Implementation of Finite State Machine to Determine The Behaviour of Non-Playabale Character in Leadership Simulation Game

Muhammad Bagus Rizqi Alvian
Faculty of Computer Science
University of Jember
Jember, Indonesia
bagusriski1@gmail.com

Saiful Bukhori
Faculty of Computer Science
University of Jember
Jember, Indonesia
saiful.ilkom@unej.ac.id

Muhammad 'Ariful Furqon
Faculty of Computer Science
University of Jember
Jember, Indonesia
ariful.furqon@unej.ac.id

***Abstract*— In today's era, games are widely enjoyed by the Indonesian society, and one of them is simulation games. Simulation games have many advantages, including allowing players to experiment freely and encouraging them to learn. Therefore, the use of simulation games can be utilized as a training medium, such as leadership training. There are five levels of leadership based on The 5 Levels of Leadership: position, permission, production, people development, and pinnacle. Direct practice is necessary in training these levels of leadership through the implementation of Artificial Intelligence (AI) in simulation games. One of the AIs used for this implementation is the Finite State Machine (FSM). FSM will be implemented in Non-Playable Characters (NPCs) to determine behavior that is adjusted to the 5 levels of leadership. There are three State Machines (SM) applied to NPCs: Core Game SM, Movement SM, and Status SM. The use of FSM in NPCs results in dynamic NPC behavior in terms of physical movement and changes in NPC status according to 5 Levels of Leadership.**

Keywords*— *Simulation Game, Leadership, Artificial Intelligence, Finite State Machine

I. Introduction

In the present era, games are widely enjoyed by the Indonesian community. This statement is supported by data obtained by Katadata Media Network, which explains that Indonesia is the third country with the highest number of gamers[1]. There are various game categories available, such as action games, simulations, sports, strategy, adventure, and others [2]. Simulation games have the advantage of allowing experimentation without the consequences of those experiments [3]. Simulation games are widely played because they provide players with the motivation to learn [4]. The benefits offered through simulation games can be utilized as a training tool, one of which is for leadership development.

There are five levels of leadership that an individual possesses: position, permission, production, people development, and pinnacle. It is explained that direct practice is necessary to train and assess the leadership capabilities of an individual [5]. With this idea, the implementation of artificial intelligence in a game becomes essential. The goal of AI is to make the game intelligent with the aim of making it engaging to play [6]

. One application of AI in games is the use of Finite State Machine (FSM). In this context, FSM will be employed to determine the behavior of Non-Playable Characters (NPCs). Each NPC behavior will be influenced by the player's gameplay. FSM can be utilized to simplify the program code for Artificial Intelligence, especially for supporting characters that move or respond to specific conditions [7]. Each behavior is based on the book "The 5 Levels of Leadership" by John C. Maxwell, so it can be utilized as a leadership training tool for players. Based on the background information provided, the research conducted by the author aims to explore the implementation of Finite State Machine to determine NPC behavior in the development of a leadership simulation game.

II. Literature Review

A. Simulation games

Game simulation is a type of game that replicates real-world aspects for players to engage with [8]. These simulation games immerse players in activities reflective of real-life experiences, allowing them to partake in these activities without physically doing so.

The development of simulation games involves combining elements of both gaming and simulation. Thus, in addition to providing entertainment, simulation games offer other benefits to players, such as active player engagement, adaptability, self-paced experiences, feedback mechanisms, standardization, and cost-effectiveness [9].

B. Finite State Machines

Finite State Machine (FSM) is a state machine that consists of a limited set of states or conditions connected by transitions, forming a data structure referred to as a graph. Every game begins with an initiation state. Subsequently, the conditions within the game lead to transitions to other states. There are four main components of FSM: states, transitions, rules, and events [6].

Finite State Machine is commonly used to determine the behavior of AI in the game [10]. Utilizing Finite State Machine in game development makes the game more interactive and enjoyable to play [11].

TABLE I. FSM COMPONENTS

FSM Components	Description
States	Explaining the condition or what an object does in a game or NPC.
Transitions	This component explains the relationships between different states.
Rules	This component explains when transitions will occur.
Events	This component explains the activities that exist within states.

The components mentioned will form a Finite State Machine (FSM) design that regulates the behavior of game objects, including NPCs. FSM will shape a behavioral response for a game object into a state [12]. In games, the typical form of Finite State Machine applied is as shown in Table I. States are connected to each other using transitions, and each transition is triggered by activated rules.

C. Leadership

Leadership is a process of influential interaction that occurs when several people agree to designate someone as their leader to achieve a common goal [13]. Leadership is commonly associated with an individual's ability to lead. The definition indicates that the ability for leadership is necessary to influence others to achieve a common goal. In John C. Maxwell's book titled "The 5 Levels of Leadership," it explains that leadership abilities consist of five levels: Position, Permission, Production, People Development, and Pinnacle.

