendship	6	15	8	-	1	-	6	36
Disgust	3	5	1	1	-	1	8	19
Hate	5	5	1	1	-	1	5	18
Totals	31	74	62	14	19	9	37	246

Figure 3.25: Possible Dialogue options for the sorceress broken down by attribute and topics of discussions

3.3.3 Changing the Override Playstyle

Since the sorceress contained a more complex protocol for changing relationship tones, a new form of override was created to simulate playstyles. Each interaction with the sorceress began with a prompt on how the player would approach the conversation. Figure 3.26 lists all nine approaches to any given interaction. The goal was to simulate gameplay styles quickly and efficiently while monitoring Kati and the dialogue produced. Five distinct play styles were tested using this scheme. The first used scheme or strategy relied on the default settings in the Module System and game. The second was based on a friendly approach using friendly, polite, and funny overrides (Figure 3.26. numbers 2, 3, and 4). The third was based on romance using romantic and confident approaches to each iteration (Figure 3.26 numbers 5 and 8). The fourth was based on a rude play style that incorporated being rude, mean, and uninterested in the conversation (Figure 3.26 numbers 6,7 and 9). The final playstyle was the strategic method. The sorceress's dialogue pool is most extensive for the romantic attribute and the

romantic attribute was designed to be the most challenging relationship to build as per the game's protocol. The strategic method used all conversation approaches at various points except for the mean and uninterested options (Figure 3.26 number 7 and 9).

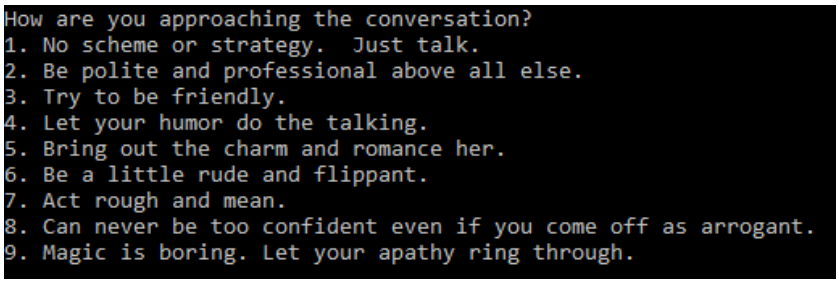
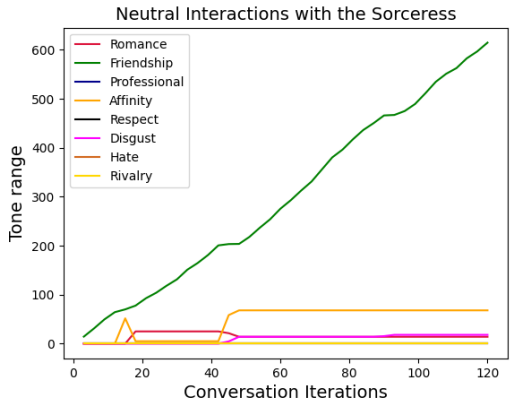


Figure 3.26: The nine possible approaches the player can interact with the sorceress

A record was kept when running each of the five playstyles through 120 conversation iterations. Each output was logged along with the change in the relationship between the player and the sorceress. It was important to see how the dialogue changed compared to the change in relationship, what topics were discussed as the relationship changed, and the overall mood of the conversations as the relationship changed. Figures 3.27 through 3.31 graphs the relationships between the player and the sorceress through the five playstyles, and figure 3.32 lists the total ending values of each relationship attribute. Figure 3.33 shows the favorite topic of discussion broken down by blocks of twenty iterations and the percent of the outputs from that topic. Accompanying Figure 3.33, Figure 3.34 shows the overall tone based on the number of positive leaning dialogue phrases used. The dialogue contained in the sorceress's module was roughly divided into three categories positive, negative and neutral. All neutral values were reserved for interactions with low relationship values and housed in the module's greetings topic. Positive and negative dialogue spanned across all topics and was linked to their appropriate attribute branch. Lastly, figures 3.35, 3.36, and 3.37 show dialogue outputs from the module's 'Magic Lesson' topic from the strategic play style, rude play style, and the friendly play style. These figures are a good representation of dialogue delivered by the system following the relationship state of the player and the sorceress. They are separated by the number of iterations undergone before the dialogue phrase was delivered by the system.



