



TRIBHUVAN UNIVERSITY
Institute of Science and Technology
RPG game: Path of wanderer
Mid Defense Report

Submitted to:
Department of Computer Science and Information Technology
Academia International College

In partial fulfillment of the requirement for the degree of B.Sc.
Computer Science and Information Technology (B.Sc. CSIT)

Submitted by:
Kritan Shrestha (T.U. Symbol No. 29012/078)
Smriti Tamang (T.U. Symbol No. 29031/078)



Tribhuvan University

Institute of Science and Technology

Academia International College

Department of Computer Science and Information Technology

Email: mail@academiacollege.edu.np

Supervisor's Recommendation

I hereby recommend that the project work report prepared under my supervision Mr. Kritan Shrestha (T.U. Symbol No. 29012/078), Ms. Smriti Tamang (T.U. Symbol No. 29031/078) entitled "RPG game: Path of wanderer" be accepted as fulfilling in partial requirements for the degree of Bachelor of Science in Computer Science and Information Technology. In my best knowledge, this is an original work in Computer Science and Information Technology.

.....
Er. Anup Shrestha

Project Supervisor

Department of Computer Science and Information Technology

Academia International College,

Gwarko, Lalitpur

Acknowledgement

We owe our most profound appreciation to Academia International College for giving us a chance to work on this project as part of our syllabus.

Special thanks to our supervisor, Er. Anup Shrestha (Academia International College), for his consistent guidance, support, and feedback throughout the report's creation. We are generously obligated to him for providing this excellent opportunity to expand our knowledge. It helped us a lot to realize what we studied for.

We would like to express our sincere gratitude to all those individuals, families, friends, colleagues, and teachers for supporting and helping us a lot in finalizing this project within the limited time frame by providing valuable insights and feedback on the report.

Thanking You,

Kritan Shrestha (T.U. Symbol No. 29012/078)

Smriti Tamang (T.U. Symbol No. 29031/078)

Abstract

The "Path of Wanderer" is a single-player RPG game application that serves to provide an immersive and engaging fantasy gaming experience to its users. It is a story-driven role-playing game that caters towards players seeking narrative depth, character progression, and strategic combat mechanics without the burden of pay-to-win models or excessive monetization.

This game features a comprehensive fantasy world where players can explore diverse environments, complete meaningful quests, engage in tactical combat with various monsters, and ultimately face a challenging boss battle to achieve victory. The game includes essential RPG elements such as character progression, inventory management, quest tracking, and NPC interactions, all wrapped in an intuitive user interface designed for seamless navigation.

The game will be developed using Unity engine with C# as the programming language. Visual assets will be created using industry-standard tools, and the project will utilize version control systems for collaborative development. The game is designed to run efficiently on PC platforms, ensuring accessibility for players with varying hardware specifications.

Path of Wanderer represents a return to traditional RPG values, emphasizing storytelling, exploration, and player agency over competitive multiplayer mechanics. The project demonstrates the potential of Unity for creating engaging single-player experiences that prioritize player enjoyment and narrative immersion over profit-driven design decisions.

Abbreviation

RPG	Role-Playing Game
NPC	Non-Player Character
UI	User Interface
UX	User Experience
XP	Experience Points
OOP	Object-Oriented Programming
PC	Personal Computer
LTS	Long-Term Support
Git	Version Control System (Git)

Table of Contents

Supervisor's Recommendation	i
Acknowledgement	ii
Abstract.....	iii
Abbreviation	iv
List of figures.....	vii
List of tables.....	viii
Chapter 1: Introduction.....	1
1.1. Introduction.....	1
1.2. Problem Statement.....	2
1.3. Objectives	3
1.4. Scope and Limitations.....	3
1.4.1. Scopes	3
1.4.2. Limitations	3
1.5. Methodology	4
1.6 Report Organization.....	4
Chapter 2: Background Study and Literature Review	6
2.1 Background Study.....	6
2.2 Literature Review.....	6
Chapter 3: System Analysis	8
3.1. System Analysis.....	8
3.1.1. Requirement Analysis	8
3.1.2. Feasibility Study	9
□ Technical:.....	9
a. Operational:.....	9
b. Economic:.....	9
c. Gantt Chart:	9
3.1.3 Analysis (Object Oriented).....	10
Chapter 4: System Design.....	13
4.1 Design (Object Oriented).....	13
4.2. Algorithm Details.....	13
4.2.1 Finite State Machines (FSM).....	13
4.2.2 Pathfinding (A* Algorithm).....	13
4.2.3 Dialogue Trees	14
4.2.4 Behavior Trees	14
4.2.5 Quest Tracking Algorithm.....	14

4.2.6 Combat Mechanics Algorithm	15
4.2.7 Event-Driven Architecture	15
Chapter 5: Implementation and Testing	16
5.1. Implementation	16
5.1.1. Tools Used.....	16
5.1.2. Implementation Details of Modules.....	17
5.2 Testing.....	18
5.2.1 Unit Testing.....	18
5.2.2 Integration Testing	18
5.2.3 System Testing	19
Play entire game.....	19
Smooth gameplay with no errors	19
Still some bugs in gameplay	19
Ongoing.....	19
5.3 Result Analysis.....	19
Chapter 6: Conclusion and Future Recommendations.....	20
6.1 Conclusion	20
6.2 Future Recommendations	20
References.....	21
Appendix.....	22

List of figures

Figure 1: Agile Methodology of Software Development	4
Figure 2 Use Case Diagram	8
Figure 3:Gantt chart	9
Figure 4: Use Case Diagram	10
Figure 5: Sequence Diagram.....	11
Figure 6: Activity Diagram	12
Figure 7 : Creating Game in Unity	22
Figure 8: Enemy of Game.....	22
Figure 9 : Follow Camera C# Script.....	23
Figure 10: Enemy AI Controller	23

