UTA Department of Computer Science and Engineering

CSE 1320 Project Documentation

TERMINAL BATTLE

Students names, surnames, IDs:
1. Alex Kowis 1001794789
2. Roxie Barnett 1002090515
3. Suhas Gillipelli 1002054171
4. Seezayn Bishwokarma 1002168842
5. Samuel Uwakwe 100

Mentor: Marika Apostolova
TA: Dahak Arjun

CSE 1320 GROUP 1

Intermediate programming CSE 1320

Student declaration:

We declare that:

- *We understand what is meant by plagiarism*
- *The implication of plagiarism has been explained to me by our professor*
- *This assignment is all team own work and we have acknowledged any use of the published and unpublished works of other people.*

1 Student name, surname, ID and **signature: Roxie Barnett 1002090515**
2 Student name, surname, ID and **signature: Alex Kowis 1001794789**
3 Student name, surname, ID and **signature: Suhas Gillipelli 1002054171**
4 Student name, surname, ID and **signature: Seezayn Bishwokarma 1002168842**
5 Student name, surname, ID and **signature: Samuel Uwakwe 100**

	Total number of pages including this cover page	17
Class Code / Group	CSE 1320	
Lecturer's Name	MARIKA APOSTOLOVA	

Table of Contents

CHAPTER ONE

Project introduction

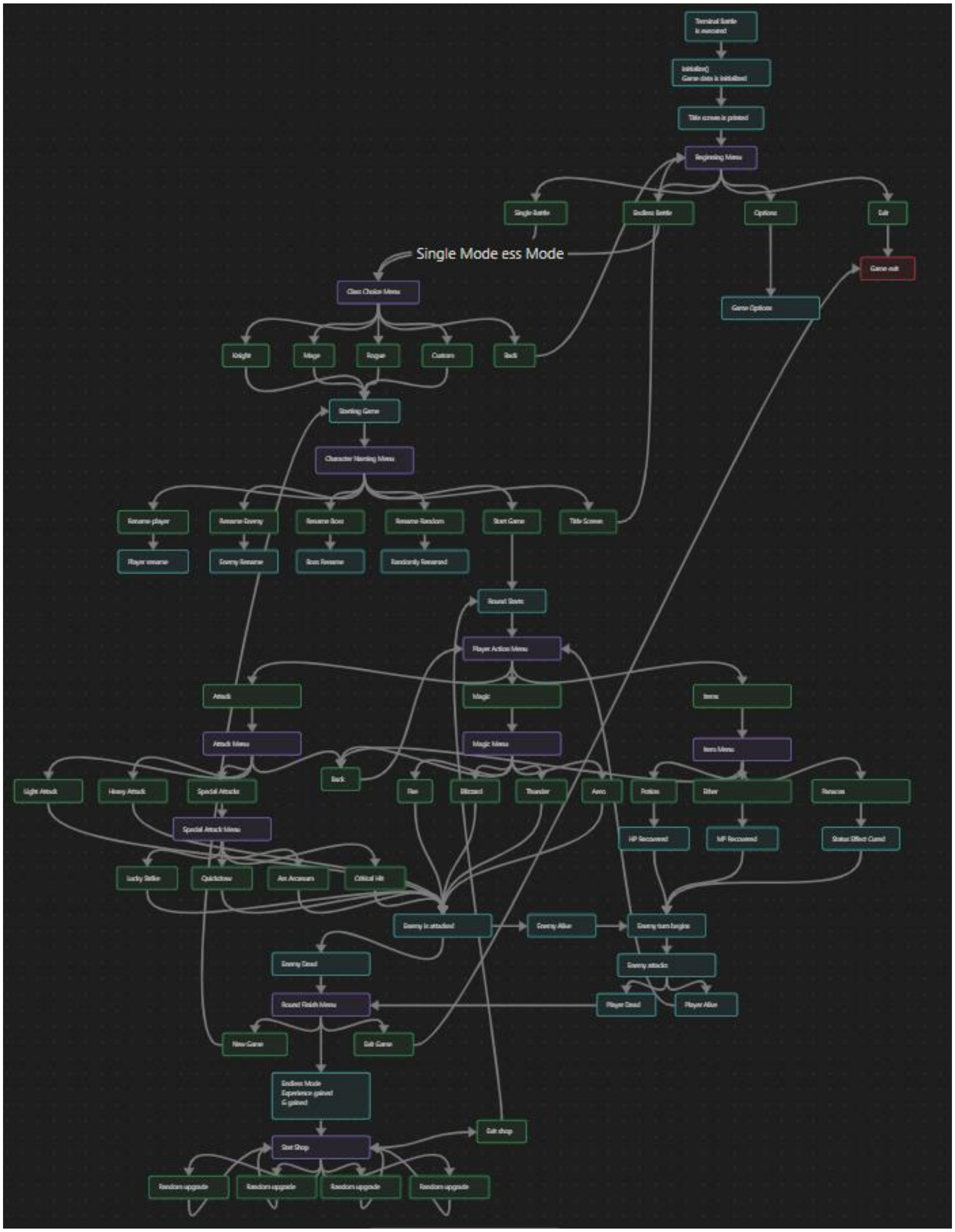
Terminal Battle is a video game programmed in the C language. Terminal Battle is in the Turn-Based-Combat genre of video games, where the player takes turns with a programmed enemy in a battle in which the main goal is to deplete the enemy’s health before the player’s health can be depleted.

