گزارش کار پروژه بازی Covid 2030 **درس مبانی کامپیوتر**

استاد راهنما: دکتر داوود ابادی

اعضای تیم:

مانی مستعلی – عماد معمار

ترم پاییز ۱۴۰۲

روند طراحی پروژه

- بخش ابتدایی بازی شامل طراحی منوی اولیه بود که عبارت COVID 2030 با کاراکتر # در ابتدا به رنگ قرمز نمایش داده شده و سپس منوی بازی در پایین آن میاید.
- ضمنا منوی اولیه را به صورت تابع void نوشتیم تا بتوانیم هرجا در ادامه برنامه لازم بود, ان را call کنیم که به نام printMenu() تعریف شده است.

C:\Users\Salam\Downloads\project final.cpp - Embarcadero Dev-C++ 6.3

File Edit Search View Project Execute Tools AStyle Window Help

TDM-GCC 9.2.0 64-bit Release

(globals)

Project Clas

project final.cpp

```
385
386     DWORD mode;
387     GetConsoleMode(hConsole, &mode);
388
389     mode |= ENABLE_VIRTUAL_TERMINAL_PROCESSING;
390
391     SetConsoleMode(hConsole, mode);
392
393     string RESET="\033[0m";
394     string RED="\u001b[31m";
395     cout<<RED<<" ######    #######   ##       ##  #### ########          #######    #####     #######    #####"<<RESET<<endl;
396     cout<<RED<<"##     ## ##     ##  ##       ##   ##  ##     ##        ##     ##  ##   ##   ##     ##  ##   ##"<<RESET<<endl;
397     cout<<RED<<"##        ##     ##  ##       ##   ##  ##     ##               ## ##     ##         ## ##     ##"<<RESET<<endl;
398     cout<<RED<<"##        ##     ##  ##      ##    ##  ##     ##         ######   ##     ##   #######  ##     ##"<<RESET<<endl;
399     cout<<RED<<"##        ##     ##   ##    ##     ##  ##     ##        ##        ##     ##         ## ##     ##"<<RESET<<endl;
400     cout<<RED<<"##     ## ##     ##    ##  ##      ##  ##     ##        ##         ##   ##   ##     ##  ##   ##"<<RESET<<endl;
401     cout<<RED<<" ######    #######     ###      #### ########          #########   #####     #######    #####"<<RESET<<endl;
402     cout<<"Welcome to Covid2030 game!"<<endl;
403     cout<<endl;
404
405     // delcaring neccessary variables
406     int level=1;
407     int Vaccine=0;
408     int Credit=0;
409     int Round=0;
410     int Kill=0;
411     int Gunammo=3;
412     int pocketammo=1;
```

Compiler Resources Compile Log Debug Find Results Console

Line: 392 Col: 1 Sel: 0 Lines: 1300 Length: 54719 Insert Done parsing in 6.109 seconds

Type here to search

3:56 PM
1/30/2024

طراحی ظاهری پروژه

- نکته قابل توجه در این برنامه این است که برای نمایش آیتم ها در صفحه بازی از دستورات زیر استفاده است. البته این بخش یک چالش بزرگ داشت. آن هم اینکه به طور عادی تنها مختص به لینوکس بودند و در کامپایلر های ویندوزی اجرا نمی شد. پس از تحقیق و بررسی, متوجه شدیم که دستورات و setup های اولیه ای برای اینکه بتوان در هر کامپایلری آن را اجرا نمود باید نوشته شود.

```
//enable ansi codes on windows consoles
HANDLE hConsole = GetStdHandle(STD_OUTPUT_HANDLE);

DWORD mode;
GetConsoleMode(hConsole, &mode);

mode |= ENABLE_VIRTUAL_TERMINAL_PROCESSING;

SetConsoleMode(hConsole, mode);

string RESET="\033[0m";
string RED="\u001b[31m";
cout<<RED<<" ######    #######   ##       ##  #### ########         #######    #####     #######    #####"<<RESET<<endl;
cout<<RED<<"##     ## ##     ##  ##       ##   ##  ##     ##       ##     ##  ##   ##   ##     ##  ##   ##"<<RESET<<endl;
cout<<RED<<"##        ##     ##  ##       ##   ##  ##     ##              ## ##     ##         ## ##     ##"<<RESET<<endl;
cout<<RED<<"##        ##     ##  ##      ##    ##  ##     ##         ######   ##     ##   #######  ##     ##"<<RESET<<endl;
cout<<RED<<"##        ##     ##   ##    ##     ##  ##     ##        ##        ##     ##         ## ##     ##"<<RESET<<endl;
cout<<RED<<"##     ## ##     ##    ##  ##      ##  ##     ##        ##         ##   ##   ##     ##  ##   ##"<<RESET<<endl;
cout<<RED<<" ######    #######      ###       #### ########         #########   #####     #######    #####"<<RESET<<endl;
cout<<"Welcome to Covid2030 game!"<<endl;
cout<<endl;
```

