National College of Ireland

Honours in Computing

Software Development

2024/2025

Karl Miller

X21522489

X21522489@student.ncirl.ie

Tales of Atheria

Technical Report

Contents

Executive Summary

Tales of Atheria is a side story in the world known as Atheria, these tales focus on Important figures within the world. This particular story is about a famous Blacksmith called Selum P.Unker in the time before his breakthrough within the Metal work industry. However, in this current time the world is in an uncertain spot. The void is ravaging the many mines within Atheria causing problems in the city of Quarrine where Mining is their main trade and while Selum isn't an experienced fighter he wants to save his town from the decline it faces and restore it to its previous glory! The goal of the game is to venture down into the dangerous depths of the Quarrine Mines and gather all the resources you can while Fighting whatever monsters may stop you, If you successfully manage to retrieve resources from the mines you might be able to help your allies in town or create some new gear!

1.0 Introduction

1.1. Background

I've taken up this project as since i was a child I've had a big interest in video games and later a large interest in RPG and Roguelikes as they have lots or replayability and immersive stories that I could play repeatedly. I made an immersive story in my head over the years and while I'm not focusing on that with this project because the scope is too large so I used one of my smaller plots in the same world, The countless hours of thinking about this story will help me develop it properly as I know how it will look and play. I feel as though working on something that I'm truly interested in will motivate me to put my all into this project and make something I can be proud of.

1.2. Aims

I intend to create a Videogame with rougelike elements such as randomised floors and item drops from enemies. A Rougelite is a type of Role-Playing Game (RPG) that is described as a Dungeon crawl through procedurally generated levels with Grid based movement and most Rougelikes have permadeath of the player character, however most modern Rougelikes have other styles meshed into them too. For example, Slay the Spire is a rougelike with card game mechanics or The Binding of Isaac being a Rougelike and a Twin-stick shooter. My game Tales of Atheria is a Rougelike and a Village management game where you can upgrade the buildings in the town to unlock better items in the shops. I also want to leave a lot of the gameplay up to player discovery and let them figure things out since a lot of games hold your hand and I feel it's more rewarding to play and learn.

1.3. Technology

RPGMaker MV: It is my main software for developing the game when it comes to programming the game and all the mechanics and gameplay. RPGMaker is mainly for creating RPG but many you can create just about anything with it if you know how to use it properly. Another thing RPGmaker can do is allow players to use their own sprites for creating their own games but it also has templates that come with a copy of the software. Something that makes it a software I love is that its super easy to understand how to create things within it but it has enough power to create incredibly complex events and mechanics.

Aesprite: Aesprite is a Pixel art software that I am using for creating and modifying sprites for tiles such as Weapon or Ore sprites, Aesprite is also very easy to use but its one of the best pixel art softwares out there as it has many different tools that can allow users to create beautiful pieces of art or just use it as a simple software, It also is extremely accessible to users as its UI is easy to take in.

Github: Github is a developer platform that allows people to create repositories for file storage and development as people can push and pull files easily acting as a bridge of sorts to easily grab files and work on projects. It's also a great safety net in case file corruption or anything of that sort

1.4. Structure

This document will contain all the necessary Functional requirements that are associated with the design of my game, these requirements will outline key components that will have to be in the game and working without issues for the game to work as intended. Further down they are showcased with Use case diagrams which will outline how a User/Actor interacts with the system.

2.0 System

2.1. Requirements

Ones labelled as Functional requirements are required and are needed to be shown off in the final showcase of the game, A second section will be added for the Non-Functional requirements. The Non-Functional requirements are features that I would like to have in the game, but they aren't needed for the game to function.

2.1.1. Functional Requirements

The Requirements listed here will be ordered from Highest to Lowest priority.

Functional Requirements:

1. Walking around:
    a. Highest Priority
        i. Allows the user to move around within the game.
        ii. This is the Highest priority as if the user cannot move around then the user cannot play the game and the game cannot continue
2. Opening the Inventory/Menu:
    a. High Priority
        i. Allows the player to pause the game, check their resources they have collected and save their progress
        ii. This is right below the priority of walking around as if the player cannot use this feature, then they cannot do a lot of important functions in the game.
3. Saving/Loading progress:
    a. High Priority

i. Saving/Loading the game is an extremely important feature that needs to exist and function properly.
ii. This is one of the higher priorities as if the player cannot save the game, then they would repeatedly lose their progress.

4. Level progression:
    a. High Priority
        i. The user moves down a level in the mines by interacting with the ladder
        ii. If this does not function, then the main gameplay cannot be accessed by the player
5. Exiting the Mines:
    a. High Priority
        i. The user Interacts with the ladder to exit the mines and return to the surface
        ii. If this doesn’t function, then the user wouldn’t be able to return to the town without dying and respawning
6. Interaction:
    a. High Priority
        i. The user uses the “Enter” key to interact with near every object in the game, NPCs, Rocks, Chests, Moving areas.
        ii. If this is not functional then the player cannot do most functions the game requires
7. Battles:
    a. Medium Priority
        i. Battles will happen randomly while the User is within the cave as they walk around.
        ii. While it's not as important as the above, If battles don't work then the player will walk around open caves without a challenge.

Non-functional requirements:

1. Fleeing battles:
    a. Medium Priority
        i. While this feature is not required it would be better if It was to be added as it gives the player freedom within battles if they want to escape if they are low on HP or if they just don’t want to fight
        ii. The player would progressively have a higher chance at escape the more attempts they do in battle, and it resets when there is a successful escape attempt
2. Items in battle:
    a. Medium Priority
        i. This would also give players another chunk of freedom in battle as they can use items in various ways to assist them within battle

ii. If this was to not exist it could make fights difficult as they would not be able to heal etc, which would be a huge problem in boss fights.