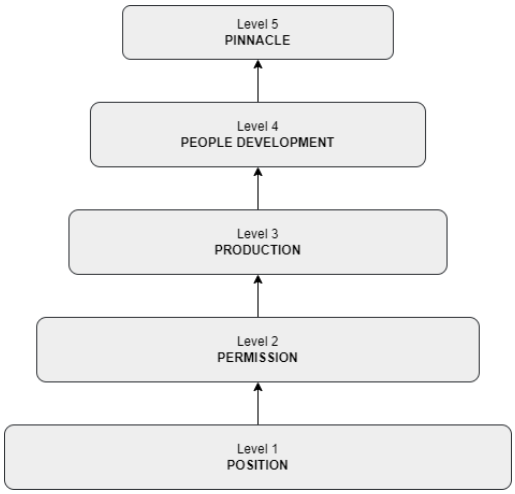

Fig. 1. Leadership Levels

In Fig.1. , the sequence of the 5 levels of leadership is explained. Each level presents increasing difficulties as the leadership level rises. As one climbs higher in leadership levels, the time required to advance to the next level becomes longer. There is a need for high commitment to reach higher levels. However, achieving higher levels will provide ease in leading an organization. High leadership capabilities also indicate a level of success for the organization [14].

TABLE II LEVEL DESCRIPTIONS

Leadership Levels	Description
Level 1 “Position”	In this level, the leader is followed by their members due to their position, as the person holds a leadership role by virtue of their position.
Level 2 “Permission”	In this level, the leader is followed by their members because the leader is recognized and known by the members.
Level 3 “Productive”	In this level, the leader is followed by their members because the leader is recognized for their contributions to the organization.
Level 4 “People Development”	In this level, the leader is followed by their members because the leader has developed their team members.
Level 5 “Pinnacle”	In this level, the leader is followed by their members because the leader has produced leaders who have reached level 4 of leadership.

In the book "The 5 Levels of Leadership," it is explained that each member has a different perspective on their leader's level. Therefore, to elevate the leadership level within an organization, there is a need for leadership skills directed toward every member of the organization. At each level, there are negative aspects, added value, best practices, and the belief required to ascend to a higher level.

III. RESEARCH METODOLOGY

In this research, the author employs the GDLC (Game Development Life Cycle) method. For development using the GDLC method, there are three main stages: initiation, production cycle, system testing, and release [15].

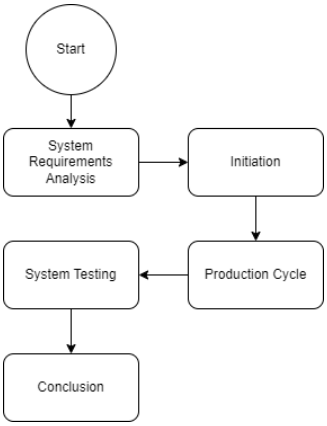

Fig. 2. Research methodology

A. System Requirements Analysis

In this stage, a system needs analysis will be conducted. The analysis performed involves determining specifications that include the necessary components to build and implement the system.

B. Initiation

The initiation stage is the initial phase in the development of a game. This stage involves outlining the game, identifying its concept, and determining the target audience. The output

of the initiation stage is a rough concept of the game under development.

There are two main outputs in this stage: the game concept and wireframe. The game concept represents the rough idea of the designed game and the inspiration behind it. The wireframe is a simple representation of the designed game, serving as the foundation for the UI/UX of the game.

C. Production Cycle

In this stage, we enter the production cycle, which is divided into three parts: pre-production, production, and alpha testing. The cycle begins with the pre-production phase, during which a clear plan for the game under development is established. This is followed by the production phase to produce and develop the game.

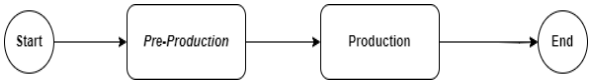

Fig. 3. Production Cycle

1) Pre-Production: In the Pre-Production stage, the author determines the Game Design that will serve as the main design for the game under development. Additionally, in this stage, the steps of game production are outlined. The product of this stage is the Game Design Document (GDD). The Game Design Document is a collection of documents used to design, plan, and outline the elements of the game, serving as the foundation for game development [16].

2) Production: In this section, the focus is on the development of the game planned in the previous stage. Game development typically involves creating game assets, implementing the GDD in Unity, and programming the game using C#. The output of this stage is the actual game application that has been structured and built as a whole. During this stage, the author implements the finite state machine in the developed game. The implementation results in two outputs: the architecture design of the finite state machine and the written code developed for the game.

D. System Testing

In this evaluation, the author employs a testing method called black box testing. Black box testing focuses on the input and output of the software being tested. In its application, the author utilizes the type of Functional Testing within this Black box testing. The testing is centered around the functions of the FSM implementation on NPCs. Several test cases will be created to assess each functionality of the FSM. The results of these tests will then be analyzed, and conclusions will be drawn.

IV. RESULT

A. System Requirement Analysis

In this stage, the system requirements in the development of the simulation are analyzed. This stage includes the Statement of Purpose, system functions, functional requirements, and non-functional requirements..

1) Statement of Purpose: The leadership simulation game focuses on training players in leadership without sacrificing entertainment value. The concept of leadership in this game is based on the leadership book titled 'The 5 Levels of Leadership.' In the game, players will lead an organization in a university setting. This game is designed for players who enjoy management, simulation, and strategy games. It is intended for the PC platform as one of the main platforms to be used

2) System Function: The main function of this game is to train and allow players to practice leadership skills within the game. Players will gain an understanding of leadership theories based on the book used as the foundation for leadership concepts in the game. Players will then apply these theories within the game. The game is capable of identifying and providing a rating of the player's leadership level based on their gameplay.

