Snake Game V2

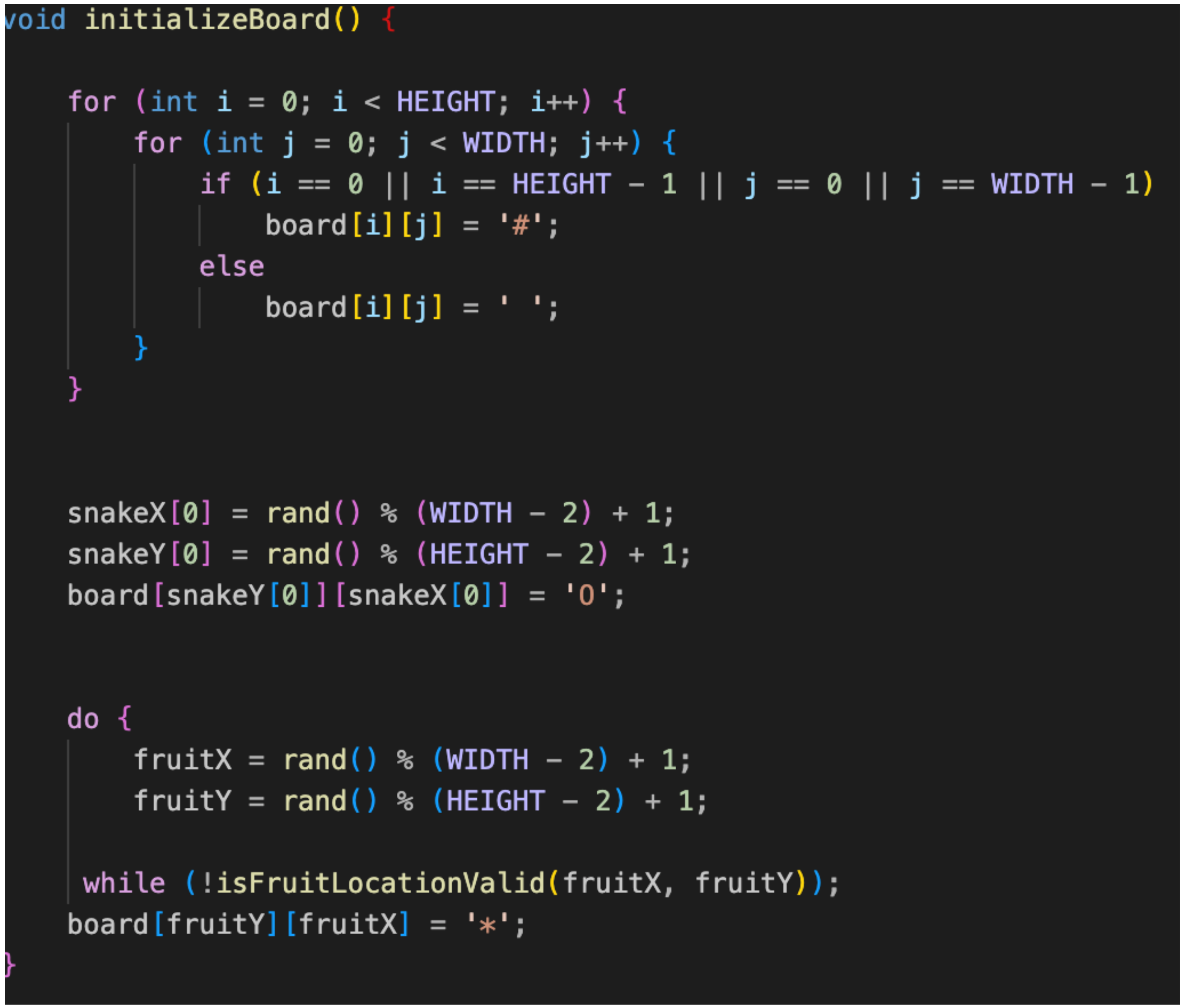
By Nikunj and Melvince

Initialization

- This section of the code focuses on setting the board using a 2d array

- The snake's position is randomly set using the RAND command

- This all occurs within a for-loop

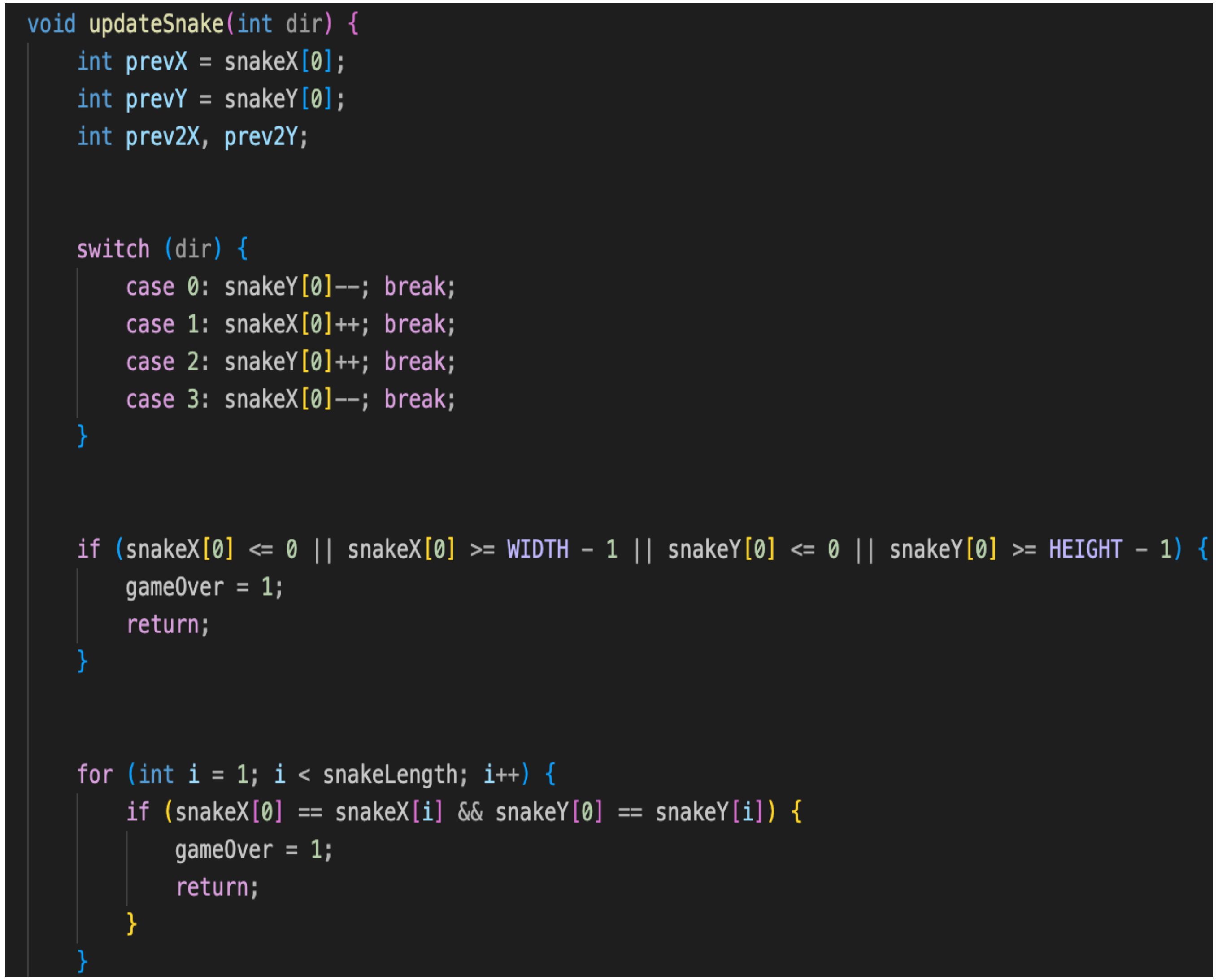


```
void initializeBoard() {

    for (int i = 0; i < HEIGHT; i++) {
        for (int j = 0; j < WIDTH; j++) {
            if (i == 0 || i == HEIGHT - 1 || j == 0 || j == WIDTH - 1)
                board[i][j] = '#';
            else
                board[i][j] = ' ';
        }
    }


    snakeX[0] = rand() % (WIDTH - 2) + 1;
    snakeY[0] = rand() % (HEIGHT - 2) + 1;
    board[snakeY[0]][snakeX[0]] = 'O';


    do {
        fruitX = rand() % (WIDTH - 2) + 1;
        fruitY = rand() % (HEIGHT - 2) + 1;

     while (!isFruitLocationValid(fruitX, fruitY));
    board[fruitY][fruitX] = '*';
}
```

Snake Mechanics

- Using the switch dir. function game updates the coordinates of the snake head based on the input

