

**Tribhuvan University**

**Faculty of Humanities and Social Sciences**

**“Adventure of the Mystic Isles: 2D Game”**

**A PROJECT REPORT**

**Submit to**

**Department of Computer Application Kathmandu**

**BernHardt College**

**Bafal, Kathmandu Nepal**

***In partial fulfilment of requirements for the Bachelors in Computer Application***



**Submitted by:**



|  |  |
| --- | --- |
| **Name** | **Roll No.** |
| **Prashanna Pun Magar** | **46102020** |

**November, 2023**

**Under the supervision of**

**Ram Babu Mahato**

# ACKNOWLEDGEMENT

I extend my heartfelt gratitude to **Mr. Rambabu Mahato**, our esteemed Coordinator and Project Supervisor, who continuously helped and guided me through different problem faced during this whole Project titled “Adventure of the Mystic Isles:2D Game”.

I am profoundly thankful to **Kathmandu Bernhardt College** for its unwavering support, provision of essential resources, and fostering an ideal environment that greatly facilitated the development of this project. Furthermore, I wish to express our sincere appreciation to Tribhuvan University for affording us the opportunity to showcase and assess our programming skills through this project.

Prashanna Pun Magar 46102020

# ABSTRACT

"Adventure of the Mystic Isles:2D Game " is an action-adventure game that immerses players in a world filled with magic, mystery, and heroism. The game's core elements combine physical challenges, exploration, and a compelling storyline, providing players with a rich and immersive gaming experience. As players navigate through diverse and visually stunning environments, from lush forests to desolate wastelands, they face mystic creatures not seen in the normal world, defeating them to gain valuable points used to upgrade their character's stats. This character progression system adds a sense of growth and empowerment to the gameplay. Action-adventure games like this offer players a welcome escape from the stresses of real life, allowing them to embark on thrilling virtual adventures where their choices shape the outcome of the world. In this project report, we explore the development of "Adventure of the Mystic Isles" and its significance in the context of action-adventure gaming, alongside a review of well-known titles such as "The Legend of Zelda," "God of War," "Assassin's Creed," "Ghost of Tsushima," and "Tomb Raider," exemplifying the rich diversity of experiences this genre has to offer.

***Keywords: Action-adventure, mystery,*** ***The Legend of Zelda, God of War, Ghost of Tsushima, Tomb Raider.***

**TABLE OF CONTENT**

[ACKNOWLEDGEMENT iv](#_Toc152844810)

[ABSTRACT v](#_Toc152844811)

[LIST OF ABBREATIONS vii](#_Toc152844812)i

[LIST OF FIGURES ix](#_Toc152844813)

[LIST OF TABLES x](#_Toc152844814)

[CHAPTER 1: INTRODUCTIONS 1](#_Toc152844815)

[Introduction 1](#_Toc152844816)

[1.2 Problem Statements 1](#_Toc152844817)

[1.3 Objectives 2](#_Toc152844818)

[1.4 Scopes and limitation 2](#_Toc152844819)

[1.4.1 Scope 2](#_Toc152844820)

[1.4.2 Limitation 2](#_Toc152844821)

[1.5 Development Methodology 2](#_Toc152844822)

[1.6 Report Organization 4](#_Toc152844823)

[CHAPTER 2: BACKGROUND STUDY AND LITERATURE REVIEW 5](#_Toc152844824)

[2.1 Background Study 5](#_Toc152844825)

[2.2 Literature Review 7](#_Toc152844826)

[CHAPTER 3: SYSTEM ANALYSIS AND DESIGN 9](#_Toc152844827)

[3.1 System Analysis 9](#_Toc152844828)

[3.1.1 Requirement analysis 9](#_Toc152844829)

[I. Functional requirements 9](#_Toc152844830)

[II. Non-functional requirements 10](#_Toc152844831)

[3.1.2 Feasibility Analysis 10](#_Toc152844832)

[3.1.3 Object Oriented Approach 11](#_Toc152844833)

[3.1.4 Dynamic modelling: state and sequence diagram 12](#_Toc152844834)

[3.1.4.1 Sequence diagram 12](#_Toc152844835)

[3.1.4.2 State Diagram 13](#_Toc152844836)

[3.1.5 Process modelling: Activity diagram 14](#_Toc152844837)

[3.2.1 Object Modelling: Class Diagram 16](#_Toc152844838)

[3.2.2 Component diagram 17](#_Toc152844839)

[3.2.3 Deployment diagram 18](#_Toc152844840)

[3.3 Algorithms 18](#_Toc152844841)



[CHAPTER 4: IMPLEMENTATION AND TESTING 24](#_Toc152844842)

[4.1 Implementation 24](#_Toc152844843)

[4.1.1 Tools Used 24](#_Toc152844844)

[4.1.2 Implementation Details of modules 24](#_Toc152844845)

[4.2 Testing 26](#_Toc152844846)

[4.2.1 Test cases for unit testing 26](#_Toc152844847)

[4.2.2 Test cases for System testing 27](#_Toc152844848)

[CHAPTER 5: CONCLUSION AND FUTURE RECOMMENDATION 29](#_Toc152844849)

[5.1 Lesson learnt/outcomes 29](#_Toc152844850)

[5.2 Conclusion 29](#_Toc152844851)

[5.3 Future recommendations 29](#_Toc152844852)

[REFERENCES](#_Toc147437122)

**APPENDICES**

# LIST OF ABBREATIONS



**2D** Two Dimension

**AI** Artificial Intelligence

**FPS** Frames Per Second

**GUI** Graphical User Interface

**IDE** Integrated Development Environment

**OOP**  Object Oriented Program

**RPG**  Role-Playing Games

**SDK** Software Development Kit

**UI**  User Interface

# LIST OF FIGURES

[Figure 1.1 Waterfall Model 3](#_Toc147437150)

[Figure 3.1 Use Case Diagram 10](#_Toc147437151)

[Figure 3.2 Class Diagram 11](#_Toc147437152)

[Figure 3.3 Object Diagram 12](#_Toc147437153)

[Figure 3.4 Sequence Diagram 13](#_Toc147437154)

[Figure 3.5 State Diagram 14](#_Toc147437155)

[Figure 3.6 Activity Diagram 15](#_Toc147437156)

[Figure 3.7 Class Diagram 17](#_Toc147437157)

[Figure 3.8 Component Diagram 18](#_Toc147437159)

# LIST OF TABLES

Table 4.1: Unit Testing …………………………………………………………………...26

Table 4.2: System Testing……………………………………..…………………………27

# CHAPTER 1: INTRODUCTIONS

## Introduction

Action-adventure games are a hybrid genre that combines elements of both action and adventure games. They typically involve exploration, puzzle-solving, and combat. What adventure games are about exploration and solving situational problems, action games are fast-paced and require quick reflexes and hand eye coordination. Action-adventure games combine these genres by engaging both physical and conceptual challenges.[1]

“Adventure of the Mystic Isles” is an Action-adventure game. The game is a thrilling action-adventure filled with magic, mystery, and a compelling story of heroism, sacrifice. Players will navigate a richly detailed world, uncover hidden secrets, and ultimately decide the fate of the world. The hero is lost and has to find his way out of the mystic Isles, by fighting different mystic creatures never seen in the normal world.

