Application of Graph Theory and Combinatorics to Design Dialogue Systems in Video Games

Muhammad Ra'if Alkautsar - 13523011[1]
Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung, Jl. Ganesha 10 Bandung 40132, Indonesia
[1]mraifalkautsar@gmail.com, 13523011@std.stei.itb.ac.id

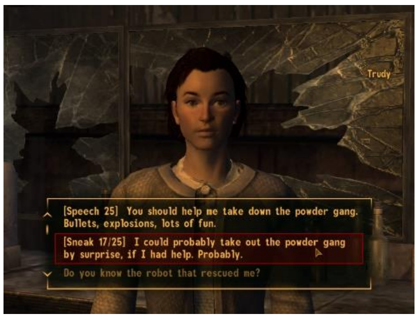
***Abstract*—In narrative-heavy games such as RPGs, dialogue is an integral part. A good dialogue system design is necessary to ensure that programmers, designers, and writers are able to work together effectively. This paper explores and analyzes the design of dialogue systems in videogames through a graph theory and combinatorics lens. Two idealized types are conceived to categorize dialogue systems: the unique paths and the pseudochoice. A combination of both and elements such as question loop and checks are necessary to ensure that the game can bring a good experience to the player while still in a programmer's and writer's capability.**

***Keywords*—dialogue, video game, story, graph, combinatorics**

I. Introduction

In a story-based game, dialogue can be a major component in conveying narrative elements. The advantage that video games have over linear story media such as films, books, and entertainment is the interactivity of the player. In various games that offer story interactivity as their main attraction, dialogue is often one of the main pillars.

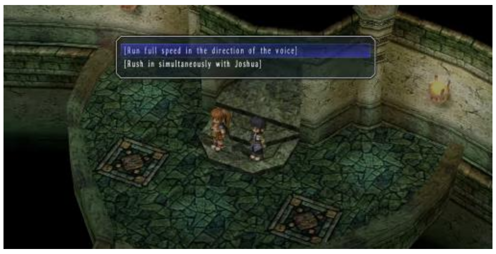
To make the player feel that they have the active ability to participate in directing the storyline, the player, playing a character, is able to make decisions in the narrative. Examples of this decision selection are deciding to do one of several actions when the narrative confronts the player with a situation or choosing what sentence the player wants to say to another character. Depending on the player's decision, in a game that is committed to its interactivity, there will be differences in the storyline between one decision and another.



[Speech 25] You should help me take down the powder gang. Bullets, explosions, lots of fun.
[Sneak 17/25] I could probably take out the powder gang by surprise, if I had help. Probably.
Do you know the robot that rescued me?

Fig. 1 Dialogue in the game Fallout New Vegas. The play can choose between several options to say.
(Source: https://problemmachine.wordpress.com/wp-content/uploads/2015/11/nvdialogue.png)

Currently, decisions in narrative have become a staple in roleplaying games (RPGs) and the likes such as visual novels. RPGs are a type of videogame which players assume the role of a specific character and play through a story, usually with a lot of autonomy on how they want to play their character. These games put a heavy emphasis on narrative—world, characters, lore, and such. Developers want to create an engaging world and environment for their players. To do that, they need to implement various ways for the players to interact with the world. This interaction is usually done with dialogue systems, which give the player control over their character's actions and allow them to talk to other non-player characters that exist in the world. Within the dialogues, there are often decisions—what the player wants to say through the character to the non-player character(s). The number of decisions offered by the game and how much influence those decisions have on the gameplay vary greatly from game to game.



[Run full speed in the direction of the voice]
[Rush in simultaneously with Joshua]

Fig. 2 Choosing between two decisions in the JRPG The Legend of Heroes Trails in The Sky. The decision the player choose doesn't affect much of the plot in the scenario.
(Source: https://www.youtube.com/watch?v=pAhWsfHS9uo)

For example, in most JRPGs (Japanese roleplaying games, a subgenre of RPGs, which can be a term for RPGs that are made in Japan or RPGs that are inspired by RPGs made in Japan), there are many small choices scattered throughout the narrative. However, these decisions do not have a big impact and are often just a taste.

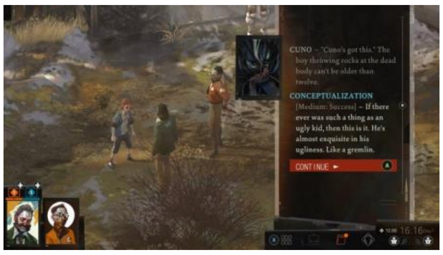
This is different in RPGs which emphasize player freedom. In CRPGs (computer role-playing games, a term that is now obsolete for reasons to be elaborated), a subgenre of RPGs that has root in adapting pen-and-paper roleplaying games such as Dungeon and Dragons, the authority of the player to decide what to do in the narrative is heavily emphasized. The emphasis comes from the fact that on pen-and-paper RPGs, the game is overseen by the Dungeon Master (DM) and the players are free to do whatever they want, creating a dynamic narrative environment.

Early CRPGs' main aim is to emulate this experience on the personal computer, hence the name computer role-playing game, to differentiate it from tabletop or pen-and-paper role-playing game. But now, with the vast amount of RPGs that are run on the computer, the term computer role-playing game more or less refers to old school RPGs of the 90s or modern games that take after their formulas.

In classic CRPGs such as Baldur's Gate or Fallout, on each encounter with characters that somewhat has a significance in the world (not mere bystanders or insignificant NPCs), the player is faced with an array of dialogue options and with the possibility that each choice might have an impact on the narrative. The player can also extensively customize their character, whether at the beginning of the game on the character creation phase, or as the game progresses via gained experience points that can be spent to level up the skills of the character, up to the player's liking, based on the kind of character that the player wants to roleplay, or based on the most optimal build.

This heavy commitment to the freedom of the player to roleplay their character however they want also leads to the refinement of the dialogue system. The purpose of dialogues in a narrative is to convey story arc, world-building, and characters. In order to ensure that the experience of the player stays solid in spite of the vast amount of narrative possibilities, designers and writers of many games have ideated and developed various systems. Most popular among the systems are the so-called "dialogue tree" or "conversation tree". When interacting with a non-player character, the player is given a choice of what to say and makes subsequent choices until the conversation ends. A lot of video game genres, such as visual novels and dating sims, are revolved around these interactions.