Terminal Battle is inspired by games such as Pokémon and Final Fantasy, which are both in the Turn-Based-Combat genre and have very similar mechanics, featuring a variety of choices the player can make which change dynamically during the player’s experience.

The goal of the project was to create a suitable user experience in which users can think both strategically and creatively through the multitude of decision-making opportunities presented by the game. Dynamically challenging the player based on conditions which change based on randomness is a key component in maintaining a consistent and enjoyable challenge.

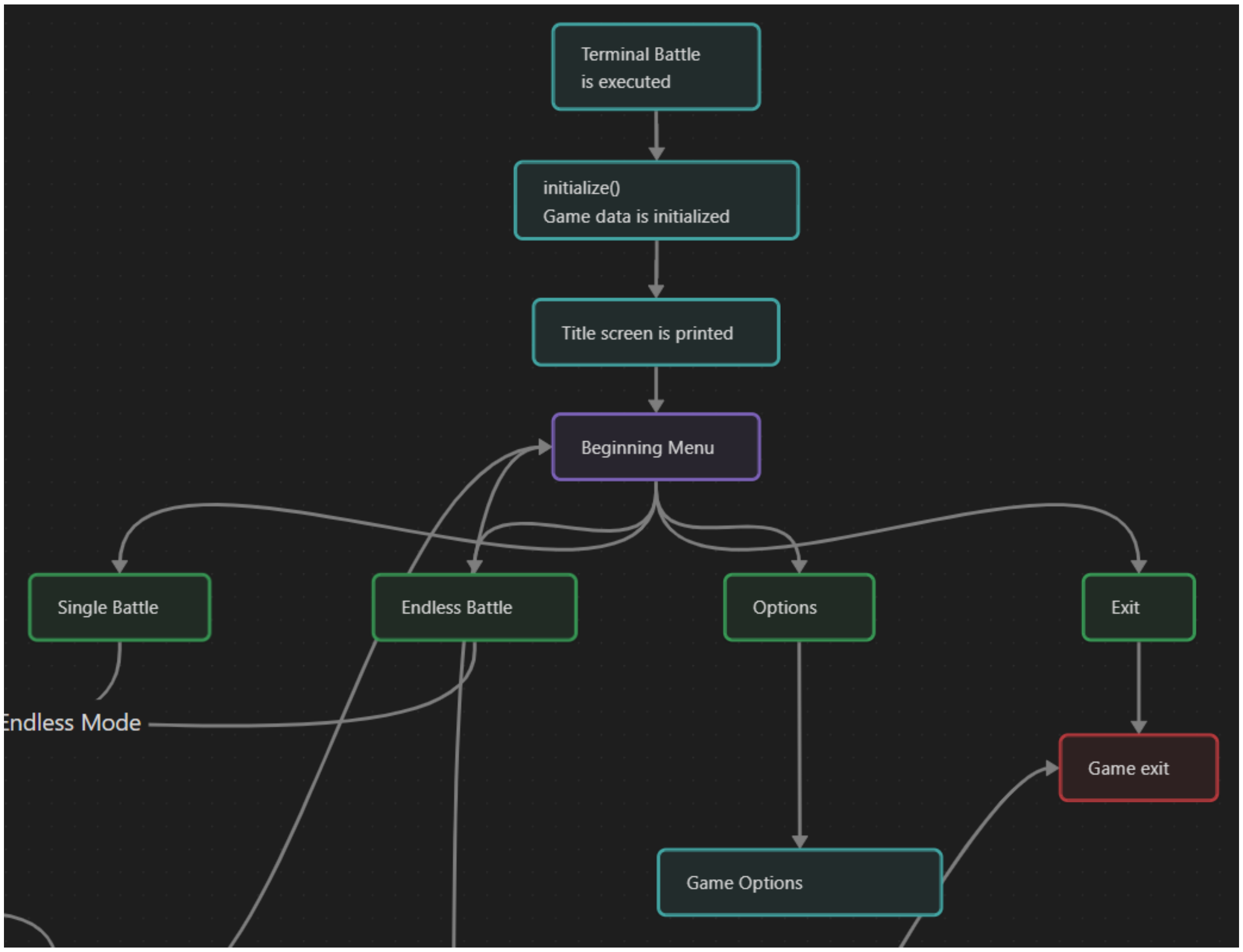
Current Systems

CSE 1320 GROUP 1



Blue represents the program mechanics. Purple represents a menu for the player. Green represents a potential player input. Note the expansive options the player has.

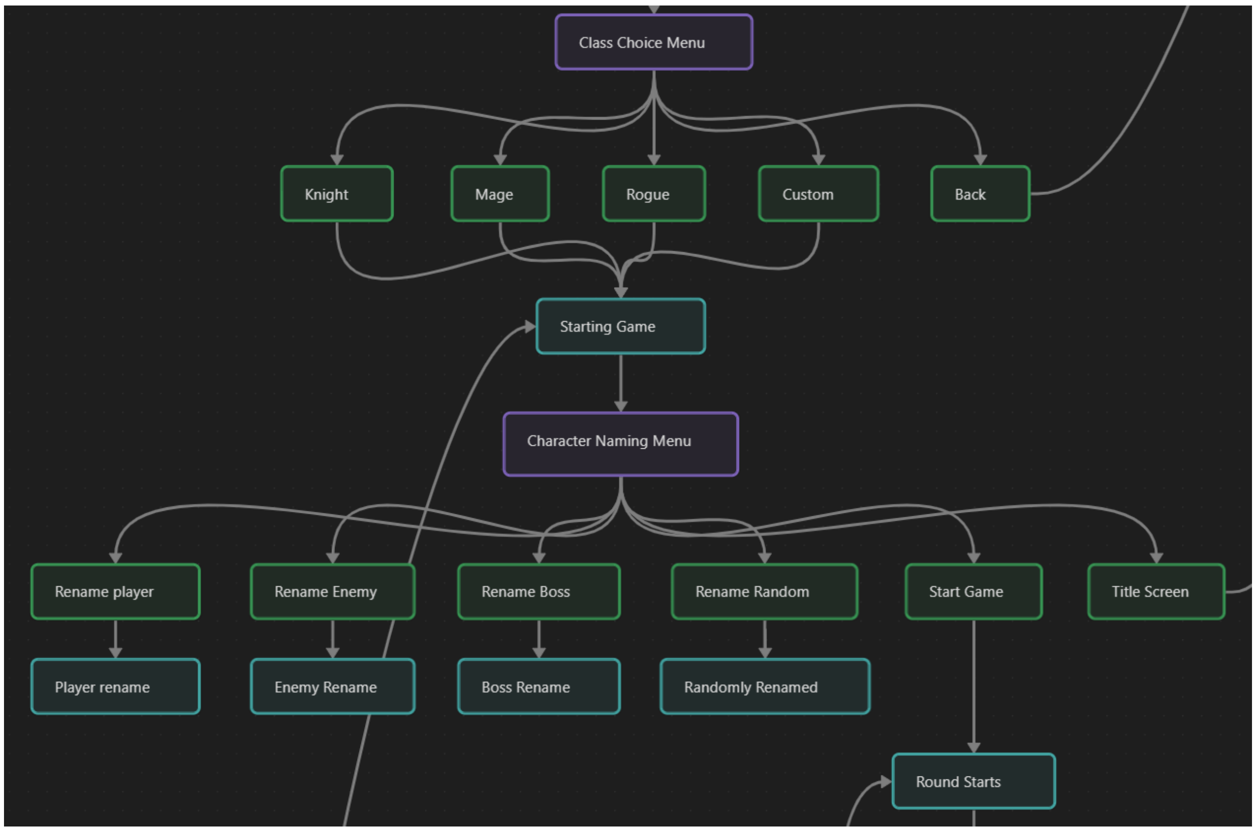
Project Specification/ Function Modules



The starting sequence of the program initializes variables and prepares for the first player input, where the player can choose to either begin the game immediately or modify the playing experience through the Options menu.