The player has to defeat different enemies to gain points. The points can then be used to upgrade the players stats. The game feature diverse and visually stunning environments, from lush forests to desolate wastelands. The settings contribute to the overall immersion and provide a backdrop for the game's events. Many action-adventure games feature character progression systems. Players can level up their characters, acquire new skills, and upgrade equipment as they advance in the game. This progression adds a sense of growth and empowerment to the gameplay.

So, when creating Action-adventure Games, the key is making sure that player sicks around. The story should be immersive and interesting for the player to get hooked on. Video Game is specially played to gain the sense of adventure. With action-adventure games, this is especially true, the adventure you take on as a player, has its highs and lows, ups and down and is highly dependent on the role you are playing. Action-adventure game give the gamers a break from stress of life, allowing us to focus in on a world that may be better that our own. Essential powers chosen by the player helps them to become the best version possible through gaming.[2]

## 1.2 Problem Statements

Just about all of the most popular video games have some requirement for problem- solving and / or critical thinking? This promotes adaptability and cognitive flexibility. These are really important skills to have in any kind of problem-solving task.

Studies have indicated that compared to non-gamers, experienced gamer are better at: Task objects; keeping track of several objects simultaneously; filtering out irrelevant information; switching from task to task; detecting changes in visual layouts; and 3D mental rotation.

## Objectives

The major objectives of the project are:

* To create accessible gameplay using collision detection algorithm and enemy follow algorithm
* To improve cognitive skills

## 1.4 Scopes and limitation

### 1.4.1 Scope

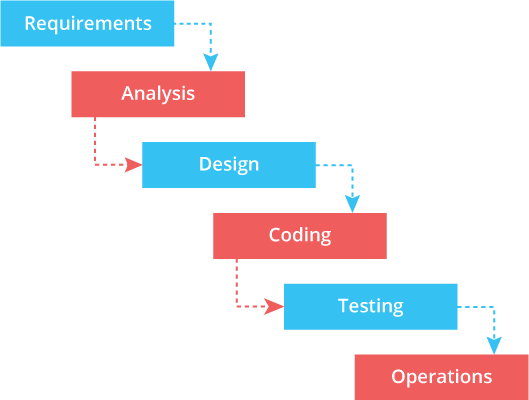
When people talk about video games development, they often read that that rate of project abandonment is very high and it’s true; it’s fun, attractive, with the right functionality, and with enough marketing to be seen among the many that are published every day.

### 1.4.2 Limitation

* May cause epilepsy
* Does not support color blind mode

## 1.5 Development Methodology

Waterfall model has been used to develop the game “Adventure of the Mystic Isles”. The software development activity is divided into different phase and each phase consist of a series of tasks and has different objectives. It is divided into phases and output of one phase becomes the input of the next phase. It is mandatory for a phase to be completed before the next phase start. There is no overlapping in the waterfall model.



**Figure 1.1 Waterfall Model**

The **Waterfall model** is a linear sequential approach to project management that is used to complete projects within a given deadline, cost, and quality. It is a breakdown of project activities into linear sequential phases, where each phase depends on the deliverables of the previous one and corresponds to a specialization of tasks. The approach is typical for certain areas of engineering design. In software development, it tends to be among the less iterative and flexible approaches, as progress flows in largely one direction through the phases of requirements, analysis, design, coding, testing and operations.

The Waterfall model is simple and easy to understand and use. It works well for smaller and low-budget projects where requirements are very well understood. However, the Waterfall model has some disadvantages. It is less flexible and less adaptable to changes in requirements. It is difficult to estimate the time and cost of each stage accurately. It is also difficult to make changes once a stage is completed. In conclusion, the Waterfall model is a useful approach for projects with well-defined requirements and a fixed scope. It is less suitable for projects with changing requirements or where flexibility is required.

Waterfall model was used during the making of the project because the approach for projects had well defined requirements and a fixed scope.

## 1.6 Report Organization

This report consists of five chapters, with different main topic and their sub topic

**Chapter 1:** discusses the need of the game along with the problem statement and objectives of the project. Here, the tools that are used while completing the project are also introduced.

**Chapter 2:** analyses the existing system along with background study and literature review of other systems.

**Chapter 3:** summarizes the system design along with the requirement analysis and feasibility analysis.

**Chapter 4:** explain the tools that are used on our project. The modules and the development model used are also explained in this part. And unit testing and system testing are also included.

**Chapter 5:** discusses the conclusion of how the project is accomplished, its findings and many more. Recommendation for future enhancements of the project is also discussed here. In conclusion, this chapter overview’s purpose of doing this project including its scopes and objectives.

# CHAPTER 2: BACKGROUND STUDY AND LITERATURE REVIEW

## 2.1 Background Study

Action-adventure games is the combination of both action and adventure games. An action-adventure game can be defined as a game with a mix of elements from an action game and an adventure game. Action-adventures require many of the same physical skills as action games, but also offer a storyline, numerous characters, an inventory system, dialogue, and other features of adventure games. They are faster-paced than pure adventure games, because they include both physical and conceptual challenges.

An action game is a video game genre that emphasizes physical challenges, including hand–eye coordination and reaction time. The genre includes a large variety of sub-genres, such as fighting games, beat 'em ups, shooter games, and platform games. Multiplayer online battle arena and some real-time strategy games are also considered action games. The action genre includes any game where the player overcomes challenges by physical means such as precise aim and quick response times. Action games can sometimes incorporate other challenges such as races, puzzles, or collecting objects, but they are not central to the genre. Players may also encounter tactical and exploration challenges, but these games first-and-foremost require high reaction speed and good hand–eye coordination. The player is often under time pressure, and there is not enough time for complex strategic planning. In general, faster action games are more challenging. Action games may sometimes involve puzzle solving, but they are usually quite simple because the player is under immense time pressure. [3]

An adventure game (rarely called a quest game [citation needed]) is a video game genre in which the player assumes the role of a protagonist in an interactive story, driven by exploration and/or puzzle-solving. The genre's focus on story allows it to draw heavily from other narrative-based media, such as literature and film, encompassing a wide variety of genres. Most adventure games (text and graphic) are designed for a single player, since the emphasis on story and character makes multiplayer design difficult. The term "adventure game" originated from the 1970s text computer game Colossal Cave Adventure, often referred to simply as Adventure, which pioneered a style of gameplay which many developers imitated and which became a genre in its own right. The video game genre is therefore defined by its gameplay, unlike the literary genre, which is defined by the subject it addresses: the activity of adventure.[4]

