Game Project Report

H00468402

Fox Murray

Contact details:

Fox Murray

H00468402

fjm4000@hw.ac.uk

Revision Table:

Contributor:	Date	Version number	Notes
Fox Murray	03/04/25	1	Flowchart Created
Fox Murray	04/04/25	1.1	Wrote main body

Contents

Game Overview:

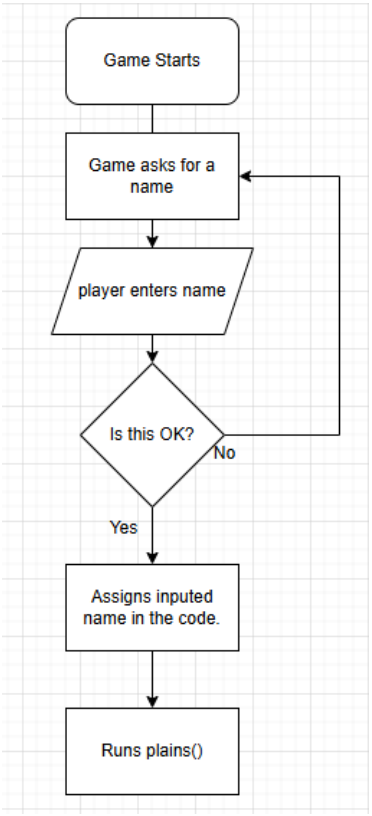
To begin the game, after running the compiled code, the player is prompted to enter their name and confirm it.

Everything in the game is done by typing specific works into the terminal and pressing enter. The game then uses the scanf(“”) function and compares it against its known actions, then does the chosen one. There are no graphics libraries or images with this game, hence the genre name, “Text based adventure”.

```c
printf("\nEnter your name, adventurer! (no spaces!)\n");
// Use %13s to limit input length and avoid buffer overflow - reads the 12 letters put in in sequence. Doesnt allow for spaces though.
 scanf("%13s", &playername);

printf("Ah, so your name is %s\n", playername);


printf("Are you happy with the name, %s ?\n Y / N\n", playername);  //asks the question
scanf("%1s", &contYN); //reads 1 character and sets the continue question to that letter. only 1 letter stops overfow

if (strcmp(contYN,"Y") == 0 || (strcmp(contYN,"y") == 0) ) {
   // || splits options it accepts, allowing Y and y respectively
   printf("Very well, %s \n", playername);
   break; //exits the loop
} else if (strcmp(contYN,"N") == 0 || (strcmp(contYN,"n") == 0) ) {
   continue; //if not happy with name, shortcut to start of loop.
} else {
   printf("\n");
};
```

Its record's the players name to the "playername" variable, and to confirm, the player must enter Y or N. the code also accounts for lowercase y or n. If No is entered or something other than Y/ Yes the code will loop around to the top due to a "continue" function, looping back to the top of the whole(1) loop.