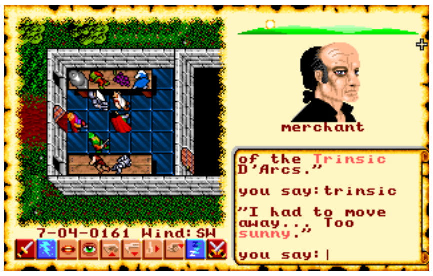
The main trade-off for a system that emulates realism (each dialogue choice will lead to a completely different branch of dialogues) is the amount of work that needs to be done by the writer(s) of the game. The amount of text that the game contains could increase exponentially on such systems. As such, most RPGs does not really have a dialogue tree, but rather a dialogue graph, since a tree does not have a cycle. In these dialogue graph systems, on a lot of conversations, the player cycles back to the same dialogue state he was in before choosing a dialogue. Only select dialogue choices are truly irreversible and adept RPG players have an intuition for these dialogue choices. This dialogue graph system ensures that the player can try all the choices and its result before progressing the story, exhausting all that the story has to offer. This system is very optimal for RPG players that have an affinity for complex, long, and intertwining lore. Most avid RPG players earn great satisfaction when learning all about a character, like fitting a piece of puzzle in the grand jigsaw of the fictional world.


CUNO – "Cuno's got this." The boy throwing rocks at the dead body can't be older than twelve.

CONCEPTUALIZATION [Medium: Success] – If there ever was such a thing as an ugly kid, then this is it. He's almost exquisite in his ugliness. Like a gremlin.

Fig. 4 Dialogue system in the game Disco Elysium. Narratively speaking, the protagonist is talking to his own "thought". The player can choose which specific "thought" to put skill points into that will affect how often the player will speak to this thought.

Many spins are often also included in an RPG's dialogue system. The skill check is a mechanic where to choose some dialogue options, the character needs to have sufficient skill depending on the option. As an example, for a dialogue option that has the character talking about science, the character needs to have its science skill stat above a certain threshold for it to succeed or have a higher chance of succeeding. Other mechanics that often appear are the party member chiming in (where a non-player character party member can interject in the middle of a conversation), inner monologues (where the character can have a conversation with its own inner thoughts, an integral part in the game Disco Elysium), and the relationship system, where the dialogues depend on the player's character relationship to the conversing character, a core part in games with dating elements such as Persona.)


of the Trinsic D'Arcs."

you say:trinsic

"I had to move away... Too sunny."

you say:|

Fig. 4
Dialogue from Ultima, where the player types in the word to say. (Source: https://gigi.nullneuron.net/gigilabs/wp-content/uploads/2017/11/ultima6_000.png)

The dialogue tree or graph wasn't always the only dialogue system in popular use. In old CRPGs like Ultima, specific words were highlighted and entering those words would provide additional conversation with the NPC. Dialogue trees, more or less, are replacing those single keywords with length lines of dialog. This deterministic system has allowed game designers to provide interactive conversations with nonplayer characters without the challenges of dealing with natural language processing or a more sophisticated artificial intelligence system. Writers have great freedom in presenting the narrative to the players.

Designing and planning the dialogue system is an integral part when developing a narrative heavy game, especially for large projects. For AAA games (a term for games with high production value or budget), the project needs to account for voice actors as well, which would accrue a cost in the production, especially if all of the characters are fully-voiced. The integration of the dialogue system to the games that have multiple genres (like action first-person shooter combined with heavy narrative elements) would also necessitate a ripe design.

This paper will analyze the design of dialogue systems from a discrete mathematics view, with heavy emphasis on the systems' relation with graph theory and combinatorics. It will also discuss the ways of planning and designing the system and its implementation with a simple program.

II. THEORETICAL BASIS

A. Graph

A graph can be used to represent discrete objects and its relationship with each other. The formal definiton of a graph G is G = (V,E), where:

- V is a finite non-empty set of vertices (synonomously known as nodes or points) = {v_1, v_2, … v_n}
- E is a finite set of edges that connects two vertices = {e_1, e_2, … e_n}

Based on the availability of loops, a graph can be:

a) Simple graph
A graph that does not contain parallel edges or loops.

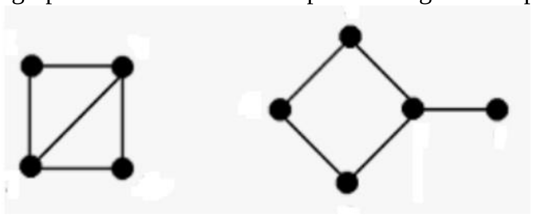


Fig. 5 Simple graph [1]

b) Not simple graph
A graph that contain parallel edges and/or loops A graph that contains parallel edges is called multi-graph. A graph that contains loops is called pseudo-graph.

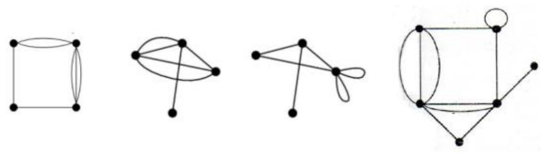


Fig. 6 Unsimple graph [1]

Based on the directional orientation of the edges, a graph can be:

a) Undirected graph
An undirected graph is a graph that does not have a direction orientation on its edges.

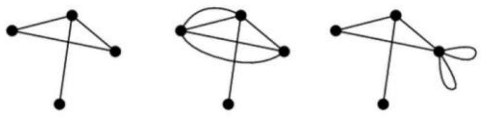


Fig. 7 Undirected graph [1]

b) Directed graph
A directed graph is a graph that each of its edges has a directional orientaition.

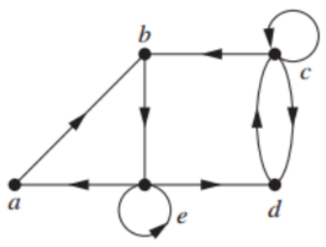


Fig. 8 Undirected graph [1]

Based on the availability of edges weights, a graph can be:

a) Weighted graph
A graph that has weight on each edges is called weighted graph.

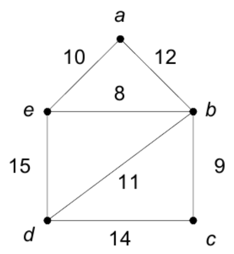


Fig. 9 Weighted graph [1]

b) Non-weighted graph
A graph that does not has weight on each edges is called weighted graph.

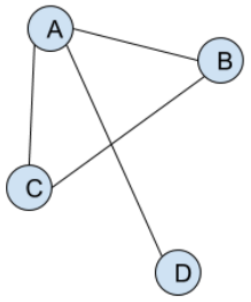


Fig. 10. Unweighted graph [1]

There are many terminologies that can be used to describe various things:

1. Adjacency
Adjacency refers to the relationship between two vertices (nodes) in a graph. Two vertices are adjacent if there is an edge directly connecting them. In an undirected graph, adjacency is mutual, meaning if u is adjacent to v, then v is also adjacent to u.
In a directed graph, adjacency depends on the direction of the edge. The vertex u is adjacent to v only if there is an edge from u to v.
2. Incidency
Incidency describes the relationship between an edge and the vertices it connects. An edge is incident on a vertex if the vertex is one of its endpoints. In a directed graph, an edge is incident to its destination vertex and incident from its source vertex.
3. Degree
The degree of a vertex is the number of edges connected to it. On an undirected graph, the degree of a vertex v denoted as deg(v) is the total number of edges incident to v. On a directed graph, the degree is separated into two: (i) In-degree ($\deg^-(v)$): The number of edges directed into the vertex v. (ii) Out-degree (deg+(v)): The number of edges directed out of the vertex v.
4. Path
A path is a sequence of edges that connects a series of vertices in a graph without any vertex repeated (in simple paths). The formal definition a path from vertex u to vertex v is a sequence (u, $v1$, $v2$, …, vk, v) such that each consecutive pair of vertices is connected by an edge.