List of tables

Table 1: Player Movement Test.....	18
Table 2: Obstacle Collision Test	18
Table 3:Player Data Persistence Test	19
Table 4: Full Game Playthrough Test	19

Chapter 1: Introduction

1.1. Introduction

One of the very first games ever released was the 1976 Colossal Cave Adventure. With advancements in technology, the visual adventure games started to appear. Considered one of the first visual adventure games, Mystery House by Sierra On-line was created in 1980s. Yet, the input texts were still required. In time's passage, new generation games began development of 3D gameplay. Technologies surround everything now. That is quite factual within this age. A wide variety of adventure games is available across the world as well as on the play store. However, according to my research, most games do not provide educational purposes for their users or players. At this moment, the market provides a wide variety of games. These games mainly engage their users, which represents one of the market's largest weaknesses. For that specific purpose, this 3D RPG (Role Playing Game) has been developed in order to fill up that gap and to provide educational benefits while it entertains the users and the players who are actively engaging in the game.

This 3D adventure game is a single player educational fantasy game. In this game, the player controls the actual character to freely walk and climb around the platforms. The game has several quests completely different from the first one to make it interesting for the player and to keep them engaged in playing the game. As the game progresses, the players can experience various levels in the game and objectives, difficulty varying according to the quests of the game. In the game "Path of Wanderer", players can explore a fantasy world, complete quests, fight monsters, and face off against a boss to achieve victory. The game will feature simple RPG mechanics, such as quest tracking, combat systems, and player progression, without any pay-to-win elements. The game is designed for casual players, specifically for a small group of friends and teachers, and will be available on PC only.

This game will utilize Unity and C# programming language to create the core gameplay and mechanics, with the game's art style consisting of 3D or 2D models set in a fantasy world inspired by old-time themes. The goal of the project is to develop an engaging yet simple RPG experience that showcases the potential of Unity for creating accessible and fun games.

1.2. Problem Statement

With the rise in competitive, monetized, and fast-paced games, there's been a noticeable decline in story-driven RPGs that emphasize exploration, immersion, and creativity. Most modern titles prioritize multiplayer mechanics and monetization over core gameplay. This project aims to bring back the essence of traditional RPGs through a single-player experience built using Unity, focusing on design quality over profit models.

Some key problems addressed in this project are:

- **Lack of Narrative-Driven Games**

Many recent games focus heavily on online multiplayer and competition. Story-based single-player RPGs are increasingly rare, especially those that allow immersive world exploration and player choice.

- **Pay-to-Win and Monetization Models**

Modern games often rely on paid upgrades or in-game purchases. This project removes all forms of monetization, offering a complete experience without microtransactions or locked features.

- **Graphics-Hungry Game Engines**

Many modern games demand high-end hardware, limiting accessibility for players with modest systems. This project optimizes visual fidelity to provide immersive graphics while maintaining broad hardware compatibility.

- **Complex Learning Curve**

RPGs often involve intricate systems such as branching dialogue, stat progression, and open-world navigation. This complexity can overwhelm new players. The project simplifies gameplay mechanics without compromising depth, making the experience approachable.

1.3. Objectives

The primary objective of this project is to create a simple yet engaging RPG game. The objectives we hope to achieve within this project are as follows:

- To develop a basic RPG game in Unity with core RPG features: character movement, combat systems, quests, and enemy AI.
- To create a program that can read user input provided by players and respond appropriately.
- To build a simple but detailed environment that will be both educational and entertaining.

1.4. Scope and Limitations

1.4.1. Scopes

The scopes of this game include:

- Player character movement and animation
- Basic combat system and ranged attacks
- Inventory system with equipment slots
- NPC interaction system
- Quest tracking system
- Save/load functionality

All the above-mentioned components will be combined to form a complete game. Some of the other features and functionalities that the project might include are given below:

- Actual background music of the game
- Special sound effect. For example: character jumping, dying, etc.
- Actual world gravitational physics

1.4.2. Limitations

The limitations include:

- Single-player gameplay only
- Doesn't support WASD movement
- Fixed camera perspective

1.5. Methodology

The Agile methodology has been used during the development of this project, as it focuses more on iterative and flexible process of development, also dividing the projects into smaller pieces of steps known as sprints. Each sprint was carried out to focus on quick development of working, manageable and functioning part of the game system. The project was also carried out during development by communicating closely with the supervisor by gaining 3 insightful advice and feedback regularly. The main focus maintained during the project was finalizing the scope and requirements of the game development project as well as figuring out how long it would take to complete the entire project. Initially, the planning phase comprised of writing down the project objectives and the requirements. Then, strategic planning was done on how to achieve the plans set out to be carries during the project. On upcoming days, development of each component of the game system will go through proper testing.