Brett Weiss cites Atari's Superman (1979) as an action-adventure game, with Retro Gamer crediting it as the "first to utilize multiple screens as playing area". Mark J.P. Wolf credits Adventure (1980) for the Atari VCS as the earliest-known action-adventure game. The game involves exploring a 2D environment, finding and using items which each have prescribed abilities, and fighting dragons in real-time like in an action game. Muse Software's Castle Wolfenstein (1981) was another early action-adventure game, merging exploration, combat, stealth, and maze game elements, drawing inspiration from arcade shoot 'em ups and maze games and war films (such as The Guns of Navarone). [5]

According to Wizardry developer Roe R. Adams, early action-adventure games "were basically arcade games done in a fantasy" setting. Tutankham, debuted by Konami in January 1982, was an action-adventure released for arcades. It combined maze, shoot 'em up, puzzle-solving and adventure elements, with a 1983 review by Computer and Video Games magazine calling it "the first game that effectively combined the elements of an adventure game with frenetic shoot 'em up gameplay. "It inspired the similar Time Bandit (1983). Action Quest, released in May 1982, blended puzzle elements of adventure games into a joystick-controlled, arcade-style action game, which surprised reviewers at the time. [6]

## 2.2 Literature Review

Several games had to be review before making “Adventure of the Mystic Isles”. different game form different time was used while reviewing the games:

**The Legend of Zelda: The tears of the kingdom** is an action-adventure game developed and published by Nintendo for the Nintendo Switch in 2023. The sequel to The Legend of Zelda: Breath of the Wild (2017), Tears of the Kingdom retains aspects from it, including the open world of Hyrule, which has been expanded to allow for more vertical exploration. The player controls Link as he searches for Princess Zelda and fights to prevent the Demon King from destroying the world.[7]

**God of War: Ragnarök** is an action-adventure game developed by Santa Monica Studio and published by Sony Interactive Entertainment. It was released worldwide on November 9, 2022. Loosely based on Norse mythology, the game is set in ancient Scandinavia and features series protagonist, Kratos, and his now teenage son, Atreus. Concluding the Norse era of the series, the game covers Ragnarök, the eschatological event which is central to Norse mythology and was foretold to happen in the previous game after Kratos killed the Aesir god Baldur. [8]

**Assassin’s Creed** **Mirage** is a 2023 action-adventure game developed by Ubisoft Bordeaux and published by Ubisoft. The game is the thirteenth major installment in the Assassin's Creed series and the successor to 2020's Assassin's Creed Valhalla. While its historical timeframe precedes that of Valhalla, its modern-day framing story succeeds Valhalla's own. Set in 9th-century Baghdad during the Islamic Golden Age—in particular during the Anarchy at Samarra—the story follows Basim Ibn Ishaq (a character first introduced in Valhalla), a street thief who joins the Hidden Ones to fight for peace and liberty, against the Order of the Ancients, who desire peace through control. The main narrative focuses on Basim's internal struggle between his duties as a Hidden One and his desire to uncover his mysterious past. [9]

**Ghost of Tsushima** is a 2020 action-adventure game developed by Sucker Punch Productions and published by Sony Interactive Entertainment. The player controls Jin Sakai, a samurai on a quest to protect Tsushima Island during the first Mongol invasion of Japan. Jin must choose between following the warrior code to fight honorably, or using practical but dishonorable methods of repelling the Mongols with minimal casualties. The game features a large open world which can be explored either on foot or on horseback. When facing enemies, the player can choose to engage in a direct confrontation using Jin's katana or to become a legendary warrior known as "the Ghost" by using stealth tactics to assassinate opponents. [10]

**Shadow of the Tomb Raider** is a 2018 action-adventure video game series game developed by Eidos-Montréal and published by Square Enix's European subsidiary. The game is the sequel to Rise of the Tomb Raider and is the twelfth mainline entry in the Tomb Raider series, as well as the third and final entry of the Survivor trilogy. In this game the player assumes the role of archaeologist-adventurer Lara Croft, who navigates through a series of ancient ruins and tombs in search of an ancient artefact. The game follows Lara Croft's intense origin story from a young woman to a hardened survivor, as she explores ancient ruins and fights off deadly enemies in gorgeous worlds. [11]

# CHAPTER 3: SYSTEM ANALYSIS AND DESIGN

## 3.1 System Analysis

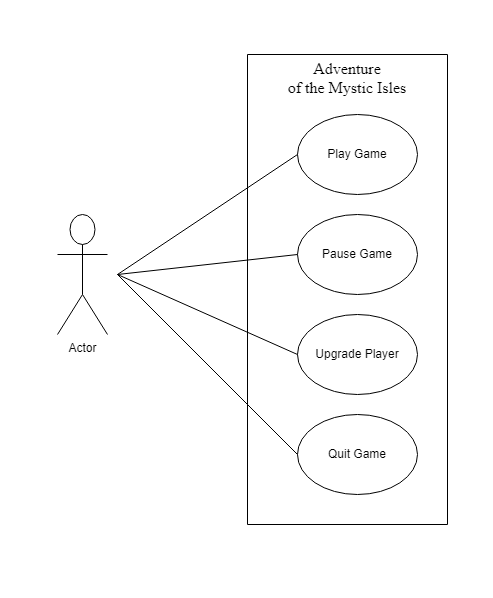
System analysis is a process of gathering and interpreting facts, diagnosing problems and the information about the project. It is done to recommend improvements on the system. It is a problem-solving activity that requires intensive communication between the system users and system developers. It is an important phase of any system development process. The system is studied to the minutest detail and analyzed. The system analyst plays the role of the interrogator and dwells deep into the working of the present system. The system is viewed as a whole and the input to the system is identified. The outputs from the organizations are traced to the various processes. System analysis is concerned with becoming aware of the problem, identifying the relevant and decisional variables, analyzing and synthesizing the various factors and determining an optimal or at least a satisfactory solution or program of action. The conclusion is an understanding of how the system functions. This system is called the existing system.

## 3.1.1 Requirement analysis

Requirement analysis results in the specification of operational characteristics of software: indicates interface of software with other system elements and establishes constrains the software must meet. The requirement analysis is mainly categorized into two types functional and non-functional:

### Functional requirements

* Player Controls: The user can control the player character using the keyboard input. The player can move left, right and jump along with a shooting button using mouse click.
* Enemies and Obstacles: The application game contains a variety of enemies which the player can either avoid or defeat. It includes various types of enemies that can attack, push and shield from the player.
* User Interface: The game has a user interface that allows the user to start, pause and exit the game



**Figure 3.1 Use Case Diagram**

### Non-functional requirements

* Performance: The game application runs smoothly and responsively even on the low-end devices. The game is compatible with different operating systems, devices, and screen resolutions to ensure a wide reach and accessibility for users. No special permission for admin is required
* Portability: The game is easily portable to different platforms without having to do a significant change to code.
* User experience: The game provides a user-friendly interface that is easy to understand and navigate. It allows users to customize aspects of their gaming experience, such as graphics settings, audio preferences, and control options

## 3.1.2 Feasibility Analysis

1. Technical feasibility

Technical feasibility assesses the current resources (hardware and software) and technologies, which are required to accomplish user requirements.