The length of a path is the number of edges in the path.

5. Circuit
A circuit is a path that starts and ends at the same vertex, with no other vertex repeated. Simple Circuit: A closed loop where no vertices (other than the starting/ending vertex) and no edges are repeated. In a directed graph, the edges in the circuit must respect the direction of edges.
6. Connectedness
Connectedness refers to whether or not there is a path between any two vertices in the graph. There are three types of connectedness for a graph. (i) Connected graph: In an undirected graph, the graph is connected if there is a path between every pair of vertices. (ii) Strongly connected: In a directed graph, the graph is strongly connected if there is a directed path between every pair of vertices. (iii) Weakly connected: A directed graph is weakly connected if by replacing all directed edges with undirected edges will result in a connected graph.

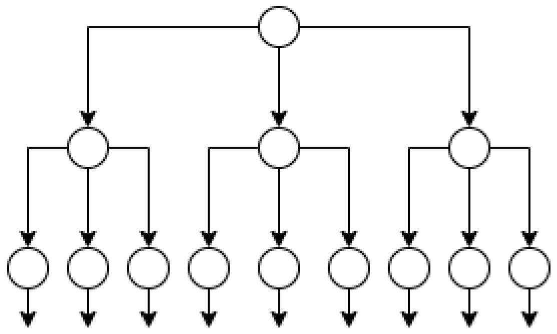
B. Combinatorics
Combinatorics is a branch of mathematics that studies counting, arrangement, and combination of objects in a systematic way. Several key concepts in combinatorics are applied in the analysis and experiment.
1. Counting Principles
   a. Addition Principles
   If there are n_1 ways to do one task and n_2 ways to do another (mutually exclusive), the total number of ways is
   $$n_1 + n_2$$
   b. If there are n_1 ways to do one task and n_2 ways to do another (independent tasks), the total number of ways is
   $$n_1 \times n_2$$
2. Permutations
   a. Number of permutations of n objects
   $$P(n) = n!$$
   b. Permutations of r objects from n objects
   $$P(n,r) = \frac{n!}{(n-r)!}$$
   c. Permutations of n objects with repetitions
   $$\frac{n!}{p_1!\,p_2!\dots p_3!}$$
   where $p_1, p_2, \dots, p_k$ are the frequencies of repeated objects.
3. Combinations
Combinations involve selecting objects where the order does not matter:
$$C(n,r) = \frac{n!}{r!\,(n-r)!}$$
4. Inclusion-Exclusion Principle
This following principle is used to count the number of elements in the union of overlapping sets:
$$|A \cup B| = |A| + |B| - |A \cap B|,$$
$$|A \cup B \cup C| = |A| + |B| + |C| - |A \cap B| - |B \cup C| - |C \cap A| + |A \cap B \cap C|$$
5. Pigeonhole Principle
If n items are placed into m containers, and $n > m$, at least one container must hold more than one item.

III. COMMON DIALOGUE TYPES

The goal of this research is to find ways to make the design of a dialogue system much clearer by using a mathematical approach, especially from graph theory and combinatorics. The first step of this analysis would be to categorize possible idealized dialogue types. The easiest structure and closest to reality is The Unique Paths structure.

A. The Unique Paths

In the Unique Paths structure, each choice would branch off into completely different prompts and different choices.


(Source: Author)

In this analysis, prompt is defined as the text segment that is presented to the player originating from the non-player character that the player talks to. Choice is defined as the text segment that can be said from the player's character to the non-player character.

For each prompt, there can be several c choices for the player. A c with the value of 1 would mean that the conversation is linear (the player does not have an autonomy to choose what to say).

From that, the sum of the possible prompts in a conversation in this structure can be derived from the following:
$$P(c,d) = 1 + c + c^2 + c^3 + \cdots + c^d$$
$$P(c,d) = \sum_{i=0}^{d} c^i$$
$$P(c,d) = \begin{cases} \dfrac{1 - c^{d+1}}{1 - c} & if\ c > 1 \\ d + 1, & if\ c = 1 \end{cases}$$

Where d is the depth of the conversation and c is the number of choices on each segment of the dialogue.

To find out the number of choices in a conversation, we can trace to the fact that each prompt would have c choices except for the last prompt. With that fact, we derive the following:

$$C(c,d) = P(c,d-1) \cdot c$$

Thus, the total text segments that must be written for the conversation:

$$A(c,d) = P(c,d) + C(c,d)$$
$$A(c,d) = P(c,d) + P(c,d-1) \cdot c$$
$$A(c,d) = \begin{cases} \dfrac{1+c-2c^{d+1}}{1-c} & if\ c \neq 1 \\ 2d+1, & if\ c = 1 \end{cases}$$

This system proves to be quite unfeasible because the number of text segments to be written for a conversation that have more than 1 choices increase exponentially with the increase of the number of choices. For example, with four (4) distinct choices for each prompt, with the conversation having the player choose five (5) times, this would be:

$$A(4,5) = \frac{1+4-2\cdot 4^{5+1}}{1-4} = \frac{1+4-2\cdot 4^{6}}{-3} = 2729$$

If we increase the depth of choice to seven times, this would be:

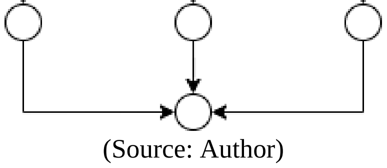
$$A(4,7) = \frac{1+4-2\cdot 4^{7+1}}{1-4} = \frac{1+4-2\cdot 4^{8}}{-3} = 43689$$

The drastic exponential increase would be unfeasible for the limited amount of the writer's capability. To deal with this, games usually implement the Pseudo Choice system, where the player is given a choice, but the result is the same regardless of the chosen.

B. The Pseudo Choice

To have a choice yet still get the same outcome might feel like a betrayal to the interactivity that is offered by the game. But most games do not have a uniform and easily categorized dialogue types. Instead, most games mix and match.

In the Pseudo Choice type, players are given several choices but all of them resolve to the same outcome. This dialogue type is an easy way to deal with the combinatorial explosion that happens in the Unique Paths structure.