مطابق آنچه در اینجا آمده است خطوط اول تا هشتم برای کراس پلتفرم کردن نمایش رنگ ها است که بعد از تحقیق و بررسی به دست آمده است. اما در اینجا یک چالش دیگر وجود داشت. در بعضی از کامپایلر ها متغیر ENABLE_VIRTUAL_TERMINAL_PROCESSING تعریف نشده است. در این صورت نیاز بود تا آن را در بالای کد تعریف کنیم.

```
#ifndef ENABLE_VIRTUAL_TERMINAL_PROCESSING
#define ENABLE_VIRTUAL_TERMINAL_PROCESSING 0x0004
#endif
```

```
//show
for(int i=0;i<17;i++)
{    for(int j=0;j<16;j++)
    {
        if(game_screen[i][j]=='P')
        {
            string RESET="\033[0m";
            string GREEN = "\033[32m";
            cout<<GREEN<<game_screen[i][j]<<RESET;
        }
        else if(game_screen[i][j]=='D')
        {
            string RESET="\033[0m";
            string MAGNETA="\u001b[35m";
            cout<<MAGNETA<<game_screen[i][j]<<RESET;
        }
        else if(game_screen[i][j]=='Z')
        {
            string RESET="\033[0m";
            string RED="\u001b[31m";
            cout<<RED<<game_screen[i][j]<<RESET;
        }
        else if(game_screen[i][j]=='A')
        {
            string RESET="\033[0m";
            string BLUE="\u001b[34m";
            cout<<BLUE<<game_screen[i][j]<<RESET;
        }
        else if(game_screen[i][j]=='V')
        {
            string RESET="\033[0m";
            string YELLOW="\u001b[33m";
            cout<<YELLOW<<game_screen[i][j]<<RESET;
         }
            else
        cout<<game_screen[i][j];

    }
cout<<endl;
}
```

در این بخش از کد که مربوط به تابع show_game_screen است و برای راه اندازی اولیه صفحه بازی استفاده و جلوتر بیشتر معرفی خواهد شد و همچنین در تابع show_updated_game_screen() که مرتبط با نمایش صفحه بعد از شروع بازی است, برای طراحی ظاهری و استفاده از رنگ ها از کد های ANSI رنگ های مربوطه استفاده شده.

- بعد از پرینت کردن منو, باید انتخاب پلیر را ورودی میگرفتیم و با استفاده از سوییچ کیس,منطق منو را پیاده میکردیم.
- کیس اول مربوط به شروع بازی(new game) است که جلوتر توضیح داده خواهد شد.
- کیس دوم مربوط به تنظیمات(settings) بازی است که شماره مرحله آخر و مرحله کنونی را نمایش می دهد.
- کیس سوم سازندگان بازی (credits) را نمایش می دهد و با استفاده از تابع sleep_sec() که در عکس های بعدی کد آن آورده شده است به مدت پنج ثانیه باقی می ماند و بعد از آن منو مجددا نمایش داده می شود.
- کیس چهارم مربوط به منوی راهنما(help) بازی است. این منو هم به مدت ۵ ثانیه نمایان است و بعد از آن منوی اصلی مجددا نمایان می شود.
- کیس پنجم مربوط به خروج از بازی(exit) است که اگر کاربر y را انتخاب کند بازی بسته می شود و اگر n را انتخاب کند مجددا منوی اصلی نمایش داده می شود.(در حین بازی هم با زدن دکمه e بازی تمام می شود.)
- اگر کاربر گزینه غیر مجاز انتخاب کند, ابتدا از او درخواست می شود که ورودی مجاز وارد کند و سپس منوی بازی مجددا نمایش داده می شود.