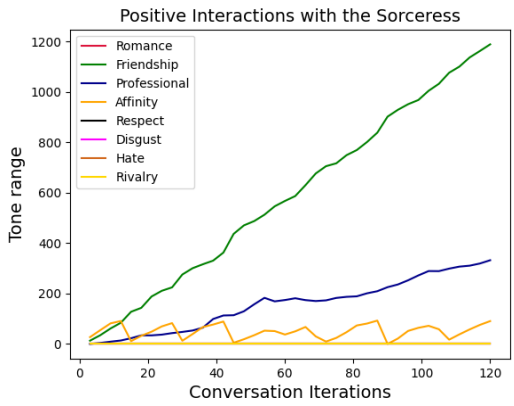
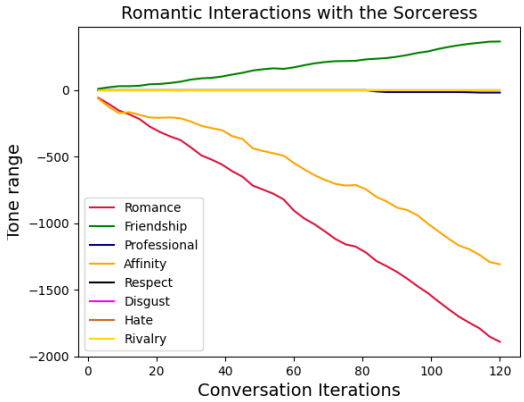


Figure 3.27 (Left): Relationship values from generic playstyle with the sorceress

Figure 3.28 (Right): Relationship values from positive playstyle with the sorceress



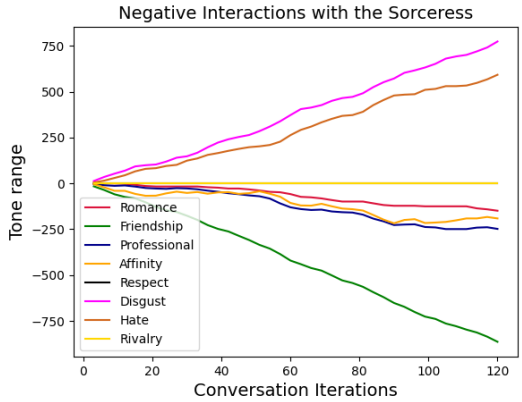


Figure 3.29 (Left): Relationship values from romantic playstyle with the sorceress

Figure 3.30 (Right): Relationship values from negative playstyle with the sorceress

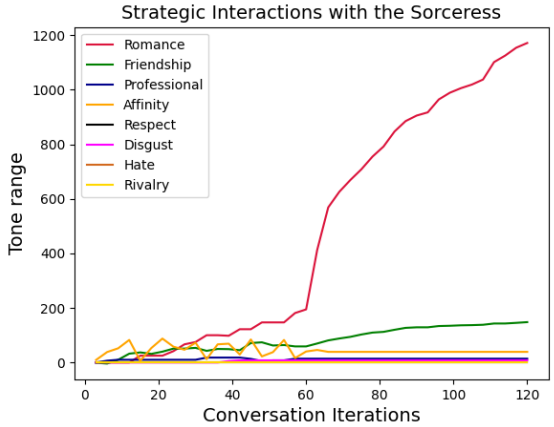


Figure 3.31: Relationship values from strategic playstyle with the sorceress

Final Scores after 120 Dialogue Iterations

Play Style	Romance	Friendship	Professional	Affinity	Respect	Disgust	Hate	Rivalry
None	14	623	0	68	0	18	0	0
Nice Guy	0	1193	340	96	0	0	0	0
Pushy Romantic	-1923	372	-18	-1341	0	0	0	0
Mean	-149	-883	-248	-191	0	796	592	0
Strategic	1186	148	14	39	0	8	0	0