1. Economic feasibility

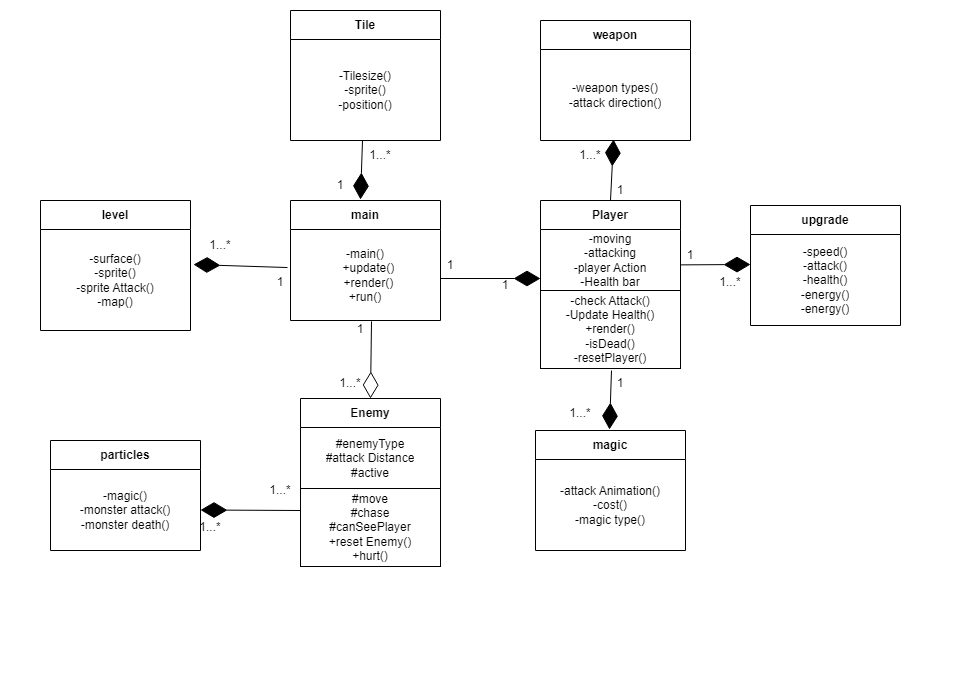
It is economically feasible because the project is created using free tools and application like PyCharm.

1. Operational feasibility

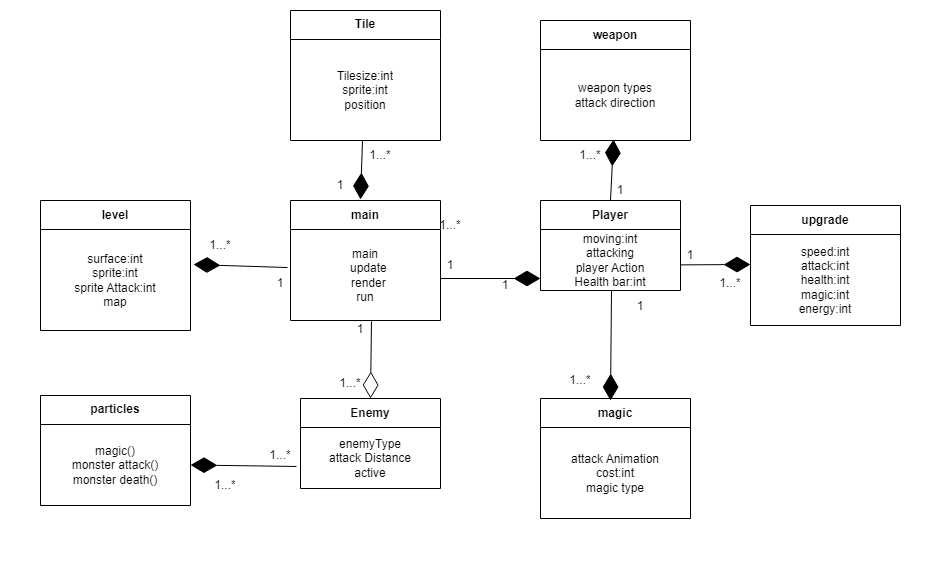
Since the game is user-friendly and it provides tutorials to get started while playing the game, no external tutorials are necessary.

## 3.1.3 Object Oriented Approach

Class diagram represents the static view of an application. Class diagram describes the attributes and operations of a class and also the constraints imposed on the system. It shows a collection of classes, interfaces, associations, collaborations and constraints.