3. Opening Chests:
   a. Medium Priority
      i. This one is needed for the gameplay experience but not needed for the game to function as a whole, this will be how players find items like Potions etc.
      ii. Having the player find items allows for more freedom in playstyles and show a little more professionalism with development

2.1.1.1. Requirement 1 <Walking>

U

2.1.1.2. Priority

t

2.1.1.3. Requirement 1 Walking

Use case for walking via User interaction

2.1.1.4. Priority

The user must press the arrow keys to move around within the overworld, While the user is within the overworld they should be able to walk around with their own free will. If the user cannot do that, they would be unable to play the game and as such this would have the Highest priority of all functions within the game.

2.1.1.5. Use Case

Name

Walking use case

Number

Use case 001

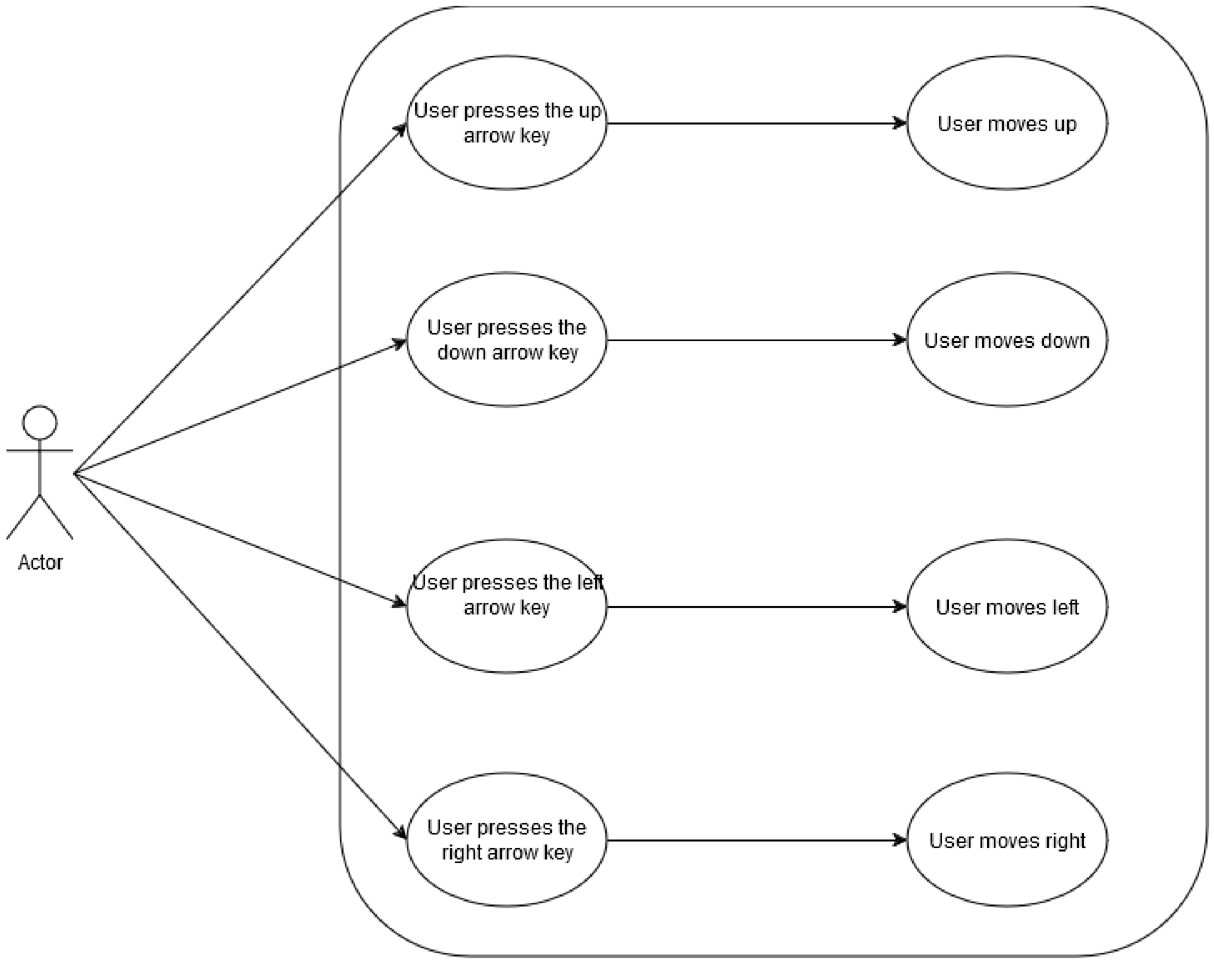
Scope

The scope of this use case is to showcase the user Walking around

Description

This use case describes the user hitting one of the arrow keys and in response the character in game should move corresponding to the arrow key pressed by the user

Use Case Diagram


User presses the up arrow key
User moves up
User presses the down arrow key
User moves down
Actor
User presses the left arrow key
User moves left
User presses the right arrow key
User moves right

Flow Description

Precondition

The system is in a wait state on the “overworld”

Activation

This use case starts when the user presses an arrow key

Main flow

1. The user presses any arrow key (Up, Down, Left, Right)

2. The system responds to the button press and moves the character in the corresponding direction.

Exceptional flow

1. The system crashes.
2. The system autosaves
3. The game reopens

Termination

The system recognises the game has been closed

Post condition

The system goes into a wait state awaiting the player to press another arrow key

2.1.1.6. Requirement 2 Saving/Loading

Use case for Saving/Loading via User interaction

2.1.1.7. Priority

Saving/Loading the game is an extremely important feature that allows the user to continue playing from a save point they have created and as such it's one of the Highest priority features that needs to be functional

