

StarWars

گزارش کار پروژه نهایی



اسفند ماه ۱۴۰۲ – دانشگاه بوعلی سینا

مبانی کامپیوتر و برنامه سازی

استاد : دکتر بشیری

محمد متین سلیمانی

شماره دانشجویی : ۴۰۲۱۲۳۵۸۰۲۱

Git Hub:

<https://github.com/M-M-Soleimani/StarWars>

Gmail: m.m.soleimani84@gmail.com

بسم الله الرحمن الرحيم

فهرست :

۲	معرفی بازی :
۳	چالش های اجرای پروژه :
۳	چالش ها در سطح پیاده سازی :
۳	چالش های منطقی :
۴	توابع موجود در کد :
۴	تابع main :
۵	تابع Menu :
۶	تابع Game_Mode :
۷	تابع Initializer_Basic :
۸	تابع Initializer_Advanced :
۹	تابع positioning :
۱۰	تابع display :
۱۱	تابع Move_Spaceship :
۱۲	تابع Move_Energy_Spaceship :
۱۳	تابع Shoot :
۱۴	تابع is_dead :
۱۵	تابع Show_information :
۱۶	تابع save :
۱۷	تابع Continue_Game :
۱۸	تابع Run_game_basic :
۱۹	تابع Enemy_is_in :
۲۰	تابع End_Game :

۲۱	تابع user_level :
۲۲	تابع Run_game_Advanced :
۲۳	تابع positioning_Advanced :
۲۴	تابع Save_History :
۲۵	تابع display_History :
۲۶	تابع SpaceShips_Description :
۲۷	کتابخانه های مورد استفاده :
۲۷	کتابخانه ی iostream :
۲۷	کتابخانه ی time.h و stdlib.h :
۲۷	کتابخانه ی windows.h :
۲۷	کتابخانه ی conio.h :
۲۷	کتابخانه ی vector :
۲۷	کتابخانه ی string :
۲۸	کتابخانه fstream :
۲۸	کتابخانه ncurses :
۲۸	برآمد های آموزشی :
۲۸	سوال در استک اور فلو :

معرفی بازی :

در این بازی یک سفینه خودی داریم که در ردیف اول از پایین قرار دارد و میتوان به چپ و راست حرکت کند و به سمت بالا گلوله شلیک کند و دارای سلامتی سه می باشد . در این بازی ما چهار نوع سفینه ی دشمن داریم که در ابعاد و سلامتی دارای تفاوت اند.

بازی در دو مود پیشرفته و پایه ای وجود دارد ، که کاربر حق انتخاب بین این دو را داراست.

بازیکن در ابتدای بازی سائز مپ ، حد نصاب امتیاز را وارد می کند در ادامه در صورت رسیدن به حد نصاب امتیاز بازیکن برنده می شود می تواند به صورت بدون محدودیت بازی را ادامه دهد. در صورت از دست دادن تمامی سلامتی ها شما می بازید.

چالش های اجرای پروژه :

چالش ها در سطح پیاده سازی :

۱. طراحی گرافیکی زمین بازی
۲. ماژولاریتی توابع
۳. عدم استفاده از متغیر های گلوبال
۴. بهینه کد زدن
۵. تعریف دو مود برای بازی
۶. حجم بالای کد

چالش های منطقی :

۷. زمان بر بودن حرکات
۸. ساده بودن بازی

توابع موجود در کد :

تابع **main** :

در این تابع ، تابع منو فراخوانی می کنیم.

```
65  int main()
66  {
67      while (Menu())
68      {
69      }
70      return 0;
71  }
```

تابع Menu:

تابعی که مسئول نمایش منو به کاربر جهت تعامل و ادامه ی روند بازی. در این تابع با استفاده از switch case یه ورودی از کاربر دریافت و او را به مرحله ی بعدی می برد.

این تابع دارای ۵ وضعیت است :

۱. انتخاب مدل بازی
۲. ادامه ی بازی قبلی
۳. اطلاعات سفینه ها و اجزای بازی
۴. تاریخچه ی بازی ها
۵. خروج

```
73 bool Menu()
74 {
75     system("cls || clear"); // This function clears the console
76     bool Invalid_Selection = false;
77     do
78     {
79         // In the next few lines, the menu will be displayed on the console
80         cout << "1- Game Mode " << endl;
81         << "2- Continue Game " << endl;
82         << "3- Description of SpaceShips " << endl;
83         << "4- History" << endl;
84         << "5- Exit " << endl;
85         char user_selection;
86         user_selection = getch(); // We get an option chosen by the user
87         switch (user_selection)
88         {
89             case '1':
90                 if (Game_Mode()) // In this case, the game mode function is called
91                 {
92                     return false;
93                 }
94                 else
95                     return true;
96                 break;
97             case '2':
98                 Continue_Game(); // The continue game function is called
99                 return true;
100                 break;
101             case '3':
102                 SpaceShips_Description(); // The function of the description of spaceships is called
103                 return true;
104                 break;
105             case '4':
106                 display_History(); // call the history function
107                 return true;
108                 break;
109             case '5':
110                 return false;
111                 break;
112             default:
113                 Invalid_Selection = true;
114                 system("cls || clear"); // This function clears the console
115                 cerr << Red << "Invalid Selection !" << Reset << endl; // In this line, if an invalid choice is made by the user, an error will be displayed on the console
116                 break;
117         }
118     } while (Invalid_Selection);
119     return false;
120 }
121
122
123
124
125
126
```

تابع Game_Mode :

تابعی که به کاربر اجازه می ده از بین دو مود بازی یکی را انتخاب کند.
این تابع نیز مانند تابع با از یک switch case تشکیل شده که انتخاب ها به شرح زیر است :

