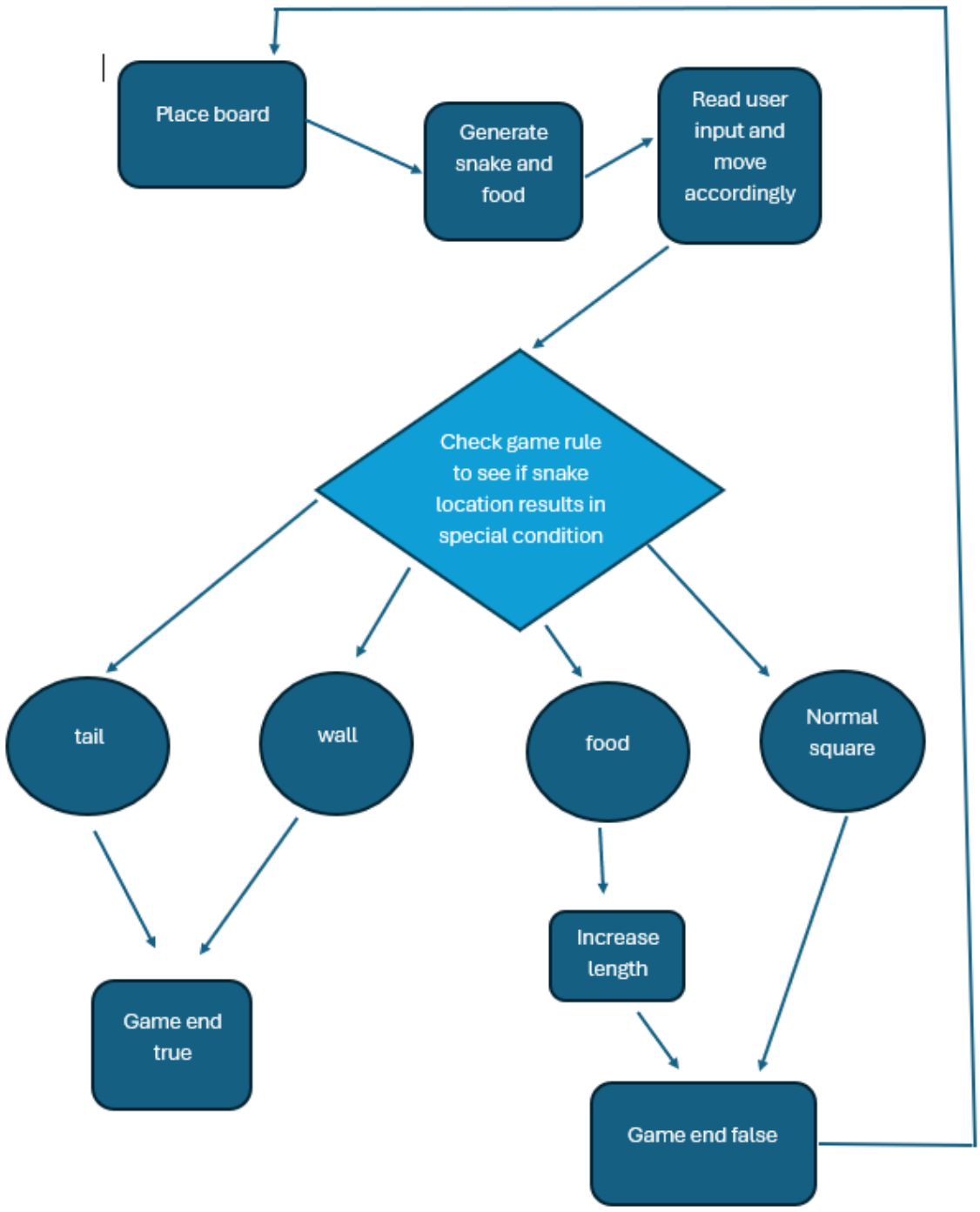
Introduction

The given code is a C implementation of the Snake game. The game ends if the snake collides with the walls or itself but you score by eating and getting longer. This report explains the design the implementation and the issues.

Game Design and Flowchart



The game follows a structured flow placing the initial components then entering a loop to read the users input move the snake and then check what happens at the new location after which it returns to the start of the loop.

```
setup_snake();
setup_food();

while(!isGameOver) {
    fill_board();
    draw_food();
    draw_snake();
    game_rules();
    printf("Snake Game, Score: %d\n", snake.length * 100);
    print_board();
    if( !isGameOver ) {
        int dx = 0, dy = 0;
        read_keyboard(&dx, &dy);
        move_snake(dx, dy);
    Sleep(1000);

    }
```

```
struct SnakePart {
    int x, y;
};
struct Snake {
    int length;
    struct SnakePart part[SNAKE_MAX_LEN];
};
struct Snake snake;

struct Food {
    int x, y;
    int consumed;
};
struct Food food[foods];
```

figure 1(right). The loop used to run the main operations of the game

figure 2(left).

- Fill board generates at 25 by 25 board by filling the edges in with #.
- Draw food places each piece of food represented by a + this is run before draw snake so that if a piece of snake overlaps the food the snake ends up being visible.
- Draw snake uses the snakes length and the position of the snake parts. (Figure 2) according to the adjustments from the user input
- Checks the position of the “head of the snake” to see if the game continues
- Prints everything waits 1 second and then starts again if the game is still going

How to Play the Game

Start the code click on the terminal and after the first game screen appears use w,a,s,d to change the direction of the snake pick up the + to grow the snake while avoiding the walls.

Improvements

1. The game generates each board one after another the original idea was that the board would clear itself after every board using “system("cls");” unfortunately this didn’t work for reasons undetermined other methods where tried as well but nun where successful so it was left to generate in this way making it flicker.
2. Adding a way to change the speed to add difficulty would make it more engaging
3. Instead of running it in a terminal if I did it again id look at using the resources to have a graphic output.

Conclusion

The game works relatively smoothly but uses windows specific libraries which reduce its availability. The improvements allow room for growth and a mild increase in complexity.