3) Problem Identification : In the development of leadership simulation games, various challenges arise that need to be addressed to produce a game that aligns with its intended functions. These challenges include:

- The leadership simulation game requires the formulation of leadership theories and practices that align with the book The 5 Levels of Leadership.
- Players need to play this game within a specified time frame to identify the player's appropriate level of leadership.

4) Functional Requirments: Functional requirements are the needs of the system to produce the desired system functions. Some of these requirements include:

- The leadership simulation game is capable of implementing leadership theories and practices.
- The leadership simulation game is capable of assessing the player's leadership abilities.
- This game requires players to choose an answer. Each answer has different consequences.
- Players are required to manage an organization with managerial skills in this game.

5) Non-Functional Requirements: Non-functional requirements are requirements that emphasize the behavior of the system. These requirements are related to the services offered by the system, such as:

- The game can run on PC with Windows 10 OS.
- The game requires low hardware requirements.
- The game can assess a player's leadership abilities with just one playthrough.
- The game has a small file size.

B. Initiation

In this stage, the rough concept of the game will be designed and created to serve as the core reference for the game design in the next stage. The outputs generated at this stage are the game concept and the game wireframe.

1) Game Concept : The development of this simulation game draws inspiration from various games of the same genre. Several references serve as conceptual foundations for the development of this game.

TABLE III. GAME IDEAS

Game Name	The concepts drawn from include
Paradox games	Game design concept and the features
Kairosoft games	Art design concept
Nobunaga ambition games	Main feature turn system

In essence, all three games share a common game design, namely a strategy game. The ideas from these games are then processed and combined with leadership concepts based on the book 'The 5 Levels of Leadership,' implementing the Finite State Machine algorithm.

2) Wireframe : From the rough concept created, a wireframe is then structured for the developed game. The produced wireframe will serve as the main design concept for the UI/UX of this simulation game.

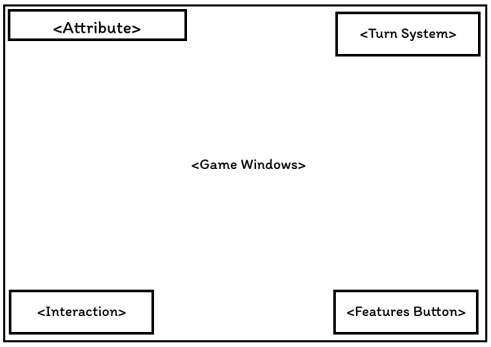

Fig. 4. Wireframe

The initial wireframe for this simulation game is the game window wireframe. This wireframe illustrates the main screen layout of the game being played. The written elements on the image indicate the locations of the features implemented in the simulation game.

C. Production Cycle

In the production cycle, there are two main stages implemented, namely Pre-Production and Production. This stage focuses on the execution of game development from the design phase to the actual game creation phase.

1) Pre-Production: In this stage, the Game Design Document (GDD) is outlined for the upcoming game development. The GDD for the leadership simulation game includes:

TABLE IV GAME DESIGN DOCUMENT

Category	Description
Genre	Simulation Strategy
Controls	Mouse and Keyboard
Tech Stack	Game Engine: Unity Documentation: Notion Asset Editor: Adobe Illustrator, Figma
Game Summary	You are tasked with becoming a leader in a campus organization. Hone your leadership skills to reach the highest level. The more you apply your leadership abilities, the higher your experience level in leadership becomes. There are specific criteria for advancing your leadership level. Seek and ascend to the highest level.
Core Player Experience	Organizing and leading a student organization from the perspective of the organization's chairperson.

Within the Game Design Document (GDD), there is also something known as the Core Loop. The Core Loop is a design of actions in the game that is repeated continuously as the main flow for the player's experience. It is a foundational design within a game's design, serving as a tool to motivate players to engage with the game continuously.

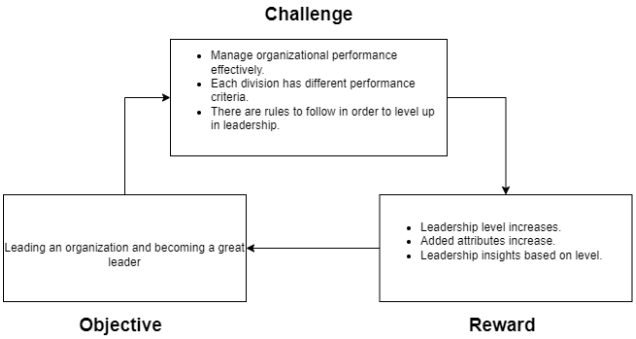

Fig. 5. Core Loop

The game concludes when the total number of turns in the game reaches 20. Upon completion of the game, it will display the player's leadership level throughout the gameplay. This level is influenced by the player's choices in the game. Each player choice has varying impacts. If a player chooses options that reflect leadership values, they will achieve a higher leadership level, and vice versa.