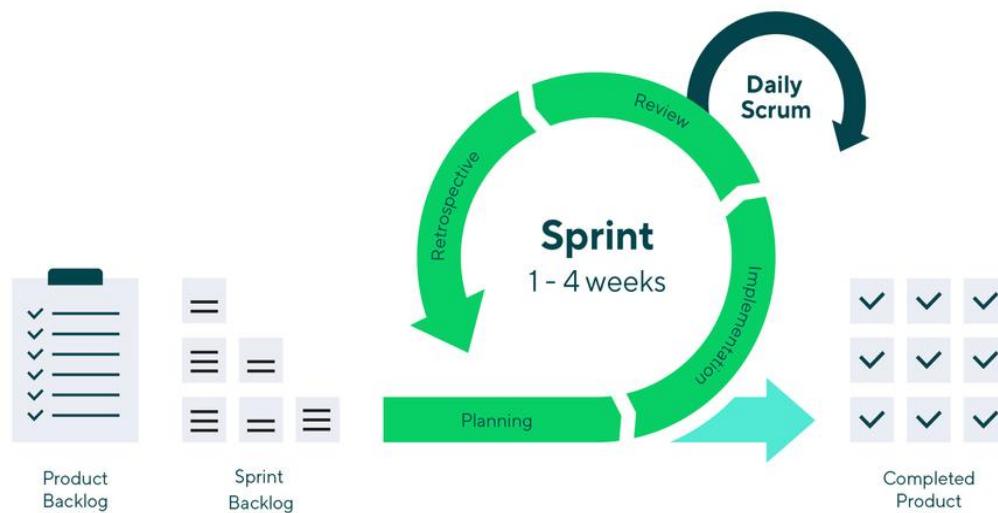


Figure 1: Agile Methodology of Software Development

1.6 Report Organization

Project Organization refers to the structured management and coordination of all tasks, activities, and resources involved in successfully delivering the project. A clear and logical organization is essential to ensure that every phase of the project is completed efficiently, on schedule, and according to the outlined objectives.

The **main report** is organized into six chapters, each with its respective headings and content, as outlined below:

1. **Chapter 1: Introduction**

Introduces the project, discusses the background, defines the scope and objectives, and outlines the problem that the project aims to address.

2. **Chapter 2: Literature Review**

Summarizes existing research and related work, explores similar systems and technologies, and identifies gaps that this project aims to fill.

3. **Chapter 3: Methodology and System Design**

Details the tools, techniques, and design methodologies adopted, including software requirements, system architecture, and conceptual models such as use case and sequence diagrams.

4. **Chapter 4: Analysis and Design**

Presents detailed analysis and design specifications. This includes data flow diagrams, entity-relationship diagrams, and interface designs, illustrating the structural and functional aspects of the system.

5. **Chapter 5: Implementation and Testing**

Describes the implementation process, tools and platforms used, module-wise functionality, and testing strategies. It also discusses unit testing, integration testing, and system testing with corresponding test cases and results.

6. **Chapter 6: Conclusion and Future Recommendations**

Summarizes the entire project's accomplishments, discusses its limitations, and suggests potential improvements or features that could enhance the project in future iterations.

Chapter 2: Background Study and Literature Review

2.1 Background Study

The background study involves exploring the fundamental theories, general concepts, and terminologies related to our project. The genre of our game project is fantasy adventure. Here player character can move according to users will. It's an open world. These games consist of very basic and simple mechanics:

- Movement: The player can move with the help of any input from the user as it is open world.
- Quests: The game spawns' various quests that the player has to solve by various means.
- Collectables: The game consists of various collectables like maps, potions and powerups that the player can collect for various purposes.
- Inventory system: Collectables that are collected are stored in inventory system.

2.2 Literature Review

The Role-Playing Game (RPG) genre has long been a cornerstone of the video game industry, offering players immersive narratives, character development, and strategic gameplay.[1] RPGs range from complex open-world experiences to minimalist indie titles, adapting to evolving player preferences and hardware capabilities. This literature review explores key RPG titles across various sub-genres, highlighting core mechanics, user experience design, and their relevance to our project's scope and objectives.[4]

- The Witcher 3: Wild Hunt (2015, CD Project Red)
This game is renowned for its narrative depth and morally complex choices. Its character progression system and branching dialogue trees contribute to high replay value. However, the game's length and dense storytelling may not cater to users seeking short, casual gameplay sessions. Its relevance lies in demonstrating how quality storytelling can elevate user engagement an area we aim to simplify for our casual target audience.
- Final Fantasy Series (1987–Present, Square Enix)
Spanning multiple sub-genres from turn-based combat to action-RPGs, this franchise demonstrates the RPG genre's adaptability. Notably, Final Fantasy XV's hybrid combat and open-world structure reflect modern trends toward real-time action. However, the

games often involve complex systems and lore-heavy narratives, making them less ideal for new or casual players.

- Genshin Impact (2020, miHoYo)

As a recent action-RPG, Genshin Impact showcases high-quality visuals, real-time combat, and a gacha monetization model. While it excels in polish and world-building, its heavy monetization and grind-centric progression systems present barriers to entry for casual players. These drawbacks further support our decision to exclude monetization from our own project, ensuring accessibility and fair gameplay.

Chapter 3: System Analysis

3.1. System Analysis

3.1.1. Requirement Analysis

i. **Functional Analysis:**

- Character navigation and movement
- Interaction with NPCs and objects in the game world
- Inventory system
- Quest system for tracking and managing tasks
- Basic combat mechanics (attacking monsters and boss)
- UI for health, inventory, and settings

ii. **Non-functional:**

- Optimized performance for smooth gameplay on mid-range PCs
- Modular and maintainable codebase using OOP and Unity best practices
- Accessibility through readable fonts, intuitive UI, and basic color contrast

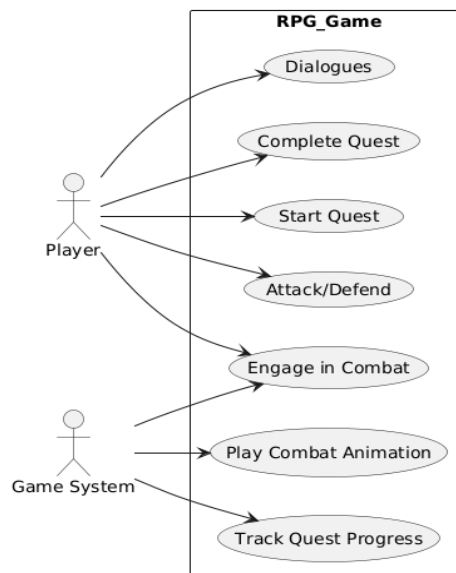


Figure 2 Use Case Diagram

3.1.2. Feasibility Study

- **Technical:**

- Game development using Unity (LTS version) with C# for scripting [3]
- Visual Studio as the development environment
- PC-only build targeting low to mid-range hardware