Figure 3.32: Scalar value of relationship attributes after 120 interactions with each play style

Favorite Topic by Iteration Count

Play Style	1-20	21-40	41-60	61-80	81-100	101-120
None	Greeting (50%)	Location, History (40%)	History (45%)	History (65%)	History (50%)	History(55%)
Nice Guy	Greeting(50%)	Location (55%)	Townsfolk (40%)	History (40%)	History (35%)	Location (55%)
Pushy Romantic	Greeting (90%)	Greeting (45%)	Townsfolk (30%)	History (40%)	Greeting (30%)	Location (40%)
Mean	Greeting (100%)	Greeting (100%)	Greeting (95%)	Greeting (100%)	Greeting (100%)	Greeting (95%)
Strategic	Greeting (95%)	Quest (35%)	Quest, Close (25%)	Close (70%)	Close (75%)	Close (65%)

Figure 3.33: Most used topics of discussion based on number of interactions and playstyle

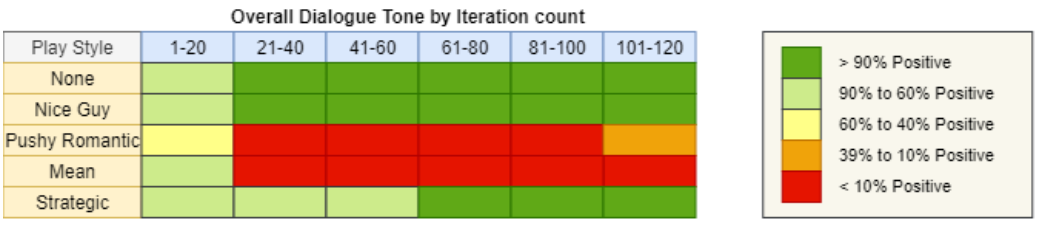


Figure 3.34: Overall tonal values of discussion based on number of interactions and playstyle

Magic Lessons Dialogue During Strategic Play Style		
Iteration Sequence	Dialogue	Branch
1-20	If you are looking for magic lessons you will need to travel to Oxenfel and join the Wizards Academy.	Neutral
21-40	Sorry, I will not be teaching magic techniques. You will need to look elsewhere if you are interested in learning the arcane.	Neutral
41-60	I don't know you very well yet but you seem nice, so despite my better judgement I'll teach you the basics. #unlock_magicLesson_01#	Romance
61-80	#player_name# you are growing on me. Let's resume your training and expand on the basics I taught you last time. #unlock_magicLesson_02#	Romance
61-80	I have a real soft spot for you #player_name#. #new_textbox# #art_blush# You give me butterflies in my stomach when you are around. #new_textbox# I was thinking that I would pass along more of my knowledge with you. #unlock_magicLesson_03#	Romance
81-100	#art_blush# You are quickly becoming one of my favorite people and I view you as more than a friend. #new_textbox# I want to elaborate on some of the concepts and practices we've discussed last time. #unlock_magicLesson_04#	Romance
81-100	#player_name#, I am starting to find it difficult to live without you so I want to draw you closer to me by teaching you some secrets I've discovered during my time as a sorceress. #new_textbox# I will reveal things that I have never told anyone before. #unlock_magicLesson_05#	Romance
101-120	You are so precious to me. Let me give you a piece of myself, a secret piece of knowledge from my craft. #unlock_magicLesson_06#	Romance
101-120	#player_name# I love you dearly. Of course I will teach you. #unlock_magicLesson_07#	Romance