Additionally, besides the Core Loop in the GDD, there is the Core Feature. The Core Feature is a crucial aspect provided in a game, representing its distinctive characteristics and gameplay style. In the leadership simulation game, there are three core features, namely:

a) Turn-base simulation : The primary system in the leadership simulation game will use a turn-based structure, with each turn representing a unit of a week. When the player presses the play button, the game will progress for a duration of seven days within the game.

b) Focused character play and event system : The game will focus on the perspective of the character within the game or the player's perspective as a leader. Events will occur

within the game, each presenting a scenario that the player must respond to. Every answer chosen by the player will have an impact on both themselves and the organization. The emphasis here is on the leader's perspective in leading the organization. Each scenario that arises is based on real-life experiences commonly encountered in student organizations. Thus, the stories will convey a sense of real-life situations. In total, there are 26 scenarios that appear in the game.

c) Scenario The 5 levels of leaderhsip : In this leadership simulation game, the main attributes used to train leadership are based on the book 'The 5 Levels of Leadership.' The five leadership levels to be implemented in the game are position, permission, production, people development, and pinnacle.

TABLE V LEADERSHIP ATTRBUTE DESIGN

Attribute	Description
Trust	Demonstrates the level of trust in the leader. This attribute is closely related to all levels of leadership.
Influence	Indicates the level of influence the leader has on their members. This attribute is closely related to level 4 of leadership.
Relation	Indicates the level of relationship between the leader and their members. This attribute is closely related to level 2 of leadership.
Morale	Indicates the level of morale among the members. This attribute is closely related to level 3 and level 2 of leadership.

In the leadership attribute of the game, there are four main attributes applied: Trust, Influence, Relation, Morale. These key attributes are commonly mentioned and have an impact on the book 'The 5 Levels of Leadership.' Overall, these attributes have values that impact the entire leadership spectrum. Therefore, these attributes are considered as the main parameters for leadership in the leadership simulation game.

TABLE VI LEADERSHIP LEVELS SCENARIO

Leadership levels	Requirements
Level 1 “Position”	
Level 2 “Permission”	1. 50% of all NPCs meet the level-up requirements. 2. Leadership attribute requirements for NPCs: • Trust: 100 • Influence: 100 • Morale: 100 • Relation: 120 3. Completing questions at level 2.
Level 3 “Productive”	1. 50% of all NPCs meet the level-up requirements. 2. Leadership attribute requirements for NPCs: • Trust: 200 • Influence: 200 • Morale: 220 • Relation: 220 3. Completing questions at level 3.
Level 4 “People Development”	1. 50% of all NPCs meet the level-up requirements. 2. Leadership attribute requirements for NPCs: • Trust: 300 • Influence: 320 • Morale: 320 • Relation: 320 3. Completing questions at level 4.
Level 5 “Pinnacle”	1. 50% of all NPCs meet the level-up requirements. 2. Leadership attribute requirements for NPCs: • Trust: 500 • Influence: 500 • Morale: 500 • Relation: 500 3. Completing questions at level 5.

This attribute will be applied as a leadership attribute for each NPC in the game. Players will acquire this attribute through their gameplay, utilizing the Meet Mechanic, Action Mechanic, and Decision System features. In this scenario, players can level up their leadership if they meet the requirements. Then, players must answer questions to advance to the next level. If there is a wrong choice, the player will receive a penalty and cannot level up for three turns. Additionally, if a player fails to plan an action, that action will not be executed and will have a negative effect on the organization.

At the end of each turn, insights will appear in the form of sentences. These insights are derived from the sub-chapters in the book 'The 5 Levels of Leadership,' providing information on each level. There will be a total of 20 insights for each level that will appear randomly at the end of the turn. Players will gain knowledge directly related to their leadership level from these insights.

2) Production : In this stage, the development of the game designed in the pre-production stage will take place. The outputs from this stage include the UI/UX design of the game, the architecture design of the finite state machine, and the implementation of the finite state machine.

a) UI/UX Design : In this section, the UI/UX design that will be used in the leadership simulation game is outlined. The UI/UX components include the game windows, action UI, meet UI, and interaction UI.

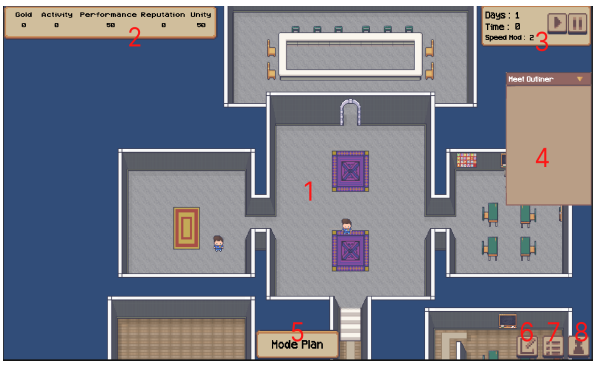

Fig. 6. Game Windows

In Figure 4, the design of the Game Windows for the development of the leadership simulation game is presented. Each number corresponds to different explanations and functions. The following is an explanation of the functions of each UI/UX element.

TABLE VII GAME WINDOWS FUNCTION

No.	Name	Function
1	Game Interface	Displaying the current state within the game and interacting with NPCs or organizational divisions.
2	Organization Attribute UI	Displaying real-time values of organizational attributes.
3	Turn System UI	Showing the day, time, and turn speed. It can be used to play or pause turns
4	Outliner UI	Displaying the scheduled meetings planned by the player.
5	Game State UI	Showing the current game state, including plan mode, pause mode, and play mode.
6	Action Board Button	Clicking will display the Action Board
7	Decision Menu Button	Clicking will display Leadership Task
8	Player Profile Button	Clicking will display Player Profile.