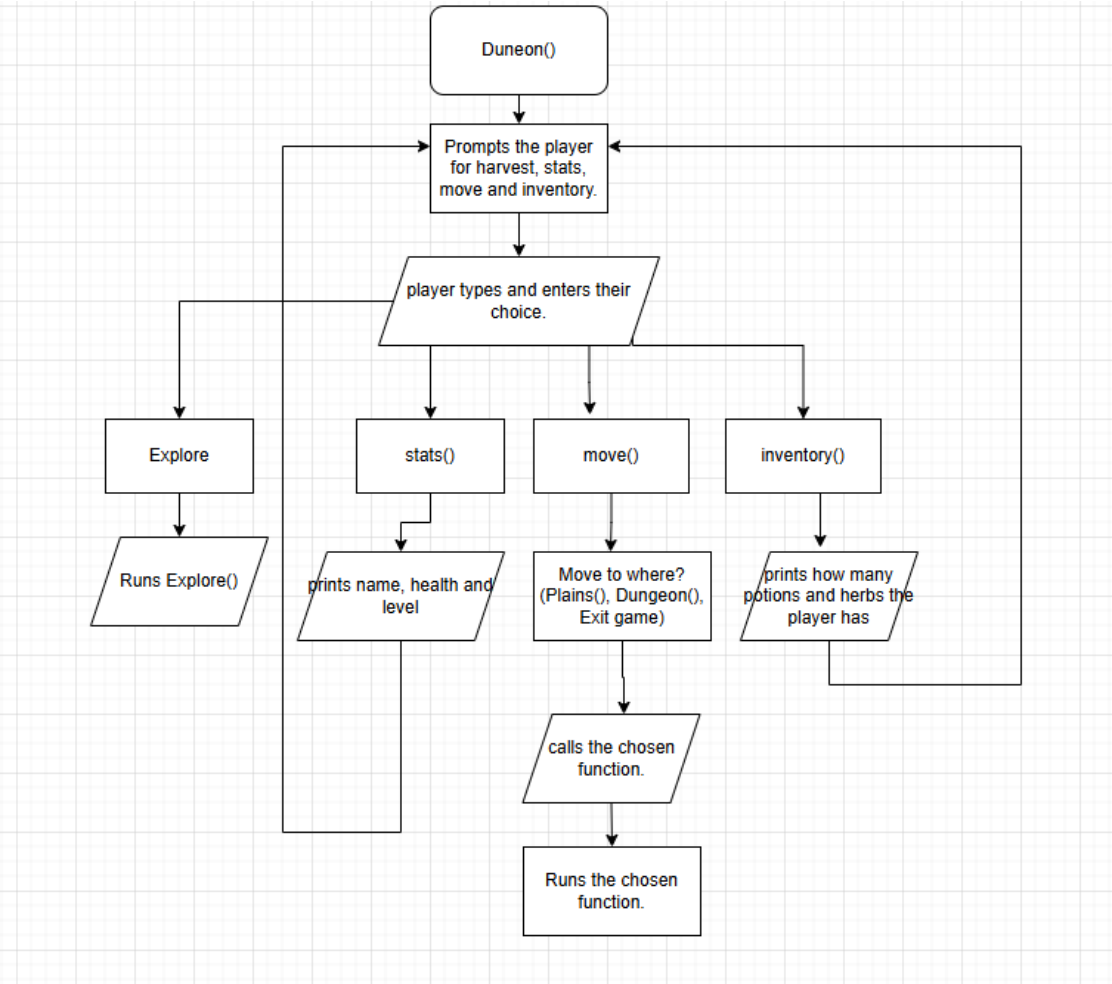
Here is a spreadsheet detailing this process:



Next, the game calls the function stats() and plains(). Stats() shows various information recorded in different variables. These are Health, Level and Name.
The main areas of the game as called as functions, like plains(). This allowed me to code each area separately and allow free movement between them and other processes the game wants to handle, such as checking your inventory, healing through rests, your health or going to fight zombies.

The next function, to progress the game, is to explore the dungeon, so the code calls dungeon(). When dungeon is called it presents an identical option list, except “explore” replaces“harvest.” when explore is chosen the player descends into the dungeon, where (on every floor) they are met with a zombie! Now they’re given more options:
Attack the zombie, heal their wounds with potions, check their stats, check the enemy’s health or run away.

```
printf("\n");
enemyident = 3;
printf("Dungeon - Floor %d\n",dungeonfloor);
printf("A zombie blocks the way!\n");
printf("- - - - - FIGHT - - - - -\n");
printf("attack\n");
printf("heal\n");
printf("stats\n");
printf("enemy (check's enemy stats)\n");
printf("escape\n");
printf("- - - - - - - - - - - - -\n");
```



Same as before, the player just types in what they want to do and presses enter, then it will run that action.

```
void fight() {
    srand(time(NULL));  // Seed the random number generator once
    // Player attack
    int damage = rand() % 11 + 10;  // Random damage between 10 and 20
    damage *= lvl;  // Multiply damage by level
    printf("%s attacks!\n", playername);
    enemyhealth -= damage;
    printf("Enemy took %d damage! Enemy's remaining health: %d\n", damage, enemyhealth);

    // Enemy attack
    int incomingdamage = rand() % 11 + 10;  // Random damage between 10 and 20
    playerhealth -= incomingdamage * 0.5;  // Subtract incoming damage from player health
    printf("Zombie attacks! Player's remaining health: %d\n\n", playerhealth);

    if (playerhealth <= 0) {
        printf("Game over!\n");
        exit(1);
        main();
    }

    else if (enemyhealth < 0)
    {
     printf("Zombie slain!\n");

     lvlticker += 1; //adds 1 to level ticker / level progress
     dungeonfloor += 1; // go deeper int othe dungeon
     if (lvlticker >= lvl)
         {
         levelup(); //if level has been completed, run level up script.
        }
     dungeon();
    }
```

Which calculates damage dealt to the zombie based on your level and then the zombie attacks the player back. Once you kill the zombie, if you've killed enough zombies (counted by "lvlticker") then the player levels up and earns a potion.

Once the fight is over the player is prompted by the normal location menu once more, where they can exit the dungeon, continue going deeper or leave to heal by resting. Each time you return to the dungeon you have to begin from the top!

Flowchart Detailing Major Processes:

A flowchart explaining the general function of the game: