**Snake Game**

Sytem

System is the skeleton view behind the GUI part of a game. System defines the working methodology of the game and shows the components, their relationships and how they evolve to make the game work.

* The snake is represented with an \*(asterisk) symbol.
* The fruit is represented with an @ (at sign) symbol.
* The snake can move in any direction according to the user with the help of the keyboard (⬇️, ⬆️, ⬅️, ➡️ keys).
* When the snake eats a fruit, the score will increase by 1 point ✔️.
* The fruit will generate automatically within the boundaries 🍓 🍒 🍇.
* Whenever the snake will touch the boundary, the game is over ❌.

Module

#include <stdio.h>

#include <time.h>

#include <stdlib.h>

#include <conio.h>

#include<time.h>

#include<ctype.h>

#include <time.h>

#include <windows.h>

#include <process.h>

Interface

Used structure for interface.

struct coordinate

{

int x;

int y;

int direction;

};

Integration Testing

In this individual software, modules are combined and tested as a group.

Functions and Non-Functions

void record();

void load();

int life;

void Delay(long double);

void Move();

void Food();

int Score();

void Print();

void gotoxy(int x, int y);

void GotoXY(int x,int y);

void Bend();

void Boarder();

void Down();

void Left();

void Up();

void Right();

Background colour is black(this is non function).

Functional Requirements:

1. WASD keys for W-Up, A-Left, S-Down, D-Right change the snake's direction.
2. The sake has a velocity in the direction it is moving. That velocity increases with time. (Provide function of time for detailed requirement.)
3. Randomly place power-ups:
   1. Slow snake velocity.
   2. Food makes the snake longer.
   3. Bonus targets gain bonus points.
   4. etc.
4. If the sake collides with the snake the round ends and a new round starts.
5. At the end of rounds, the score is registered as the player’s highest if it is better than any recorded before. The highest registered score is tracked between all players on a leaderboard.
6. Log in to identify the player. (Details should include username/password requirements, is two-factor auth needed, should obscene words be allowed as usernames? etc..?)
7. Options menu to set music and sound effects volumes. Color scheme?

Non-Functional Requirements:

1. On cutting-edge computers, as of 2010, it should get a 40 FPS refresh rate.
2. The leaderboard needs to support 9 million concurrent users.
3. The leaderboard must be updated within 2 min after a new high score is registered by a play.

This is the idea. For real game development documentation, you would also want some discussion about monetization and promotion. You would want some details about the look and feel. (Is it neon colors, does it use a disco beat to the music/effects, should be it slower and more casual or fast and intense? Etc…)

A good game design document is more than requirements. But, the requirements are a major part of it. Functional requirements are about what it should do. Non-Functional requirements are about how well it should perform on which platforms.