Then, the Interaction UI functions as a player interface for various interactions with the divisions within the game. Currently, there are two offered interactions: Meeting and Action.

Fig. 7. Interaction Room UI

Furthermore, it also displays the condition of the division's room that the player will interact with. If a meeting is conducted, the condition of the room will change significantly. Clicking the 'meet' button will reveal the Meeting UI, while clicking 'action' will bring up the Action UI.

Fig. 8. Meeting UI

In the Meeting UI, players can choose between two meeting categories: regular and event. Regular meetings involve routine discussions, while event meetings are designed to prepare for planned actions. Players can select the meeting time within the range of 1-to-3-time units. Once confident, they can proceed by clicking 'plan.'

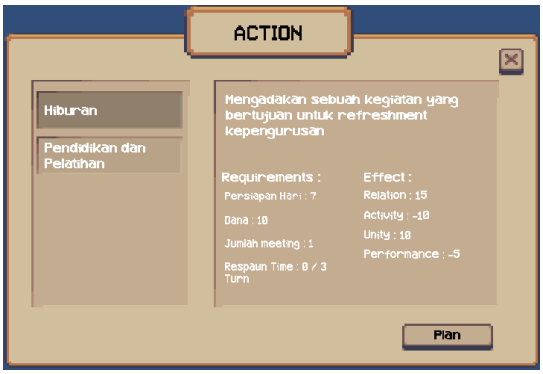

Fig. 9. Action UI

In the Action UI, players can plan a work program for each division. Each division offers two distinct work programs. Each program has specific requirements and effects on the organization. Once the player has chosen a work program, they can click the 'plan' button to schedule the program.

Fig. 10. Decision Sytem UI

In the Decision System UI, information is displayed regarding the player's eligibility to advance to the next leadership level. Once the player meets the criteria, they can press the 'upgrade' button to progress to the next stage.

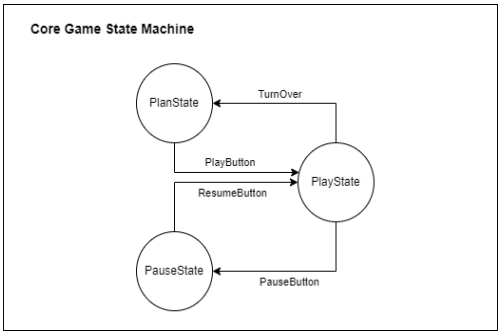

Fig. 9. Core game state machine diagram

Subsequently, a series of questions will appear that the player must answer to ascend to the next leadership level.

b) Design architecture of finite state machine : The architecture design of the Finite State Machine to be implemented in this leadership simulation game consists of three main FSM designs: plan, play, and pause.

TABLE VIII CORE GAME STATE MACHINE

State	Events	Rules	Action
Plan	In this state, the player can: 1. Plan meetings 2. Plan actions or work programs 3. Take tests to advance to the next leadership level in the Decision System	Initial state of the game	First state
		After the seventh day of the game or turn is completed	Play state
Play	In this state, players can: 1. Conduct planned meetings 2. Implement planned actions or work programs	If the player presses the "play" button during the planning state.	Plan state
		If the player presses the "play" button during the pause state.	Pause state
Pause	In this state, the player can pause momentarily during the play state with the aim of reviewing the game and being cautious in case something has been overlooked.	If during the play state, the player presses the "pause" button,	Play state

In each main state, there is a state machine within it, and the transition between its states is influenced by the main state. In the implementation of FSM on NPCs in the game, there are two FSMs in the NPC: Movement State Machine and Status State Machine.

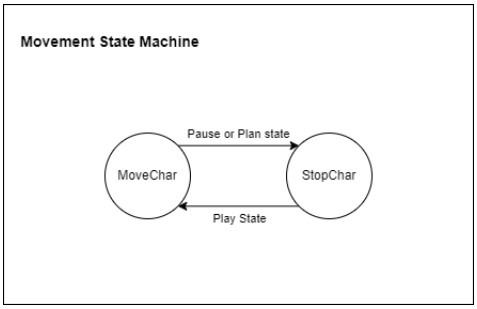

Fig. 10. Movement state machine diagram

The Movement State Machine is a behavior FSM applied to NPCs in this game, and its function is to control the movement behavior of NPCs in the game. There are two states in this state machine: MoveChar and StopChar.

TABLE IX MOVEMENT STATE MACHINE

State	Events	Rules	Action
MoveChar	In this state, the NPC moves freely, such as walking to its designated location and performing tasks assigned by the player	Transition to this state occurs when the game state enters the Play state.	StopChar
StopChar	In the StopChar state, the NPC remains stationary, with only idle animations.	Transition to this state occurs when the game state enters the Pause or Plan state.	MoveChar

The transition between states in the Movement State Machine is influenced by the transition of states in the Game State. Utilizing the Movement State Machine will result in dynamic NPC movements that adapt to the states present in the game.

