



گزارش کار مینی پروژه

**تدوین گران :**

متین امیرپناه فر40212358003\_

نیما مخملی40212358035\_

زمستان 1402

# فهرست

# [**معرفی بازی -----------------------------------------------------**](#_معرفی_بازی)**3**

# [**ساختار پروژه ----------------------------------------------------5**](#_ساختار_پروژه)

## [**کتابخانه‌ها----------------------------------------------------5**](#_کتابخانه‎ها)

### [**iostream-------------------------------------------------------------5**](#_<iostream>)

### [**windows.h----------------------------------------------------------5**](#_<windows.h>)

### [**ctime--------------------------------------------------------------5**](#_<ctime>)

## [**تعریف‌ها-----------------------------------------------------6**](#_تعریف_ها)

## [**ساختمان‌ها -----------------------------------------------------------------6**](#_ساختمان‌ها)

### [**SpaceShip-----------------------------------------------------------**](#_Spaceship)**6**

## [**توابع ----------------------------------------------------------------------7**](#_توابع)

### [**gameRun-----------------------------------------------------------7**](#_gameRun())

### [**horizontalDraw-------------------------------------------------------8**](#_horizontalDraw())

### [**grandDraw----------------------------------------------------------9**](#_granddraw())

### [**action-------------------------------------------------------------10**](#_Action())

### [**Move-------------------------------------------------------------11**](#_Move())

### [**lossHealth---------------------------------------------------------13**](#_lossHealth())

### [**Shoot-------------------------------------------------------------14**](#_Shoot())

# [**منابع و پیوند‌ها--------------------------------------------------16**](#_پیوندها_و_منابع)

# معرفی بازی

بازی به صورت انفرادی در یک زمین ۱۰×۱۰ انجام می‌شود.

در سطر اول نام بازی درج شده است.

بازیکن در بازی دارای یک سفینه است که با کاراکتر# در صفحه مشخص می‌شود که قابلیت حرکت و یا شلیک در بازی را دارا است و سفینه بازیکن دارای سه فرصت بازی می‌باشد که به نام سلامتی در سطر دوم بازی مشخص شده است.

در صفحه بازی ده دشمنان وجود دارند که با کاراکتر(\*) مشخص شده‌اند.



سفینه با دستور حرکت(m) و سپس با تعیین کردن مسیر حرکت خود به راست(r)،چپ(l)،بالا(u) و یا پایین(d) می‌تواند موقعیت خود در زمین بازی را تغییر دهد در صورتی که سفینه به اشتباه بر روی یکی از سفینه‌های دشمن قرار بگیرد از سلامتی سفینه کم شده و پیغام “!—ops--” چاپ می‌شود و زدن کلید اینتر سفینه به مکان شروع بازی باز می‌گردد.

up or down or left or right[u/d/l/r]:

با دستور شلیک(s) و سپس تعیین جهت شلیک به راست(r) یا چپ(l) سفینه می‌تواند سفینه‌‌ای از دشمن که در جهت شلیک قرار دارد را نابود کند.

shoot right or left[r/l]:

در صورتی که دستور تعریف نشده‌ای به بازی داده شود پیغام زیر چاپ می شود و بعد از کلید اینتر منتظر دستور بعدی باقی می ماند

"undefined please enter to try again"

بازی تا زمانی ادامه پیدا می‌کند که سفینه پیروز شود و یا شکست بخورد.

سفینه زمانی موفق می‌شود به پیروزی دست یابد که تمامی سفینه‌های دشمن موجود در نقشه را نابود کند و پس از پیروزی پیغام زیر چاپ می شود.



در صورتی که بازیکن تمام سلامتی خود را از دست بدهد شکست می خورد و پیغام زیر چاپ می شود.



# ساختار پروژه

بـرنـامـه بـه صـورت مـاژولار و از توابع و ساختمان‌های جـداگـانـه طراحی شده تـا قـابلیت بهینه سازی و تـوسـعه را داشته باشد.

