• Project Description

◦ !!!IMPORTANT!!! The game should be run on :

1. Windows system
2. Codeblocks
3. VScode, but run on Big5-HKSCS type word.(A small portion of the words might still not show properly.)

◦ The background of this game depicts a civilized world, yet people are brainwashed by Virtual Youtubers, also known as VTubers.

◦ Player represents a hero created by the justice force, and is obliged to save the world.

◦ The only method to win this game is by beating the final boss.

◦ Notwithstanding, if the player's health bar hits zero, then game over.

• Playing Method

◦ Before playing, we start from game setting section. The player is asked to opt for a preferred size of game map, as well as specified occupation, etc. Further information will be presented in the game automatically.

◦ After game settings, the game officially starts. By several number keys, the player can perform actions such as but not limited to : interaction, movement, and what not. More comprehensive and exhaustive info will be shown in later parts or in-game.

- ## Function Description

The functions’ names are specially designed, so that their names already describe how it works.

```
    void select_role(info * player);
    void name(info * player);
    void create_map(info2 * map);
    bool are_you_sure();
    void settings_before_game(info * player);
    void check_level_up(info * player);
    void settle(info * player);
    void print_map(info player, info2 * map);
    bool check_occupied(info2 * map, int tempx,
int tempy);
    void set_up_map_objects(info * player, info2 *
map);
```

```c
    bool check_boundary(int tempx, int tempy, info player, info2 map);
    void go_up(info * player, info2 * map);
    void go_down(info * player, info2 * map);
    void go_left(info * player, info2 * map);
    void go_right(info * player, info2 * map);
    void movement(info * player, info2 * map);
    void trap(info * player);
    void gun(info * player);
    void village(info *player);
    info3 generate_monster(info player);
    void berserker(info * player, int health);
    void player_attack(info * player, info3 * monster, info4 initial);
    void monster_attack(info * player, info3 * monster, info4 initial);
    void battle(info * player);
    info3 generate_boss();
    void boss_attack(info * player, info3 * boss, info4 initial);
    void player_attack_boss(info * player, info3 * boss, info4 initial);
    void battle_boss(info * player);
    void game_introduction();
    void overall_introduction();
    void check_object(info * player, info2 * map, int tempx, int tempy);
    bool check_level(info player, info3 boss);
    bool dodge();
    void adventure(info * player, info2 * map);
```

- ## Variable Description

◦ To conclude, most variables used in this game can be

amalgamated into 4 struct-type variables.

◦ Info : An integration of the player’s up-to-date information,

which includes :

```
struct player_information{
    int health;
    int strength;
    int role;
    int level;
    int exp;
    int passive_ability;
    char name[11];
    int x;
    int y;
    bool gameover;
};
```

Note that :

```
int x;int y;
```

take responsibility of recording the player’s current position.

```
bool gameover;
```

flags whether the game is over.

◦ Info2 : An integration of map details.

```
struct map_information{
    int row;
    int col;
    char **arr;
};
```

It should be clear that :

```
    char **arr;
```

stands for the two-dimentional map.

- Info3 : An integration of the information of monster

encountered.

```
struct monster_information{
    int health;
    int strength;
    int level;
    char name[100];
    bool dead;
};
```

- Initial : An integration of special values that need to be

recorded at the beginning of a fight.

```
struct battle_information{
    int player_health;
    int monster_health;
    int player_strength;
};
```

They are so imperative and pivotal since some occupations

might need them during or after the battle.

Trivial variables such as tempx, tempy, are temporarily used for

testing some checks, suck as boundary.

What’s more, variables with similar names, including : option, test, flag … are oriented to record player’s choice, and take actions after which.

• Version History

<0.1 - Initial Release>