The Status State Machine represents the FSM behavior applied to NPCs to influence the leadership level effects on the player. There are 5 states in this State Machine: Member, Friend, Partner, Comrade, and Golden. Each state represents the 5 leadership levels achieved by the player.

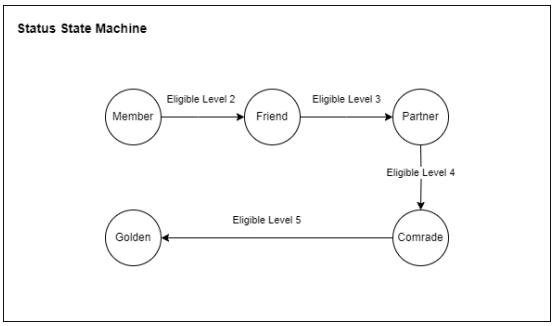

Fig. 11. Movement state machine diagram

In the implementation of each state in the Status State Machine, it is based on the interpretation and simple modeling of each leadership level. The following is an explanation of the implementation of the 5 levels of leadership in the Status State Machine.

TABLE X LEADERSHIP IMPLEMENTATION ON FSM

Leadership Levels	State Implementation	Description
Level 1 "Position"	*Member*	At this level, the implementation in the game serves as the initial state initiation for NPC.
Level 2 "Permission"	*Friend*	At this level, the spotlight is on the Relation attribute because this level focuses on the leader's relationship with the player.
Level 3 "Productive"	*Partner*	At this level, the spotlight is on the Morale attribute as it will boost the morale of the members.
Level 4 "People Development"	*Comrade*	At this level, the spotlight is on the Influence attribute because this level is related to the influence exerted.
Level 5 "Pinnacle"	*Golden*	At this level, it is the pinnacle level. For its implementation, the player needs to elevate all leadership attributes to their highest levels.

The correlation of each leadership level to be applied to each state in the Status State Machine is explained in Table 4.8. Each implemented state will reflect each leadership level, having specific requirements and benefits for the player. Each state has its conditions, and there are advantages for the player. These states are then incorporated into the Status State Machine, adjusted for events, rules, and transitions for each state. The implementation form in the Status State Machine is as follows.

TABLE XI STATUS STATE MACHINE

State	Events	Rules	Action
Member	No events	First state	First state
Friend	NPC will receive an additional 1.2 times for each increase in the leadership attribute Relation.	Enter this state if the player has reached level 2 of leadership.	Member state
Partner	NPC will receive an additional 1.2 times for each increase in the leadership attribute Trust.	Enter this state if the player has reached level 3 of leadership.	Friend state
Comrade	NPC will receive an additional 1.2 times for each increase in the leadership attribute Influence.	Enter this state if the player has reached level 3 of leadership.	Partner state
Golden	NPC will receive an additional 1.5 times for each increase in all leadership attributes.	Enter this state if the player has reached level 5 of leadership.	Comrade state

Each state has different explanations for events, rules, and transitions. The Status State Machine is then implemented into the NPC's behavior related to leadership levels. Overall, each state provides benefits in the form of additional

attributes. However, as players progress to higher leadership levels, the difficulty level they face increases.

c) Finite state machine implementation : The implementation of the three State Machines, namely Core Game SM, Movement SM, and Status SM, has its own script code. This is intended to facilitate the system to execute different State Machines simultaneously. Here is the implementation of each State Machine. In the Core Game SM, the focus is on the core game behavior in this leadership simulation game. The implementation of this State Machine is divided into three states: plan, play, and pause.

```
using Leadership.Action;
using TMPro;
using UnityEngine;


namespace Leadership.Core
{
    public class GameSM : StateMachine
    {
        [HideInInspector]
        public Plan planState;
        public Play playState;
        public Pause pauseState;

        [SerializeField] public TurnSystem turnSystem;
        [SerializeField] TextMeshProUGUI statusTextNow;

        private LeadershipMechanic leadershipMechanic;
        private ActionDatabase actionDatabase;


        private void Awake()
        {
            planState = new Plan(this);
            playState = new Play(this);
            pauseState = new Pause(this);

            leadershipMechanic = FindObjectOfType<LeadershipMechanic>();
            actionDatabase = FindObjectOfType<ActionDatabase>();

        }

        public override void PlayButton()
        {
            currentState = playState;
            base.PlayButton();

        }

        public override void PauseButton()
        {
            if (IsPlanState()) return;
            base.PauseButton();

            currentState = pauseState;
        }

        public bool IsPlanState()
        {
            return currentState == planState;
        }

        public bool ChangeWeek()
        {
            return turnSystem.CalenderTime >= 8;
        }

        protected override BaseState GetInitialState()
        {
            return planState;
        }

        public void PrintNow(string value)
        {
            print(value);
        }

        public LeadershipMechanic GetLeadershipMechanic()
        {
            return leadershipMechanic;
        }

        public ActionDatabase GetActionDatabase()
        {
            return actionDatabase;
        }



        public void ChangeModeText()
        {
            statusTextNow.text = currentState.name;
        }

        // private void OnGUI()
        // {
        //     string content = currentState != null ? currentState.name : "(no current state)";
        //     GUILayout.Label($"<color='black'><size=40>{content}</size></color>");
        // }
    }
                                }
```