- همچنین در حین بازی یک منوی مجزا در اختیار بازیکن است که با زدن دکمه m قابل دسترس است که شامل موارد زیر است:
- ۱- بازگشت به بازی: در این حالت منو بسته شده و صفحه بازی دوباره نمایان می شود.
- ۲-شروع بازی جدید: در این حالت تمامی متغیر ها به حالت اولیه بر می گردند و صحه بازی جدید نمایش داده می شود.
- ۳- تنظیمات: مشابه تنظیمات منوی اصلی مرحله کنونی و تعداد کل مرحله ها را نشان می دهد.
- ۴- خروج: در این حالت از بازی خارج شده تمامی متغیر ها پیش فرض می شوند و منوی اصلی نمایش داده می شود.
- همچنین اگر ورودی غیر استاندارد دریافت شود تا زمانی که بازیکن مقدار درست وارد کند از او درخواست ورودی دارد.

C:\Users\Salam\Downloads\project final.cpp - Embarcadero Dev-C++ 6.3

File Edit Search View Project Execute Tools AStyle Window Help

TDM-GCC 9.2.0 64-bit Release

(globals)

Project Clas

project final.cpp

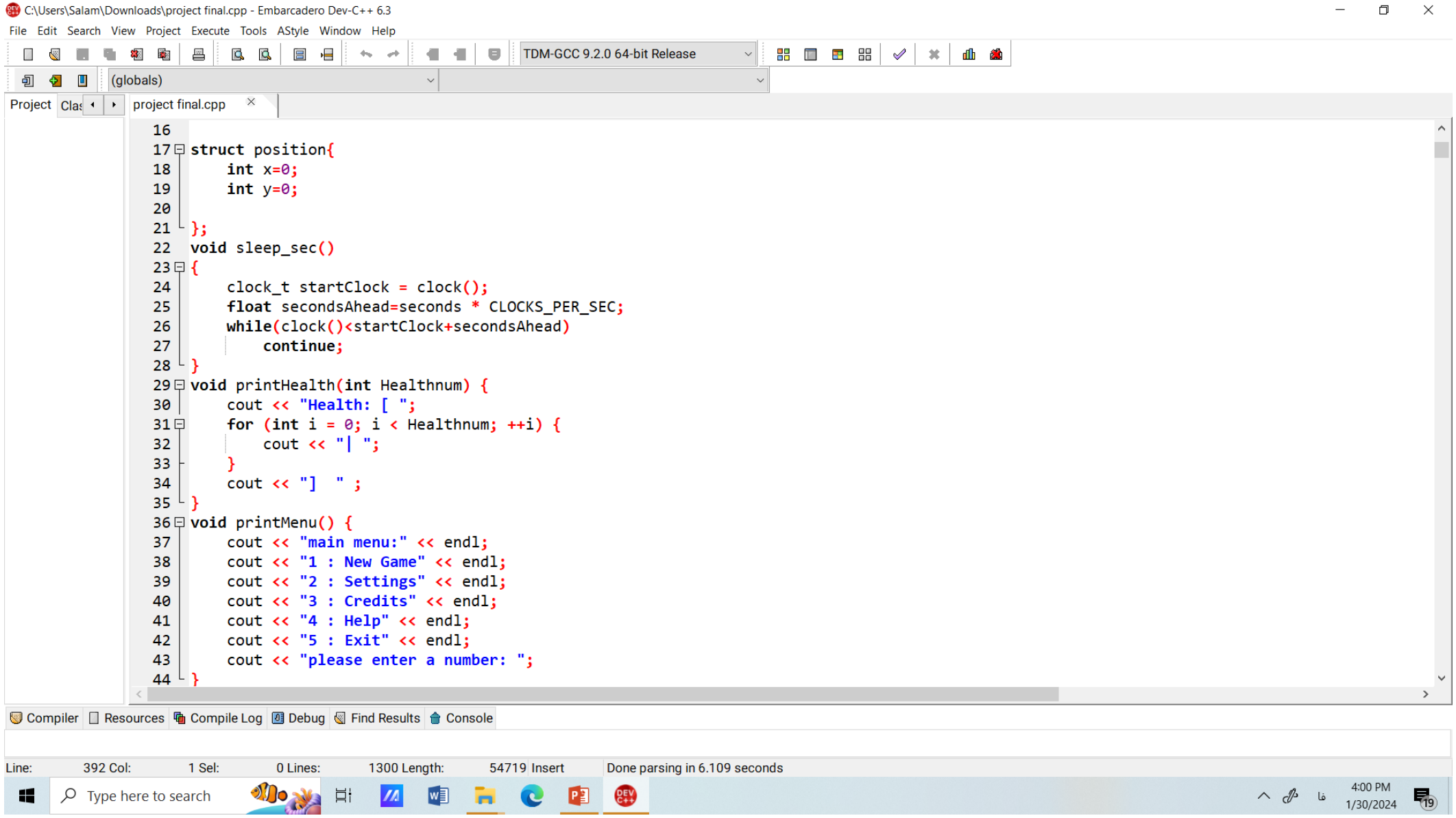
```
1243    }
1244
1245                case '2':
1246                    cout << "Current level: " << level << endl;
1247                    cout<<"Final level is : 20"<<endl;
1248                    break;
1249
1250                case '3':
1251                    cout << "creators: EMAD MEMAR & MANI MASTEALI" << endl;
1252                    sleep_sec();
1253                    break;
1254                case '4':
1255                    cout << "This is a survival game. You should try to keep away from zombies and collect ammo and vaccines." << endl;
1256                    cout << "Player's location is shown by character (P), zombies(Z), vaccines(V), and ammos(A)" << endl;
1257                    cout << "You gain credit by killing zombies, collecting ammos, and vaccines." << endl;
1258                    cout<<endl;
1259                    cout <<"The player moves with W(up), S(down), D(right), A(left)"<<endl;