(Source: Author)

In an idealized pure Pseudo Choice dialogue system, the number of possible prompts in a conversation would be the following:

$$P(c,d) = 1 + d$$

where d is the depth of the conversation or the number of choices that the player has to make. Then, the amount of text segments for all choices in a conversation would be as follow:

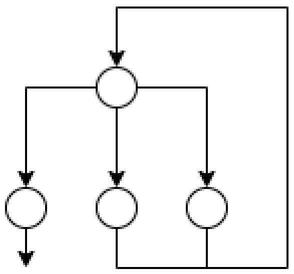
$$C(c,d) = cd$$

where c is the number of choices on each segment of the dialogue. Combining the two equations, the total number of text segments that must be written is the following:

$$A(c,d) = 1 + d + cd$$
$$A(c,d) = 1 + (1+d)c$$

C. The Question Loop

Often players want to know or need to know everything a character can say. Dialogue is used as exposition or as a means for the player to gain information. A more interactive approach involves allowing the player to gather information by repeatedly asking a character questions. This is typically presented with a prompt followed by multiple choices. Most choices trigger another prompt that eventually loops back to the original set of options, enabling the player to explore additional (or repeat the same) questions.

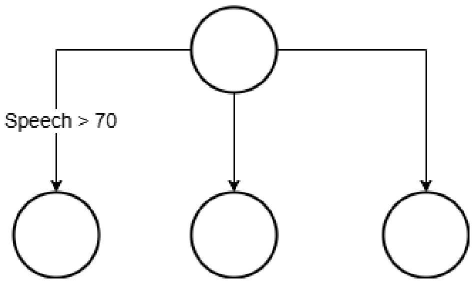

(Source: Author)

This mechanic is prominently used in detective games, as asking questions is a core aspect of being a detective. It allows for delivering exposition while maintaining the player's involvement in the story.

However, this design has a trade-off: questions can be retried indefinitely, producing the same responses each time. While this provides players the opportunity to reread information, it may feel unnatural and risk breaking immersion.

D. The Checks

Skill/speech check is a system where the player needs to have a specific statistic to be above a certain threshold in order to choose and go through a dialogue. When these checks are visible, certain dialogue options become accessible based on the player's character attributes, such as speech skill or reputation.



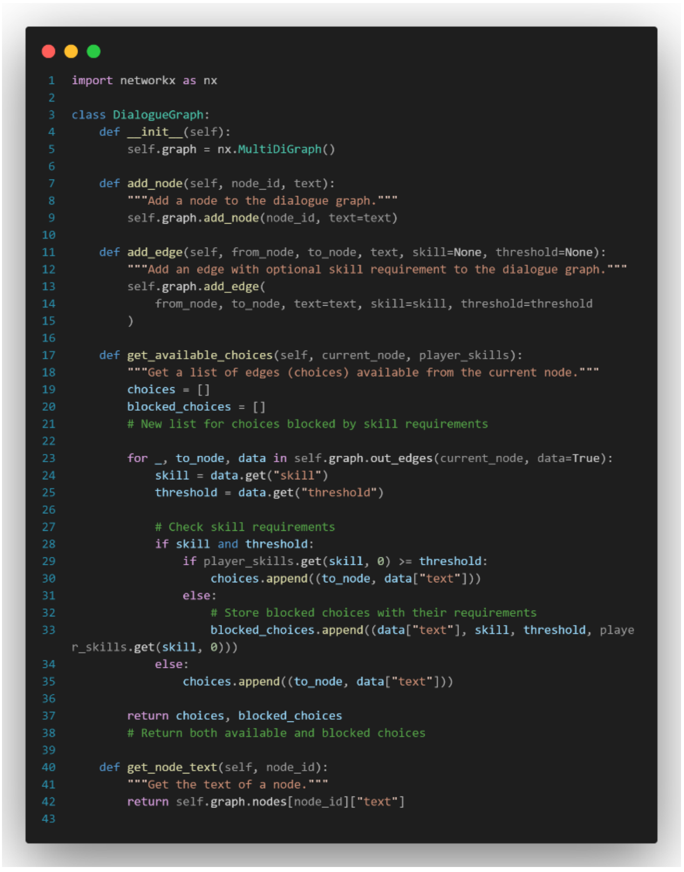
(Source: Author)

These checks can rely on absolute thresholds (e.g., Speech > 70, Reputation > 50) or incorporate random dice rolls alongside these values to determine success. Hidden checks, on the other hand, make dialogue feel more dynamic, less predictable, and often more engaging.

IV. MODELING A DIALOGUE SYSTEM WITH GRAPH STRUCTURE

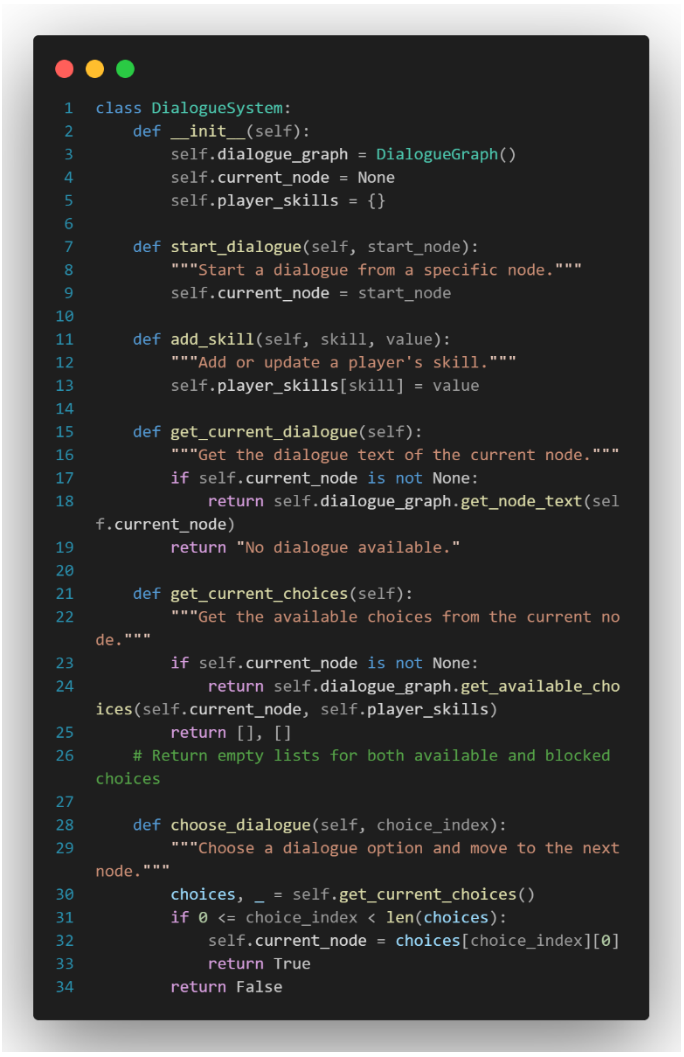
A dialogue system can be modeled using a graph structure. An edge would represent the player's choice, whereas the node would represent what the character the player's talking to has to say.

This implementation will use the MultiDiGraph (or Multi-Directed Graph) class from the networx library in Python as the underlying data structure for the dialogue system. It is a class in the networkx library that represents a graph with directed edges. A directed edge means there is a specific direction from one node to another, which is perfect for modeling dialogue systems where choices (edges) lead to specific outcomes (nodes).