Name

Saving/Loading use case

Number

Use case 002

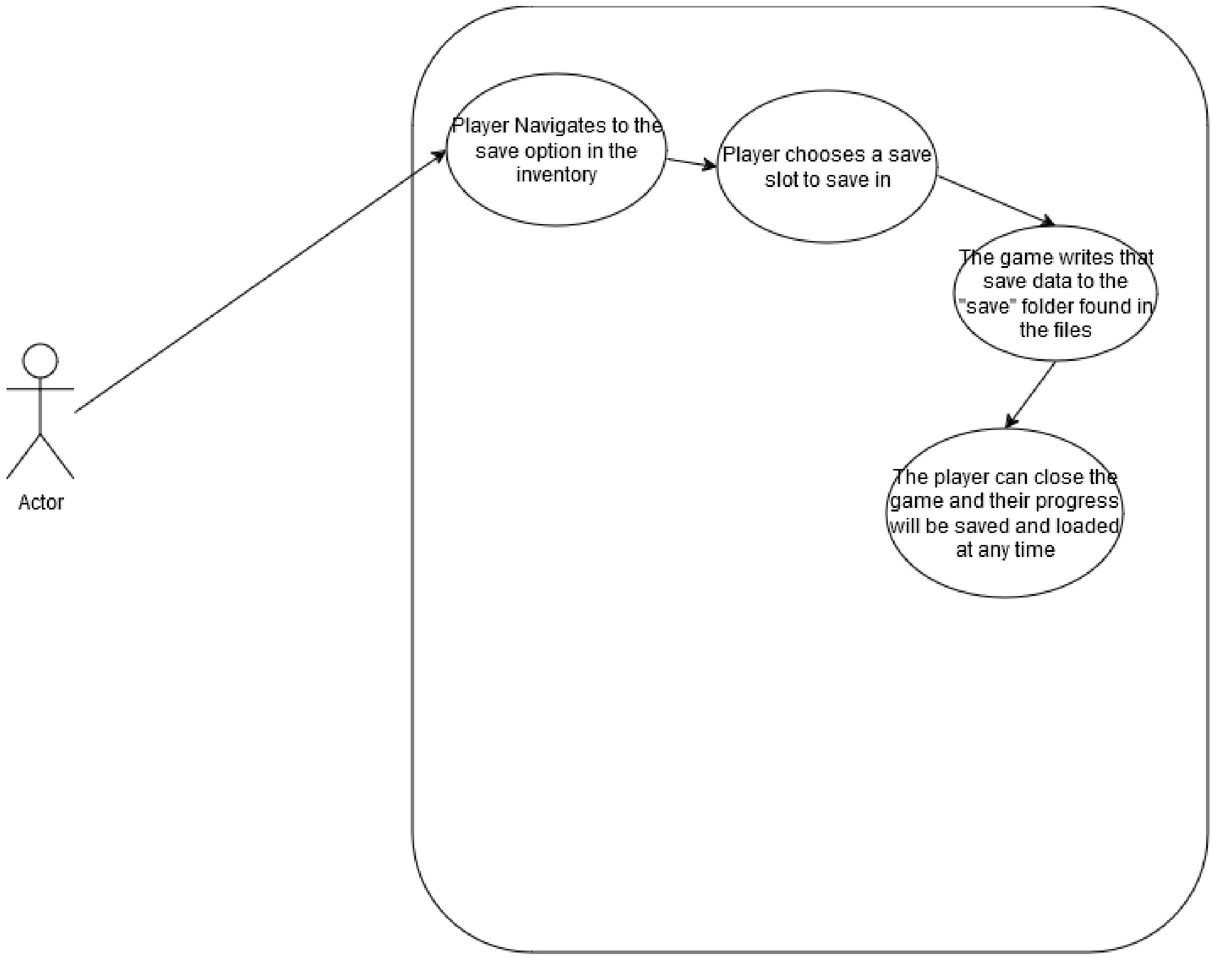
Scope

The scope of this use case is to show off saving and loading game data

Description

This use case describes the process of saving in game loading save data to continue playing

Use Case Diagram


Player Navigates to the save option in the inventory
Player chooses a save slot to save in
The game writes that save data to the "save" folder found in the files
The player can close the game and their progress will be saved and loaded at any time
Actor

Flow Description

Precondition

The user is in the save category within the inventory and the system allows the user to save (Area dependant)

Activation

This use case starts when the user chooses the slot to save their progress

Main flow

1. The user saves their game with the save option in the inventory
2. The user confirms the slot they want to save in
3. The system writes the save data to the save folder in the game files
4. The player can close the save section in the inventory and continue playing

Alternate flow

1. The user loads their save data in the main menu by hitting the “Continue” option in the main menu

2. The system finds the selected save file from the “saves” folder
3. The system loads that file and opens the game at the point the player saved at previously.

Exceptional flow

4. The system crashes.
5. The system autosaves
6. The game reopens

Termination

The system has saved the data successfully/ The system has loaded the data correctly

Post condition

The system goes into a wait state until the next time the user saves/loads data

2.1.1.8. Requirement 3 Opening/Closing the Inventory

Use case for opening the Inventory via User interaction

2.1.1.9. Priority

The user opens the Inventory with the ESC key and the System displays the Inventory as a result of the user pressing ESC and vice versa if the inventory is already open. This is one of the Highest priority features as if this doesn't work the player can access anything within the inventory, Including saving the game

