```c
#include <stdio.h>
#include <windows.h>
#include <time.h>
#define screen_x 80
#define screen_y 25
#define scount 20
HANDLE rHnd;
HANDLE wHnd;
DWORD fdwMode;
CHAR_INFO consoleBuffer[screen_x * screen_y];
COORD bufferSize = { screen_x, screen_y };
COORD characterPos = { 0,0 };
SMALL_RECT windowSize = { 0, 0, screen_x - 1, screen_y - 1 };
COORD star[scount];
char ship[10] = "<-0->";
int life = 10, color=7;

int setConsole(int x, int y)
{
    wHnd = GetStdHandle(STD_OUTPUT_HANDLE);
    SetConsoleWindowInfo(wHnd, TRUE, &windowSize);
    SetConsoleScreenBufferSize(wHnd, bufferSize);
    return 0;
}

int setMode()
{
    rHnd = GetStdHandle(STD_INPUT_HANDLE);
    fdwMode = ENABLE_EXTENDED_FLAGS | ENABLE_WINDOW_INPUT |
        ENABLE_MOUSE_INPUT;
    SetConsoleMode(rHnd, fdwMode);
    return 0;
}

void clear_buffer() {
    for (int i = 0; i < screen_x * screen_y; i++) {
        consoleBuffer[i].Char.AsciiChar = ' ';
    }
}

void fill_buffer_to_console()
{
    WriteConsoleOutputA(wHnd, consoleBuffer, bufferSize, characterPos,
        &windowSize);
}

void init_star()
{
    for (int i = 0; i < scount; i++) {
        star[i].X = rand() % 80;
        star[i].Y = rand() % 10;
    }
}

void star_fall()
{
    int i;
    for (i = 0; i < scount; i++) {
        if (star[i].Y >= screen_y - 1) {
            star[i].X = (rand() % 80);
            star[i].Y = 0;
        }
        else {
            star[i].Y = star[i].Y + 1 ;
        }
    }

}

void fill_star_to_buffer()
{

    for (int i = 0; i < scount; i++) {
        consoleBuffer[star[i].X + star[i].Y * screen_x].Char.AsciiChar = '*';
        consoleBuffer[star[i].X + star[i].Y * screen_x].Attributes = 7;
    }
}

void fill_life(int life) {
    if (life > 9) {
        consoleBuffer[78].Char.AsciiChar = '1';
        consoleBuffer[78].Attributes = 7;
        consoleBuffer[79].Char.AsciiChar = '0';
        consoleBuffer[79].Attributes = 7;
    }
    else {
        consoleBuffer[79].Char.AsciiChar = life + 48;
        consoleBuffer[79].Attributes = 7;
    }

}

void draw_ship(int x, int y) {
    for (int i = 0; i < 5; i++) {
        consoleBuffer[x + screen_x * y + i].Char.AsciiChar = ship[i];
        consoleBuffer[x + screen_x * y + i].Attributes = color;
    }

}

void check_colission(int x, int y) {
    for (int i = 0; i < scount; i++) {
        if (star[i].Y == y && star[i].X >= x && star[i].X <= x+4) {
            star[i].X = (rand() % 80);
            star[i].Y = 0;
            life -= 1;
        }
    }
}


int main()
{
    int posy=25, posx=0;
    bool play = true;
    DWORD numEvents = 0;
    DWORD numEventsRead = 0;
    setConsole(screen_x, screen_y);
    setMode();
    srand(time(NULL));
    init_star();

    while (play)
    {
        GetNumberOfConsoleInputEvents(rHnd, &numEvents);
        if (numEvents != 0)
        {

            INPUT_RECORD* eventBuffer = new INPUT_RECORD[numEvents];
            ReadConsoleInput(rHnd, eventBuffer, numEvents, &numEventsRead);
            for (DWORD i = 0; i < numEventsRead; ++i)
            {
                if (eventBuffer[i].EventType == KEY_EVENT &&
                    eventBuffer[i].Event.KeyEvent.bKeyDown == true)
                {
                    if (eventBuffer[i].Event.KeyEvent.wVirtualKeyCode == VK_ESCAPE)
                    {
                        play = false;
                    } else if (eventBuffer[i].Event.KeyEvent.uChar.AsciiChar == 99) {
                        color = 1 + rand() % 9;
                    }
                }
                else if (eventBuffer[i].EventType == MOUSE_EVENT)
                {
                    posx = eventBuffer[i].Event.MouseEvent.dwMousePosition.X;
                    posy = eventBuffer[i].Event.MouseEvent.dwMousePosition.Y;
                    if (eventBuffer[i].Event.MouseEvent.dwButtonState &&
                        FROM_LEFT_1ST_BUTTON_PRESSED) {
                        color = 1 + rand() % 9;
                    }
                    if (eventBuffer[i].Event.MouseEvent.dwEventFlags & MOUSE_MOVED)
                    {
                        check_colission(posx, posy);
                    }
                }
            }
            delete[] eventBuffer;
        }
        clear_buffer();
        star_fall();
        check_colission(posx, posy);
        draw_ship(posx, posy);
        fill_star_to_buffer();
        fill_life(life);
        fill_buffer_to_console();
        if (life <= 0) {
            clear_buffer();
            fill_buffer_to_console();
            printf("GAME OVER");
            return 0;
        }
        Sleep(100);
    }
    return 0;
}
```


C:\code\project\lab8\x64\Debug\lab8.exe
9
<-0->