- a. **Operational:**

- Agile development cycle with iteration and playtesting
- Use of version control (Git) to track changes and maintain project history
- Play tested internally by friends and teachers

- b. **Economic:**

- All tools used are free and open source (Unity, Visual Studio)
- No licensing cost or third-party paid assets involved
- No monetization planned; game is for academic learning only
- Assets are custom-made or free-to-use, making the project cost-effective

- c. **Gantt Chart:**

Key activities	1st	2nd	3rd	4th	5th	6th	7th	8th	9th	10th	11th	12th
Project Initiation & Planning												
System Design												
Environment Setup & Prototyping												
Character & Movement System												
Integration												
Testing												
Documentation												

Figure 3:Gantt chart

3.1.3 Analysis (Object Oriented)

This project of developing a mobile game follows a object-oriented approach as the programming language used (C#) is an object oriented programming language.

i. Object modelling using Class and Object Diagrams

The Class and Object diagram for our project is as follows:

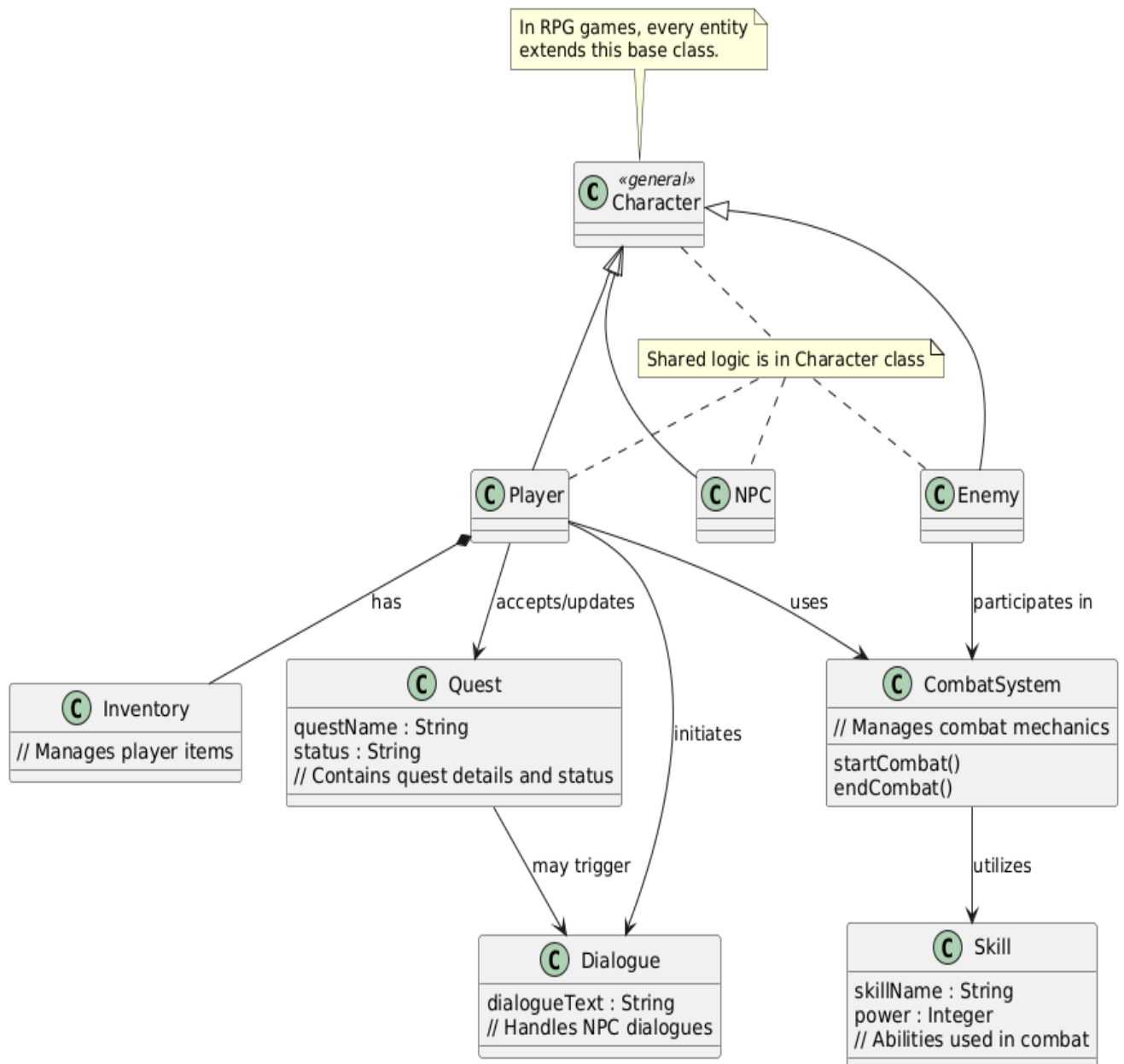


Figure 4: Use Case Diagram

ii. Dynamic modelling using Sequence Diagrams

The sequence diagram for our project is as follows:

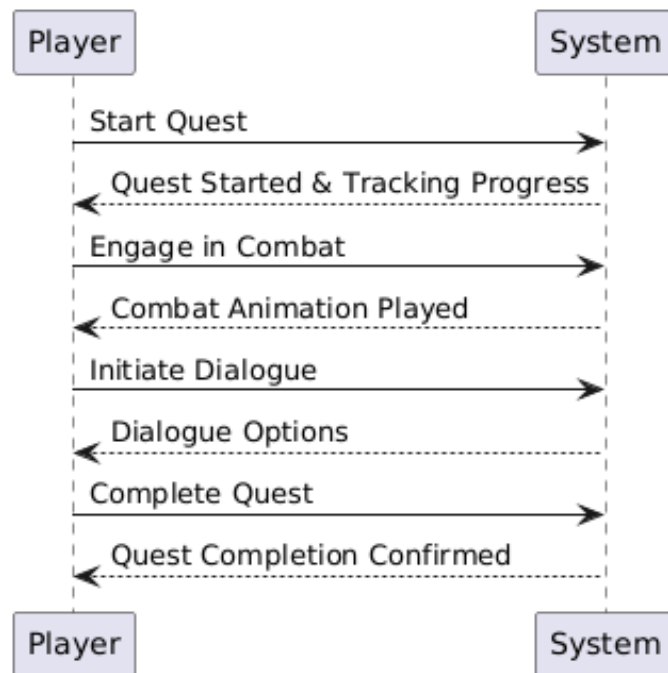


Figure 5: Sequence Diagram

iii. Process modelling using Activity Diagrams

Activity diagrams are simply an advanced form of flow-chart diagram that model the working of the system from one activity to another. An activity diagram for our project is as follows:

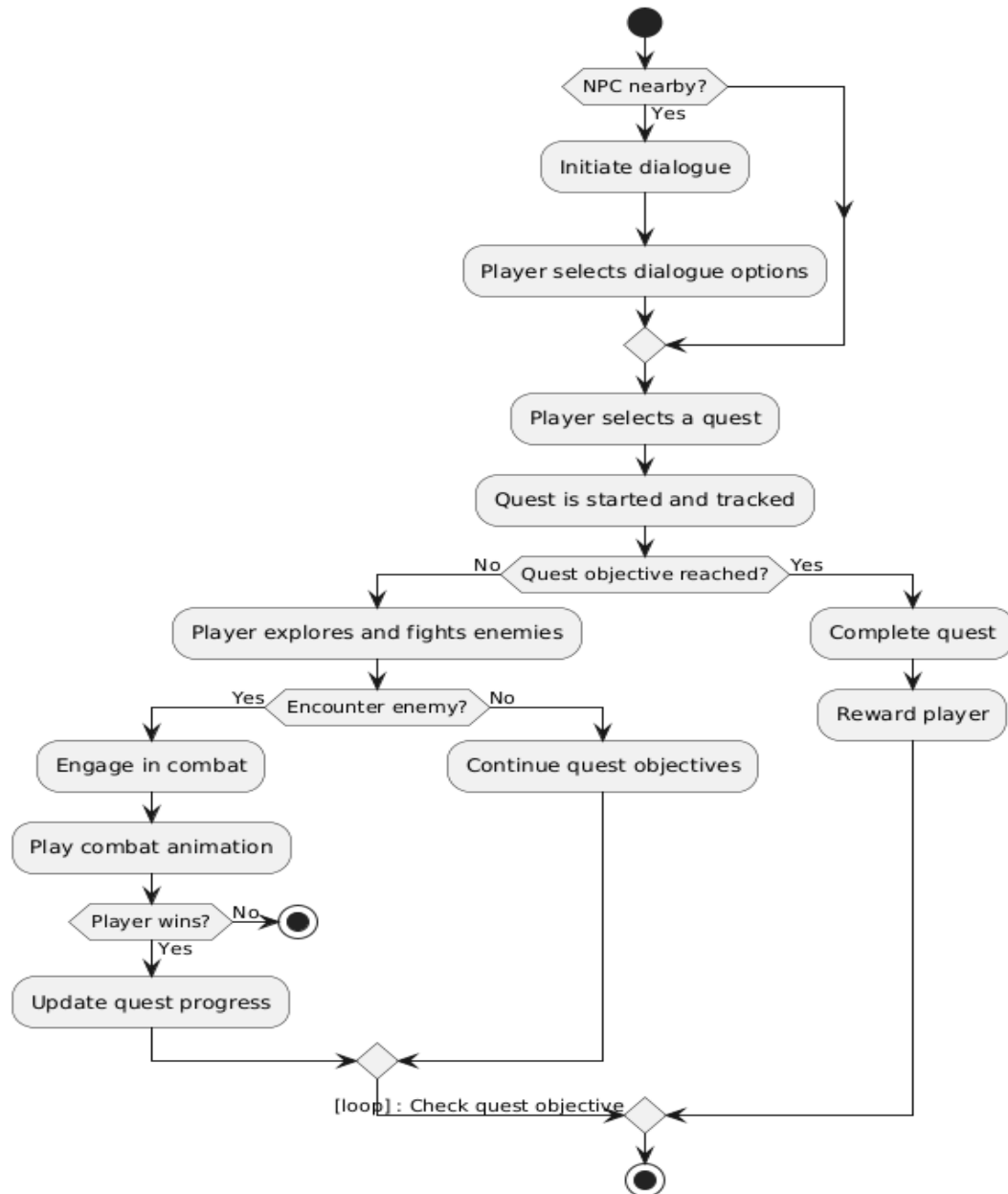


Figure 6: Activity Diagram

Chapter 4: System Design

4.1 Design (Object Oriented)

As discussed in the analysis chapter, the system design also follows an object-oriented approach.

- **Refinement of Class, Object and Activity Diagram**

Since this is a very basic project with not much complexity, the class diagrams, object diagrams and activity diagrams designed earlier are already refined enough to be applicable within the project.

4.2. Algorithm Details

4.2.1 Finite State Machines (FSM)

- **Purpose:**

Manage different states of gameplay elements such as quests, combat, and NPC behavior.