Name

Inventory use case

Number

Use case 003

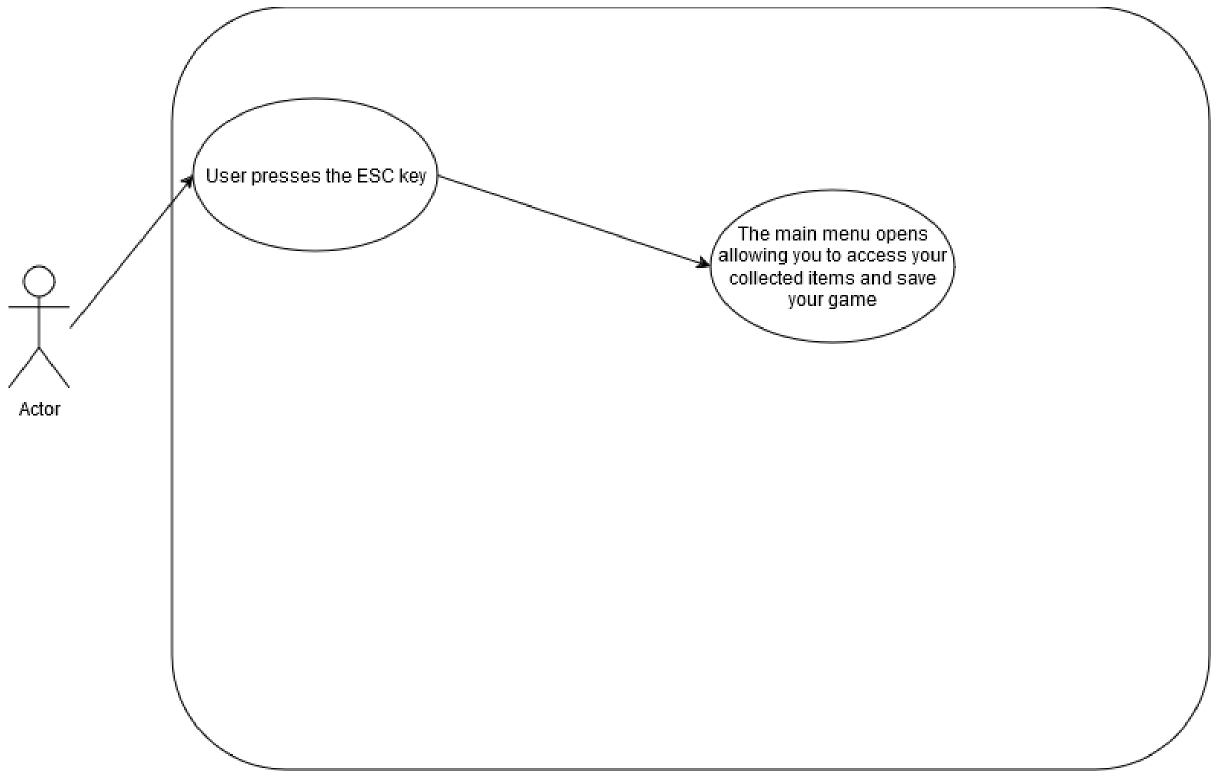
Scope

The scope of this use case is to give a rough explanation of the inventory and how it works

Description

This use case describes the way the player can interact with the Inventory system

Use Case Diagram


User presses the ESC key
The main menu opens allowing you to access your collected items and save your game
Actor

Flow Description

Precondition

The system is currently outside of battle and on the overworld without the menu already being open

Activation

This use case starts when an Actor presses the ESC key, pressing this will make the system open the inventory

Main flow

1. The user presses the ESC key
2. The system recognises the user pressed the ESC key
3. The system opens the Inventory
4. The player may access the various options in the inventory menu

Alternate flow

1. The player presses the ESC key while the Inventory is open already
2. The system recognises the user pressed the ESC key
3. The system closes the Inventory

Exceptional flow

7. The system crashes.
8. The system autosaves
9. The game reopens

Termination

The system either recognises the game was terminated or it a battle started

Post condition

The system awaits the next input of the ESC key

2.1.1.10. Requirement 4 Battle: Attacking

This shows off the attack command within battle for the player

2.1.1.11. Priority

If battling is to be in the game, then all commands must be functional as intended this will include Attacking, including all the commands and this one it will all be classed as medium priority as the battles are a core gameplay mechanic that need to work