1260                    cout<<endl;
1261                    cout <<"If you want to shoot the zombies use this instruction: "<<endl;
1262                    cout<< " T(shooting up), G(shooting down), H(shooting left), F(shooting right)" <<endl;
1263                    cout<<"Tip: you can enter either upper case letters or lowercase ones for shooting and movement "<<endl;
1264                    cout<<endl;
1265                    cout << "You lose if your health reaches 0." << endl;
1266                    sleep_sec();
1267                    break;
1268                case '5':
1269                    cout << "Are you sure you want to exit the game?" << endl;
1270                    cout << " n " << endl;
1271                    cout << " y " << endl;
1272
1273                    cin >> exitChoice;
1274
1275                    if (exitChoice == 'n' || exitChoice == 'N') {
1276                        displayMenu = true;
1277                    } else if (exitChoice == 'y' || exitChoice == 'Y') {
1278                        cout << "Good Bye!" << endl;
1279                        return 0;
1280                    } else {
1281                        cout << "Please enter valid letters: y or n." << endl;
1282                        validInput = false;
1283                    }
1284                    break;
1285                default:
1286                    cout << "Enter appropriate number." << endl;
1287                    validInput = false;
1288                    Round++;
1289                    break;
1290            if (validInput) {
1291                displayMenu = true;
1292            }
1293        }
```

Compiler Resources Compile Log Debug Find Results Console

Line: 698 Col: 33 Sel: 0 Lines: 1300 Length: 54719 Insert Done parsing in 6.109 seconds

