***The Vengeful Vampire***

A Text-Based Game

**Abstract**

With C++ as its foundation *The Vengeful Vampire* provides a simple yet fascinating gaming experience for the role-playing enthusiasts.

The game presents the player with a choice of three heroes: The Warrior, The Mage and The Rogue. After selecting their hero the player begins their journey. The player must face the army of *Vlads* .First he has to cut through the minions summoned by *Vlads*. That’s not all. As the player faces the tiresome horde of bandits, he must also confront the *Vlads’s* three *Gargoyle’s* in between.

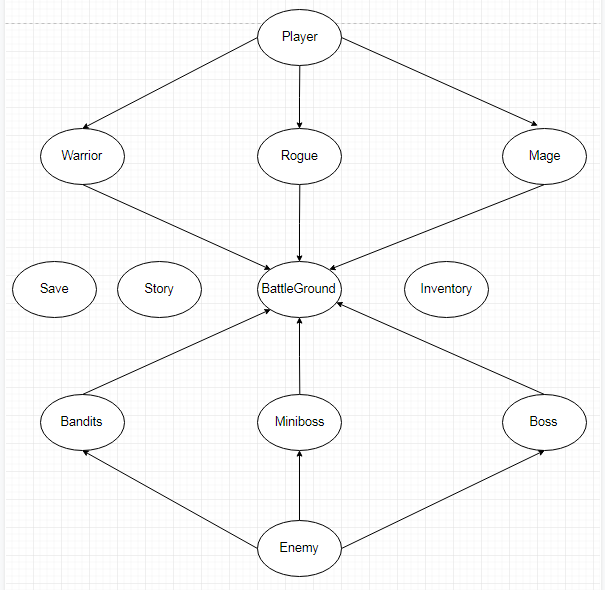
Throughout the game, the player levels up as he defeats his foes to unlock new character attacks and abilities. Limited Health and Special Power forces the player to act smartly to achieve their goal.

**Introduction**

This project uses C++ and its respective concepts and features to create a complex text-based game.

This project implements Object Oriented Programming concepts such as data encapsulation, data hiding, inheritance, reusablity and polymorphism.

**ER Diagram**



**Class Descriptions:**

**Class Player{**

This class consists of common player attributes, like health, gold, experience, damage, inventory/bag and the player’s level. Simple get and set functions are employed in this class.

**}**

**Class Warrior/Mage/Rogue**{

Each of these classes inherit from the Player class, and they hold the specific attributes of a special player type. For e.g; Warrior’s rage, Mage’s mana, Rogue’s energy, which serve as the player’s resource to use his attacks/abilities. Simple get and set functions are employed in this class.

**}**

**Class Enemy{**

This class consists of common enemy attributes, like health and damage. Simple get and set functions are employed in this class.

**}**

**Class Bandits/Miniboss/Boss**{

Each of these classes inherit from Enemy, and they consits of abilities specific to their type, as in, the Bandit only punches. The Miniboss on the other hand has multiple attacks, such as Axe Slash, Charge etc which a Miniboss uses randomly. Similiarly the Boss has the highest number of attacks which he randomly uses. Each of these attacks have different scaling and damage attributed to them. Simple get and set functions are employed in this class.

**}**

**Class BattleGround{**  
BattleGround inherits all the classes like Warrior, Mage, Rogue, Bandit, Miniboss, Boss.

1. Int levelup():

This method deals with player level up which triggers when a player acquires certain number of experience points.

1. Void HUD():

Provides the core interface for our game, including health bar, resource bar, experience bar/points, the player’s level, also the enemy’s health bar.

1. Void AttackMenu():

This method is overloaded 3 times for each playable type; Warrior, Mage and Rogue. It displays abilities of each player type (Warrior’s abilities and so on) also displays the damage dealt and the damage received by the player and to the player respectively.

1. Void EnemyAttack() / Void MiniBossAttack() / Void BossAttack():

These methods deal with enemy damage dealt to the player and the value of damage dealt by each enemy type respectively.

1. Void WarriorPlay() / Void RoguePlay() / Void MagePlay():

These methods control the entire gameplay, of each player type in their respective methods. This employs the logic as to when a miniboss will attack the player, and makes sure the boss is the last enemy the player faces, while the bandits attack the player continously.

This also adds items dropped by the enemy to the inventory of the player, and also makes sure the game runs until the player health reaches 0.

This method also deals with granting experience, health regeneration, resource regeneration, and increases the max health/resource of player at Levelup, and re-initializes the health with maxhealth. Also provides the player with access to shop at each level up.

1. Void Interface():

Allows player to select the player type, from the character selection screen of the game, using arrow keys. Coordinated Console is also implemented here.

1. Void Menu():

Allows the player to start a new game or load the last session of his/her game, using arrow keys.

Coordinated Console is mplemented here aswell.

**}**

**Class Inventory{**

This class adds and deletes items from the player’s inventory, and also deals with the consumption of items in the bag.

1. Void BAG():

Deals with consumption of items.

1. Void Add()/Void DeleteItem():

Deals with acquiring dropped items and removing items from inventory.

**}**

**Class Story{**

Provides Story for the game. Each Narrative is called at a set time, like at the start of the game, and when a miniboss/boss arrives.

**}**

**Class Save{**

This class uses a save function to auto-save the game at each level up using filing, and saves all the player’s information including health, resource, level, gold, inventory information. This allows player to continue his journey from the last level up.

Uses the load function to read the contents of the save file and use that information to start the game from the given level.

**}**

A global function Shop also exists that enables the functionality of a Shop holding items, and returns the damage of the item that the player buys.

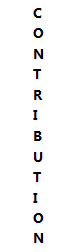
**Libraries:**

* Iostream
* Fstream
* Conio.h
* Windows.h
* Cstdlib
* Cstring

\

**Distribution of Work:**

|  |  |  |  |
| --- | --- | --- | --- |
| **Mahad Khalid** | **Abdullah Raheel** | **M. Ammar** | **Jahanzeb Raza** |
| Prototype | Prototype | Prototype | Prototype |
| Debugging | Debugging | Debugging | Debugging |
| HUD | Levelup | Save | Class MiniBoss |
| Interface | Story | Filing | Class Boss |
| Menu | Shop | Class Bandit | MiniBossAttack |
| AttackMenux3 | Project Proposal | BanditAttack | BossAttack |
| Inventory |  |  |  |
| Player Class | Player Class | Player Class | Player Class |
| Warrior Class  MageClass  RogueClass | Warrior Class  MageClass  RogueClass | Warrior Class  MageClass  RogueClass | Warrior Class  MageClass  RogueClass |
| WarriorPlay  MagePlay  RoguePlay | Addition to:  WarriorPlay  MagePlay  RoguePlay | Addition to:  WarriorPlay  MagePlay  RoguePlay | Addition to:  WarriorPlay  MagePlay  RoguePlay |
| PlayersAbilities | PlayersAbilities |  |  |
| Casting |  |  | Casting |
| Project Report | CD | Project Report | CD |



**Conclusion:**

We tried our level best to implement all the concepts of Object Oriented Programming.

**Future Work:**

Increasing the complexity of the game and making it better.

Adding GUI.

**References:**

* Lab Manuals
* <https://stackoverflow.com/questions/4053837/colorizing-text-in-the-console-with-c>
* Geeksforgeeks