## کتابخانه‎ها

### **<iostream>**

#### این کتابخانه کلاس‌ها و توابع کنترل ورودی و خروجی در ترمینال را در اختیار ما قرار می‌دهد.

### **<windows.h>**

این کتابخانه دستورات کلی کار با ترمینال ویندوز را در اختیار ما قرار می‌دهد.

توابع استفاده شده از این کتابخانه موارد زیر است.

**Sleep()**

مسئولیت ایجاد تاخیر زمانی در برنامه را دارا است.

**System()**

پاک کردن صفحه ترمینال را با پارامتر cls و ایجاد توقف را با پارامتر pause را انجام می‌دهد.

### **<ctime>**

از دستور time(0) را برای دادن دانه اولیه به تابع srand() استفاده شده است.

## تعریف ها

ما در این پروژه از ثابت‌هایی با کلمات کلیدی #define استفاده کرده ایم تعدادی از این ثابت‌ها رنگ ها و برخی پررنگ‌های آنان است و سه ثابت دیگر مختص به صفحه ده در ده بازی است. row و col که برای نمایش صفحه هستند را مقدار 10 داده ایم و همچنین تعداد دشمنان ما که به صورت رندوم چیده خواهند شد هم مقدار 10 قرار داده‌ایم.

#define RESET   "\033[0m"

#define BLACK   "\033[30m"      /\* Black \*/

#define RED     "\033[31m"      /\* Red \*/

#define GREEN   "\033[32m"      /\* Green \*/

#define YELLOW  "\033[33m"      /\* Yellow \*/

#define BLUE    "\033[34m"      /\* Blue \*/

#define MAGENTA "\033[35m"      /\* Magenta \*/

#define CYAN    "\033[36m"      /\* Cyan \*/

#define WHITE   "\033[37m"      /\* White \*/

#define BOLDBLACK   "\033[1m\033[30m"      /\* Bold Black \*/

#define BOLDRED     "\033[1m\033[31m"      /\* Bold Red \*/

#define BOLDGREEN   "\033[1m\033[32m"      /\* Bold Green \*/

#define BOLDYELLOW  "\033[1m\033[33m"      /\* Bold Yellow \*/

#define BOLDBLUE    "\033[1m\033[34m"      /\* Bold Blue \*/

#define BOLDMAGENTA "\033[1m\033[35m"      /\* Bold Magenta \*/

#define BOLDCYAN    "\033[1m\033[36m"      /\* Bold Cyan \*/

#define BOLDWHITE   "\033[1m\033[37m"      /\* Bold White \*/

## ساختمان‌ها

#### Spaceship

این ساختمان داده‌های سفینه ما را از جمله موقعیت اولیه،موقعیت کنونی و سلامتی را نگهداری می‌کند.

struct *spaceship*

{

    int xFirst;

    int yFirst;

    int x, y;

    char c = '#';

    int health = 3;

};

## توابع

### **gameRun()**

این تابع وظیفه ایجاد مقدمات بازی را برعهده دارد ,و پیشنیازها برنامه را برای بازی کردن کاربر آماده می‌کند.

void gameRun(int *condition*[col][row],*spaceship* &*mySpaceShip*){

*mySpaceShip*.xFirst = rand()%col;

*mySpaceShip*.yFirst = rand()%row;

*condition*[*mySpaceShip*.xFirst][*mySpaceShip*.xFirst] = 1;

*mySpaceShip*.x = *mySpaceShip*.xFirst;

*mySpaceShip*.y = *mySpaceShip*.yFirst;

    for (int i = 0;i < NumberOfEnemys;i++)

    {

        while (true)

        {

            int x = rand()%col,y = rand()%row;

            if (*condition*[x][y] != 0)

                continue;

            else

            {

*condition*[x][y] = 2;

                break;

            }

        }

    }

    for (int i = 0; i < 32; i++)

    {

        cout << YELLOW << "\*" << RESET;

        Sleep(5);