۱. مود پایه ای

۲. مود پیشرفته

۳. خروج

مقدار بازگشتی این تابع درست و نادرست است .

```
127 bool Game_Mode()
128 {
129     system("cls || clear"); // This function clears the console
130     bool Invalid_Selection = false;
131     do
132     {
133         // In the next few lines, the menu will be displayed on the console
134         cout << "1- Basic Mode " << endl
135              << "2- Advanced Mode " << endl
136              << "3- Exit " << endl;
137         char user_selection;
138         user_selection = getch(); // We get an option chosen by the user
139         switch (user_selection)
140         {
141             case '1':
142                 Initializer_Basic(); // call the basic initializer function
143                 return true;
144                 break;
145             case '2':
146                 Initializer_Advanced(); // call the basic Advanced function
147                 return true;
148                 break;
149             case '3':
150                 return false;
151                 break;
152             default:
153                 Invalid_Selection = true;
154                 system("cls || clear"); // This function clears the console
155                 cerr << Red << "Invalid Selection !" << Reset << endl; // In this line, if an invalid choice is made by the user, an error will be displayed on the console
156                 break;
157         }
158     } while (Invalid_Selection);
159     return false;
160 }
```


تابع Initializer_Basic:

این تابع وظیفه‌ی آماده‌سازی برای مود پایه‌ی ای را دارد. این تابع با دریافت ابعاد مپ و حد نصاب امتیاز و تعریف متغیرها بستر را برای فراخوانی تابع Run_game_basic مهیا می‌کند. این تابع مقدار بازگشتی ندارد.

```
165 void Initializer_Basic()
166 {
167     system("cls || clear"); // This function clears the console
168     bool Invalid_Selection = false;
169     int map_size;
170     do
171     {
172         Invalid_Selection = false;
173         cout << "Enter the map size : "; // In these few lines, the dimensions of the playground are received from the user
174         cin >> map_size;
175         if (map_size <= 15)
176         {
177             Invalid_Selection = true;
178             system("cls || clear"); // This function clears the console
179             cerr << Red << "Invalid size ! (Map size should be bigger than 15) " << Reset << endl; // In this line, if an invalid choice is made by the user, an error will be displayed on the console
180         }
181     } while (Invalid_Selection);
182     vector<vector<Map_Components>> map;
183     map.resize(map_size, vector<Map_Components>(map_size)); // Creates a two-dimensional vector with the dimensions of map size * map size
184
185     system("cls || clear"); // This function clears the console
186     cout << "Enter the map size : " << map_size << endl;
187     cout << "Enter the quorum for the win : "; // In these few lines, a quorum of points to win is received from the user
188     int quorum_point;
189     cin >> quorum_point;
190     double point = 0;
191     int spaceship_health;
192     int Spaceship_position;
193     int level;
194     int Spaceship_type;
195     vector<string> Enemies_history;
196     Run_game_basic(map, map_size, quorum_point, point, spaceship_health, Spaceship_position, Enemies_history, level, Spaceship_type); // Calling the run game function
197 }
```

تابع Initializer_Advanced :

این تابع وظیفه ی آماده سازی برای مود پایه ای را دارد. این تابع با دریافت ابعاد مپ ، حد نصاب امتیاز ، تایین نوع سفینه ی خودی و تعریف متغیر ها بستر را برای فراخوانی تابع Run_game_Advanced مهیا می کند.

```
199 void Initializer_Advanced()
200 {
201     system("cls || clear"); // This function clears the console
202     bool Invalid_Selection = false;
203     int map_size;
204     do
205     {
206         Invalid_Selection = false;
207         cout << "Enter the map size : "; // In these few lines, the dimensions of the playground are received from the user
208         cin >> map_size;
209         if (map_size <= 15)
210         {
211             Invalid_Selection = true;
212             system("cls || clear"); // This function clears the console
213             cerr << Red << "Invalid size ! (Map size should be bigger than 15) " << Reset << endl; // In this line, if an invalid choice is made by the user, an error will be displayed on the console
214         }
215     } while (Invalid_Selection);
216     vector<vector<Map_Components>> map;
217     map.resize(map_size, vector<Map_Components>(map_size)); // Creates a two-dimensional vector with the dimensions of map size * map size
218
219     cout << "Enter the quorum for the win : "; // In these few lines, a quorum of points to win is received from the user
220     int quorum_point;
221     cin >> quorum_point;
222     int spaceship_type;
223     Invalid_Selection = false;
224     do
225     {
226         string Spaceship_Type_name;
227         Invalid_Selection = false;
228         // In these few lines, we take the type of spaceship from the user
229         cout << "Select the type of spaceship : " << endl
230              << "1- spaceship 1" << endl
231              << "2- spaceship 2" << endl
232              << "3- spaceship 3" << endl;
233         spaceship_type = getch() - 48;
234         switch (spaceship_type)
235         {
236             case 1:
237                 system("cls || clear"); // This function clears the console
238                 cout << "Select the type of spaceship : spaceship 1 " << endl;
239                 Spaceship_Type_name = "spaceship 1"; // In this line, we assign the name of a type of spaceship to a string variable to be used in subsequent functions.
240                 break;
241             case 2:
242                 system("cls || clear"); // This function clears the console
243                 cout << "Select the type of spaceship : spaceship 2 " << endl;
244                 Spaceship_Type_name = "spaceship 2"; // In this line, we assign the name of a type of spaceship to a string variable to be used in subsequent functions.
245                 break;
246             case 3:
247                 system("cls || clear"); // This function clears the console
248                 cout << "Select the type of spaceship : spaceship 3 " << endl;
249                 Spaceship_Type_name = "spaceship 3"; // In this line, we assign the name of a type of spaceship to a string variable to be used in subsequent functions.
250                 break;
251             default:
252                 Invalid_Selection = true;
253                 system("cls || clear"); // This function clears the console
254                 cerr << Red << "Invalid Selection !" << Reset << endl; // In this line, if an invalid choice is made by the user, an error will be displayed on the console
255                 break;
256         }
257     } while (Invalid_Selection);
258     cout << "Enter the map size : " << map_size << endl;
259     cout << "Enter the quorum for the win : " << quorum_point << endl;
260     double point = 0;
261     int spaceship_health;
262     int Spaceship_position;
263     int level;
264     vector<string> Enemies_history;
265     Run_game_Advanced(map, map_size, quorum_point, point, spaceship_health, Spaceship_position, Enemies_history, level, spaceship_type); // Calling the run game function
266 }
```