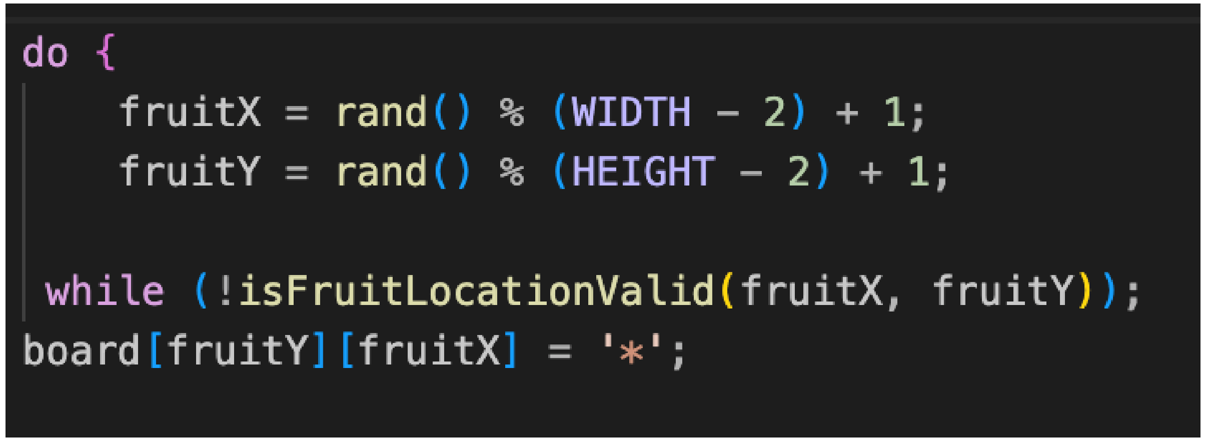
- If statement ensures the game is over when the x or y position of snake is upon boundary



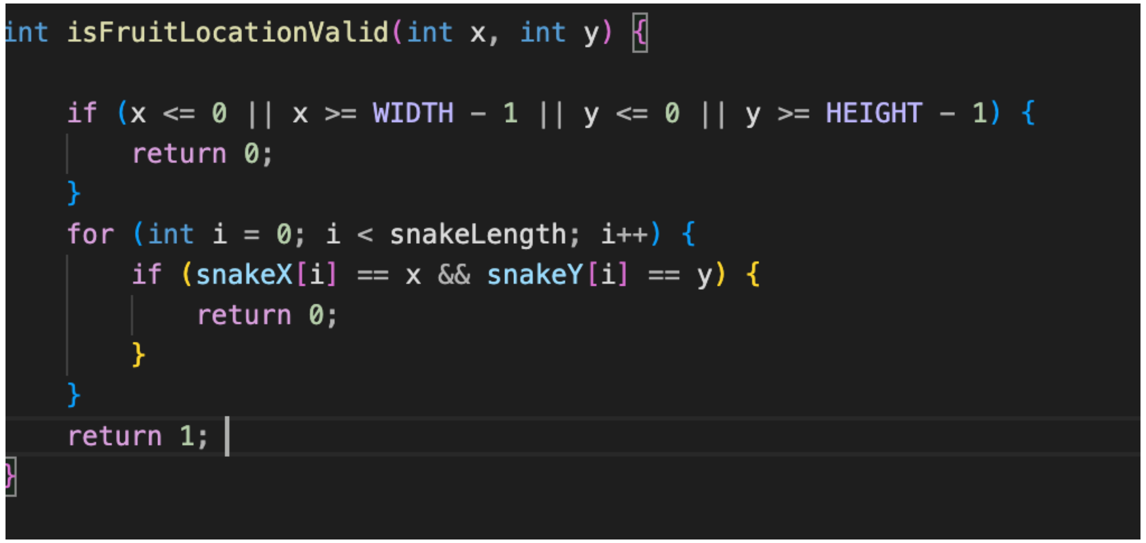
```c
void updateSnake(int dir) {
    int prevX = snakeX[0];
    int prevY = snakeY[0];
    int prev2X, prev2Y;


    switch (dir) {
        case 0: snakeY[0]--; break;
        case 1: snakeX[0]++; break;
        case 2: snakeY[0]++; break;
        case 3: snakeX[0]--; break;
    }


    if (snakeX[0] <= 0 || snakeX[0] >= WIDTH - 1 || snakeY[0] <= 0 || snakeY[0] >= HEIGHT - 1) {
        gameOver = 1;
        return;
    }


    for (int i = 1; i < snakeLength; i++) {
        if (snakeX[0] == snakeX[i] && snakeY[0] == snakeY[i]) {
            gameOver = 1;
            return;
        }
    }
```

Food Generation

- Fruit is placed using a do while loop, RAND function ensures it is placed in random locations
- A separate loop checks to make sure food is not placed on the snake itself (SECOND PIC)