Name

Player attack use case

Number

Use case 004

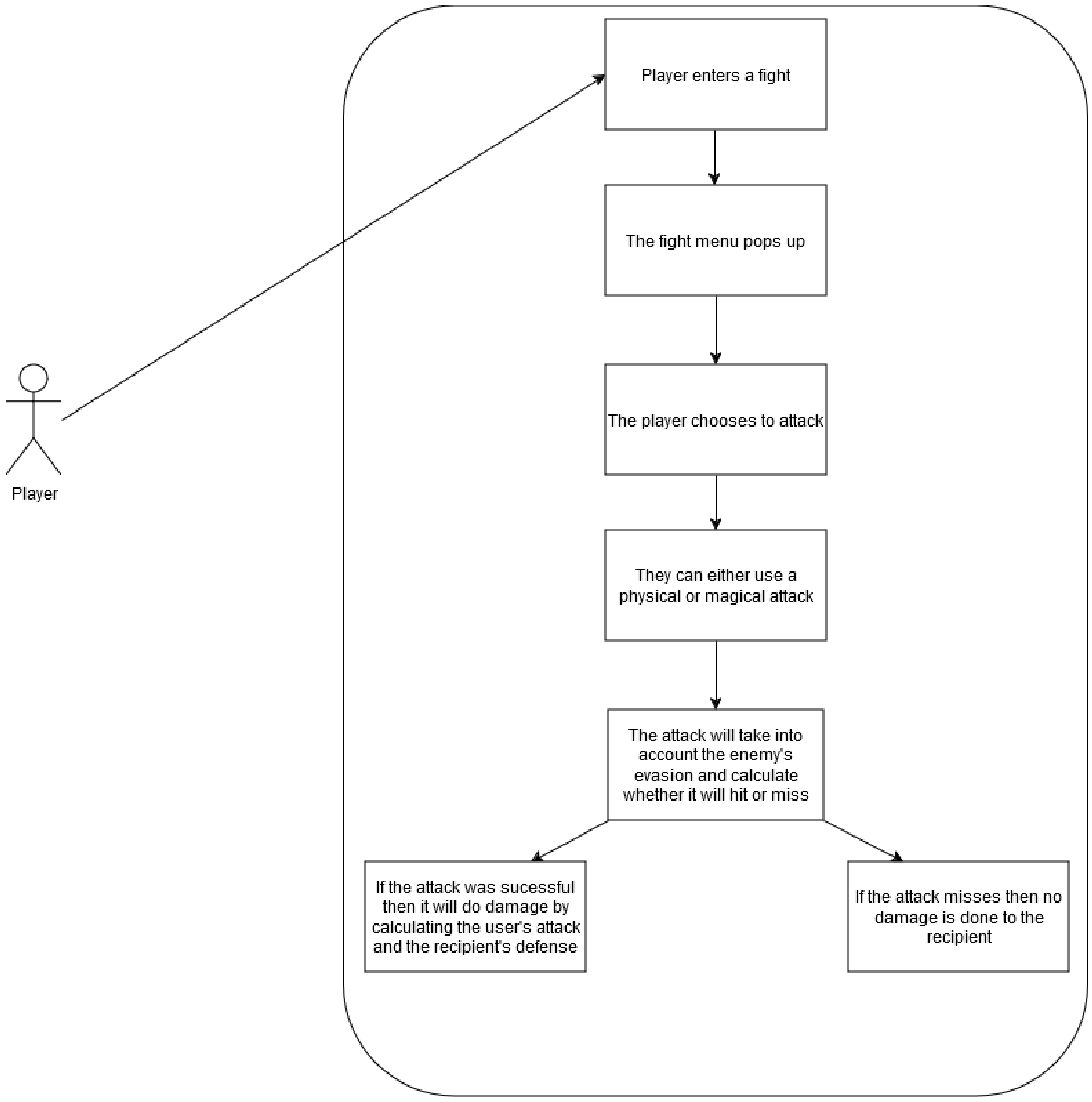
Scope

The scope of this use case is to explain the processes of attacking an enemy.

Description

This use case describes the processes of attacking

Use Case Diagram


Player enters a fight
The fight menu pops up
The player chooses to attack
They can either use a physical or magical attack
The attack will take into account the enemy's evasion and calculate whether it will hit or miss
If the attack was sucessful then it will do damage by calculating the user's attack and the recipient's defense
If the attack misses then no damage is done to the recipient
Player

Flow Description

Precondition

The player must have entered a battle with an enemy

Activation

The player must currently be in a battle with an enemy.

Main flow

1. The system identifies that the battle has started
2. The system pulls up the battle menu for the User
3. The Player gets to pick the options of Attack, Magic, Item and Flee

4. The Player chooses Attack which then the Player can choose their target for the attack
5. The attack either hits or misses.

5i. If the attack hits it deals damage to the enemy

5ii. If the attack misses nothing happens

6. The Player's turn ends and It becomes the enemy's turn
7. It becomes the enemy's turn (Look at Enemy Attack use case)

Exceptional flow

1. The system crashes.
2. The game reopens at home page.
3. The game autosaves.

Termination

The battle ending via the player defeating the enemy or the player dying in battle.

Post condition

The system goes into a wait state until the next battle begins.

List further functional requirements here, using the same structure as for Requirement1.

2.1.1.12. Requirement 5 Death process

This shows off what happens when the player dies

2.1.1.13. Priority

The player dying is an important feature that must be implemented alongside battling, If death isn't added it give the player no sense of worry when they partake in battles as they wouldn't be punished for failure, Death would have the same priority as the battle mechanics as its in integral part of the game

Name

Player death use case

Number

Use case 005

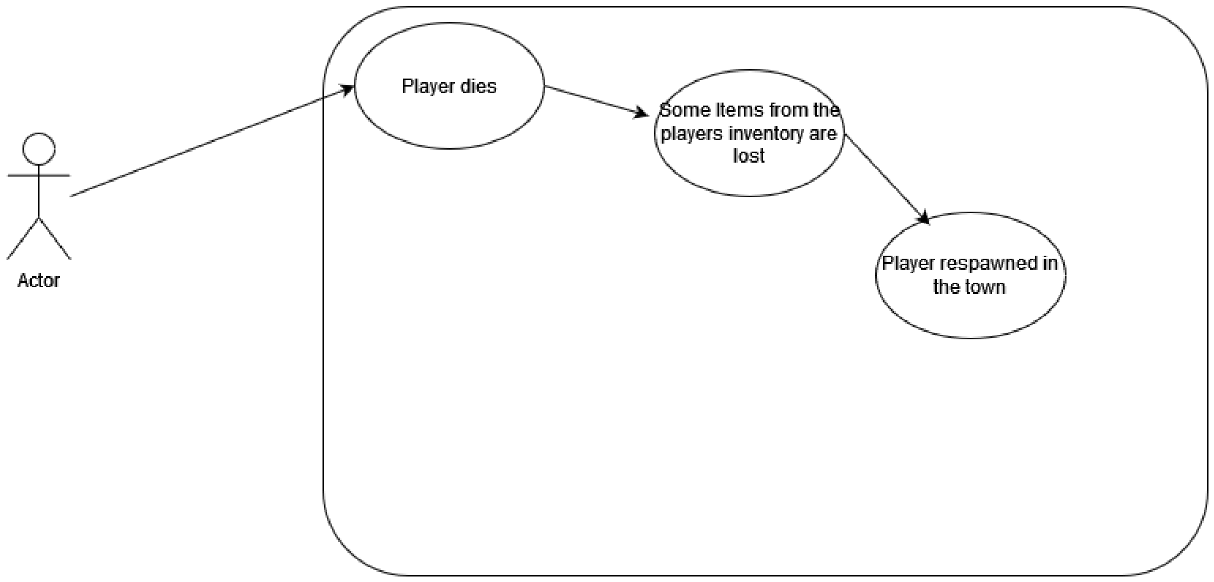
Scope

The scope of this use case is to showcase the process of dying in the game.

Description

This use case describes the process of what happens when you die

Use Case Diagram


Actor
Player dies
Some Items from the players inventory are lost
Player respawned in the town

Flow Description

Precondition

The player must have entered a battle with an enemy

Activation

The player must have lost a battle with an enemy

Main flow

1. The player’s HP drops to 0
2. The System displays a message informing the Player they lose the fight
3. The system chooses at random 10% of their collected items to be removed from inventory.
4. The system respawns the player at their base
5. The System displays a message informing the Player of item lost from inventory

Alternate flow

1. The player tries to flee battle
2. The player fails to escape
3. The enemy hits the player
4. The player’s HP is dropped to 0
5. The Player dies
6. The system chooses at random 10% of their collected items to be removed from inventory.
7. The system respawns the player at their base
8. The System displays a message informing the Player of item lost from inventory

Exceptional flow

4. The system crashes.
5. The game reopens at home page.
6. The game autosaves.

Termination

The system respawns the player at their base and allows them to continue playing.