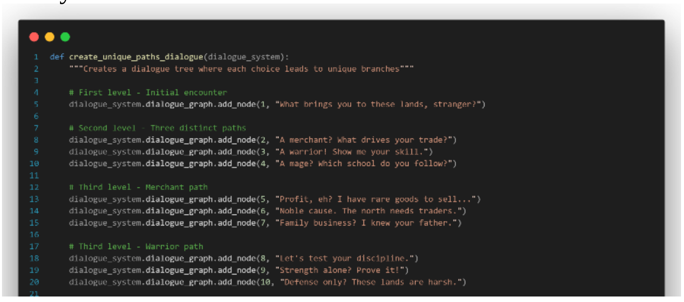
```python
import networkx as nx

class DialogueGraph:
    def __init__(self):
        self.graph = nx.MultiDiGraph()

    def add_node(self, node_id, text):
        """Add a node to the dialogue graph."""
        self.graph.add_node(node_id, text=text)

    def add_edge(self, from_node, to_node, text, skill=None, threshold=None):
        """Add an edge with optional skill requirement to the dialogue graph."""
        self.graph.add_edge(
            from_node, to_node, text=text, skill=skill, threshold=threshold
        )

    def get_available_choices(self, current_node, player_skills):
        """Get a list of edges (choices) available from the current node."""
        choices = []
        blocked_choices = []
        # New list for choices blocked by skill requirements

        for _, to_node, data in self.graph.out_edges(current_node, data=True):
            skill = data.get("skill")
            threshold = data.get("threshold")

            # Check skill requirements
            if skill and threshold:
                if player_skills.get(skill, 0) >= threshold:
                    choices.append((to_node, data["text"]))
                else:
                    # Store blocked choices with their requirements
                    blocked_choices.append((data["text"], skill, threshold, player_skills.get(skill, 0)))
            else:
                choices.append((to_node, data["text"]))

        return choices, blocked_choices
        # Return both available and blocked choices

    def get_node_text(self, node_id):
        """Get the text of a node."""
        return self.graph.nodes[node_id]["text"]

```

The code implements a dialogue graph that contains nodes and edges. Both contain a text property for the dialogue and the edge can contain a skill stat to be checked and its threshold. The graph requires a dialogue system class that encompasses it and will interface with the user.

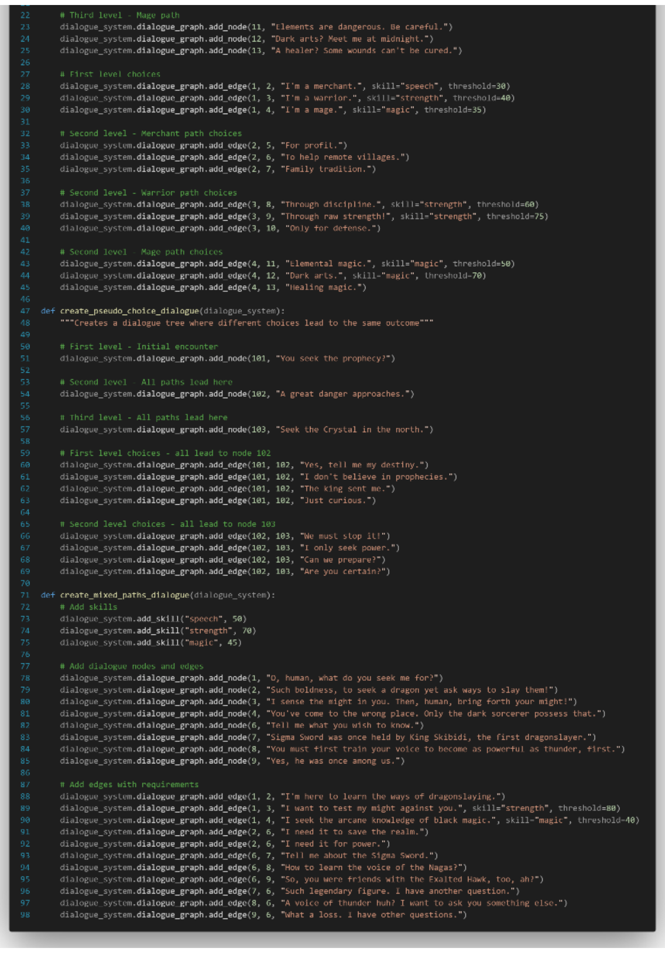


```python
class DialogueSystem:
    def __init__(self):
        self.dialogue_graph = DialogueGraph()
        self.current_node = None
        self.player_skills = {}

    def start_dialogue(self, start_node):
        """Start a dialogue from a specific node."""
        self.current_node = start_node

    def add_skill(self, skill, value):
        """Add or update a player's skill."""
        self.player_skills[skill] = value

    def get_current_dialogue(self):
        """Get the dialogue text of the current node."""
        if self.current_node is not None:
            return self.dialogue_graph.get_node_text(self.current_node)
        return "No dialogue available."

    def get_current_choices(self):
        """Get the available choices from the current node."""
        if self.current_node is not None:
            return self.dialogue_graph.get_available_choices(self.current_node, self.player_skills)
        return [], []
    # Return empty lists for both available and blocked choices

    def choose_dialogue(self, choice_index):
        """Choose a dialogue option and move to the next node."""
        choices, _ = self.get_current_choices()
        if 0 <= choice_index < len(choices):
            self.current_node = choices[choice_index][0]
            return True
        return False
```

The code implements a dialogue system that contains the dialogue graph as one of its attributes, alongside the current node and the player skill key-value pairs dictionary. From the dialogue system class, the driver or main can call methods to interact with the dialogue graph.
For this paper, three dialogue graphs are written to demonstrate the two dialogue types above and a mix of both with skill checks and cycle in it.