    }

    cout << YELLOW << "STARWARS" << RESET;

    for (int i = 0; i < 32; i++)

    {

        cout << YELLOW << "\*" << RESET;

        Sleep(5);

    }

    system("cls");

}

### **horizontalDraw()**

این تابع وظیفه طراحی قسمت افقی صفحه بازی را دارد.

void horizontalDraw(void)

{

    cout << BOLDBLUE;

    for (int i=0;i < col;i++)

        cout << " ---";

    cout << RESET;

}

### **granddraw()**

این تابع وظیفه طراحی صفحه بازی را بر عهده داردکه کاربر میتواند با اعمال دستورات سفینه را روی آن به حرکت یا حمله وا دارد.

void grandDraw(int *condition*[col][row],*spaceship* &*mySpaceShip*)

{

    system("cls");

    bool win = false;

    cout << BOLDGREEN <<"\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*STARWARS\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*" << RESET;

    cout << "\nhealth:" << RED << *mySpaceShip*.health << RESET << endl;

    for (int i = 0;i < row;i++)

    {

        horizontalDraw();

        cout << endl;

        for (int j = 0;j < col;j++)

        {

            cout << BOLDBLUE <<"| " << RESET;

            if (j == *mySpaceShip*.x &&  i == *mySpaceShip*.y)

                cout << BOLDGREEN << *mySpaceShip*.c << ' ' << RESET;

            else if (*condition*[j][i] == 2)

            {

                cout << BOLDRED << "\* " << RESET;

                win = true;

            }

            else

                cout << "  ";

        }

        cout << BOLDBLUE<< "|" << RESET << endl;

    }

    horizontalDraw();

    if (win == false)

    {

        system("cls");

        cout << BOLDGREEN << "\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*YOU WIN\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*"<< endl << RESET;

        system("pause");

        exit(0);

    }

}

### **Action()**

این تابع وظیفه گرفتن و انتقال دستور از کاربر و تعیین حرکت یا شلیک را بر عهده دارد.

void action(int *condition*[col][row],*spaceship* &*mySpaceShip*)

{

    char moveOrShoot;

    bool flag =true;

    do

    {

        cout << "\nmove or shoot[m/s]:";

        cin >> moveOrShoot;

        switch (moveOrShoot)

        {

        case 'm':

            move(*condition*,*mySpaceShip*);

            flag = true;

            break;

        case 's':

            shoot(*condition*,*mySpaceShip*);

            flag = true;

            break;

        default:

            cout << RED << "undefined please enter to try again" << RESET << endl;

            getchar();

            getchar();

            flag = false;

        }

    } while (flag = false);

}

### **Move()**

این تابع وظیفه گرفتن و انتقال دستور تعیین نوع حرکت را برعهده دارد,کاربر در این بازی میتواند با اعمال دستورات راست(r)،چپ(l)،بالا(u)،پایین (d)سفینه را به حرکت درآورد

void move(int *condition*[col][row],*spaceship* &*mySpaceShip*)

{

    char move;

    bool flag = true;

    do

    {

        cout << "up or down or left or right[u/d/l/r]:";

        cin >> move;

        switch (move)

        {

        case 'u':

            if (*condition*[*mySpaceShip*.x][*mySpaceShip*.y - 1] == 2)

            {

                lossHealth(*condition*,*mySpaceShip*);;

            }

            else if(*mySpaceShip*.y - 1 >= 0 && *mySpaceShip*.y - 1 < row)

            {

*condition*[*mySpaceShip*.x][*mySpaceShip*.y]=0;

*mySpaceShip*.y--;

*condition*[*mySpaceShip*.x][*mySpaceShip*.y]=1;

                flag = true;

            }

            break;

        case 'd':

            if (*condition*[*mySpaceShip*.x][*mySpaceShip*.y + 1] == 2)

            {

                lossHealth(*condition*,*mySpaceShip*);

            }

            else if(*mySpaceShip*.y + 1 >= 0 && *mySpaceShip*.y + 1 < row)

            {

*condition*[*mySpaceShip*.x][*mySpaceShip*.y]=0;

*mySpaceShip*.y++;

*condition*[*mySpaceShip*.x][*mySpaceShip*.y]=1;

                flag = true;

            }

            break;

        case 'r':

            if (*condition*[*mySpaceShip*.x + 1][*mySpaceShip*.y] == 2)

            {

                lossHealth(*condition*,*mySpaceShip*);

            }

            else if(*mySpaceShip*.x + 1 >= 0 && *mySpaceShip*.x + 1 < col)

            {

*condition*[*mySpaceShip*.x][*mySpaceShip*.y]=0;

*mySpaceShip*.x++;

*condition*[*mySpaceShip*.x][*mySpaceShip*.y]=1;

                flag = true;

            }

            break;

        case 'l':

            if (*condition*[*mySpaceShip*.x - 1][*mySpaceShip*.y] == 2)

            {

                lossHealth(*condition*,*mySpaceShip*);

            }

            else if(*mySpaceShip*.x - 1 >= 0 && *mySpaceShip*.x - 1 < col)

            {

*condition*[*mySpaceShip*.x][*mySpaceShip*.y]=0;

*mySpaceShip*.x--;

*condition*[*mySpaceShip*.x][*mySpaceShip*.y]=1;

                flag = true;

            }

            break;

        default:

            cout << RED << "undefined please enter to try again" << RESET << endl;

            getchar();

            getchar();

            flag = false;

        }

    }while (flag == false);

}

### **lossHealth()**

این تابع در صورتی که سفینه در حرکت خود با دشمن برخورد کند یک سلامتی از بازی‌کن کسر می‌کند و سفینه بازی‌کن را به محل شروع بازی بر می‌گرداند.

void lossHealth(int *condition*[col][row],*spaceship* &*mySpaceShip*)

{

*mySpaceShip*.health--;

    cout << RED << "!--ops--" << RESET;

    getchar();

    getchar();

*condition*[*mySpaceShip*.x][*mySpaceShip*.y] = 0;

*mySpaceShip*.x = *mySpaceShip*.xFirst;

*mySpaceShip*.y = *mySpaceShip*.yFirst;

*condition*[*mySpaceShip*.x][*mySpaceShip*.y] = 1;

}

### **Shoot()**

این تابع مسئولیت گرفتن و انتقال دستور شلیک و نابودی سفینه‌های دشمن را برعهده دارد,که کاربر میتواند با دستورs به سمت چپ و راست خود شلیک کند تا در صورت وجود دشمن آن را از بین ببرد.

void shoot(int *condition*[row][col],*spaceship* &*mySpaceShip*)

{

    char shoot;

    bool flag = true;

    do

    {

        int x = *mySpaceShip*.x;

        cout << "\nshoot right or left[r/l]:";

        cin >> shoot;

        switch (shoot)

        {

        case 'r':

            do

            {

                x++;

                if (2 == *condition*[x][*mySpaceShip*.y])

                {

*condition*[x][*mySpaceShip*.y] = 0;

                    break;

                }

            }   while(x < col);

            flag = true;

            break;

        case 'l':

            do

            {

                x--;

                if (2 == *condition*[x][*mySpaceShip*.y])

                {

*condition*[x][*mySpaceShip*.y] = 0;

                    break;

                }

            }while(x >= 0);

            flag = true;

            break;

        default:

            cout << RED << "undefined please enter to try again" << RESET << endl;

            getchar();

            getchar();

            flag = false;

        }

    }while (flag == false);

}

# پیوندها و منابع

Book

cpp-How to program Deitel 1997 edition

website

c++ programming

<https://www.w3schools.com/cpp/default.asp>

https://www.geeksforgeeks.org/c-plus-plus/?ref=gcse

git

<https://faradars.org/courses/fvgit9609-git-github-gitlab>

پیوند گیت پروژه

<https://github.com/Matin0789/StarWar-mini-project-of-FCP-.git>