تابع positioning :

تابعی که با استفاده از تابع srand با استفاده از seed اولیه ی time شروع به تولید اعداد تصادفی برای تولید دشمن ها و مکان یابی آنها انجام می دهد.

```
101 void positioning(vector<vector<map<string, int>>> &map, int &map_size, int &spaceship_health, vector<string> &spaceship_history)
102 {
103     srand(time(0)); // A function that generates random numbers with an initial seed of time 0
104     unsigned int row, column;
105     bool is_it = false;
106     for (size_t i = 0; i < map_size; i++) // In these few lines, if there is no native ship, a ship will be created
107     {
108         if (map[map_size - 1][i].name == "spaceship")
109         {
110             is_it = true;
111             spaceship_health = map[map_size - 1][i].health;
112             break;
113         }
114     }
115     if (!is_it)
116     {
117         map[map_size - 1][map_size - 1 / 2].name = "spaceship";
118         map[map_size - 1][map_size - 1 / 2].health = 4;
119         map[map_size - 1][map_size - 1 / 2].damage = 200000000; // the largest possible value for an integer data type
120         map[map_size - 1][map_size - 1 / 2].size = 1;
121         map[map_size - 1][map_size - 1 / 2].color = "bright_green";
122         map[map_size - 1][map_size - 1 / 2].character = "[S]";
123         spaceship_health = map[map_size - 1][map_size - 1 / 2].health;
124     }
125     int enemy_type;
126     bool invalid_position = false;
127     enemy_type = rand() % 4; // This function generates random numbers between 0 and 4
128     switch (enemy_type) // This section selects the type of enemy to create in the map
129     {
130         case 0: // In each section of the switch case, a type of enemy is created
131             column = rand() % map_size;
132             map[0][column].name = "hurt";
133             map[0][column].health = 1;
134             map[0][column].damage = 1;
135             map[0][column].size = 1 * 1;
136             map[0][column].color = "bright_pink";
137             map[0][column].character = "[*]";
138             spaceship_history.push_back("hurt");
139             break;
140         case 1:
141             do
142             {
143                 invalid_position = false;
144                 column = rand() % map_size;
145                 if (column + 2 <= map_size)
146                 {
147                     for (size_t i = 0; i < 2; i++)
148                     {
149                         for (size_t j = 0; j < 2; j++)
150                         {
151                             map[i][column + j].name = "strider";
152                             map[i][column + j].health = 2;
153                             map[i][column + j].damage = 1;
154                             map[i][column + j].size = 2 * 2;
155                             map[i][column + j].color = "bright_blue";
156                             map[i][column + j].character = "[*]";
157                         }
158                     }
159                 }
160                 else
161                 {
162                     invalid_position = true;
163                 }
164             } while (invalid_position);
165             spaceship_history.push_back("strider");
166             break;
167         case 2:
168             do
169             {
170                 invalid_position = false;
171                 column = rand() % map_size;
172                 if (column + 4 <= map_size)
173                 {
174                     for (size_t i = 0; i < 4; i++)
175                     {
176                         for (size_t j = 0; j < 4; j++)
177                         {
178                             map[i][column + j].name = "warth";
179                             map[i][column + j].health = 4;
180                             map[i][column + j].damage = 1;
181                             map[i][column + j].size = 4 * 4;
182                             map[i][column + j].color = "blue";
183                             map[i][column + j].character = "[*]";
184                         }
185                     }
186                 }
187                 else
188                 {
189                     invalid_position = true;
190                 }
191             } while (invalid_position);
192             spaceship_history.push_back("warth");
193             break;
194         case 3:
195             do
196             {
197                 invalid_position = false;
198                 column = rand() % map_size;
199                 if (column + 6 <= map_size)
200                 {
201                     for (size_t i = 0; i < 6; i++)
202                     {
203                         for (size_t j = 0; j < 6; j++)
204                         {
205                             map[i][column + j].name = "moshoo";
206                             map[i][column + j].health = 6;
207                             map[i][column + j].damage = 1;
208                             map[i][column + j].size = 6 * 6;
209                             map[i][column + j].color = "magenta";
210                             map[i][column + j].character = "[*]";
211                         }
212                     }
213                 }
214                 else
215                 {
216                     invalid_position = true;
217                 }
218             } while (invalid_position);
219             spaceship_history.push_back("moshoo");
220             break;
221     }
222 }
```

تابع display :

تابعی که مسئول نمایش رنگی اجزای مپ می باشد.

```
398 void display(vector<vector<Map_Components>> map, int map_size)
399 {
400     system("cls || clear"); // This function clears the console
401     for (size_t i = 0; i < map_size; i++)
402     {
403         for (size_t j = 0; j < map_size; j++)
404         {
405             if (map[i][j].color == "Bright_Green") // In the following conditions, the color for printing the map components is specified and they are displayed in the console
406             {
407                 cout << Bright_Green << map[i][j].character << Reset;
408             }
409             else if (map[i][j].color == "Bright_Cyan")
410             {
411                 cout << Bright_Cyan << map[i][j].character << Reset;
412             }
413             else if (map[i][j].color == "Bright_Blue")
414             {
415                 cout << Bright_Blue << map[i][j].character << Reset;
416             }
417             else if (map[i][j].color == "Blue")
418             {
419                 cout << Blue << map[i][j].character << Reset;
420             }
421             else if (map[i][j].color == "Magenta")
422             {
423                 cout << Magenta << map[i][j].character << Reset;
424             }
425             else if (map[i][j].color == "Bright_Red")
426             {
427                 cout << Bright_Red << map[i][j].character << Reset;
428             }
429             else
430             {
431                 cout << White << map[i][j].character << Reset;
432             }
433         }
434         cout << endl;
435     }
436 }
```

تابع Move_Spaceship :