Post condition

The system goes into a wait state until the player falls in battle again or dies on some other way.

2.1.1.14. Requirement 6 Battle: Items

This shows off the Items command within battle for the player

2.1.1.15. Priority

If battling is to be in the game, then all commands must be functional as intended this will include using Items, including all the commands and this one it will all be classed as medium priority as the battles are a core gameplay mechanic that need to work

Name

Player Item use case

Number

Use case 006

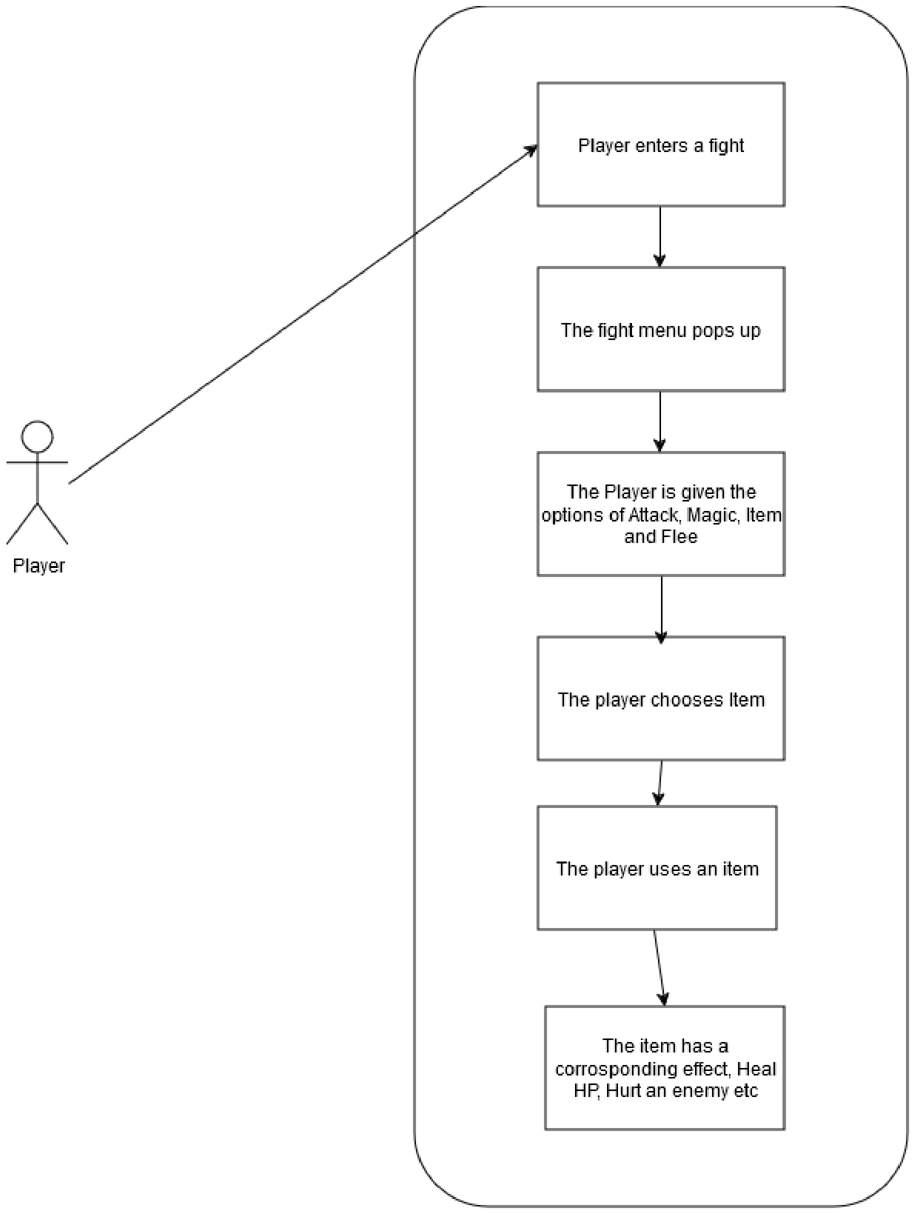
Scope

The scope of this use case is to show the use of items in battle

Description

This use case describes the process of using an item in battle

Use Case Diagram


Player enters a fight
The fight menu pops up
The Player is given the options of Attack, Magic, Item and Flee
The player chooses Item
The player uses an item
The item has a corrosponding effect, Heal HP, Hurt an enemy etc
Player

Flow Description

Precondition

The player must have entered a battle with an enemy

Activation

The player must currently be in a battle with an enemy.

Main flow

1. The player opens item menu
2. The Player chooses item from menu
3. The player uses an item
4. The item has a specific effect (Heal user, Harm enemy etc)
5. The system subtracts 1 instance of that specific item from the player's inventory

6. The Player's turn ends
7.

Exceptional flow

1. The system crashes.
2. The game reopens at home page.
3. The game autosaves.

Termination

The system ends the Player's turn and changes it to the Enemy's turn

Post condition

The system goes into a wait state until the player gets into another battle

2.1.1.16. Requirement 7 Battle: Fleeing

This shows off the attack command within battle for the player

2.1.1.17. Priority

If battling is to be in the game, then all commands must be functional as intended this will include Fleeing, including all the commands and this one it will all be classed as medium priority as the battles are a core gameplay mechanic that need to work