Figure 3.35: Dialogue from the Magic Lesson topic from the strategic playstyle separated by interaction iterations

Magic Lessons Dialogue During Friendly Play Style		
Iteration Sequence	Dialogue	Branch
1-20	Unfortunately, I am not looking to take on any pupils and I doubt that I ever will. You may want to find a tutor somewhere else.	Neutral
21-40	You are fairly trust worthy. I guess I can teach you a thing or two. #unlock_magicLesson_01#	Friend
41-60	You have been a good friend to me and well, friends help each other out so I guess I can train you further. #unlock_magicLesson_02#	Friend
61-80	I don't usually give out my knowledge like this to everyone. Since you are one of the few here that I respect, I'll take your magic skill to the next level. #unlock_magicLesson_03#	Friend
81-100	#player_name# you've been a good loyal friend to me and what good is knowledge you it is not shared. Let me reveal a few arcane secrets to you. #unlock_magicLesson_04#	Friend
101-120	You've been a good friend to me but I have my limits and my secrets. If you are serious about learning the arcane trade, you should travel to Oxenfel.	Friend

Figure 3.36: Dialogue from the Magic Lesson topic from the friendly playstyle separated by interaction iterations

Magic Lessons Dialogue During Rude Play Style		
Iteration Sequence	Dialogue	Branch
1-20	If you are looking for magic lessons you will need to travel to Oxenfel and join the Wizards Academy.	Neutral
21-40	Sorry, I will not teach you my craft. Try taking up another, easier trade, like ditch digging or panhandling.	Disgust
41-60	If you're interested in learning magic forget it. #new_textbox# I don't think you're cut out for it.	Disgust
61-80	I will not teach you my craft so don't ask. #new_textbox# If you do not have an appointment with me, I have nothing more to say.	Hate
81-100	I've already told you that I won't teach you anything. Please go away!	Hate
101-120	You are trying my patience. I've told you I will not teach you anything, and I'd rather not talk to you for any reason if I can help it.	Disgust

Figure 3.37: Dialogue from the Magic Lesson topic from the Rude playstyle separated by interaction iterations

4 Reviewing the Results

The eleven townsfolk outputs and the sorceress produced positive results regarding delivered dialogue relative to the character's relationships. The text game focuses primarily on this point. The shortcomings of the text game's results stemmed from the sense of randomness that the conversation sets seemed to have. Each conversation was not connected to the previous conversation. This fact is understandable considering that the text game test was not created with any specific conversation organization schemes in mind but was targeting dialogue delivery based on a rule structure, character relationships, and the use of multiple modules through the Module Hub. This fact juxtaposed with the Job Interview game results offers passing test cases for dialogue delivery based on character relationship, a sense of cohesion and logical dialogue progression, and flexibility for custom rules. Overall, Kati seems to be successful.

4.1 Limitations

The Module System faces a few problems. One such issue is that the amount of dialogue required is non-trivial. For a large, rich pool of dialogues between several characters to have a hope of prolonging the player's enjoyment, there must be an equally large pool of dialogue possibilities to exist. The system does not minimize the amount of quality dialogue required for a game. Instead, it focuses on the how and the why of the dialogue presentation. Metadata must be considered for each dialogue string as well. This factor packs additional overhead into the dialogue authorship space. Each dialogue string can contain various dialogue requirements and the "leads to" effects of the dialogue. Though the initial cost might be steep, the possible variation of dialogue combinations produced by character relationships and game states can account for a large amount of dialogue reuse.

The system allows and encourages developers to override predefined functionality to suit a particular module. This customizability is a great feature that allows for a greater use-case with Kati; even so, it introduces problems. The developer must become familiar with the system and fully understand how each element needed to be altered functions in the default system. A robust developer documentation body is required to allow individuals ease of use. This documentation currently doesn't exist.