این تابع ابتدا با پیمایش بر روی سطر آخر سفینه ی کاربر را پیدا کرده و سپس با دریافت ورودی از کاربر سفینه را به چپ و راست حرکت می دهد و همچنین این تابع با فشردن space بازی را متوقف و در صورت درخواست کاربر آن را save می کند. مقدار بازگردانی شده ی این تابع یک استرینگ است که با استفاده از شروطی با کاربر به تعامل می پردازد.

```
100 string Move_Spaceship(vector<vector<Map_Component>> &map, int map_size, int spaceship_position, int quorum_point, double point, int spaceship_health, vector<string> &enemies_history, int &level, int spaceship_type)
101 {
102     int User_Selection;
103     User_Selection = getch();
104     int row, column;
105     for (size_t i = 0; i < map_size; i++) // In these few lines we find the position of the spaceship
106     {
107         if (map[map_size - 1][i].name == "Spaceship")
108         {
109             row = map_size - 1;
110             column = i;
111             spaceship_position = column;
112             break;
113         }
114     }
115     switch (User_Selection)
116     {
117     case 100:
118         // If we choose the letter D, we move to the right
119         if ((column + 1) < map_size) // This condition checks not to leave the game map
120         {
121             map[row][column + 1].name = map[row][column].name;
122             map[row][column + 1].health = map[row][column].health;
123             map[row][column + 1].damage = map[row][column].damage;
124             map[row][column + 1].size = map[row][column].size;
125             map[row][column + 1].color = map[row][column].color;
126             map[row][column + 1].character = map[row][column].character;
127
128             map[row][column].name = "empty";
129             map[row][column].health = 0;
130             map[row][column].damage = 0;
131             map[row][column].size = 1 * 1;
132             map[row][column].color = "white";
133             map[row][column].character = " ";
134         }
135         else
136         {
137             cerr << Red << "Invalid Move !" << Reset << endl; // In this line, if an invalid move is made by the user, an error will be displayed on the console
138             Sleep(200); // This function freezes the console for 200 milliseconds
139         }
140         return "The move was successful";
141         break;
142     case 97:
143         // If we choose the letter A, we move to the left
144         if (column - 1 >= 0) // This condition checks not to leave the game map
145         {
146             map[row][column - 1].name = map[row][column].name;
147             map[row][column - 1].health = map[row][column].health;
148             map[row][column - 1].damage = map[row][column].damage;
149             map[row][column - 1].size = map[row][column].size;
150             map[row][column - 1].color = map[row][column].color;
151             map[row][column - 1].character = map[row][column].character;
152
153             map[row][column].name = "empty";
154             map[row][column].health = 0;
155             map[row][column].damage = 0;
156             map[row][column].size = 1 * 1;
157             map[row][column].color = "white";
158             map[row][column].character = " ";
159         }
160         else
161         {
162             cerr << Red << "Invalid Move !" << Reset << endl; // In this line, if an invalid move is made by the user, an error will be displayed on the console
163             Sleep(200); // This function freezes the console for 200 milliseconds
164         }
165         return "The move was successful";
166         break;
167     case 115:
168         // If you press the S key, you will remain without moving
169         break;
170     case 27: // By pressing the esc key, the game is saved and the program is closed
171         save(map, map_size, quorum_point, point, spaceship_health, enemies_history, level, spaceship_type);
172         Save_History(level, point, spaceship_health, quorum_point);
173         return "saved successfully";
174         break;
175     case 32:
176         // In the next few lines, the game will be saved by pressing the space bar, then if you press the S key, the game will be saved and the game will be closed.
177         int user_selection;
178         do
179         {
180             system("cls || clear"); // This function clears the console
181             cout << Bright_Yellow << "Press space to continue the game or press S to save" << Reset << endl;
182             user_selection = getch();
183             if (user_selection == 115)
184             {
185                 save(map, map_size, quorum_point, point, spaceship_health, enemies_history, level, spaceship_type);
186                 Save_History(level, point, spaceship_health, quorum_point);
187                 return "saved successfully";
188                 break;
189             }
190             while (user_selection != 32 && user_selection != 115);
191         } while (true);
192         break;
193     default:
194         cerr << Red << "Invalid Selection !" << Reset << endl; // In this line, if an invalid choice is made by the user, an error will be displayed on the console
195         Sleep(200); // This function freezes the console for 200 milliseconds
196         break;
197     }
198     return "The move was successful";
199 }
```

تابع Move_Enemy_Spaceship :