Name

Player flee use case

Number

Use case 007

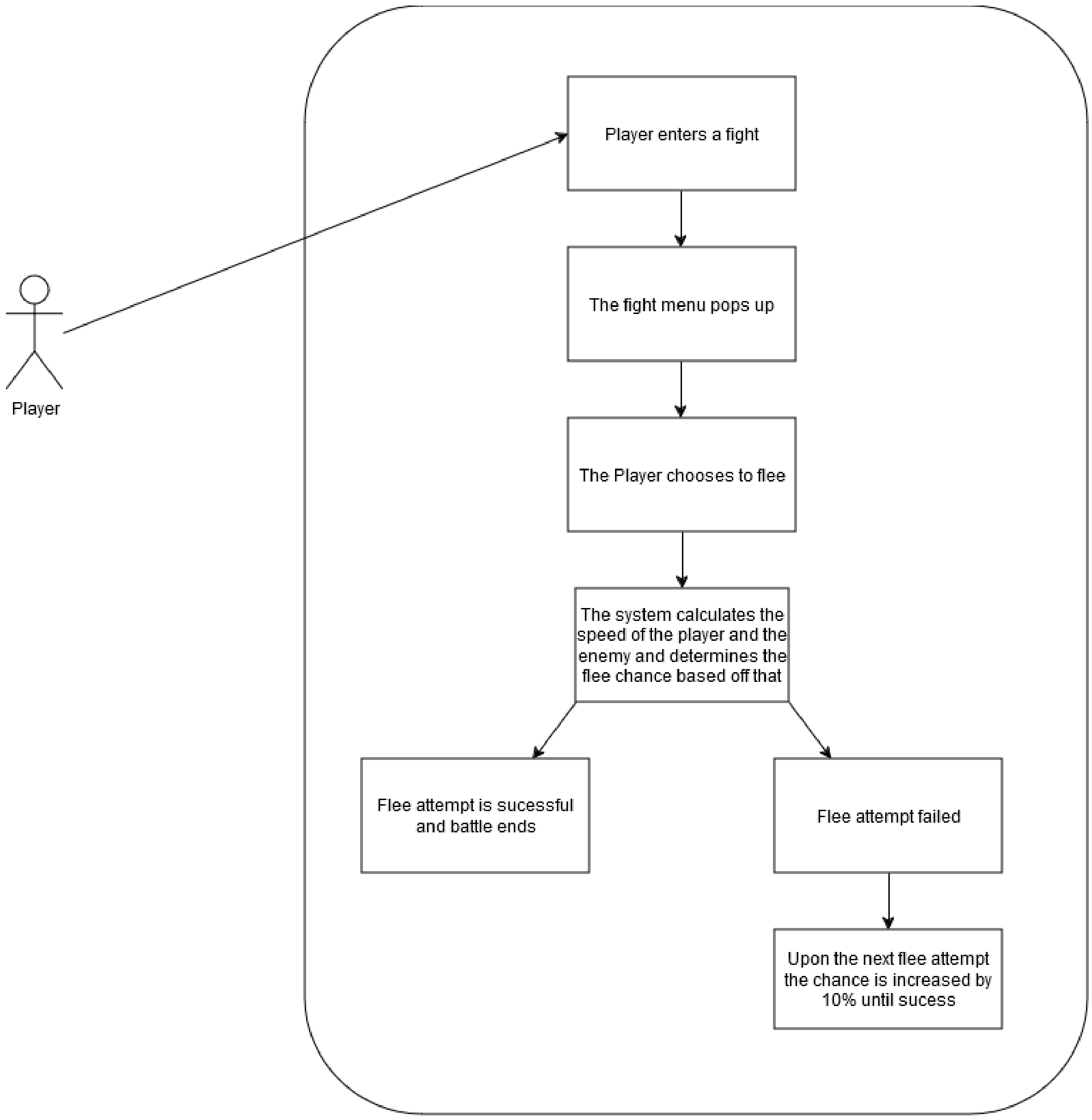
Scope

The scope of this use case is to show the flee mechanic in battle

Description

This use case describes the flee mechanic in battle

Use Case Diagram


Player enters a fight
The fight menu pops up
The Player chooses to flee
The system calculates the speed of the player and the enemy and determines the flee chance based off that
Flee attempt is sucessful and battle ends
Flee attempt failed
Upon the next flee attempt the chance is increased by 10% until sucess
Player

Flow Description

Precondition

The player must have entered a battle with an enemy

Activation

The player must currently be in a battle with an enemy.

Main flow

1. The Player chooses the option to flee.
2. The system checks the speed of the Player and the Enemy
3. The system calculates the chance of a successful flee attempt
4. The system displays that the flee attempt was successful

Alternate flow

10. The Player chooses the option to flee.

1. The system checks the speed of the Player and the Enemy
2. The system calculates the chance of a successful flee attempt
3. The system displays the flee attempt was a failure
4. The system adds an extra 10% chance to the next flee attempt when the player tries again
5. The system ends the player's turn and it becomes the enemy's turn

Exceptional flow

1. The game crashes
2. The game reopens at home page.
3. The game autosaves.

Termination

The system ends the battle by either the player escaping the battle or if they die in battle

Post condition

The system goes into a wait state until the player gets into another battle

List further functional requirements here, using the same structure as for Requirement1.