**Figure 3.2 Class Diagram**

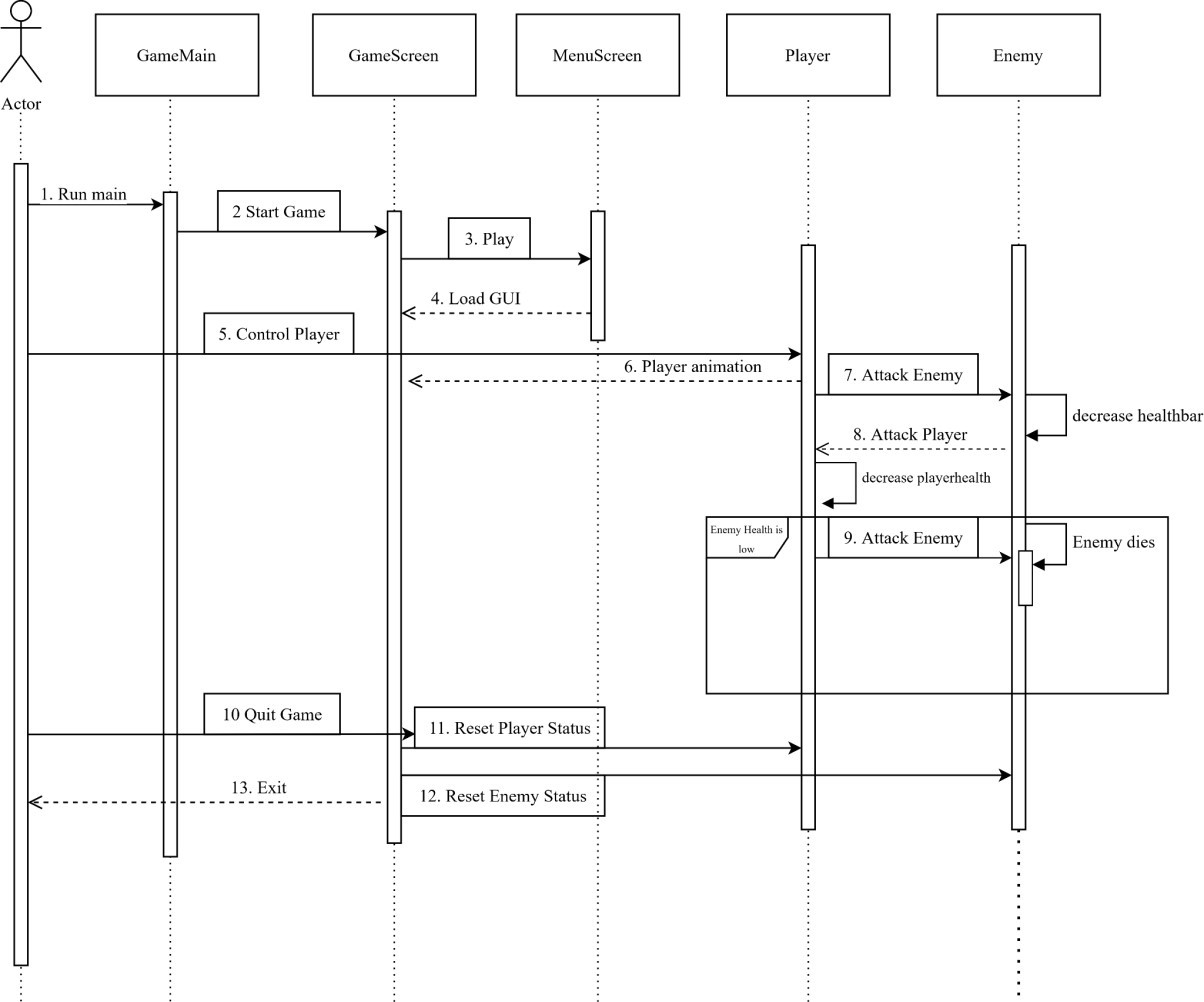


**Figure 3.3 Object Diagram**

## 3.1.4 Dynamic modelling: state and sequence diagram

### 3.1.4.1 Sequence diagram

Sequence diagrams are interaction diagram that detail how operations are carried out. It is a type of UML (Unified Modeling Language) diagram that depicts interactions between objects or components in a software system over a specific period. It illustrates how various entities in the system communicate with each other to achieve a particular function or scenario.

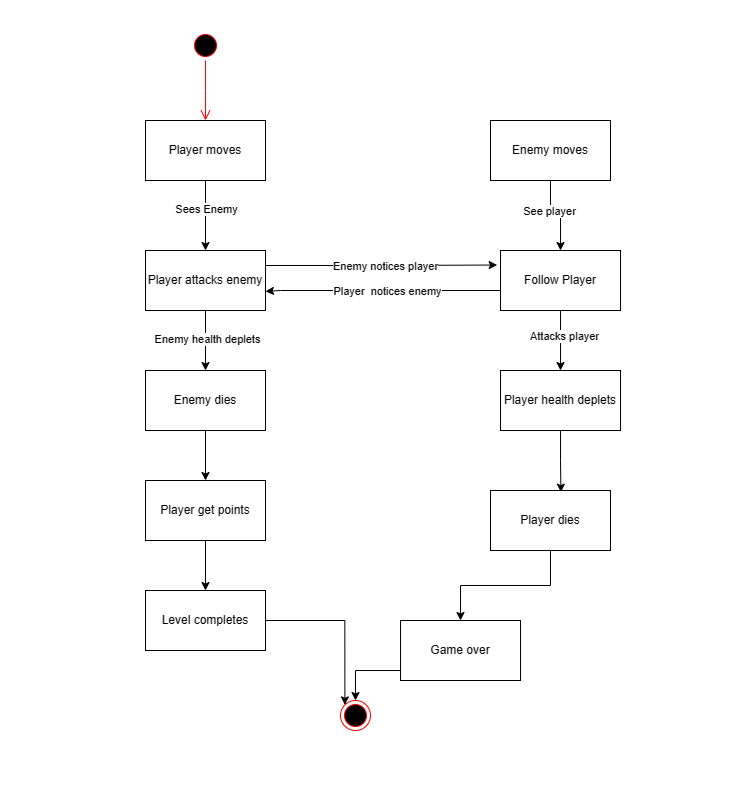


**Figure 3.4 Sequence Diagram**

Actor starts/ loads the game and the system load the UI. The actor can give input to move the hero, they can pause, resume and quit the game, with the help of different UI.

## 3.1.4.2 State Diagram

A state diagram, also known as a state machine diagram, is a visual representation used in system modeling to depict the various states that an object or system can exist in and the transitions between those states. State diagrams are part of the Unified Modeling Language (UML) and are widely used in software engineering, control systems, and various other domains to model the behavior of complex systems.

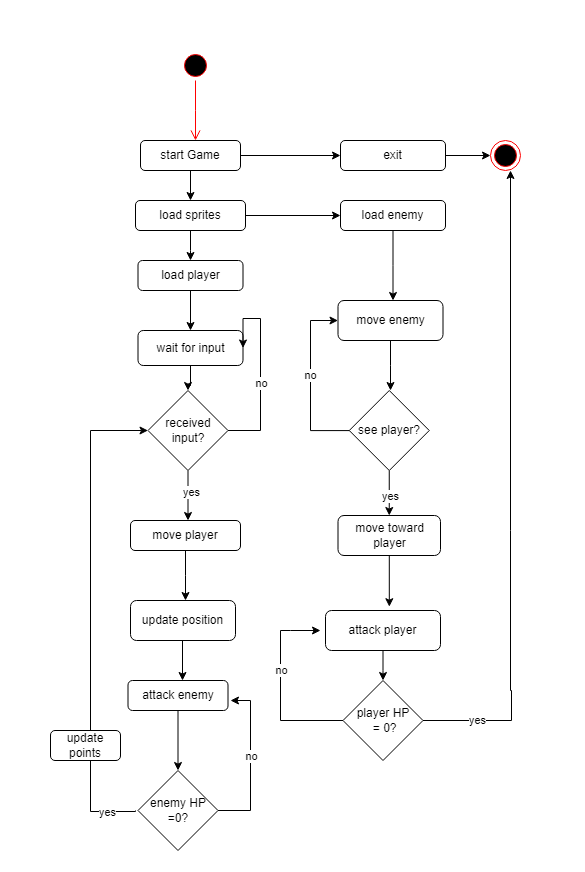


**Figure 3.5 State Diagram**

## 3.1.5 Process modelling: Activity diagram

An activity diagram is a type of Unified Modeling Language (UML) diagram used to visualize the flow of activities or actions within a system, process, or workflow. It is particularly useful for modeling the dynamic aspects of a system, focusing on how different activities are performed and how they relate to each other. The activity can be described as an operation of the system.

It consists of activities that are made up of smaller actions. It is an advancement of a flowchart that contains some unique capabilities.