The Single Battle option allows for a demo of the Terminal Battle experience, where a user may attempt to defeat a single enemy.

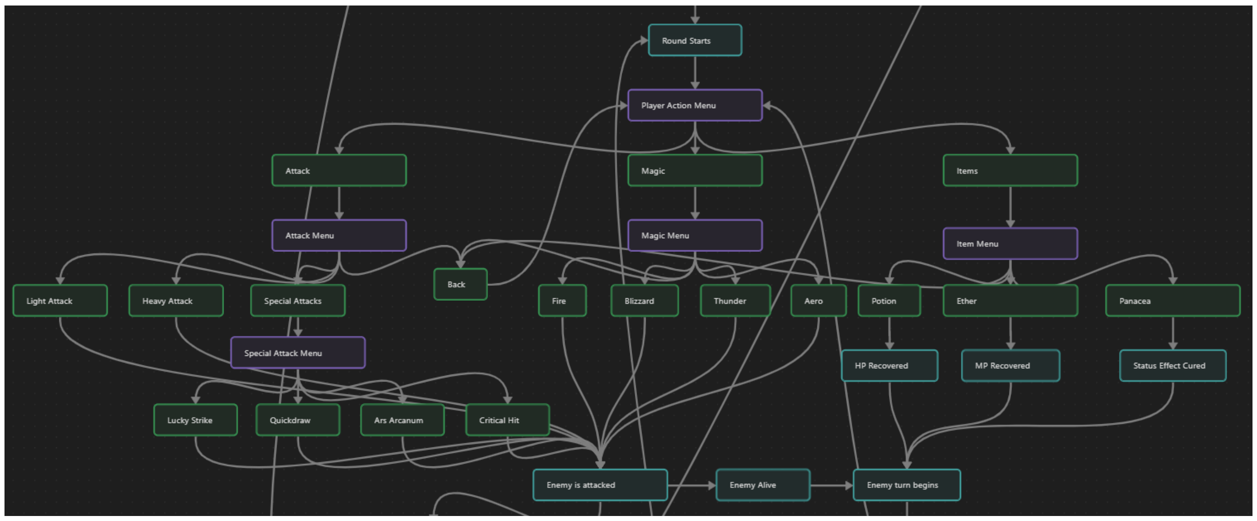
The Endless Battle option is a much more involved and complex system, where enemies indefinitely appear and the goal of the game changes from simply defeating enemies to strategically collecting currency in order to upgrade and last as long as possible.



When beginning a new game, the player is asked to choose starting conditions for the game, more specifically the “class” of the player which may come with different benefits or disadvantages. This allows players greater variability in playstyle and flexibility to suit the needs of many different players.

Users may also rename their player character or “enemy” characters.

Users are also able to return to the Title Screen at this point if they do not which to begin the Terminal Battle experience or have decided to perform modifications using the Options menu in the Title Screen.



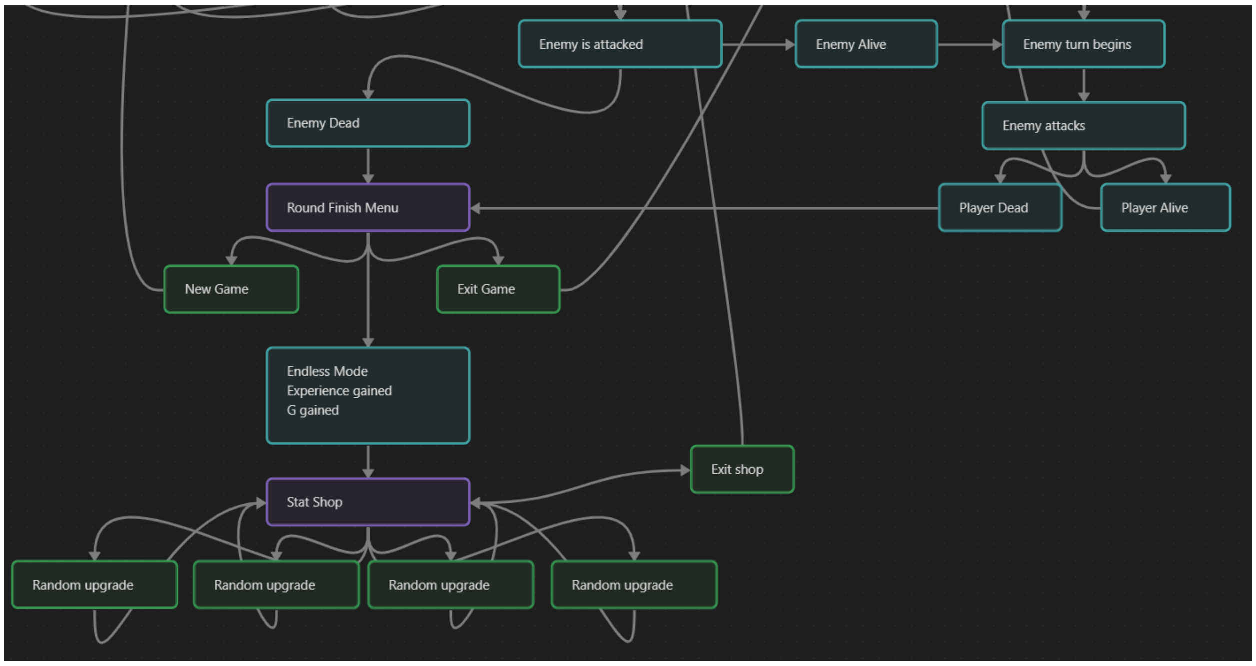
Menus and player choices depicted above:

- Player Attack Menu
  - Attack
    - Light Attack
    - Heavy Attack
    - Special Attack
      - Lucky Strike
      - Quickdraw
      - Ars Arcanum
      - Critical Hit
    - Back
  - Magic
    - Fire
    - Blizzard
    - Thunder
    - Aero
  - Items
    - Potion
    - Ether
    - Panacea

The above sequencing represents a player turn in a round of Terminal Battle, and shows the main game experience. Players have a variety of options to us Attacks, Magic, or Items in a strategic and creative manner in order to satisfy the win condition of depleting the enemy's health to zero.

What is not shown above is the many conditionals which may dynamically change menu options, such as an enemy causing a status effect on the player called "freeze", which prevents the player from being able to Attack or use Magic unless the unfreeze condition has been met or a special curing item is used.

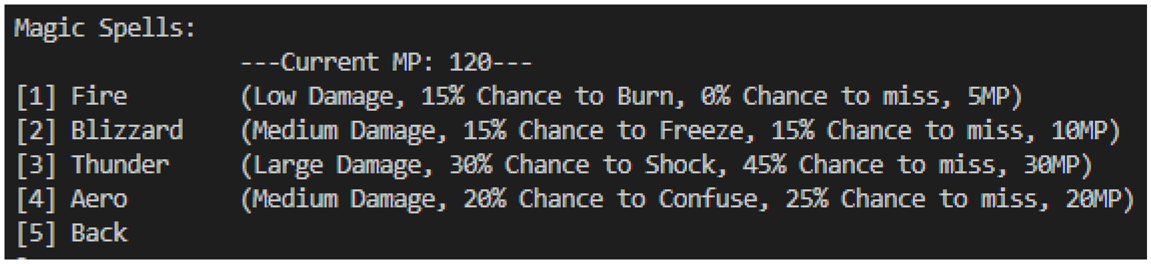
The player sequence above and the enemy sequence below both occur cyclically until either two conditions are met: the player health value depletes to zero or the enemy health value depletes to zero.



The enemy sequence is similar to the player sequence, however the decisions made by the enemy are randomized.