2.1.1.18. Requirement 8 Battle: Enemy Attacking

This shows off the attack command within battle for the enemies

2.1.1.19. Priority

If the enemies do not attack, then the player wouldn't have any level of challenge when it comes to battles and enemies would be nothing more that HP sponges that just waste the players time

Name

Enemy attack use case

Number

Use case 008

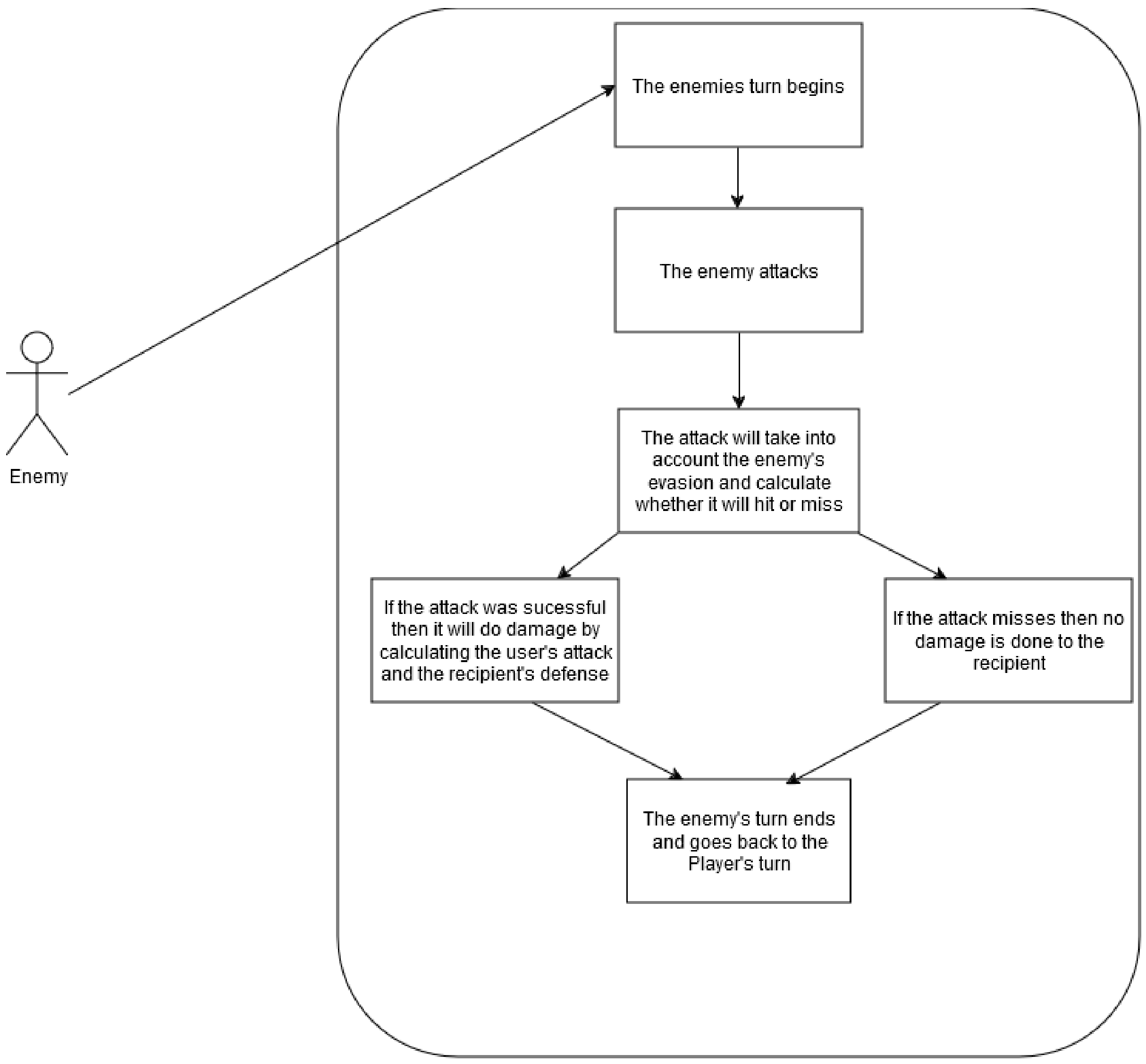
Scope

The scope of this use case is to show off the turn of an enemy

Description

This use case describes the how an enemy's turn plays out

Use Case Diagram


The enemies turn begins
The enemy attacks
The attack will take into account the enemy's evasion and calculate whether it will hit or miss
If the attack was sucessful then it will do damage by calculating the user's attack and the recipient's defense
If the attack misses then no damage is done to the recipient
The enemy's turn ends and goes back to the Player's turn
Enemy

Flow Description

Precondition

The player must be in a battle with an enemy

Activation

This use case starts when the player has done an action and the system ends

Main flow

1. It becomes the enemy's turn
2. The enemy's attack either hits or misses
3. The system calculates the user's accuracy and the recipient's evasion
4. The attack hits and the player takes damage from the enemy
5. The system ends the enemy's turn and starts the player's turn

Alternate flow

1. If the attack misses then the recipient takes no damage

2. The system ends the enemy's turn and starts the player's turn

Exceptional flow

11. The system crashes
12. The game reopens at home page.
13. The game autosaves.

Termination

The system ends the process if either the enemy or player dies in battle

Post condition

The system goes into a wait state until the player gets into another battle

2.1.2. Data Requirements

2.1.3. User Requirements

2.1.4. Environmental Requirements

2.1.5. Usability Requirements

2.2. Design & Architecture

Describe the design, system architecture and components used. Describe the main algorithms used in the project. (Note use standard mathematical notations if applicable).

An architecture diagram may be useful. In case of a distributed system, it may be useful to describe functions and/or data structures in each component separately.

2.3. Implementation

Describe the main algorithms/classes/functions used in the code. Consider to show and explain interesting code snippets where appropriate.

2.4. Graphical User Interface (GUI)

Provide screenshots of key screens and explain what can be seen in each one.

2.5. Testing

Describe any testing tools, test plans and test specifications used in the project. Provide evidence for and results of all Unit, Integration and End User testing that is carried out.

2.6. Evaluation

How was the system evaluated and what are the results? This may consist of usage data. It may also include performance evaluations, scalability, correctness, etc. depending on the focus of the project. Quantative results may be reported in tables or figures.

3.0 Conclusions

Describe the advantages/disadvantages, strengths and limitations of the project

4.0 Further Development or Research

With additional time and resources, which direction would this project take?

5.0 References

Please include references throughout your document where appropriate. See here for a guide on referencing from the NCI library.

6.0 Appendices

This section should contain information that is supplementary to the main body of the report.

6.1. Project Proposal

6.2. Reflective Journals

6.3. Other materials used

Any other reference material used in the project for example evaluation surveys etc.