Other issues exist in the Kati Module System. The system is still under development and has not been thoroughly tested organically. The dialogue can have any number of inputs from any number of writers but is still a static entity. The system is designed with a particular game design in mind and is not a fit-all application. It is primarily focused on character relationships and providing dialogue to reflect these relationships. Kati may not be the right choice for games using a purely linear story or for games that require large amounts of user-defined inputs for dialogue sequences.

For the case of predictability, the Kati Module System relies on authored content. In certain circumstances, the repetition of dialogue will occur when variants do not exist in the composed content. The system uses a small amount of randomness when deciding on dialogue phrases. This is included as an attempt to disrupt predictability without being overly random. The term predictability has a twofold meaning in this context. Firstly, the inclusion of randomness will help prevent characters with a strong tonal relationship, such as "Romance," from only ever conversing in that tone. The second and more difficult to control is the repetition of the same dialogue over and over again. The disruption of this repetition falls on Kati's robustness and the quantity of authored content that any specific module has

to reference. Even with guards in place, the likelihood that dialogue repetition will manifest under certain conditions is inevitable.

Lastly, the quality of the dialogue delivered by Kati is only as good as the authored dialogue written and the quality of the defined rules established by the game and the content authors. Kati is not intend or able to replace the artistry that comes with creating breath-taking narratives or the control of engineered plots. The Kati Module System is tool that is meant for an engineered ruleset to accurately deliver artistically produced dialogue in a dynamic setting.

4.2 Conclusion

Kati's design creates a dialogue delivery system that reflects the game world's state and the participating characters' relationships. The goal is to produce dynamic dialogue structures created by gameplay and present them to fit the current game circumstances.

Besides the effects on gameplay through in-game dialogue, specific modules can be manufactured with minimal effort. Using the module template, creating a custom module on a range of topics can be as easy as writing the dialogue and as complex as rewriting and extending every decision mechanism. The Kati Module System design allows for the fine-tuning and re-defining schemas at the discretion of the game developers and content authors.

The Kati Module System was designed with a "Stardew Valley"(Barone, 2016) style game design in mind. This design may restrict the type and style of games that could benefit from the system. Notwithstanding, the results are positive, and the system functions are so far successful. The notion of a good game is cohesive, varied, has good user interaction, and offers some form of social interaction (Bond & Beale, 2009).

Kati attempts to provide cohesive dialogue through user interactions in pseudo-social settings. In regards to the initial trial run and testing of the system, Kati shows the potential as a valuable asset to authors and game developers in the assistance of crafting dynamic dialogue in games.

5 References

Allen, J. F., Byron, D. K., Dzikovska, M., Ferguson, G., Galescu, L., & Stent, A. (2001). Toward conversational human-computer interaction. *AI Magazine*.

Allwood, J. (1992). On dialogue cohesion. *Gothenburg Papers in Theoretical Linguistics 65*.

Arellano, D., Rauh, R., Krautheim, B., Spicker, M., Schaller, U. M., Helzle, V., & Deussen, O. (2018). Interactive testbed for research in autism—the SARA project. *Universal Access in the Information Society*. https://doi.org/10.1007/s10209-016-0521-9

Barone, E. (2016). *Stardew Valley*. Chucklefish.

Bond, M., & Beale, R. (2009). What makes a good game? Using reviews to inform design. *People and Computers XXIII Celebrating People and Technology - Proceedings of HCI 2009*. https://doi.org/10.14236/ewic/hci2009.52

Boyd, R. L., Blackburn, K. G., & Pennebaker, J. W. (2020). The narrative arc: Revealing core narrative structures through text analysis. *Science Advances*. https://doi.org/10.1126/sciadv.aba2196

Brusk, J., & Björk, S. (2009). Gameplay design patterns for game dialogues. *Breaking New Ground: Innovation in Games, Play, Practice and Theory - Proceedings of DiGRA 2009*.