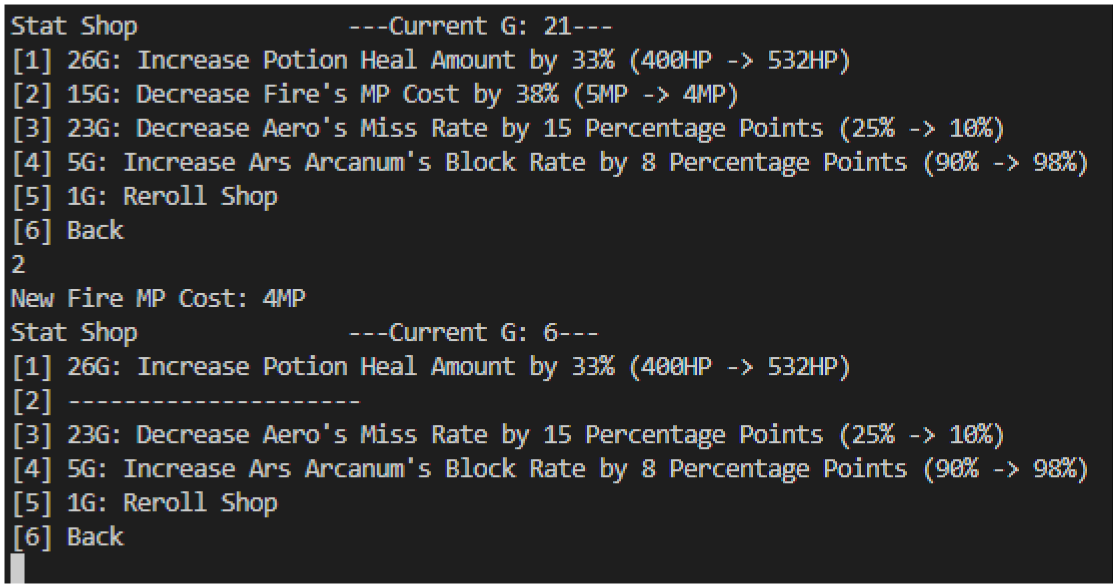
If the player is in Endless Mode and the condition of defeating the enemy is achieved, then the player may spend a currency called G which is collected at the end of a successful round in the Stat Shop, which provides four random upgrades the player can choose from in order to increase their own abilities. The Stat Shop presents an extra layer of complexity, strategy, creativity, and flexibility for the user, allowing for a more enjoyable experience.

Program (Input/ Output) Specification



```
Magic Spells:
                ---Current MP: 120---
[1] Fire        (Low Damage, 15% Chance to Burn, 0% Chance to miss, 5MP)
[2] Blizzard    (Medium Damage, 15% Chance to Freeze, 15% Chance to miss, 10MP)
[3] Thunder     (Large Damage, 30% Chance to Shock, 45% Chance to miss, 30MP)
[4] Aero        (Medium Damage, 20% Chance to Confuse, 25% Chance to miss, 20MP)
[5] Back
```

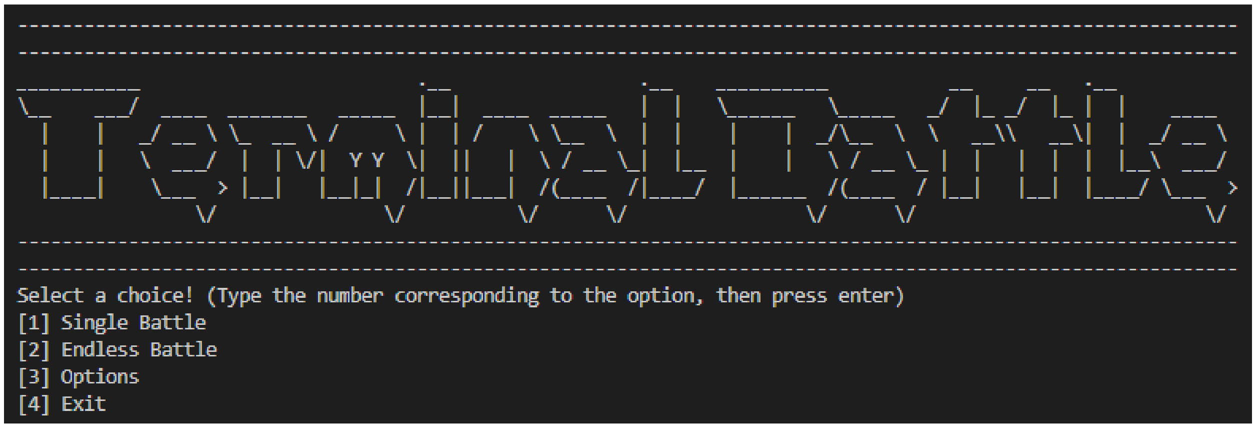
Program input is managed through menus such as the above, arranged in lists which requests a player input of an integer as the control flow conditional. I/O management such as the above is used throughout the entire program as the SOLE method of player input.



```
Stat Shop               ---Current G: 21---
[1] 26G: Increase Potion Heal Amount by 33% (400HP -> 532HP)
[2] 15G: Decrease Fire's MP Cost by 38% (5MP -> 4MP)
[3] 23G: Decrease Aero's Miss Rate by 15 Percentage Points (25% -> 10%)
[4] 5G: Increase Ars Arcanum's Block Rate by 8 Percentage Points (90% -> 98%)
[5] 1G: Reroll Shop
[6] Back
2
New Fire MP Cost: 4MP
Stat Shop               ---Current G: 6---
[1] 26G: Increase Potion Heal Amount by 33% (400HP -> 532HP)
[2] ---------------------
[3] 23G: Decrease Aero's Miss Rate by 15 Percentage Points (25% -> 10%)
[4] 5G: Increase Ars Arcanum's Block Rate by 8 Percentage Points (90% -> 98%)
[5] 1G: Reroll Shop
[6] Back
```