- **Description:**

Each game component (e.g., quests, combat system) transitions between defined states. For example, a quest may have states like NotStarted, InProgress, and Completed.

- **Example:**

When a player accepts a quest, the quest state transitions from NotStarted to InProgress. During combat, the state machine controls player and enemy actions like Idle, Attacking, and Defending.

4.2.2 Pathfinding (A* Algorithm)

- **Purpose:** Enable NPCs and enemies to navigate the game world efficiently.

- **Description:** The A* algorithm computes the shortest or optimal path around obstacles from a start to a target position.

- **Example:**

Enemies use A* to chase the player through a complex map while avoiding walls or traps.

4.2.3 Dialogue Trees

- **Purpose:**
Manage complex NPC-player conversations.
- **Description:**
Dialogues are represented as trees or graphs where nodes are dialogue lines and edges represent player choices leading to different branches.
- **Example:**
Player selects dialogue options which influence quest outcomes or NPC reactions.

4.2.4 Behavior Trees

- **Purpose:**
Control complex NPC and enemy AI behavior.
- **Description:**
Behavior trees organize AI decisions hierarchically using sequences, selectors, and decorators to handle behaviors like patrolling, chasing, attacking, or fleeing.
- **Example:**
An enemy patrol until the player is detected, then switches to aggressive attack behavior.

4.2.5 Quest Tracking Algorithm

- **Purpose:**
Monitor quest progress and update objectives dynamically.
- **Description:**
The system checks for specific player actions (e.g., item collection, enemy defeat) and updates quest states accordingly.
- **Example:**
When the player collects 5 herbs, the quest progress is updated, and upon completion, rewards are granted.

4.2.6 Combat Mechanics Algorithm

- **Purpose:** Resolve combat interactions and calculate damage.
- **Description:**
Damage calculation considers attacker stats, defender stats, random factors, and skill effects.
- **Example:**
$$\text{Damage} = (\text{AttackerAttackPower} - \text{DefenderDefense}) \times \text{RandomMultiplier} + \text{SkillBonus}$$

4.2.7 Event-Driven Architecture

- **Purpose:** Decouple game components by using events for communication.
- **Description:**
Game systems emit events such as `QuestAccepted`, `EnemyEncountered`, or `DialogueStarted`. Other components listen and react accordingly.
- **Example:**
When the player accepts a quest, the quest manager listens to the `QuestAccepted` event and updates the quest log UI.

Chapter 5: Implementation and Testing

5.1. Implementation

This report outlines how key algorithms and architectural patterns are implemented practically in the 3D RPG game, including quests, combat, NPC dialogues, and event-driven interactions. The focus is on integrating these algorithms into Unity or a similar game engine.

5.1.1. Tools Used

The tools we used for implementing this project are as follows:

a. **Unity**

Unity is a cross-platform game engine developed by Unity Technologies, that supports a variety of desktop, mobile as well as console platforms. It gives users the ability to create games and experiences in both 2D and 3D, and the engine offers a primary scripting API in C# using Mono, for both the Unity editor in the form of plugins, and games themselves, as well as drag and drop functionality.

b. **C#**

C# is a general-purpose high-level programming language based on object-oriented programming created by Microsoft. It is a popular and easy to learn as well as simple to use programming language having huge community support.

c. **Illustrator**

Adobe Illustrator is a vector graphics editor and design software developed and marketed by Adobe that allows for the creation of everything from single design elements to entire compositions.

d. **GitHub**

GitHub is one of the most widely used platforms for hosting code. It allows for version control, to track the changes made to the project and the state of progress of the project. It also allows to collaborate on projects from anywhere and monitor the activities of the team members on the project.

5.1.2. Implementation Details of Modules

i. Game Core Module

This is the heart of the game, responsible for all core mechanics and systems that make the world come to life.

- **Gameplay Mechanics:**
Handles player movement, jumping, running, and special abilities.
- **Combat and AI:**
Manages enemy behavior, pathfinding, combat interactions, hit detection, and damage calculation.
- **Physics & Collision:**
Controls gravity, character-environment interaction, and collision with terrain and props.

ii. UI Module

Responsible for every visual interface the player interacts with during gameplay.

- **Main Menu:**
Entry point to the game, allowing new games, load games and quit game.
- **Inventory & Equipment Screens:**
Displays player's items, gear, and stats.
- **Dialogue & Notification Popups:**
Handles on-screen conversations.

iii. Audio Module

Manages all sound-related features to enhance immersion.

- **Background Music and Sound Effects:**
Plays ambient themes for villages, forests, dungeons, and boss battles along with sound effects such as footsteps, sword swings, spell casting and environment effects.

iv. Asset Module

Controls efficient loading, management, and organization of all game assets.

- **3D Models & Textures:**
Manages character models, NPCs, creatures, weapons, and world scenery.
- **Animations:**
Controls animation clips for movement, combat combos, emotes, and environment elements.
- **Data Storage & Optimization:**
Efficiently handles asset memory usage to ensure smooth gameplay performance.

5.2 Testing

5.2.1 Unit Testing

Unit testing checks individual components separately to ensure they behave as expected.

Test Case 1: Player Movement Function

Objective: Verify that the player character responds correctly to movement commands.

Table 1: Player Movement Test

Test Input	Expected Output	Actual Output	Status
Right Click Mouse	Character moves to that spot	Character moves to that spot	Pass

Test Case 2: Obstacle Collision Function

Objective: Verify player detects an obstacle and collides with it.

Table 2: Obstacle Collision Test

Test Input	Expected Output	Actual Output	Status
Player hits obstacle	Player collides with an obstacle.	Player collides with an obstacle.	Pass

5.2.2 Integration Testing

Integration testing ensures that independently developed modules work together seamlessly.