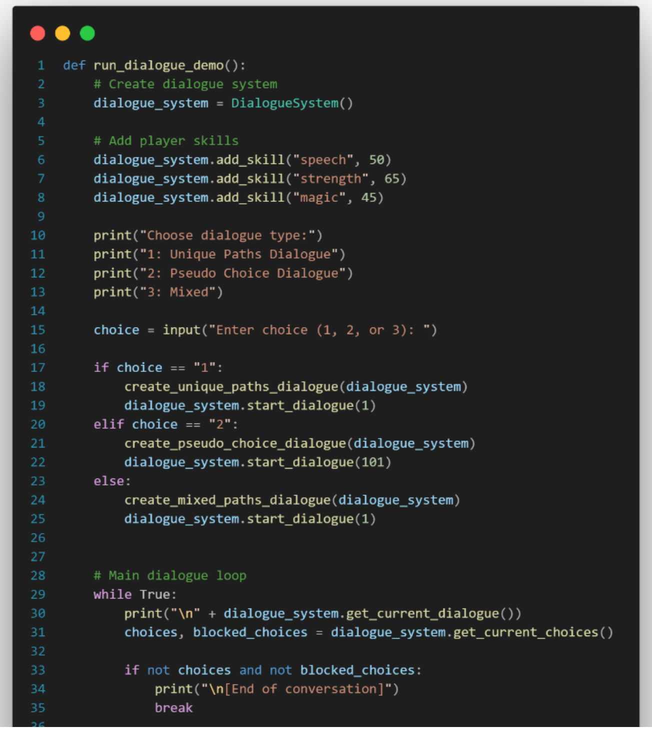


```python
def create_unique_paths_dialogue(dialogue_system):
    """Creates a dialogue tree where each choice leads to unique branches"""

    # First level - Initial encounter
    dialogue_system.dialogue_graph.add_node(1, "What brings you to these lands, stranger?")

    # Second level - Three distinct paths
    dialogue_system.dialogue_graph.add_node(2, "A merchant? What drives your trade?")
    dialogue_system.dialogue_graph.add_node(3, "A warrior! Show me your skill.")
    dialogue_system.dialogue_graph.add_node(4, "A mage? Which school do you follow?")

    # Third level - Merchant path
    dialogue_system.dialogue_graph.add_node(5, "Profit, eh? I have rare goods to sell...")
    dialogue_system.dialogue_graph.add_node(6, "Noble cause. The north needs traders.")
    dialogue_system.dialogue_graph.add_node(7, "Family business? I knew your father.")

    # Third level - Warrior path
    dialogue_system.dialogue_graph.add_node(8, "Let's test your discipline.")
    dialogue_system.dialogue_graph.add_node(9, "Strength alone? Prove it!")
    dialogue_system.dialogue_graph.add_node(10, "Defense only? These lands are harsh.")
```

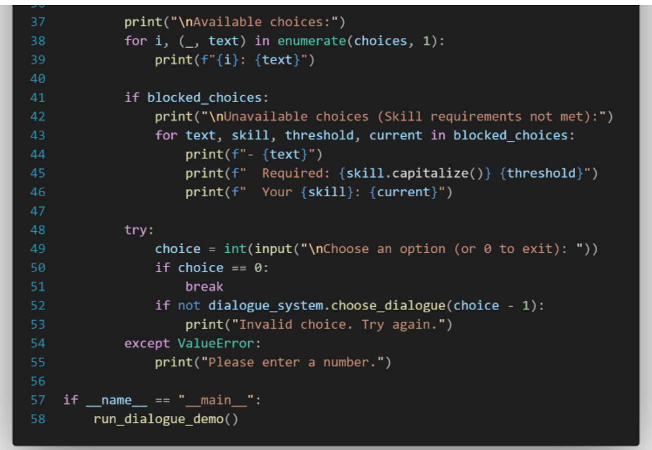


```python
    # Third level - Mage path
    dialogue_system.dialogue_graph.add_node(11, "Elements are dangerous. Be careful.")
    dialogue_system.dialogue_graph.add_node(12, "Dark arts? Meet me at midnight.")
    dialogue_system.dialogue_graph.add_node(13, "A healer? Some wounds can't be cured.")

    # First level choices
    dialogue_system.dialogue_graph.add_edge(1, 2, "I'm a merchant.", skill="speech", threshold=30)
    dialogue_system.dialogue_graph.add_edge(1, 3, "I'm a warrior.", skill="strength", threshold=40)
    dialogue_system.dialogue_graph.add_edge(1, 4, "I'm a mage.", skill="magic", threshold=35)

    # Second level - Merchant path choices
    dialogue_system.dialogue_graph.add_edge(2, 5, "For profit.")
    dialogue_system.dialogue_graph.add_edge(2, 6, "To help remote villages.")
    dialogue_system.dialogue_graph.add_edge(2, 7, "Family tradition.")

    # Second level - Warrior path choices
    dialogue_system.dialogue_graph.add_edge(3, 8, "Through discipline.", skill="strength", threshold=60)
    dialogue_system.dialogue_graph.add_edge(3, 9, "Through raw strength!", skill="strength", threshold=75)
    dialogue_system.dialogue_graph.add_edge(3, 10, "Only for defense.")

    # Second level - Mage path choices
    dialogue_system.dialogue_graph.add_edge(4, 11, "Elemental magic.", skill="magic", threshold=50)
    dialogue_system.dialogue_graph.add_edge(4, 12, "Dark arts.", skill="magic", threshold=70)
    dialogue_system.dialogue_graph.add_edge(4, 13, "Healing magic.")

def create_pseudo_choice_dialogue(dialogue_system):
    """Creates a dialogue tree where different choices lead to the same outcome"""

    # First level - Initial encounter
    dialogue_system.dialogue_graph.add_node(101, "You seek the prophecy?")

    # Second level - All paths lead here
    dialogue_system.dialogue_graph.add_node(102, "A great danger approaches.")

    # Third level - All paths lead here
    dialogue_system.dialogue_graph.add_node(103, "Seek the Crystal in the north.")

    # First level choices - all lead to node 102
    dialogue_system.dialogue_graph.add_edge(101, 102, "Yes, tell me my destiny.")
    dialogue_system.dialogue_graph.add_edge(101, 102, "I don't believe in prophecies.")
    dialogue_system.dialogue_graph.add_edge(101, 102, "The king sent me.")
    dialogue_system.dialogue_graph.add_edge(101, 102, "Just curious.")

    # Second level choices - all lead to node 103
    dialogue_system.dialogue_graph.add_edge(102, 103, "We must stop it!")
    dialogue_system.dialogue_graph.add_edge(102, 103, "I only seek power.")
    dialogue_system.dialogue_graph.add_edge(102, 103, "Can we prepare?")
    dialogue_system.dialogue_graph.add_edge(102, 103, "Are you certain?")

def create_mixed_paths_dialogue(dialogue_system):
    # Add skills
    dialogue_system.add_skill("speech", 50)
    dialogue_system.add_skill("strength", 70)
    dialogue_system.add_skill("magic", 45)

    # Add dialogue nodes and edges
    dialogue_system.dialogue_graph.add_node(1, "O, human, what do you seek me for?")
    dialogue_system.dialogue_graph.add_node(2, "Such boldness, to seek a dragon yet ask ways to slay them!")
    dialogue_system.dialogue_graph.add_node(3, "I sense the might in you. Then, human, bring forth your might!")
    dialogue_system.dialogue_graph.add_node(4, "You've come to the wrong place. Only the dark sorcerer possess that.")
    dialogue_system.dialogue_graph.add_node(6, "Tell me what you wish to know.")
    dialogue_system.dialogue_graph.add_node(7, "Sigma Sword was once held by King Skibidi, the first dragonslayer.")
    dialogue_system.dialogue_graph.add_node(8, "You must first train your voice to become as powerful as thunder, first.")
    dialogue_system.dialogue_graph.add_node(9, "Yes, he was once among us.")

    # Add edges with requirements
    dialogue_system.dialogue_graph.add_edge(1, 2, "I'm here to learn the ways of dragonslaying.")
    dialogue_system.dialogue_graph.add_edge(1, 3, "I want to test my might against you.", skill="strength", threshold=80)
    dialogue_system.dialogue_graph.add_edge(1, 4, "I seek the arcane knowledge of black magic.", skill="magic", threshold=40)
    dialogue_system.dialogue_graph.add_edge(2, 6, "I need it to save the realm.")
    dialogue_system.dialogue_graph.add_edge(2, 6, "I need it for power.")
    dialogue_system.dialogue_graph.add_edge(6, 7, "Tell me about the Sigma Sword.")
    dialogue_system.dialogue_graph.add_edge(6, 8, "How to learn the voice of the Nagas?")
    dialogue_system.dialogue_graph.add_edge(6, 9, "So, you were friends with the Exalted Hawk, too, ah?")
    dialogue_system.dialogue_graph.add_edge(7, 6, "Such legendary figure. I have another question.")
    dialogue_system.dialogue_graph.add_edge(8, 6, "A voice of thunder huh? I want to ask you something else.")
    dialogue_system.dialogue_graph.add_edge(9, 6, "What a loss. I have other questions.")
```