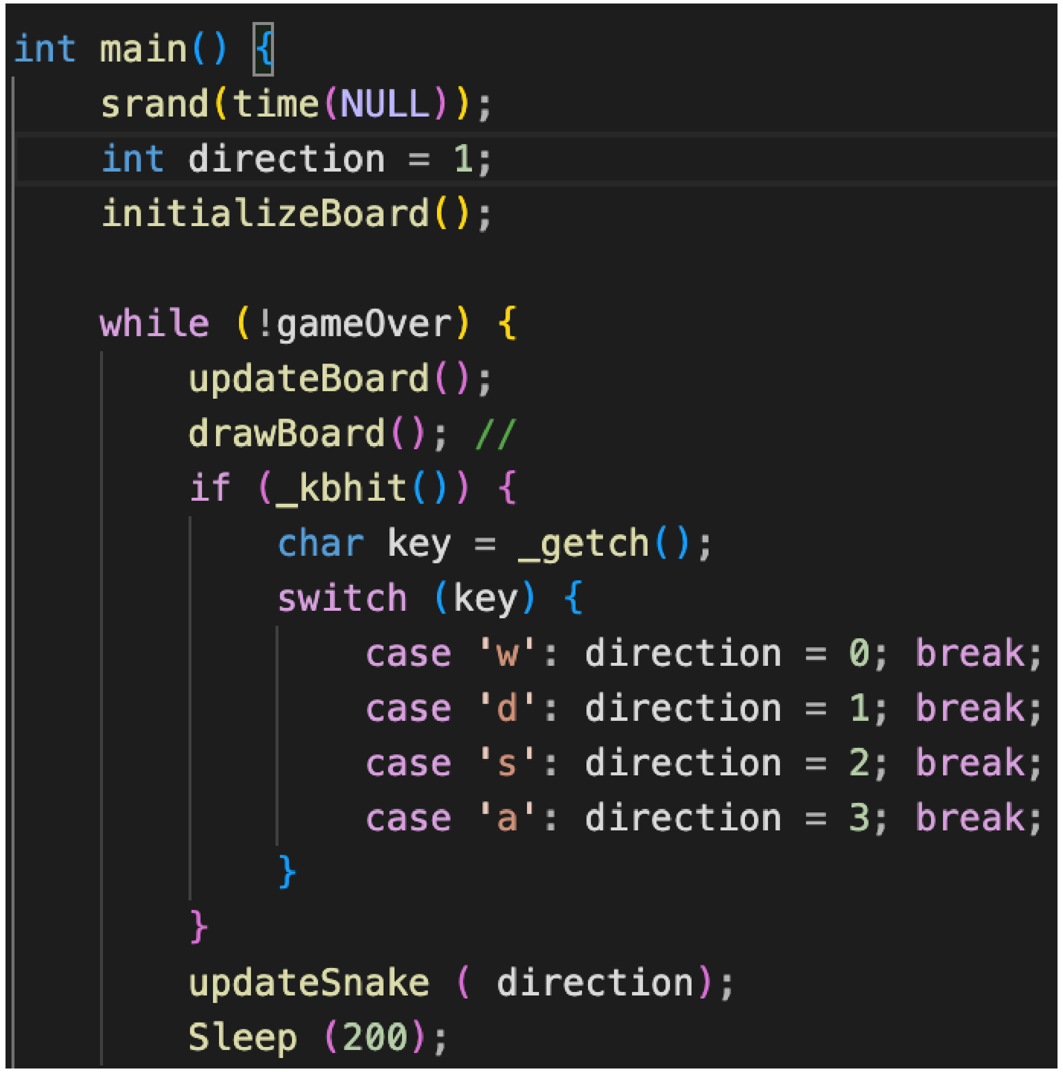
```c
do {
    fruitX = rand() % (WIDTH - 2) + 1;
    fruitY = rand() % (HEIGHT - 2) + 1;

 while (!isFruitLocationValid(fruitX, fruitY));
board[fruitY][fruitX] = '*';
```



```c
int isFruitLocationValid(int x, int y) {

    if (x <= 0 || x >= WIDTH - 1 || y <= 0 || y >= HEIGHT - 1) {
        return 0;
    }
    for (int i = 0; i < snakeLength; i++) {
        if (snakeX[i] == x && snakeY[i] == y) {
            return 0;
        }
    }
    return 1;
}
```

Main Loop

- This section represents the main gameboard, it is responsible for updating, redrawing, and user input
- This will continue to run only until a collision occurs



```
int main() {
    srand(time(NULL));
    int direction = 1;
    initializeBoard();

    while (!gameOver) {
        updateBoard();
        drawBoard(); //
        if (_kbhit()) {
            char key = _getch();
            switch (key) {
                case 'w': direction = 0; break;
                case 'd': direction = 1; break;
                case 's': direction = 2; break;
                case 'a': direction = 3; break;
            }
        }
        updateSnake ( direction);
        Sleep (200);
```

Conclusion

- This version of the Snake Game achieves all the objectives required
- Movement, collision detection, growth and food mechanics are all included using a multitude of functions

ISSUES FACED:

THANK YOU!

Do you have any questions?

hello@mail.com
555-111-222
mydomain.com