Type here to search

5:35 PM
1/30/2024


C:\Users\Salam\Downloads\project final.cpp - Embarcadero Dev-C++ 6.3
File Edit Search View Project Execute Tools AStyle Window Help
TDM-GCC 9.2.0 64-bit Release
(globals)
Project Clas
project final.cpp
16
17 struct position{
18 int x=0;
19 int y=0;
20
21 };
22 void sleep_sec()
23 {
24 clock_t startClock = clock();
25 float secondsAhead=seconds * CLOCKS_PER_SEC;
26 while(clock()<startClock+secondsAhead)
27 continue;
28 }
29 void printHealth(int Healthnum) {
30 cout << "Health: [";
31 for (int i = 0; i < Healthnum; ++i) {
32 cout << "| ";
33 }
34 cout << "] " ;
35 }
36 void printMenu() {
37 cout << "main menu:" << endl;
38 cout << "1 : New Game" << endl;
39 cout << "2 : Settings" << endl;
40 cout << "3 : Credits" << endl;
41 cout << "4 : Help" << endl;
42 cout << "5 : Exit" << endl;
43 cout << "please enter a number: ";
44 }
Compiler Resources Compile Log Debug Find Results Console
Line: 392 Col: 1 Sel: 0 Lines: 1300 Length: 54719 Insert Done parsing in 6.109 seconds
Type here to search
4:00 PM
1/30/2024

- در اسلاید قبلی,یک استراکت پوزیشن طراحی شده(پیش بینی شد که در مراحل جلوتر بازی برای قسمت های مختلف مثل برخورد زامبی و پلیر,رسیدن پلیر به نقطه D و جمع کردن واکسن و تیر,کار با مختصات ساده تر است.)
- پس از آن تابع sleep sec نوشته شده که در پی دی اف راهنمای پروژه هم قرار گرفته بود و برای انجام عملیات های مشخص با تاخیر زمانی کوچک(در این پروژه ۵ ثانیه) انجام میشود.
- تابع printhealth هم مشخصا برای نشان دادن صفحه بازی طراحی شده و به تعداد جان های باقی مانده هر شخص, کاراکتر (|) پرینت میشود. علت استفاده از این تابع چالش استفاده از استرینگ بود چرا که نمی شد استرینگ را به راحتی تغییر داد. بنابراین تعداد جان با یک متغیر تعریف شد که تعداد کاراکتر ها وابسته به آن باشند.

C:\Users\Salam\Downloads\project final.cpp - Embarcadero Dev-C++ 6.3

File Edit Search View Project Execute Tools AStyle Window Help

TDM-GCC 9.2.0 64-bit Release

(globals)

Project Clas

project final.cpp

```
37      cout << "main menu:" << endl;
38      cout << "1 : New Game" << endl;
39      cout << "2 : Settings" << endl;
40      cout << "3 : Credits" << endl;
41      cout << "4 : Help" << endl;
42      cout << "5 : Exit" << endl;
43      cout << "please enter a number: ";
44  }
45
46  void killer_bar(int Round,int Kill){
47      int condition=Kill%14;
48      switch(condition)
49     { case 1: cout<<"First Blood"<<endl; break;
50      case 2: cout<<"Double Kill"<<endl; break;
51      case 3: if(Round%2==0) cout<<"Triple Kill"<<endl;   else cout<<"Hatrick"<<endl;  break;
52      case 4: cout<<"Team Killer"<<endl; break;
53      case 5: cout<<"Headshot"<<endl; break;
54      case 6: cout<<"Rampage"<<endl; break;
55      case 7: cout<<"Killing Spree"<<endl; break;
56      case 8: cout<<"Unstopabble"<<endl; break;
57      case 9: cout<<"Monster Kill"<<endl; break;
58      case 10: cout<<"Multi Kill"<<endl; break;
59      case 11: cout<<"Ludicrous Kill"<<endl; break;
60      case 12: cout<<"Ultra Kill"<<endl; break;
61      case 13 : cout<<"Dominating"<<endl; break;
62      case 0: cout<<"Godlike"<<endl; break;
63      }
64  }
65  void show_game_screen(int level, int Vaccine, int Credit, int Round, int HealthNum, int Gunammo,int Magammo, int Kill,position &player,po:
```

Compiler Resources Compile Log Debug Find Results Console

Line: 392 Col: 1 Sel: 0 Lines: 1300 Length: 54719 Insert Done parsing in 6.109 seconds

Type here to search

4:09 PM
1/30/2024

- تابع killer bar به این منظور به کار می رود که اگر بازیکن موفق بشود یک زامبی را بکشد با توجه به تعداد kill هایی که به دست آورد بعد از هر kill یک عبارت خاص چاپ خواهد شد.