Compton, K., Kybartas, B., & Mateas, M. (2015). Tracery: An Author-Focused generative text tool. *Lecture Notes in Computer Science (Including Subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*. https://doi.org/10.1007/978-3-319-27036-4_14

Cowley, B., Charles, D., Black, M., & Hickey, R. (2008). Toward an understanding of flow in video games. *Computers in Entertainment*. https://doi.org/10.1145/1371216.1371223

Harrell, D. F. (2006). Walking blues changes undersea: Imaginative narrative in interactive poetry generation with the GRIOT system. *AAAI Workshop - Technical Report*.

Koller, A., & Petrick, R. P. A. (2011). Experiences with planning for natural language generation. *Computational Intelligence*. https://doi.org/10.1111/j.1467-8640.2010.00370.x

Lessard, J. (2016). Designing Natural-Language Game Conversations. *Proceedings of 1st International Joint Conference of DiGRA and FDG*.

Mateas, M., & Stern, A. (2004). Natural language understanding in façade: Surface-text processing. *Lecture Notes in Computer Science (Including Subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*. https://doi.org/10.1007/978-3-540-27797-2_2

McCoy, J., Treanor, M., Samuel, B., Tearse, B., Mateas, M., & Wardrip-Fruin, N. (2010). Authoring Game-based Interactive Narrative using Social Games and Comme il Faut. *Proceedings of the 4th International Conference & Festival of the Electronic Literature Organization: Archive & Innovate*.

McCoy, Josh, Treanor, M., Samuel, B., Mateas, M., & Wardrip-Fruin, N. (2011). Prom Week: Social physics as gameplay. *Proceedings of the 6th International Conference on the Foundations of Digital Games, FDG 2011*. https://doi.org/10.1145/2159365.2159425

Mirvis, P. H. (1991). Flow: The Psychology of Optimal Experience. *Academy of Management Review*. https://doi.org/10.5465/amr.1991.4279513

Oudah, M., Rahwan, T., Crandall, T., & Crandall, J. W. (2018). How AI wins friends and influences people in repeated games with cheap talk. *32nd AAAI Conference on Artificial Intelligence, AAAI 2018*.

Riedl, M. O., & Young, R. M. (2005). An objective character believability evaluation procedure for multi-agent story generation systems. *Lecture Notes in Computer Science (Including Subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*. https://doi.org/10.1007/11550617_24

Sabah, G. (2011). Natural language understanding, where are we going? Where could we go? *Computer Journal*. https://doi.org/10.1093/comjnl/bxq060

Samuel, B., Reed, A. A., Maddaloni, P., Mateas, M., & Wardrip-Fruin, N. (2015). The Ensemble Engine: Next-Generation Social Physics. *Proceedings of the 10th International Conference on the Foundations of Digital Games (FDG 2015)*.

Tonality. (n.d.). Encyclopædia Britannica. available: https://www.britannica.com/art/tonality.

Weizenbaum, J. (1983). ELIZA—A Computer Program For the Study of Natural Language Communication Between Man And Machine. *Communications of the ACM*. https://doi.org/10.1145/357980.357991

Wright, W. (n.d.). *The Sims*. Electronic Arts.

Yang, F., Li, C., Palmberg, R., Van Der Heide, E., & Peters, C. (2017). Expressive virtual characters for social demonstration games. *2017 9th International Conference on Virtual Worlds and Games for Serious Applications, VS-Games 2017 - Proceedings*. https://doi.org/10.1109/VS-GAMES.2017.8056604

Vita

The author, Stephen Marcel, obtained his bachelor's degree in fine arts from Nicholls State University. He joined the University of New Orleans in Spring 2017, and joined the Computer Science Master's program in Fall-2018. Stephen has been working under Dr. Benjamin Samuel in his L.I.G.H.T., Laboratory, Intelligence, Games, HCI, Technique, lab with in the Department of Computer Science, at the University of New Orleans.