این تابع با پیمایش مپ تمامی سفینه های دشمن در صورتی که بتوان آنها را جابجا کرد به پایین حرکت می دهد و در صورت اصابت گلوله به آنها سلامتی آنها را کاهش می دهد و یا در صورت برخورد با سفینه کاربر آنها را از بین می برد.

```
157 string Move_Spaceship(vector<vector<map_Component>> &map, int &map_size, int &Spaceship_position, int &quorum_point, double point, int &spaceship_health, vector<string> &Enemies_history, int &level, int &spaceship_type)
158 {
159     int User_Selection;
160     User_Selection = getch();
161     int row, column;
162     for (size_t i = 0; i < map_size; i++) // In these few lines we find the position of the spaceship
163     {
164         if (map[map_size - 1][i].name == "Spaceship")
165         {
166             row = map_size - 1;
167             column = i;
168             Spaceship_position = column;
169             break;
170         }
171     }
172     switch (User_Selection)
173     {
174     case 100:
175         // If we choose the letter D, we move to the right
176         if ((column + 1) < map_size) // This condition checks not to leave the game map
177         {
178             map[row][column + 1].name = map[row][column].name;
179             map[row][column + 1].Health = map[row][column].Health;
180             map[row][column + 1].damage = map[row][column].damage;
181             map[row][column + 1].size = map[row][column].size;
182             map[row][column + 1].color = map[row][column].color;
183             map[row][column + 1].character = map[row][column].character;
184
185             map[row][column].name = "empty";
186             map[row][column].Health = 0;
187             map[row][column].damage = 0;
188             map[row][column].size = 1 * 1;
189             map[row][column].color = "white";
190             map[row][column].character = '|';
191         }
192         else
193         {
194             cerr << Red << "Invalid Move!" << Reset << endl; // In this line, if an invalid move is made by the user, an error will be displayed on the console
195             Sleep(200); // This function freezes the console for 200 milliseconds
196         }
197         return "The move was successful";
198         break;
199     case 97:
200         // If we choose the letter A, we move to the left
201         if (column - 1 >= 0) // This condition checks not to leave the game map
202         {
203             map[row][column - 1].name = map[row][column].name;
204             map[row][column - 1].Health = map[row][column].Health;
205             map[row][column - 1].damage = map[row][column].damage;
206             map[row][column - 1].size = map[row][column].size;
207             map[row][column - 1].color = map[row][column].color;
208             map[row][column - 1].character = map[row][column].character;
209
210             map[row][column].name = "empty";
211             map[row][column].Health = 0;
212             map[row][column].damage = 0;
213             map[row][column].size = 1 * 1;
214             map[row][column].color = "white";
215             map[row][column].character = '|';
216         }
217         else
218         {
219             cerr << Red << "Invalid Move!" << Reset << endl; // In this line, if an invalid move is made by the user, an error will be displayed on the console
220             Sleep(200); // This function freezes the console for 200 milliseconds
221         }
222         return "The move was successful";
223         break;
224     case 115:
225         // If you press the S key, you will remain without moving
226         break;
227     case 27: // By pressing the esc key, the game is saved and the program is closed
228         save(map, map_size, quorum_point, point, spaceship_health, Enemies_history, level, spaceship_type);
229         Save_History(level, point, spaceship_health, quorum_point);
230         return "saved successfully";
231         break;
232     case 32:
233         // In the next few lines, the game will be saved by pressing the space bar, then if you press the S key, the game will be saved and the game will be closed.
234         int user_selection;
235         do
236         {
237             system("cls || clear"); // This function clears the console
238             cout << Bright_Vellow << "Press space to continue the game or press S to save" << Reset << endl;
239             user_selection = getch();
240             if (user_selection == 115)
241             {
242                 save(map, map_size, quorum_point, point, spaceship_health, Enemies_history, level, spaceship_type);
243                 Save_History(level, point, spaceship_health, quorum_point);
244                 return "saved successfully";
245                 break;
246             }
247         } while (user_selection != 32 && user_selection != 115);
248         return "play";
249         break;
250     default:
251         cerr << Red << "Invalid Selection!" << Reset << endl; // In this line, if an invalid choice is made by the user, an error will be displayed on the console
252         Sleep(200); // This function freezes the console for 200 milliseconds
253         break;
254     }
255     return "The move was successful";
256 }
```

تابع Shoot :

این تابع ابتدا با پیمایش مپ تمامی گلوله ها را در صورت امکان به بالا حرکت می دهد و سپس یک گلوله ی جدید ایجاد می کند.

```
639 void Shoot(vector<vector<Map_Components>> &map, int map_size, int &Spaceship_position, int spaceship_type)
640 {
641     for (size_t i = 0; i < map_size; i++) // In the following few lines, if there is a bullet, and if possible, we move the bullet up
642     {
643         for (size_t j = 0; j < map_size; j++)
644         {
645             if (map[i][j].name == "bullet")
646             {
647                 if (i > 0 && map[i - 1][j].name == "empty") // If possible, we move the bullet up
648                 {
649                     map[i - 1][j].name = map[i][j].name;
650                     map[i - 1][j].Health = map[i][j].Health;
651                     map[i - 1][j].damage = map[i][j].damage;
652                     map[i - 1][j].size = map[i][j].size;
653                     map[i - 1][j].color = map[i][j].color;
654                     map[i - 1][j].character = map[i][j].character;
655
656                     map[i][j].name = "empty";
657                     map[i][j].Health = 0;
658                     map[i][j].damage = 0;
659                     map[i][j].size = 1 * 1;
660                     map[i][j].color = "White";
661                     map[i][j].character = "[ ]";
662                 }
663                 else // Otherwise (the bullet is on the edge of the map) we remove it from the map
664                 {
665                     map[i][j].name = "empty";
666                     map[i][j].Health = 0;
667                     map[i][j].damage = 0;
668                     map[i][j].size = 1 * 1;
669                     map[i][j].color = "White";
670                     map[i][j].character = "[ ]";
671                 }
672             }
673         }
674     }
675     // We create a bullet in a few lines below
676     map[map_size - 2][Spaceship_position].name = "bullet";
677     map[map_size - 2][Spaceship_position].Health = 0;
678     map[map_size - 2][Spaceship_position].damage = spaceship_type;
679     map[map_size - 2][Spaceship_position].size = 1;
680     map[map_size - 2][Spaceship_position].color = "Bright_Red";
681     map[map_size - 2][Spaceship_position].character = "[^]";
682 }
```

تابع is_dead:

این تابع در مپ پیمایش می کند و در صورتی که عنصری به سلامتی 0 و یا کمتر رسیده باشد آن را پاک می کند.

```
684 void is_dead(vector<vector<Map_Components>> &map, int map_size)
685 {
686     // In the following few lines, we search all the elements of the map and if the health is less than 0, we delete it
687     for (size_t i = 0; i < map_size; i++)
688     {
689         for (size_t j = 0; j < map_size; j++)
690         {
691             if (map[i][j].Health <= 0 && map[i][j].name != "bullet")
692             {
693                 map[i][j].name = "empty";
694                 map[i][j].Health = 0;
695                 map[i][j].damage = 0;
696                 map[i][j].size = 1 * 1;
697                 map[i][j].color = "White";
698                 map[i][j].character = "[ ]";
699             }
700         }
701     }
702 }
```


تابع Show_information :

این تابع اطلاعاتی از قبیل :