**Figure 3.6 Activity Diagram**

In this activity diagram, the following activities or actions are performed:

* Start Game: Represents the activity of starting the game. It is the initial action triggered when the player begins playing the game.
* Load Sprites: Represents the activity of loading a specific sprite of the game. It involves retrieving the sprite data and setting up the animation frames. It loads NPC, player, object and all tiles.
* Process Input: Represents the activity of handling player input. It involves capturing user input from the keyboard or controller and translating it into game commands, such as moving the player character or performing actions.
* Update Game State: Represents the activity of updating the game state. It includes updating the positions, velocities, and states of game objects, handling collisions, checking for interactions between objects, and managing game logic.
* Check Game Over: Represents the activity of checking if the game has reached a game-over condition. It involves evaluating certain conditions, such as the player character's health reaching zero, character hit obstacles or a specific objective not being met, to determine if the game should end.

Display Points: Represents the activity of displaying the points player earned by killing enemies. It is used to increase the player’s status.

## 3.2.1 Object Modelling: Class Diagram

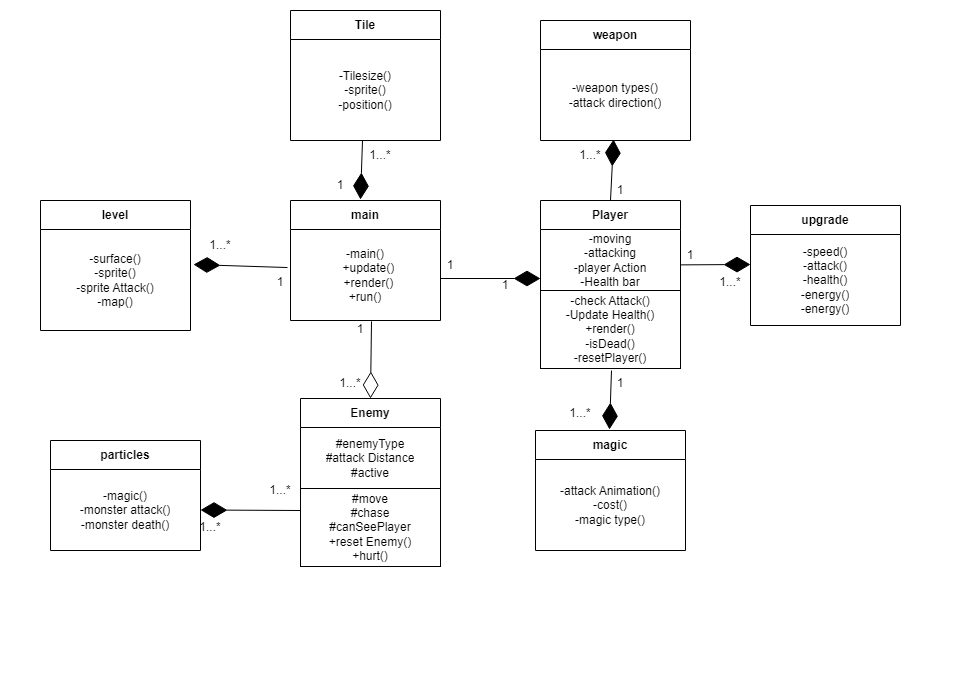
Generalization is the refinement of a class into more refined classes. It allows the developer to model objects into hierarchical structure based on their similarities. the class being refined is classed super-class and the refined versions of is are called sub-classes. Each sub-class inherits the attributes and operations from their super-class. Methods and attributes can then be refined and the sub-class specific attributes and operations. A discriminator is a variable of enumeration type, which indicate which property of an object is being abstracted. The most important use of inheritance is the conceptual simplification it makes trough the reduction of independent features in the system. Generalization refers to the relationship among classes and inheritance refers to the method of reusing attributes and operations in the hierarchic. By definition a feature with the same name and subclass can override a superclass feature.

Class Diagram:

At the top level, the main class that encapsulates the overall game logic. It contains methods for starting, updating, rendering and running the game.

The ‘Player’ class represents the game's main character. It has attributes such as health bar, action, moving and attacking. It also has methods for jumping, attacking, and interacting with objects in the game world.

The ‘Enemy’ class represents the various adversaries that the player will encounter. It has attributes like type, attack-distance, strength, and AI behavior. The 'Enemy' class also has methods for movement and attacking the player.



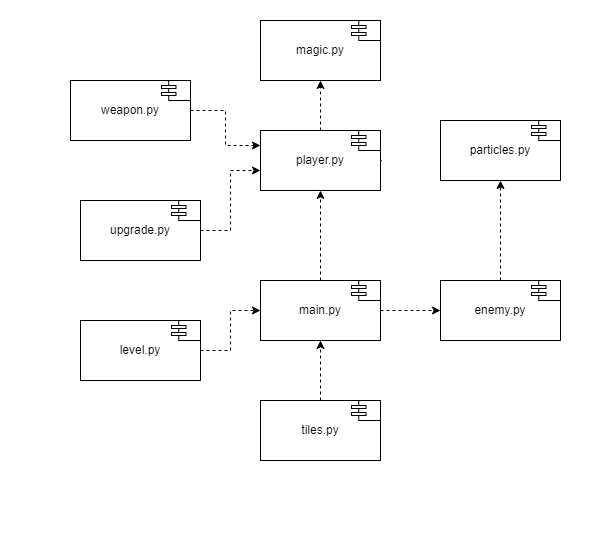
**Figure 3.7 Class Diagram**

## 3.2.2 Component diagram

Component diagram can be described as a special kind of UML as it shows static implementation view of a system. They are used to visualize the organization and relationships among components in a system. The purpose of the component diagram is to construct executables by using forward and reverse engineering. And, they are used during the implementation phase of an application.

Different components like level design which handles the creation and management of game levels, including the placement of platforms, obstacles, and collectibles. Similarly with the enemy it encompasses the intelligence system responsible for controlling enemy characters, their movement patterns, and behavior within the game.Asset Management component deals with the organization, loading, and management of game assets such as sprites, sounds, music, and other resources and likewise.







**Figure 3.8 Component Diagram**

## 3.2.3 Deployment diagram

Deployment diagrams represent the physical deployment of the system, showcasing how the software components and resources are distributed across different hardware nodes or environments.

## 3.3 Algorithms

Algorithms are step-by-step sets of instructions or procedures designed to perform specific tasks or solve particular problems. They are the foundational building blocks of computer programs and are used in various fields, including computer science, mathematics, data science, and engineering. Algorithms are essential for data processing, decision-making, problem-solving, and automation. Algorithms used in this project.

* **Sensor Range check algorithm:**

A hero detection algorithm, often referred to as a "Sensor Range Check Algorithm," is a fundamental component of many games, simulations, and applications where objects or characters (such as heroes or players) need to detect the presence of other objects or characters within a certain range or distance.