Show game screen

- در این تابع که قبل تر معرفی شد, راه اندازی اولیه صفحه بازی انجام می گیرد, که بعد از شروع اولیه بازی, و بعد از راه اندازی دوباره بازی از طریق منوی داخل بازی یا بعد از شروع مجدد زمانی که بازیکن جان هایش تمام می شود, فراخوانی می شود.

```
void show_game_screen(int level, int Vaccine, int Credit, int Round, int HealthNum, int Gunammo,int Magammo, int Kill,
                      position &player,position zombies[],position ammos[],position vaccines[])
{

   cout<<"Level: "<<level<<"  Vaccine: "<<Vaccine<<"  Credit: "<<Credit<<"  Round: "<<Round<<endl;
   printHealth(HealthNum);
   cout<<"Ammo: "<<Gunammo<<"/"<<Magammo<<"  Kill: "<<Kill<<endl;
 //  declining the game screen elements
   srand(time(NULL));
   int x,y;
   for(int i=0;i<17;i++)
   {
      for(int j=0;j<16;j++)
      {
          if(i==0 || i==16)
         { game_screen[i][j]='-';
           game_screen[i][j]='-';
          }
          else if(i==1 && j==1)
          {
              player.x=i;
              player.y=j;
          }
          else
          {
              if(j==0 || j==15)
                  game_screen[i][j]='|';
              else
                  game_screen[i][j]=' ';
          }
      }
   }
   game_screen[1][1]='P';
   game_screen[15][14]='D';
   // randomize
   int zombie_count=0;
   while(zombie_count<level)
   {
       x=rand()%17;
```

```
        x=rand()%17;
        y=rand()%16;
        if(game_screen[x][y]==' ')
        {

                game_screen[x][y]='Z';
                zombies[zombie_count].x=x;
                zombies[zombie_count].y=y;
            zombie_count++;
        }

    }
    int ammo_count=0;
    while(ammo_count<(level/2))
    {
        x=rand()%17;
        y=rand()%16;
        if(game_screen[x][y]==' ')
        {

                game_screen[x][y]='A';
                ammos[ammo_count].x=x;
                ammos[ammo_count].y=y;
            ammo_count++;
        }

    }
    int vaccine_count=0;
    while(vaccine_count<level)
    {
        x=rand()%17;
        y=rand()%16;
        if(game_screen[x][y]==' ')
        {

                game_screen[x][y]='V';
                vaccines[vaccine_count].x=x;
                vaccines[vaccine_count].y=y;
            vaccine_count++;
        }
```

```
//show
for(int i=0;i<17;i++)
{    for(int j=0;j<16;j++)
    {
        if(game_screen[i][j]=='P')
        {
            string RESET="\033[0m";
            string GREEN = "\033[32m";
            cout<<GREEN<<game_screen[i][j]<<RESET;
        }
        else if(game_screen[i][j]=='D')
        {
            string RESET="\033[0m";
            string MAGNETA="\u001b[35m";
            cout<<MAGNETA<<game_screen[i][j]<<RESET;
        }
        else if(game_screen[i][j]=='Z')
        {
            string RESET="\033[0m";
            string RED="\u001b[31m";
            cout<<RED<<game_screen[i][j]<<RESET;
        }
        else if(game_screen[i][j]=='A')
        {
            string RESET="\033[0m";
            string BLUE="\u001b[34m";
            cout<<BLUE<<game_screen[i][j]<<RESET;
        }
        else if(game_screen[i][j]=='V')
        {
            string RESET="\033[0m";
            string YELLOW="\u001b[33m";
            cout<<YELLOW<<game_screen[i][j]<<RESET;
         }
            else
        cout<<game_screen[i][j];

    }
cout<<endl;
}
```