۱. سایز مپ
۲. سلامتی سفینه
۳. حد نصاب امتیاز تایین شده توسط کاربر
۴. امتیاز لحظه ای
۵. لول
۶. و دو راهنمایی جهت save و puse بازی

```
704 void Show_information(int map_size, int quorum_point, double point, int spaceship_health, int level)
705 {
706     // In a few lines below we display the information of the game
707     cout << Yellow << "map size : " << map_size << "*" << map_size << Reset << endl;
708     cout << Green << "health : " << Reset;
709     for (size_t i = 0; i < spaceship_health; i++)
710     {
711         cout << Red << "*" << Reset;
712     }
713     cout << endl;
714     cout << Bright_Cyan << "Quorum of points : " << quorum_point << Reset << endl;
715     cout << Bright_Blue << "point : " << point << Reset << endl;
716     cout << Bright_Yellow << "level : " << level << Reset << endl;
717     cout << Bright_Yellow_new << "Select 'ecs' to exit and save" << Reset << endl;
718     cout << Bright_Yellow_new << "Press space to pause the game" << Reset << endl;
719 }
```

تابع save :

این تابع اطلاعات بازی و موقعیت اجزای بازی را در یک فایل به نام "game.txt" ذخیره می کند.

```
721 void save(vector<vector<Map_Components>> map, int map_size, int quorum_point, double point, int spaceship_health, vector<string> &Enemies_history, int level, int spaceship_type)
722 {
723     ofstream out("game.txt", ios ::out); // We create an output stream with the name "out"
724     // In the following few lines, we save the game information in the file
725     out << map_size << endl;
726     out << quorum_point << endl;
727     out << point << endl;
728     out << spaceship_health << endl;
729     out << level << endl;
730     out << spaceship_type << endl;
731     out << Enemies_history.size() << endl;
732     for (size_t i = 0; i < Enemies_history.size(); i++)
733     {
734         out << Enemies_history[i] << " ";
735     }
736     out << endl;
737     for (size_t i = 0; i < map_size; i++)
738     {
739         for (size_t j = 0; j < map_size; j++)
740         {
741             if (map[i][j].name != "empty")
742             {
743                 out << i << " " << j << " " << map[i][j].name << " " << map[i][j].Health << " " << map[i][j].character << " " << map[i][j].color << " " << map[i][j].damage << " " << map[i][j].size << endl;
744             }
745         }
746     }
747     system("cls || clear"); // This function clears the console
748     out.close();
749 }
```

تابع Continue_Game :

در این تابع با استفاده از فایلی که از بازی های قبلی باقی مانده است ، بازی را ادامه می دهیم.

در ابتدا در صورت باز نشدن فایل ارور مناسبی نمایش داده خواهد شد ، در ادامه در صورت صفر بودن سلامتی یا رسیدن به حد امتیاز اجازه ی ادامه بازی را نخواهید داشت زیرا که بازی به اتمام رسیده است.

```
751 void Continue_Game()
752 {
753     vector<vector<Map_Components>> map;
754     int map_size, quorum_point, spaceship_health;
755     int i, j;
756     double point;
757     int level;
758     int Spaceship_position;
759     int spaceship_type;
760     ifstream in; // We create an input stream with the name "in"
761     in.open("game.txt");
762     if (!in.is_open()) // If there is no appropriate error file, it will be displayed
763     {
764         cerr << Red << "Error opening file !" << Reset << endl;
765         Sleep(1000);
766     }
767     else // In the following few lines, we read the game information from the file
768     {
769         in >> map_size >> quorum_point >> point >> spaceship_health;
770         int Enemies_history_size;
771         in >> level;
772         in >> spaceship_type;
773         in >> Enemies_history_size;
774         vector<string> Enemies_history;
775         Enemies_history.resize(Enemies_history_size);
776         for (size_t i = 0; i < Enemies_history_size; i++)
777         {
778             in >> Enemies_history[i];
779         }
780         map.resize(map_size, vector<Map_Components>(map_size)); // Creates a two-dimensional vector with the dimensions of map size * map size
781         while (in >> i >> j >> map[i][j].name >> map[i][j].Health >> map[i][j].character >> map[i][j].color >> map[i][j].damage >> map[i][j].size) // Read them as long as there is information in the file
782         {
783             if (map[i][j].name == "Spaceship")
784             {
785                 Spaceship_position = j;
786             }
787         }
788         if (point >= quorum_point || spaceship_health <= 0)
789         {
790             cerr << Red << "You can not continue the game that has ended !" << Reset << endl;
791             Sleep(1000);
792         }
793         else
794         {
795             Run_game_basic(map, map_size, quorum_point, point, spaceship_health, Spaceship_position, Enemies_history, level, spaceship_type); // Calling the run game function
796         }
797         in.close();
798     }
799 }
```

تابع Run_game_basic :

این تابع مسئول اعمال صحیح منطق بازی و ایجاد گیم پلی بازی در حالت پایه ای است .

```
003 void Run_game_basic(vector<vector<Map_Components>> &map, int &map_size, int &quorum_point, double &point, int &spaceship_health, int &spaceship_position, vector<string> &Enemies_history, int &level, int spaceship_type)
004 {
005     while (true && !End_Game(quorum_point, point, spaceship_health, Enemies_history, level)) // just for test
006     {
007         if (!Enemy_is_in(map, map_size))
008         {
009             positioning(map, map_size, spaceship_health, Enemies_history); // This function specifies the position of game elements
010         }
011         display(map, map_size); // This function displays the game map
012         Show_information(map_size, quorum_point, &map_size, spaceship_health, level); // call the function that displays game information
013         string temp = Move_Spaceship(map, map_size, Spaceship_position, quorum_point, point, spaceship_health, Enemies_history, level, spaceship_type);
014         if (temp == "saved successfully") // call the Move_Spaceship function
015         {
016             cout << Bright_Green << "Game saved !" << Reset;
017             break;
018         }
019         else if (temp == "play")
020         {
021             continue;
022         }
023         else
024             Move_Enemy_Spaceship(map, map_size, spaceship_health, point, Enemies_history, spaceship_type); // Calling a function to move enemy spaceships
025         is_dead(map, map_size); // Calling a function to check health of map map Components
026         Shoot(map, map_size, Spaceship_position, spaceship_type); // Calling a function to fire bullets
027         is_dead(map, map_size); // Calling a function to check health of map map Components
028         user_level(point, level);
029         save(map, map_size, quorum_point, point, spaceship_health, Enemies_history, level, spaceship_type); // Calling a function to save game data
030     }
031 }
```