**Algorithm Steps:**

1. Initialize an empty list to store the detected objects or characters.

2. Iterate through each object or character in the environment.

3. For each object or character:

- Calculate the distance between the detecting object and the current object or character using a distance formula (e.g., Euclidean distance). The distance formula often depends on the coordinate system being used (2D or 3D).

4. Check if the calculated distance is less than or equal to the specified detection range. If it is, the object or character is within the detection range.

5. If the object or character is within the detection range, add it to the list of detected objects or characters.

6. Continue this process for all objects or characters in the environment.

7. Once the iteration is complete, the list of detected objects or characters contains all entities that are within the detection range of the detecting object.

**Pseudocode in python:**

|  |
| --- |
| Function detectObjectsInRange  (detectingObjectPosition, objectsInEnvironment, detectionRange):  detectedObjects = empty list    for each object in objectsInEnvironment:  distance = calculateDistance(detectingObjectPosition, object.position)  if distance <= detectionRange:  detectedObjects.append(object)    return detectedObjects |

This pseudocode outlines a basic hero detection algorithm. The actual implementation may vary depending on the programming language and specific requirements of the application. Additionally, optimizations and data structures (e.g., spatial partitioning) can be applied to improve the efficiency of the algorithm, especially in scenarios with a large number of objects to detect.

* **Basic Enemy Follow Algorithm:**

A basic enemy follow algorithm is used in game development to make enemies or AI-controlled characters follow and chase a target, such as the player character. This algorithm allows the enemies to pursue the target within certain constraints, such as speed, line of sight, and obstacles.

**Algorithm Steps:**

1. Calculate the direction vector from the enemy to the target:

- `direction\_vector = target\_position - enemy\_position`

2. Normalize the direction vector to maintain a constant speed:

- `normalized\_direction = normalize(direction\_vector)`

3. Calculate the new position of the enemy based on the normalized direction and the enemy's maximum movement speed:

- `new\_enemy\_position = enemy\_position + normalized\_direction \* max\_speed \* delta\_time`

- `delta\_time` is the time elapsed since the last update and is used to ensure consistent movement across different frame rates.

4. Check if the enemy has a line of sight to the target:

- You may use raycasting or other techniques to determine if any obstacles obstruct the view between the enemy and the target.

5. If there's no line of sight to the target or if obstacles block the path, the enemy may either stop moving or take evasive action (e.g., move around obstacles). The exact behavior depends on the game's requirements.

6. Update the enemy's position to the new position calculated in step 3.

**Pseudocode in python:**

|  |
| --- |
| def follow\_target(enemy\_position, target\_position, max\_speed, delta\_time):  direction\_vector = target\_position - enemy\_position  # Normalize to maintain constant speed  normalized\_direction = normalize(direction\_vector)  new\_enemy\_position = enemy\_position + normalized\_direction \* max\_speed \* delta\_time  # Check for line of sight and obstacles  if has\_line\_of\_sight(enemy\_position, target\_position) and not obstacle\_in\_path(enemy\_position, target\_position):  return new\_enemy\_position  else:  # Either stop moving or take evasive action  return enemy\_position |

In a real implementation, you would need to define the `normalize`, `has\_line\_of\_sight`, and `obstacle\_in\_path` functions based on your game's specific requirements and engine capabilities. Additionally, you would call this function repeatedly to update the enemy's position in each frame of the game.

* **Collision Detection:**

Collision detection is a type of algorithm is a fundamental technique used in computer graphics, game development, and simulations to determine whether two objects or entities are intersecting or colliding with each other based on their bounding boxes. Bounding boxes are simplified geometric shapes (often rectangles or axis-aligned boxes) that approximate the shape and size of more complex objects.

**Algorithm Steps:**

1. Define the bounding boxes for the two objects:

- Each bounding box is typically defined by its position (usually the top-left corner) and its dimensions (width and height).

2. Calculate the minimum and maximum coordinates for each bounding box:

- For each bounding box, determine its minimum x-coordinate (left edge), maximum x-coordinate (right edge), minimum y-coordinate (top edge), and maximum y-coordinate (bottom edge).

3. Check for overlap along the x-axis:

- Determine if the maximum x-coordinate of one bounding box is greater than or equal to the minimum x-coordinate of the other bounding box, and vice versa. If this condition is met, there is potential overlap along the x-axis.

4. Check for overlap along the y-axis:

- Determine if the maximum y-coordinate of one bounding box is greater than or equal to the minimum y-coordinate of the other bounding box, and vice versa. If this condition is met, there is potential overlap along the y-axis.

If there is overlap along both the x-axis and y-axis, the bounding boxes are considered to be colliding.

**Pseudocode in python:**

|  |
| --- |
| def is\_collision(box1, box2):  # Extract coordinates and dimensions of the two bounding boxes  x1, y1, width1, height1 = box1  x2, y2, width2, height2 = box2  # Calculate minimum and maximum coordinates for each bounding box  min\_x1, max\_x1, min\_y1, max\_y1 = x1, x1 + width1, y1, y1 + height1  min\_x2, max\_x2, min\_y2, max\_y2 = x2, x2 + width2, y2, y2 + height2  # Check for overlap along the x-axis and y-axis  if (max\_x1 >= min\_x2 and min\_x1 <= max\_x2) and (max\_y1 >= min\_y2 and min\_y1 <= max\_y2):  return True  else:  return False |

You would call the `is\_collision` function with the bounding boxes of two objects to determine if they are colliding. If the function returns `True`, the objects are colliding; if it returns `False`, they are not. This basic bounding box collision detection is efficient and widely used in real-time graphics and game development.

# CHAPTER 4: IMPLEMENTATION AND TESTING

## 4.1 Implementation

### 4.1.1 Tools Used

* 1. PyCharm

"PyCharm" is a popular integrated development environment (IDE) used by many Python developers. PyCharm is developed by JetBrains and is known for its features and tools tailored specifically for Python development.

* 1. Pygame

Pygame is a popular Python library used for creating 2D games and multimedia applications. It provides a set of modules and functions that simplify various aspects of game development, including graphics, audio, input handling, and more.

* 1. Tiled

Tiled is a widely-used open-source map editor that is specifically designed for creating 2D tile-based maps for games and other interactive applications. It was used to build tile-based environments/map for the game.

* 1. MS Excel

Microsoft Excel is a powerful spreadsheet application developed by Microsoft as part of the Microsoft Office. It is used by Tiled to save data like object id in the excel sheet available in the map of the game.

* 1. GitHub

GitHub is a web-based platform and service that provides a wide range of tools and features for version control, collaboration, and software development. It is was used to track changes in source code during software development.

## Implementation Details of modules

1. Movement Controller

This controller is used to move all the elements in the game. Every surface walkable comprises of a map which is used to detect collision as well as detect the location and position of the character. Whenever user inputs any movement keys (w, a, s, d, enter, arrow key, space and esc) movement controller is responsible to handle the movement.