Fig. 12. Core Game SM

In the Movement SM, the focus is on the behavior of NPC character movement in the game. The implementation of this State Machine is divided into 2 states: stop and move. The implementation form of the State Machine is shown in Fig.13

```
using Leadership.Core;
using UnityEngine;
using UnityEngine.AI;

namespace Leadership.Character
{
    public class MovementSM : StateMachine
    {
        [HideInInspector]
        public StopCharacter stopCharState;
        public MoveCharacter moveCharState;


        [HideInInspector] public GameSM gameSM;
        [HideInInspector] public Vector3 dir;
        [HideInInspector] public Animator animator;
        [HideInInspector] public NavMeshAgent agent;

        [SerializeField] public Transform targetLoc;

        private TurnSystem turnSystem;

        private void Awake()
        {
            stopCharState = new StopCharacter(this);
            moveCharState = new MoveCharacter(this);

            gameSM = FindObjectOfType<GameSM>();
            animator = GetComponent<Animator>();
            agent = GetComponent<NavMeshAgent>();

            turnSystem = FindObjectOfType<TurnSystem>();


        }

        public override void PlayButton()
        {
            currentState = moveCharState;
            base.PlayButton();

        }

        public override void PauseButton()
        {
            if(currentState == stopCharState) return;
            base.PauseButton();

            currentState = stopCharState;
        }

        protected override BaseState GetInitialState()
        {
            return stopCharState;

        }

        public void SetTransfromTarget(Transform target)
        {
            targetLoc = target;
        }

        public float  GetSpeedModifier()
        {
            return turnSystem.SpeedModifier;
        }

        // public bool IsThereTarget()
        // {
        //     return target != null;
        // }

        // private void OnGUI()
        // {
        //     string content = currentState != null ? currentState.name : "(no current state)";
        //     GUILayout.Label($"<color='blue'><size=40>{content}</size></color>");
        // }

    }
}
```

Fig 13 Movement SM

In the Status SM, the focus is on the behavior of the player's leadership level influence on NPCs in the game. The implementation of this State Machine is divided into 5 states: member, friend, partner, comrade, and golden. The implementation form of the State Machine is shown in Fig.14 .

```
using Leadership.Core;
using UnityEngine;

namespace Leadership.Character
{
    public class StatusPlayerSM : StateMachine
    {
        [HideInInspector] public MemberState memberState;
        [HideInInspector] public FriendState friendState;
        [HideInInspector] public PartnerState partnerState;
        [HideInInspector] public ComradeState comradeState;
        [HideInInspector] public GoldenState goldenState;

        private GameSM _gameSM;
        private CharacterMechanic characterMechanic;
        private LeadershipMechanic leadershipMechanic;
        private void Awake()
        {
            memberState = new MemberState(this);
            friendState = new FriendState(this);
            partnerState = new PartnerState(this);
            comradeState = new ComradeState(this);
            goldenState = new GoldenState(this);

            _gameSM = FindObjectOfType<GameSM>();
            characterMechanic = GetComponent<CharacterMechanic>();
            leadershipMechanic = FindObjectOfType<LeadershipMechanic>();
        }

        protected override BaseState GetInitialState()
        {
            return memberState;
        }

        public bool EligbleToNextLevel(int levelCheck)
        {
            return characterMechanic.CheckLevelUp(1) && levelCheck == characterMechanic.GetLevelLead();
        }

        public void NextLevel()
        {
            if(currentState == memberState)
            {
                ChangeState(friendState);

            }else if(currentState == friendState)
            {
                ChangeState(partnerState);

            }else if(currentState == partnerState)
            {
                ChangeState(comradeState);

            }else if(currentState == comradeState)
            {
                ChangeState(goldenState);

            }
        }

        public string GetCurentStatusStateText()
        {
            return currentState.name;
        }

        public CharacterMechanic GetCharacterMechanic()
        {
            return characterMechanic;
        }
        public LeadershipMechanic GetLeadershipMechanic()
        {
            return leadershipMechanic;
        }

        public void PrintText(string value)
        {
            print(value);
        }

        public bool IsCharacterLevelUpTwo(bool value)
        {
            characterMechanic.IsLevelUpTwo = value;

            return characterMechanic.IsLevelUpTwo ;
        }

        public bool GetCharacterLevelUpTwo()
        {
            return characterMechanic.IsLevelUpTwo;
        }

        public bool IsCharacterLevelUpThree(bool value)
        {
            characterMechanic.IsLevelUpThree = value;

            return characterMechanic.IsLevelUpThree ;
        }
        public bool GetCharacterLevelUpThree()
        {
            return characterMechanic.IsLevelUpThree;
        }

        public bool IsCharacterLevelUpFour(bool value)
        {
            characterMechanic.IsLevelUpFour = value;

            return characterMechanic.IsLevelUpFour ;
        }
        public bool GetCharacterLevelUpFour()
        {
            return characterMechanic.IsLevelUpFour;
        }

        public bool IsCharacterLevelUpFive(bool value)
        {
            characterMechanic.IsLevelUpFive = value;

            return characterMechanic.IsLevelUpFive;
        }
        public bool GetCharacterLevelUpFive()
        {
            return characterMechanic.IsLevelUpFive;
        }

        public bool LeadershipIsLevelUp(int levelCap)
        {
            if(leadershipMechanic.GetLevelLeadershipPlayer() >= levelCap)
            {
                return true;
            }

            return false;
        }
    }

                                }
```

Fig 14 Status SM

D. System Testing

In testing the NPC behavior, Black-Box testing is utilized. From the conducted test data, there are a total of 13 test cases aimed at evaluating NPC behavior. Each test case consists of functionality tests, expected outcomes, and the procedures followed during testing. The three implemented State Machines for NPCs will be subjected to this testing to ascertain the overall behavior alignment with the specified test cases.