A driver code is written to execute the prototype code that has been written.

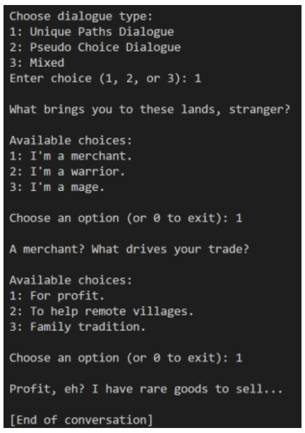


```python
def run_dialogue_demo():
    # Create dialogue system
    dialogue_system = DialogueSystem()

    # Add player skills
    dialogue_system.add_skill("speech", 50)
    dialogue_system.add_skill("strength", 65)
    dialogue_system.add_skill("magic", 45)

    print("Choose dialogue type:")
    print("1: Unique Paths Dialogue")
    print("2: Pseudo Choice Dialogue")
    print("3: Mixed")

    choice = input("Enter choice (1, 2, or 3): ")

    if choice == "1":
        create_unique_paths_dialogue(dialogue_system)
        dialogue_system.start_dialogue(1)
    elif choice == "2":
        create_pseudo_choice_dialogue(dialogue_system)
        dialogue_system.start_dialogue(101)
    else:
        create_mixed_paths_dialogue(dialogue_system)
        dialogue_system.start_dialogue(1)


    # Main dialogue loop
    while True:
        print("\n" + dialogue_system.get_current_dialogue())
        choices, blocked_choices = dialogue_system.get_current_choices()

        if not choices and not blocked_choices:
            print("\n[End of conversation]")
            break
```

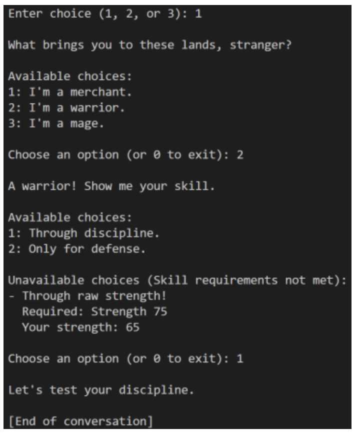


```python
        print("\nAvailable choices:")
        for i, (_, text) in enumerate(choices, 1):
            print(f"{i}: {text}")

        if blocked_choices:
            print("\nUnavailable choices (Skill requirements not met):")
            for text, skill, threshold, current in blocked_choices:
                print(f"- {text}")
                print(f"  Required: {skill.capitalize()} {threshold}")
                print(f"  Your {skill}: {current}")

        try:
            choice = int(input("\nChoose an option (or 0 to exit): "))
            if choice == 0:
                break
            if not dialogue_system.choose_dialogue(choice - 1):
                print("Invalid choice. Try again.")
        except ValueError:
            print("Please enter a number.")

if __name__ == "__main__":
    run_dialogue_demo()
```

Executing the code and choosing the unique paths dialogue type, the program will end after the player chooses two times. Each choice leads to different dialogue.

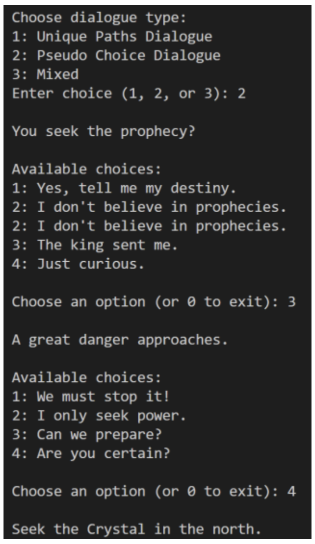


```
Choose dialogue type:
1: Unique Paths Dialogue
2: Pseudo Choice Dialogue
3: Mixed
Enter choice (1, 2, or 3): 1

What brings you to these lands, stranger?

Available choices:
1: I'm a merchant.
2: I'm a warrior.
3: I'm a mage.

Choose an option (or 0 to exit): 1

A merchant? What drives your trade?

Available choices:
1: For profit.
2: To help remote villages.
3: Family tradition.

Choose an option (or 0 to exit): 1

Profit, eh? I have rare goods to sell...

[End of conversation]
```

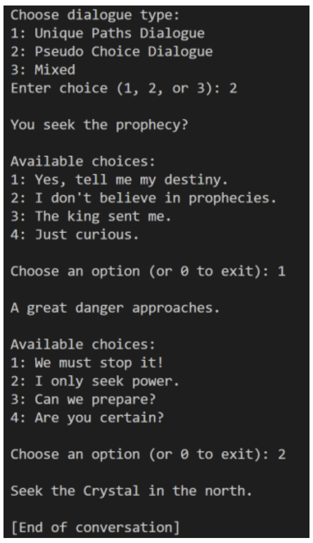


```
Enter choice (1, 2, or 3): 1

What brings you to these lands, stranger?

Available choices:
1: I'm a merchant.
2: I'm a warrior.
3: I'm a mage.

Choose an option (or 0 to exit): 2

A warrior! Show me your skill.

Available choices:
1: Through discipline.
2: Only for defense.

Unavailable choices (Skill requirements not met):
- Through raw strength!
  Required: Strength 75
  Your strength: 65

Choose an option (or 0 to exit): 1

Let's test your discipline.

[End of conversation]
```

If we choose the pseudo choice dialogue type, each choice will lead to the same response from the program.