1. Camera Controller

This module is responsible for all of the camera movements in the game. There can be multiple cameras in game to show various parts of the game window. The camera is locked on to the hero, following him so the player can only focus on him and follow his story.

1. Character State

This module is responsible to handle all the characters in the game. Character features such as Health, Damage, Movement Speed, all are managed by this module. Character State determines the state of the characters in the current frame in the game.

1. Enemy State

This module is responsible to handle all the enemies in the game. Enemy features such as Health, Damage, Movement Speed, player sensor all are managed by this module. Enemy State determines the state of the enemies in the current frame in the game.

1. Implementation of Algorithm

* hero detection algorithm (sensor range check algorithms)

sensor range check algorithms are commonly used to determine whether objects, characters, or game entities are within the detection or interaction range of a game sensor or character. These algorithms are crucial for implementing features such as interaction triggers, enemy detection, and more.

* collision detection algorithm (Bounding Box Collision Detection)

Bounding Box Collision Detection is used to detect collision between heroes and enemies’ attacks. Bounding boxes are simple axis-aligned rectangles (2D) or cubes (3D) that enclose objects. Collision detection is performed by checking if bounding boxes intersect. It is a fast and efficient method but may lead to false positives or negatives in cases where objects have irregular shapes.

* Basic Enemy Follow Algorithm

This algorithm makes an enemy character move toward the hero (player) in a 2D game, you can use this algorithm that calculates the direction vector from the enemy to the hero and then updates the enemy's position based on that direction vector.

## 4.2 Testing

### 4.2.1 Test cases for unit testing

Unit testing is a software testing technique that focuses on evaluating individual units or components of a software application in isolation to ensure they function correctly. The goal of unit testing is to verify that each unit of code, such as functions, methods, or classes, performs as expected and produces the correct output for a given set of inputs.

**Table 4.1: Unit Testing**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| s.no | Function | Input | Expected outcome | Result | Status |
| 1 | Move character | Arrow keys and a, s, d, w | Character movement | Character moved successfully | true |
| 2 | Change weapons | Q | Change between different weapons | Weapons changes | true |
| 3 | Change spells | E | Change between attacking and healing spell | Change spells | true |
| 4 | Attack enemy using weapons | Space key | Attacks using weapon | Attacks using selected weapons | true |
| 5 | Use magic | Left shift key | Use magic | Use selected magic | true |
| 6 | Pause | ESC key | pause | Pause game and open character menu | true |
| 7 | resume | ESC key | resume | Close menu and resume the game | true |

### 4.2.2 Test cases for System testing

System testing is employed to confirm that all components of a software application work together harmoniously, meet specified requirements, and operate seamlessly within the overall system architecture, ensuring the system's functionality, stability, and adherence to user expectations.

**Table 4.2: System Testing**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| s.no | Function | Input | Expected outcome | Result | Status |
| 1 | Trigger movement animations | Arrow keys and W, A, S, D | Walking animation | Animation triggers successfully | true |
| 2 | Take/dealt damage |  | Take damage | Health reduces | true |
| 3 | Die when health 0 |  | Die | Success | true |
| 4 | Gain points on enemy defeated |  | Increase points | Points increases | true |
| 5 | Improve stat using points |  | Increase stat | Stat increases | true |
| 6 | Self-Heal using healing magic |  | Hp increases | Heals heroes’ loss HP | true |
| 7 | Damage enemy using magic |  | Decrease enemies hp if hit | Damage enemy and eventually die | true |
| 8 | Sounds |  | Play different sound | Sound should play during different events | true |
| 9 | Enemy notices hero inside its notice radius |  | Notice hero and chase to attack | Enemy should notice hero inside it notice radius and chase hero | true |

# CHAPTER 5: CONCLUSION AND FUTURE RECOMMENDATION

## 5.1 Lesson learnt/outcomes

There have been several improvements in our programming language and writing skill as well as our time management skills while doing this project. I conclude that this project has helped me gain more knowledge about the topic that I researched while doing my project. A lot was learned about time management as the project had to be submitted before the deadline along with the documentation Due to time constraints, I could now add more facilities to. Although it is expectedly good, some new features to this system could be added in the upcoming days to make it more user friendly and efficient.

## 5.2 Conclusion

The description of the background and the context of the project was thoroughly researched. The purpose, scope, applicability, and requirement specifications of the system have been accurately explained. The author has included features and operations in detail including screen layouts and the limitations on which the project is being developed. Finally, the system is implemented and tested according to test cases. After the development of the system finally, it was tested and the views about results were exchanged. After testing, the limitations of the existing system were discussed. In conclusion, tools like Photoshop, PyCharm, Tiled and programming language like python was used in the development of the system.

## 5.3 Future recommendations

The applicability and usage of this project can be increased by enhancing the system based on the limitations. Entertainment is now a basic need. Every Home has their own personal Computer. There are many people who need entertainment. So, with the help of this game we can deliver a good service to customer who wants some entertainment into. This helps to uplift the mental wellbeing of people. New and effectives modules can be also added from time to time.

In future, new game modes and more maps can be added to the system where player can interact with even more objects. More interesting events can be added to the game with improved animation and graphics. Multiplayer can be introduced so people can battle and come out with awesome strategist.

# REFERENCES

[1] Rollins, A.; Morris, D. (2000). Game Architecture and Design. Coriolis Ed.

[2] Luban, Pascal (6 December 2002). "Designing and Integrating Puzzles in Action-Adventure Games". Game Developer. Retrieved 12 August 2022.

[3] Matt Barton; Bill Loguidice (2009). Vintage games: an insider look at the history of the most influential games of all time. Boston: Focal Press/Elsevier. pp. 239–255. ISBN 978-0-240-81146-8. Archived from the original on 17 January 2023.

[4] Manuel, Rob (5 February 2013). "How adventure games came back from the dead". PC World. Archived from the original on 21 July 2016. Retrieved 3 August 2016.

[5] Weiss, Brett. Classic Home Video Games, 1972–1984: A Complete Reference Guide. McFarland & Co. pp. 119, 126

[6] "Overseas Readers Column – Konami and Stern Pact On "Tutankham" Video" (PDF). Game Machine (in Japanese). No. 194. Amusement Press, Inc. 15 August 1982. p. 26. Archived (PDF) from the original on 31 January 2020.

[7] https://en.wikipedia.org/wiki/The\_Legend\_of\_Zelda

[8] https://en.wikipedia.org/wiki/God\_of\_war

[9] https://en.wikipedia.org/wiki/Assassin\_creed

[10] https://en.wikipedia.org/wiki/Ghost\_of\_Tsushima

[11] https://en.wikipedia.org/wiki/Tomb\_Rider

**APPENDICES**



Appendices A: Gameplay



Appendices B: Using Magic



Appendices C: Using Weapon



Appendices D: Pause/Stats Menu