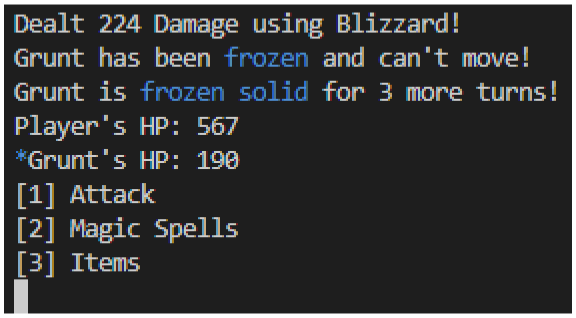
Dynamic menus are also used, such as depicted above for the Stat Shop. The options which are presented in the menu are variable and can be randomly “rerolled” by the player at a small cost.

Screen Design



```
-----------------------------------------------------------------------------------------------
-----------------------------------------------------------------------------------------------
___________                  .__              .__    __________         __    __  .__
\__    ___/__________  _____ |__| ____ _____  |  |   \______   \_____ _/  |__/  |_|  |   ____
  |    |_/ __ \_  __ \/     \|  |/    \\__  \ |  |    |    |  _/\__  \\   __\   __\  | _/ __ \
  |    |\  ___/|  | \/  Y Y  \  |   |  \/ __ \|  |__  |    |   \ / __ \|  |  |  | |  |_\  ___/
  |____| \___  >__|  |__|_|  /__|___|  (____  /____/  |______  /(____  /__|  |__| |____/\___  >
             \/            \/        \/     \/               \/      \/                    \/
-----------------------------------------------------------------------------------------------
-----------------------------------------------------------------------------------------------
Select a choice! (Type the number corresponding to the option, then press enter)
[1] Single Battle
[2] Endless Battle
[3] Options
[4] Exit
```

Terminal Battle contains some cosmetic functionality in order to make the user experience more enjoyable and comfortable, offering greater clarity and readability, especially when running the program in a standard terminal which may not be visually appealing for most users.



```
Dealt 224 Damage using Blizzard!
Grunt has been frozen and can't move!
Grunt is frozen solid for 3 more turns!
Player's HP: 567
*Grunt's HP: 190
[1] Attack
[2] Magic Spells
[3] Items
```

When enemies are afflicted with status effects, colored text is used to indicate the affliction

CHAPTER 2:
Program Design:

Overall Structure
Modularity: The software is divided into modules, each of which handles a different set of functions.
Menu-driven: The game's menu system lets users pick and navigate through various options.

Flow Control
Loop-Based Flow: To maintain a steady and regulated flow, the game's states are managed using loops.
Handling User Input: Uses scanf() to handle menu selection and user input.

Abstraction:
Function-Based Design: This approach encourages code readability and reuse by using functions to encapsulate tasks.
Separation of Concerns: For easier maintenance, each module concentrates on a single topic (such as the menu screen or character selection).

Module Menu Screen:

Objectives
Interface Management: Manages the main game options menu interface.
Flow Control: Determines how the game progresses overall by using player choices.
Important Features
Display Options: Provides the player with a list of options, such as "start game," "select character," and "game options."
User Input Processing: Reads and processes user input to move between game states.
Design Overview: In charge of showing the player the first set of menu options.
controls the game's progression according to the player's choices.

Code Function

- shows a text-based menu with options to exit the game, play an endless battle, or select a single battle.
- utilizes the use of loops to guarantee that the player's input falls within the specified range.
- invokes the relevant functions (such as battle mode selection, options, and exit) in accordance with the player's selections.
- manages the game's main loop where the player navigates the title screen.

Module Select:

Objectives
Character Selection: Handles the selection of character classes.
Attributes Initialization: Determines the character's initial attributes by selecting a class.

Important features
Class Options: Presents different character classes with their specific attributes.
Attributes Setup: Sets up character attributes based on the chosen class for use in battles.

Design Overview: Allows the player to select their character class before initiating a battle.
Differentiates between various character classes with distinct attributes and abilities.