تابع Enemy_is_in :

این تابع چک می کند که آیا سفینه ی دشمن در مپ وجود دارد یا خیر.

```
801 > void Run_game_basic(vector<vector<Map_Components>> &map, int &map_size, int &quorum_point, double &point, int &spaceship_health, int &spaceship_position, vector<string> &Enemies_history, int &level, int spaceship_type)
802
803 bool Enemy_is_in(vector<vector<Map_Components>> &map, int &map_size)
804 {
805     // In the following few lines, we navigate the map, if there is an enemy, it returns the true value
806     for (size_t i = 0; i < map_size; i++)
807     {
808         for (size_t j = 0; j < map_size; j++)
809         {
810             if (map[i][j].name == "Dart" || map[i][j].name == "Striker" || map[i][j].name == "Wraith" || map[i][j].name == "Banshee")
811             {
812                 return true;
813             }
814         }
815     }
816     return false;
817 }
```



تابع End_Game :

تابعی که با استفاده از سلامتی ، حد نصاب امتیاز و امتیاز فعلی وضعیت برد و باخت را مشخص می کند .

در صورت برد به کاربر می گوید که می خواهد به صورت بدون محدودیت به بازی ادامه دهد یا خیر.

و در آخر نتیجه را با استفاده از تابع ذخیره ی تاریخچه ذخیره می کند.

```
841 bool End_Game(int quorum_point, double point, int spaceship_health, vector<string> &enemies_history, int level)
842 {
843     // In the following few lines, the number of enemies killed by type is specified
844     int Dart_counter = 0, Striker_counter = 0, Wraith_counter = 0, Banshee_counter = 0;
845     for (auto &enemie_type : Enemies_history)
846     {
847         if (Enemie_type == "Dart")
848         {
849             Dart_counter++;
850         }
851         else if (Enemie_type == "Striker")
852         {
853             Striker_counter++;
854         }
855         else if (Enemie_type == "Wraith")
856         {
857             Wraith_counter++;
858         }
859         else if (Enemie_type == "Banshee")
860         {
861             Banshee_counter++;
862         }
863     }
864
865     static bool infinite_mode = false;
866     if (infinite_mode && spaceship_health > 0)
867     {
868         return false;
869     }
870     if (spaceship_health <= 0)
871     {
872         cout << Bright_Red << " _ " << endl
873              << " | | | " << endl
874              << " | | | W _ | | " << endl
875              << " | | | W _ | | " << endl
876              << " | | | W _ | | " << endl
877              << " | | | W _ | | " << endl;
878
879         cout << Bright_Yellow_new << "List of enemies you have fought with:" << endl
880              << Reset
881              << Bright_Cyan << "Dart : " << Dart_counter << endl
882              << Reset
883              << Bright_Blue << "Striker : " << Striker_counter << endl
884              << Reset
885              << Blue << "Wraith : " << Wraith_counter << endl
886              << Reset
887              << Magenta << "Banshee : " << Banshee_counter << endl
888              << Reset;
889         Save_History(level, point, spaceship_health, quorum_point); // The save history function is called
890         return true;
891     }
892     else if (point >= quorum_point)
893     {
894         cout << Bright_Green
895              << " _ " << endl
896              << " W W W W / | | | | " << endl
897              << " W W W W / | | | | " << endl
898              << " W W W W / | | | | " << endl;
899         cout << Bright_Yellow_new << "If you want to continue playing without restrictions, press y (press any key to finish):" << endl
900              << Reset;
901         if (getch() == 121)
902         {
903             infinite_mode = true;
904         }
905         else
906         {
907             cout << Bright_Yellow_new << "List of enemies you have fought with:" << endl
908                  << Reset
909                  << Bright_Cyan << "Dart : " << Dart_counter << endl
910                  << Reset
911                  << Bright_Blue << "Striker : " << Striker_counter << endl
912                  << Reset
913                  << Blue << "Wraith : " << Wraith_counter << endl
914                  << Reset
915                  << Magenta << "Banshee : " << Banshee_counter << endl
916                  << Reset;
917             Save_History(level, point, spaceship_health, quorum_point); // The save history function is called
918             return true;
919         }
920     }
921     else
922     {
923         return false;
924     }
925 }
```

تابع user_level :

این تابع با یک عمل ریاضی ساده امتیاز رو به لول تبدیل می کند.

```
926 void user_level(int point, int &level)
927 {
928     level = point / 200;
929 }
```

تابع Run_game_Advanced :

این تابع مسئول اعمال صحیح منطق بازی و ایجاد گیم پلی بازی در حالت پایه ای است.

```
939 void Run_game_Advanced(vector<vector<Map_Components>> &map, int &map_size, int &quorum_point, double &point, int &spaceship_health, int &spaceship_position, vector<string> &Enemies_history, int &level, int spaceship_type)
940 {
941     while (true && !End_Game(quorum_point, point, spaceship_health, Enemies_history, level)) // just for test
942     {
943         if (!Enemy_is_in(map, map_size))
944         {
945             positioning_Advanced(map, map_size, spaceship_health, Enemies_history, level); // This function specifies the position of game elements
946         }
947         display(map, map_size); // This function displays the game map
948         Show_information(map_size, quorum_point, point, spaceship_health, level); // call the function that displays game information
949         string temp = Move_Spaceship(map, map_size, Spaceship_position, quorum_point, point, spaceship_health, Enemies_history, level, spaceship_type);
950         if (temp == "saved successfully") // call the Move_Spaceship function
951         {
952             cout << Bright_Green << "Game saved !" << Reset;
953             break;
954         }
955         else if (temp == "play")
956         {
957             continue;
958         }
959         else
960         {
961             Move_Enemy_Spaceship(map, map_size, spaceship_health, point, Enemies_history, spaceship_type); // Calling a function to move enemy spaceships
962             is_dead(map, map_size); // Calling a function to check health of map map Components
963             Shoot(map, map_size, Spaceship_position, spaceship_type); // Calling a function to fire bullets
964             is_dead(map, map_size); // Calling a function to check health of map map Components
965             user_level(point, level);
966             save(map, map_size, quorum_point, point, spaceship_health, Enemies_history, level, spaceship_type); // Calling a function to save game data
967         }
968     }
969 }
```


تابع positioning_Advanced :

تابعی که با استفاده از تابع srand با استفاده از seed اولیه ی time شروع به تولید اعداد تصادفی برای تولید دشمن ها و مکان یابی آنها انجام می دهد.