Each planned test case will be executed one by one. It will then be determined whether the game passes the specified test case. If it passes, the testing process will proceed to the next test case. If it fails, the game will be debugged, and the issues addressed before retesting until it passes. Once all test cases have been executed, the system testing will be considered complete. The results from each executed test case indicate that the implemented FSM for NPCs has been successful, and functions as intended.

V. CONCLUSION

Based on the problem formulation written in this study, the results show the implementation of Finite State Machine on NPC with 3 State Machines, namely Core Game State Machine, Movement State Machine, and Status State Machine. Each state machine regulates different behaviors. The Core Game State Machine controls the flow of the game that impacts other state machines. The Movement State Machine manages NPC movement in the game. The Status State Machine regulates the leadership level of each NPC based on the 5 Levels of Leadership. Test results indicate that the implementation of Finite State Machine to determine NPC behavior in leadership simulation games produces dynamic NPC behavior in terms of NPC physical movement and changing NPC status. Thus, from the conducted research,

the implementation of a Finite State Machine is a suitable choice for determining NPC behavior in a dynamic and flexible game.

REFERENCES

[1] V. Azkiya, "The Third Highest Number of Indonesian Gamers in the World (translate)". Accessed: Oct. 28, 2022. [Online]. Available: https://databoks.katadata.co.id/datapublish/2022/02/16/jumlah-gamers-indonesia-terbanyak-ketiga-di-dunia

[2] J. J. Vargas-Iglesias, "Making Sense of Genre: The Logic of Video Game Genre Organization," *Games Cult*, vol. 15, no. 2, pp. 158–178, Mar. 2020, doi: 10.1177/1555412017751803.

[3] H. Wise, "10 Reasons Real Life Simulation Games Are So Head-Scratchingly Successful." Accessed: Feb. 08, 2023. [Online]. Available: https://www.thegamer.com/simulation-games-successful-the-sims/#you-are-the-captain

[4] J. J. Padilla *et al.*, "Using simulation games for teaching & learning discrete-event simulation," *Proceedings - Winter Simulation Conference*, pp. 3375–3384, Jul. 2016, doi: 10.1109/WSC.2016.7822368.

[5] J. Maxwell, *The 5 Levels of Leadership*, 2nd ed. Surabaya: Media Distribusi Cemerlang, 2017.

[6] D. Aversa, *Unity artificial intelligence programming : Add powerful, believable, and fun AI entities in your game with the power of Unity*, 5th ed. Birmingham: Packt, 2022.

[7] A. Taru, "Penerapan Finite State Machine Pada Perancangan Game." Accessed: Jan. 27, 2023. [Online]. Available: https://www.gamelab.id/news/206-penerapan-finite-state-machine-pada-perancangan-game

[8] J. Ellis, "What is a Sim Game?," EasyTechJunkie. Accessed: Jan. 27, 2023. [Online]. Available: https://www.easytechjunkie.com/what-is-a-sim-game.html

[9] M. Arshavskiy, "Simulations And Games: Making Learning Fun! - eLearning Industry." Accessed: Nov. 27, 2022. [Online]. Available: https://elearningindustry.com/simulations-and-games-making-learning-fun

[10] C. Buttice, "Finite State Machine: How It Has Affected Your Gaming For Over 40 Years." Accessed: Jan. 27, 2023. [Online]. Available: https://www.techopedia.com/finite-state-machine-how-it-has-affected-your-gaming-for-over-40-years/2/33996

[11] D. Jagdale, "Finite State Machine in Game Development," *International Journal of Advanced Research in Science, Communication and Technology*, pp. 384–390, Oct. 2021, doi: 10.48175/IJARSCT-2062.

[12] H. F. Ramadhan, S. H. Sitorus, and S. Rahmayuda, "Educational Game Introduction to West Kalimantan Culture and Tourism Using Android-Based Metdoe Finite State Machine (Translate)" at Coding: Journal Komputer dan Aplikasi, 7(1):108-119 2019.

[13] A. Silva, "What is Leadership?," *Journal of Business Studies Quarterly*, 8(1), 2016.

[14] M. Asrar-ul-Haq and T. Anjum, "Impact of narcissistic leadership on employee work outcomes in banking sector of Pakistan," *Future Business Journal*, vol. 6, no. 1, Dec. 2020, doi: 10.1186/s43093-020-00040-x.

[15] R. Krisdiawan, "Implementation of Gdlc System Development Model and Linear Congruential Generator Algorithm in Puzzle Game (translate)". *Jurnal Nuansa Informatika*, 12(2), Jul. 2018.

[16] M. T. Trilaksono, "Designing Game Design Document," Jan. 2022.