Code Functions

- Presents a selection menu for player classes (Knight, Mage, Rogue) with specific attributes and multipliers for HP, MP, and attack/magic abilities.
- Uses loops to validate player input and ensure the selection is within the available options.
- Initializes character attributes and abilities based on the player's class choice.
- Initiates the game after the player has selected a class for battle.

Module Add:

Objective
Additional Functionalities: Implements various additional features.
Randomization: Provides randomization for character names and potentially other game elements.
Important Features
Randomization: Selects names for characters from an external file, potentially used for player, enemy, or boss characters.
User-Driven Renaming: Allows renaming of characters based on player input.
Design Overview: Manages additional functionalities and auxiliary features in the game.
Contains functionalities to add randomness or varied elements to the game.

Code Functions

- Implements functionalities like randomizing character names using a file containing a list of names.
- Handles options such as renaming player, enemy, or boss characters based on the player's choice.

- Incorporates features like reading from files (e.g., a file containing names) to add variability or randomness to certain game elements.
- Assists in the initialization of characters or settings for battles, including endless mode.

CHAPTER THREE:

Program Testing – The program was tested throughout the coding process by running the program in the command prompt. This would tell the errors occurring. The different versions of the program represent bug fixes and additions to the code.

Test LOG

- Version 1.01 was completed on October 11th
  - This version included the uncomplete model of the battle game. It allowed the user to start the game, however it was limited as to what move was allowed next. There was no attack function, so dealing damage to the opponent was not integrated yet.

- Version 1.02 was completed on October 27th
  - This version fixed an issue that arose from the user inputting a name for their player that included a space. An option for an auto-generated name was also implemented. The attack function was created and fully functional, along with special attacks the user could decide to use. In addition to attacking, a blocking chance was also added. There was a chance the user or the enemy could block some damage based on the attack they choose to use. Regarding the Magic, Magic Points (MP) were integrated into the game as well. The user could only use magic they could afford with their MP. An additional item, called Ether, was added to the Items Function.

- Version 1.03 was completed on November 1st
  - While coding and testing up to this version of the game, there are a lot of additions to the game. A new game mode was added where the user could play an endless version of the game. In this game mode, the user will battle a new opponent

once defeating the previous one. While battling, the text in the terminal now changes color based on the magic used. Another big implementation of this version is the addition of the shop. This shop introduced an in-game currency, that can be used to purchase items. Regarding the users/opponents turn in the game, a new update was implemented where the specific player could not make a move until the magic dealt on them has finished.

- Version 1.04 was completed on November 21st
    - This is the final version of our completed game. 3 new functions were added on this version of the program. These functions are starting game, player confusion damage, and boss turn. The starting game function was implemented because the user can now choose which class they would like to play as. Each class has certain pros and cons to them. The starting game function was created to separate the naming of the characters from choosing the class type. The player confusion damage function is an aspect based on luck. If the user is "confused", they have a 50% chance of doing damage to themselves. This function calculates chance, and the damage they do. The final function that was added was the boss function. This function is only called in the endless battle, and it is called every five rounds. These specific rounds calculate new health and multipliers for the opponent.

CHAPTER 4: CONCLUSION

Weaknesses of Terminal Battle:

1. Fragile Checks
    - Throughout the program, scanf is used to get ints and occasionally chars, if someone were to input a char in place of an int, the program would break.
2. A little bit messy
    - Code inconsistencies are present and due to the size of the code and the fact that "it still works", things like this might have gone unnoticed.
3. Long Code
    - Not necessarily a weakness but the length of it could be intimidating to debug and fix certain issues if you don't know where to look
4. Luck Based
    - Though a little luck in games is okay, the program primarily relies on luck to determine specific outcomes and damage dealt

Strengths of Terminal Battle:

1. Replay ability
    - You can play with different classes and try different techniques to finish battles in the best possible outcome. You can also try to improve your score from last time
2. Infinite outcomes
    - Determining how you play your moves and on the roll of the dice, you can end battles in an infinite number of ways, some more convenient than others, and others completely unfair
3. The user can choose how to play.
    - Through the options menu, they can give themselves advantages/disadvantages in the beginning of the battle, allowing for experimentation and optimization.

How to Enhance Terminal Battle:

1. Code Condensation
    - Though the code would still be over 1500 lines, some code can likely be condensed and simplified into smaller, easier to understand batches.
2. Code organization and more notes

Use of purposeful indentions and new lines can help organize code into easier to understand batches