Test Case 3: Player Data Persistence

Objective: Verify that the player and quest progress are correctly saved and loaded.

Table 3: Player Data Persistence Test

Test Input	Expected Output	Actual Output	Status
Save game & reload	Data loaded correctly	Haven't implemented yet	Ongoing

5.2.3 System Testing

System testing validates the game as a whole, ensuring all components integrate smoothly.

Test Case 4: Full Gameplay Run-Through

Objective: Verify that the game runs without crashes from start to finish.

Table 4: Full Game Playthrough Test

Test Input	Expected Output	Actual Output	Status
Play entire game	Smooth gameplay with no errors	Still some bugs in gameplay	Ongoing

5.3 Result Analysis

Result analysis involved systematically examining the outcome of each test to identify errors or performance issues. Issues were located, documented, and resolved promptly. Once stable, we built an APK file for Android and tested the game on multiple devices. Finally, we conducted a playtest with a small group of people and gathered feedback to fine-tune game balance, controls, and bug fixes.

Chapter 6: Conclusion and Future Recommendations

6.1 Conclusion

This 3D open-world RPG project aims to create a rich and immersive role-playing experience set in a vast, explorable fantasy world. Built using the C# programming language and the powerful Unity Game Engine, the game integrates 3D models, textures, and animations created with asset tools like Blender and Substance Painter. Core gameplay mechanics—such as questing, combat, and dialogue with non-player characters—are driven by well-structured algorithms and event-driven architecture to ensure smooth, responsive interactions.

The game encourages players to explore diverse environments, uncover secrets, and grow stronger by completing missions, collecting resources, and defeating enemies. By utilizing procedural systems like random enemy spawning and dynamic quest generation, the game keeps every session engaging and unpredictable. Overall, the implementation demonstrates robust game design principles while providing a scalable and enjoyable player experience.

6.2 Future Recommendations

- **Player Customization & Progression:**
Implement a deeper customization system including unlockable gear, player skill trees, mounts, and rare items as rewards for completed quests.
- **Multiplayer & Co-Op:**
Integrate optional multiplayer features allowing friends or strangers to join each other's worlds for cooperative adventures or competitive PvP combat.
- **Economy & Trading Systems:**
Add an in-game economy with marketplaces, crafting systems, and player-to-player trading to encourage immersion and strategy.
- **Online Progress & Leaderboards:**
Implement a global leaderboard and account system to encourage competitive or cooperative play across the player base.

References

- [1] V. Karamian, Building an RPG with Unity 2018: Leverage the power of Unity, Birmingham-mumbai: Packt, 2018.
- [2] U. Learn, "Learn game development with Unity," Unity Technologies, [Online]. Available: <https://learn.unity.com>. [Accessed 2 5 2025].
- [3] H. Ferrone, Learning C# by Developing Games with Unity 2019 Fourth Edition Code in C# and build 3D games with Unity, BIRMINGHAM - MUMBAI: Packt Publishing , 2019.
- [4] J. M. & P. Buttfield-Addison, Mobile Game Development with Unity build once, Deploy Anywhere, 1005 Gravenstein Highway North, Sebastopol: O'Reilly Media, Inc., 2017.

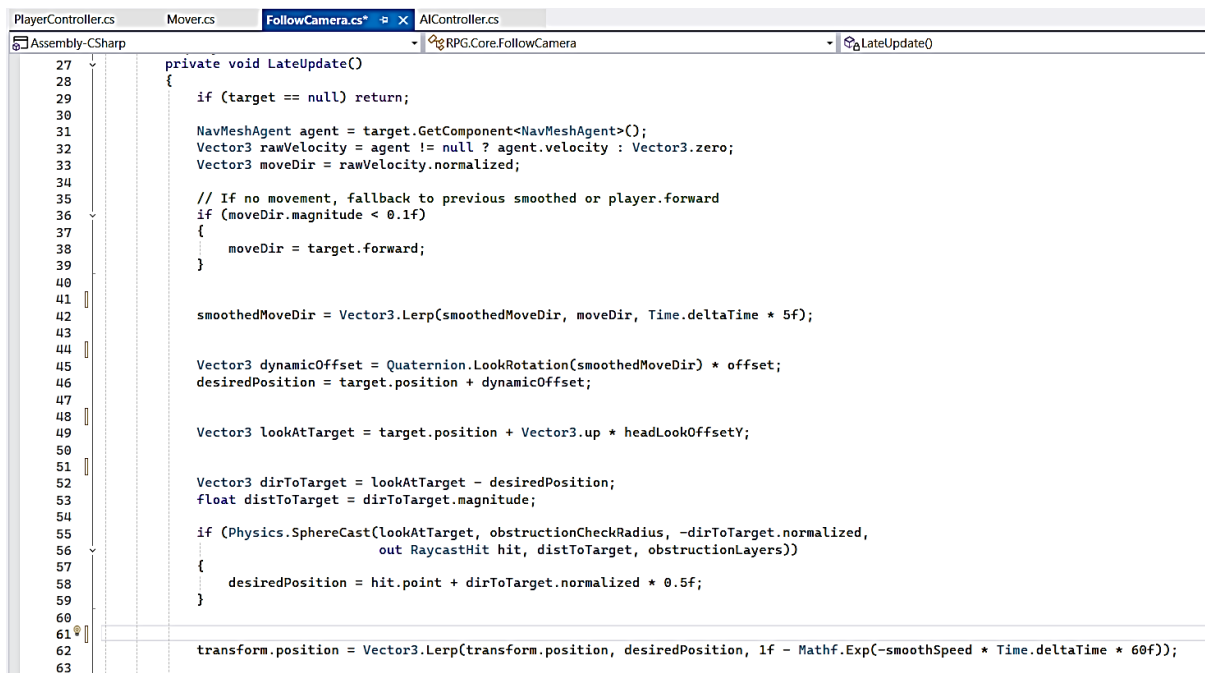
Appendix

Screenshots of Game:

Figure 7 : Creating Game in Unity

Figure 8: Enemy of Game

Algorithm:

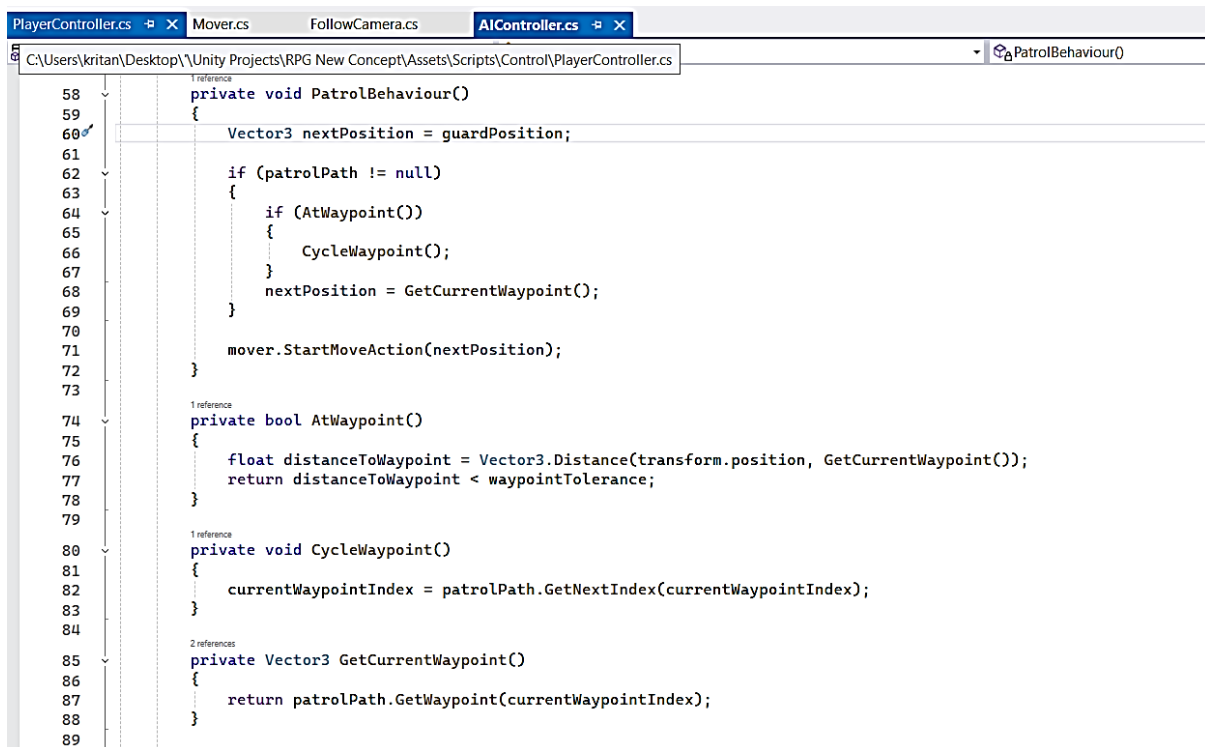


```

27 private void LateUpdate()
28 {
29     if (target == null) return;
30
31     NavMeshAgent agent = target.GetComponent<NavMeshAgent>();
32     Vector3 rawVelocity = agent != null ? agent.velocity : Vector3.zero;
33     Vector3 moveDir = rawVelocity.normalized;
34
35     // If no movement, fallback to previous smoothed or player.forward
36     if (moveDir.magnitude < 0.1f)
37     {
38         moveDir = target.forward;
39     }
40
41     smoothedMoveDir = Vector3.Lerp(smoothedMoveDir, moveDir, Time.deltaTime * 5f);
42
43     Vector3 dynamicOffset = Quaternion.LookRotation(smoothedMoveDir) * offset;
44     desiredPosition = target.position + dynamicOffset;
45
46     Vector3 lookAtTarget = target.position + Vector3.up * headLookOffsetY;
47
48     Vector3 dirToTarget = lookAtTarget - desiredPosition;
49     float distToTarget = dirToTarget.magnitude;
50
51     if (Physics.SphereCast(lookAtTarget, obstructionCheckRadius, -dirToTarget.normalized,
52         out RaycastHit hit, distToTarget, obstructionLayers))
53     {
54         desiredPosition = hit.point + dirToTarget.normalized * 0.5f;
55     }
56
57     transform.position = Vector3.Lerp(transform.position, desiredPosition, 1f - Mathf.Exp(-smoothSpeed * Time.deltaTime * 60f));
58 }
59
60
61
62
63

```

Figure 9 : Follow Camera C# Script



```

58 private void PatrolBehaviour()
59 {
60     Vector3 nextPosition = guardPosition;
61
62     if (patrolPath != null)
63     {
64         if (AtWaypoint())
65         {
66             CycleWaypoint();
67         }
68         nextPosition = GetCurrentWaypoint();
69     }
70     mover.StartMoveAction(nextPosition);
71 }
72
73
74 private bool AtWaypoint()
75 {
76     float distanceToWaypoint = Vector3.Distance(transform.position, GetCurrentWaypoint());
77     return distanceToWaypoint < waypointTolerance;
78 }
79
80 private void CycleWaypoint()
81 {
82     currentWaypointIndex = patrolPath.GetNextIndex(currentWaypointIndex);
83 }
84
85 private Vector3 GetCurrentWaypoint()
86 {
87     return patrolPath.GetWaypoint(currentWaypointIndex);
88 }
89

```

Figure 10: Enemy AI Controller