ولی تفاوت این تابع با تابع positioning در این است که این تابع با در نظر

گرفتن لول دشمن ایجاد می کند.

[illegible]

تابع Save_History :

این تابع اطلاعات هر بازی را به صورت تاریخچه نگهداری می کند.

اطلاعاتی از قبیل :

۱. لول

۲. امتیاز

۳. سلامتی

۴. نتیجه ی بازی

```
1098 void Save_History(int level, double point, int spaceship_health, int quorum_point)
1099 {
1100     ofstream out("History.txt", ios ::out | ios ::app);
1101     out << "level : " << level << " - "
1102         << "point : " << point << " - "
1103         << "health : " << spaceship_health << " - ";
1104     if (point >= quorum_point)
1105     {
1106         out << "win" << endl;
1107     }
1108     else if (spaceship_health <= 0)
1109     {
1110         out << "lose" << endl;
1111     }
1112     else
1113     {
1114         out << "Continue" << endl;
1115     }
1116     out.close();
1117 }
```

تابع display_History :

این تابع مسئول نمایش تاریخچه ی بازی ها است.

```
1119 void display_History()
1120 {
1121     ifstream in;
1122     in.open("History.txt");
1123     string data;
1124     cout << Bright_Yellow_new << "History of games : " << Reset << endl;
1125     while (getline(in, data, '\n'))
1126     {
1127         cout << data << endl;
1128     }
1129     cout << Bright_Yellow_new << "Press one of the keys to return to the menu !" << Reset << endl;
1130     int temp = getch();
1131     in.close();
1132 }
```

تابع SpaceShips_Description :

این تابع اطلاعات انواع سفینه ها را نمایش می دهد.

```
1134 void SpaceShips_Description()
1135 {
1136     system("cls || clear"); // This function clears the console
1137     cout << Bright_Green << "SpaceShips : " << endl
1138         << "Health : 3 " << endl
1139         << "damage : infinity " << endl
1140         << "size : 1*1 " << endl
1141         << "character : [#] " << Reset << endl;
1142     cout << " _____ " << endl;
1143     cout << Bright_Cyan << "Dart : " << endl
1144         << "Health : 1 " << endl
1145         << "damage : 1 " << endl
1146         << "size : 1*1 " << endl
1147         << "character : [*] " << Reset << endl;
1148     cout << " _____ " << endl;
1149     cout << Bright_Blue << "Striker : " << endl
1150         << "Health : 2 " << endl
1151         << "damage : 1 " << endl
1152         << "size : 2*2 " << endl
1153         << "character : [*][*] " << endl
1154         << "          [*][*] " << Reset << endl;
1155     cout << " _____ " << endl;
1156     cout << Blue << "Wraith : " << endl
1157         << "Health : 4 " << endl
1158         << "damage : 1 " << endl
1159         << "size : 3*3 " << endl
1160         << "character : [*][*][*] " << endl
1161         << "          [*][*][*] " << endl
1162         << "          [*][*][*] " << Reset << endl;
1163     cout << " _____ " << endl;
1164     cout << Magenta << "Banshee : " << endl
1165         << "Health : 6 " << endl
1166         << "damage : 1 " << endl
1167         << "size : 4*4 " << endl
1168         << "character : [*][*][*][*] " << endl
1169         << "          [*][*][*][*] " << endl
1170         << "          [*][*][*][*] " << endl
1171         << "          [*][*][*][*] " << Reset << endl;
1172     cout << " _____ " << endl;
1173     cout << Bright_Yellow_new << "Press one of the keys to return to the menu !" << Reset << endl;
1174     int temp = getch();
1175 }
```

کتابخانه های مورد استفاده :

کتابخانه ی **iostream** :

برای خواندن از ورودی و نوشتن در خروجی و به صورت کلی ارتباط با کاربر.

کتابخانه ی **time.h** و **stdlib.h** :

برای ایجاد seed برای تابع rand() که اعداد تصادفی تولید کند.

کتابخانه ی **windows.h** :

برای استفاده از دستورات سیستمی مانند cls که باعث پاک شدن cmd می شود.

کتابخانه ی **conio.h** :

برای استفاده از دستوراتی مانند getch و...

در واقع یه هدر غیر از استاندارد اصلی برای دریافت وردی و خروجی در کنسول است.

کتابخانه ی **vector** :

برای استفاده از وکتور و توابع وابسته به آن.

کتابخانه ی **string** :

برای استفاده از استرینگ ها و توابع وابسته به آن.

کتابخانه **fstream** :

برای ایجاد یک جریان ورودی خروجی بر روی فایل است.

کتابخانه ی **ncurses** :

برای سازگاری با سیستم عامل لینوکس

برآمد های آموزشی :

۱. یادگیری ماژولار نویسی
۲. یادگیری نحوه ی ایجاد سازگاری با سیستم عامل های متفاوت
۳. یادگیری استفاده از توابع رندوم با seed اولیه ی زمانی
۴. استفاده از دستورات سیستمی
۵. یادگیری نحوه ی کامنت گذاری
۶. طراحی محیطی گرافیکی در کنسول
۷. مدیریت استفاده از متغیر های گلوبال
۸. یادگیری تعریف و استفاده از ساختمان های داده
۹. یادگیری فراخوانی توابع با استفاده از پوینتر
۱۰. یادگیری استفاده از گیت هاب
۱۱. یادگیری ایجاد جریان ورودی و خروجی روی فایل ها

سوال در استک اور فلو :

<https://stackoverflow.com/questions/78088750/how-to-specify-in-c-to-perform-a-predetermined-task-after-a-certain-time-and-n>