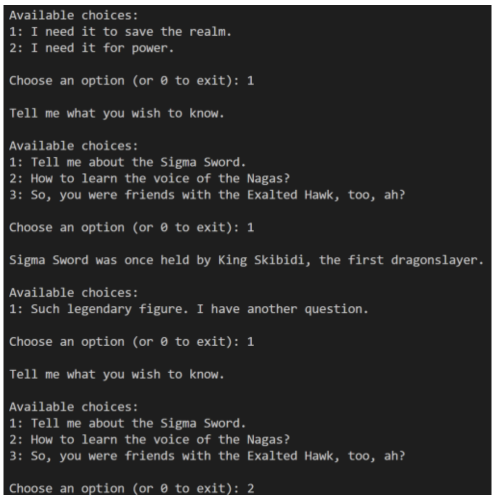


```
Choose dialogue type:
1: Unique Paths Dialogue
2: Pseudo Choice Dialogue
3: Mixed
Enter choice (1, 2, or 3): 2

You seek the prophecy?

Available choices:
1: Yes, tell me my destiny.
2: I don't believe in prophecies.
2: I don't believe in prophecies.
3: The king sent me.
4: Just curious.

Choose an option (or 0 to exit): 3

A great danger approaches.

Available choices:
1: We must stop it!
2: I only seek power.
3: Can we prepare?
4: Are you certain?

Choose an option (or 0 to exit): 4

Seek the Crystal in the north.
```

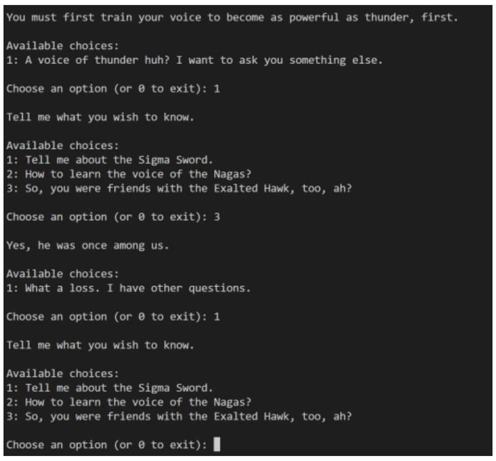


```
Choose dialogue type:
1: Unique Paths Dialogue
2: Pseudo Choice Dialogue
3: Mixed
Enter choice (1, 2, or 3): 2

You seek the prophecy?

Available choices:
1: Yes, tell me my destiny.
2: I don't believe in prophecies.
3: The king sent me.
4: Just curious.

Choose an option (or 0 to exit): 1

A great danger approaches.

Available choices:
1: We must stop it!
2: I only seek power.
3: Can we prepare?
4: Are you certain?

Choose an option (or 0 to exit): 2

Seek the Crystal in the north.

[End of conversation]
```

In the third option, that is the mixed type, there's a section where the dialogue loops and the player can check the response of each dialogue choice.



```
Available choices:
1: I need it to save the realm.
2: I need it for power.

Choose an option (or 0 to exit): 1

Tell me what you wish to know.

Available choices:
1: Tell me about the Sigma Sword.
2: How to learn the voice of the Nagas?
3: So, you were friends with the Exalted Hawk, too, ah?

Choose an option (or 0 to exit): 1

Sigma Sword was once held by King Skibidi, the first dragonslayer.

Available choices:
1: Such legendary figure. I have another question.

Choose an option (or 0 to exit): 1

Tell me what you wish to know.

Available choices:
1: Tell me about the Sigma Sword.
2: How to learn the voice of the Nagas?
3: So, you were friends with the Exalted Hawk, too, ah?

Choose an option (or 0 to exit): 2
```



```
You must first train your voice to become as powerful as thunder, first.

Available choices:
1: A voice of thunder huh? I want to ask you something else.

Choose an option (or 0 to exit): 1

Tell me what you wish to know.

Available choices:
1: Tell me about the Sigma Sword.
2: How to learn the voice of the Nagas?
3: So, you were friends with the Exalted Hawk, too, ah?

Choose an option (or 0 to exit): 3

Yes, he was once among us.

Available choices:
1: What a loss. I have other questions.

Choose an option (or 0 to exit): 1

Tell me what you wish to know.

Available choices:
1: Tell me about the Sigma Sword.
2: How to learn the voice of the Nagas?
3: So, you were friends with the Exalted Hawk, too, ah?

Choose an option (or 0 to exit): █
```

V. CONCLUSION

When designing a narrative heavy game, the design of the dialogue system is an integral part. Ripe planning and design is required to ensure that it lays a good foundation and to the narrative that is going to be written for the game. A good design is also required to ensure that game designers, programmers, and writers can work effectively together.

Analyzing common dialogue system types from various games, it is surmised that dialogue types can be idealized into two: the unique paths, where each choice leads to an array of different dialogues and choices, and the pseudo choice, where each choice leads to the same result. The unique paths is oftentimes unfeasible due to the exponential increase of text segments that needs to be written on each addition of choice. The pseudo choice is also not a good choice because of the linearity that'll result from the system, which is a bad faith to the players for games that offer interactivity as one of its selling point.

A combination of the two types, along with various elements to spice up the system, is required to craft a dialogue system that is robust, flexible and easy to work on, and is capable of serving a good experience to the player.

VI. APPENDIX

Paper Explanation Video on YouTube
https://youtu.be/F0iOiZk_0Ws

GitHub Repository
https://github.com/mraifalkautsar/MakalahMatdisDialog

VII. ACKNOWLEDGEMENT

This paper would not have been completed without the help of others. Therefore, the author would like to thank:

1. Allah SWT,
2. the author's parents and family,
3. the lecturers in charge of Discrete Mathematics, and
4. the author's friends

who have always supported the author during the making of this paper

REFERENCES

[1] https://philipphagenlocher.de/post/video-game-dialogues-and-graph-theory/ accessed on 5th January 2025
[2] https://informatika.stei.itb.ac.id/~rinaldi.munir/Matdis/2024-2025/18-Kombinatorika-Bagian1-2024.pdf accessed on 5th January 2025
[3] https://informatika.stei.itb.ac.id/~rinaldi.munir/Matdis/2024-2025/19-Kombinatorika-Bagian2-2024.pdf accessed on 5th January 2025
[4] https://informatika.stei.itb.ac.id/~rinaldi.munir/Matdis/2024-2025/20-Graf-Bagian1-2024.pdf accessed on 5th January 2025
[5] https://informatika.stei.itb.ac.id/~rinaldi.munir/Matdis/2024-2025/21-Graf-Bagian2-2024.pdf accessed on 5th January 2025
[6] https://informatika.stei.itb.ac.id/~rinaldi.munir/Matdis/2024-2025/22-Graf-Bagian3-2024.pdf accessed on 5th January 2025

PERNYATAAN

Dengan ini saya menyatakan bahwa makalah yang saya tulis ini adalah tulisan saya sendiri, bukan saduran, atau terjemahan dari makalah orang lain, dan bukan plagiasi.

Bandung, 8 Januari 2025

Raif

Muhammad Ra'if Alkautsar (13523011)