**Initial Idea**

The game will be a text based adventure game, but could alternatively become a side scrolling game if we get chance to create a GUI.

**Game Story**

(Boss Name) has taken (Item Name) from (Village Name) to (his/her) (Lair Name). It is up to (Username) to (save/retrieve) (Item Name) from the (Boss Name) by defeating many enemies along the way such as (Enemy Names).

As the game progresses (Username) will face many scenarios which will alter the course of the game and decide if (Username) has been successful in the quest. These scenarios can alter the stats of (Username) in terms of advantages or disadvantages.

**Design Layout (Prototype)**

# Opening Screen

**DUNGEON CRAWLER**

NEW GAME

CONTINUE QUEST

LOAD GAME

EXIT

Top Scores

|  |  |
| --- | --- |
| Name | Scores |
|  |  |



# New Game

STORY………

CHOOSE YOUR CHARACTER

RACE… RACE… RACE…

STATS… STATS… STATS…



CONTINUE

# Extra screen before game starts

Detailed description / story about chosen character …

Picture of character

START QUEST!!!

# Game screen

Dungeon Crawler

The scenarios you face …

The choices available …

The user input!!!

(Username Stats)

-HP

-Attack

-Defence

-Items (Storage)

-EXP

-Level

-etc.

Game Options… Save (Screen), Load (Screen), Score, Exit, New Game etc.

The outcome of your choice!!!

(Scenario stats)

-Enemies

-Items etc…

The screen will display all stats, scenarios and choices at start up. The user then can read the scenario and choices that can be selected for that scenario. Some choices will benefit you, others will cause bad consequences and some will be neutral. It will be turn based so that when the next scenario appears then the stats will be updated based on the previous outcome. Some turns you will face enemies or rooms with items inside and other turns you may be neutral and continue a turn. The idea is that we will need to build a database that C++ can interact with by making a copy of the database so the original stats are not changed when a user plays the game just the copy version will change for that save. The database will hold stats, scenarios and as the gameplay progresses the more technical and advanced it will become causing more thought to be put into the user’s choice. The game will need an end ultimately however it could be left open until the game gets to complex and resulting in a user dying causing a game over and a score entry saved.

# Game-over screen



Retry, Load Game, New Game, Exit

Your final score, Enter name, Submit

Story based on how you came to your demise ….

DATABASE – Design

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| ID | Name | HP | Attack | Defence | Special Move | Range | EXP | Level |
| 0 | Elf | 100 | 8 | 5 | 5 | 1 | 0 | 0 |
| 1 | Wizard | 90 | 6 | 3 | 8 | 8 | 0 | 0 |
| 2 | Archer | 85 | 9 | 6 | 8 | 6 | 0 | 0 |
| 3 | (Enemy Race 1) | 30 | 2 | 1 | 5 | 1 |  | 0 |
| 4 | (Enemy Race 2) | 35 | 3 | 2 | 5 | 2 |  | 0 |
| 5 | Boss | 200 | 10 | 8 | 12 | 5 |  | 0 |

The table above is an example of how the database could be created for the characters and enemies. Each of them can be edited by the game once play has started and will continue to change as the game progresses. We could also include the scenarios and item stats into the database and hold the values for the score. In theory, anything that we need to contain a value can be stored into the database and means we can just code the game to access and modify to make the gameplay more enjoyable.