- همانطور که دیده می شود در ابتدا شماره مرحله, تعداد واکسن ها , امتیاز, تعداد جان, تعداد تیر در خشاب, تعداد تیر ذخیره و تعداد کشته ها در بالا نمایش داده می شود. در تابع show_updated_game_screen() که بعد از این توضیحات می آید نیز همین منطق پیاده سازی شده شده است.
- سپس برای اینکه مکان زامبی ها, واکسن ها و تیر های اضافه به طور تصادفی در صفحه مشخص شوند از تابع srand استفاده شده.
- سپس در ادامه صفحه بازی ساخته می شود. ابتدا گوشه های صفحه و سپس محل قرارگیری بازیکن, نقطه مقصد(D) و در انتها مکان های زامبی ها تعریف می شود. چالش این بخش این بود که ما به مکان زامبی ها, واکسن ها و تیر ها در ادامه نیاز داشتیم بنابراین در تابع main آرایه هایی از مکان های آنها تعریف کرده و در این تابع مقدار دهی کردیم.
- در انتها هم جدول با توجه به رنگ های عناصر مختلف نمایش داده می شوند.

Show updated game screen

- پویایی برنامه ایجاب می کرد که ما از یک تابع برای نمایش صفحه به صورت پویا استفاده کنیم. در کد این برنامه ابتدا نوار وضعیت جان ها و غیره نمایش داده می شود و سپس هر آیتم متناسب با رنگ و موقعیت خود نمایش داده می شود.

```
void show_updated_game_screen(int level, int Vaccine, int Credit, int Round, int Healthnum, int Gunammo,int pocketammo, int Kill)
{
     cout << "Level: " << level << "  Vaccine: " << Vaccine << "  Credit: " << Credit << "  Round: " << Round << endl;
    printHealth(Healthnum);
    cout << "Ammo: " << Gunammo << "/" << pocketammo << "  Kill: " << Kill << endl;
    for(int i=0;i<17;i++)
    {    for(int j=0;j<16;j++)
        {
            if(game_screen[i][j]=='P')
            {
                string RESET="\033[0m";
                string GREEN = "\033[32m";
                cout<<GREEN<<game_screen[i][j]<<RESET;
            }
            else if(game_screen[i][j]=='D')
            {
                string RESET="\033[0m";
                string MAGNETA="\u001b[35m";
                cout<<MAGNETA<<game_screen[i][j]<<RESET;
            }
            else if(game_screen[i][j]=='Z')
            {
                string RESET="\033[0m";
                string RED="\u001b[31m";
                cout<<RED<<game_screen[i][j]<<RESET;
            }
            else if(game_screen[i][j]=='A')
            {
                string RESET="\033[0m";
                string BLUE="\u001b[34m";
                cout<<BLUE<<game_screen[i][j]<<RESET;
            }
            else if(game_screen[i][j]=='V')
            {
                string RESET="\033[0m";
                string YELLOW="\u001b[33m";
                cout<<YELLOW<<game_screen[i][j]<<RESET;
             }
                else
            cout<<game_screen[i][j];

        }
    cout<<endl;
    }
}
```

C:\Users\Salam\Downloads\project final.cpp - Embarcadero Dev-C++ 6.3

File Edit Search View Project Execute Tools AStyle Window Help

TDM-GCC 9.2.0 64-bit Release

(globals)

Project Clas project final.cpp

```
232
233 void clear_screen(){
234 #if defined _WIN32
235 system("cls");
236 #elif defined (__LINUX__) || defined(__gnu_linux__) || defined(__linux__)
237 system("clear");
238 #elif defined (__APPLE__)
239 system("clear");
240 #endif
241 }
242
243 void movePlayer( position &player, char move) {
244     int currentX = player.x;
245     int currentY = player.y;
246
247     if ( (move == 'W' || move=='w') && player.x>1) {
248         player.x--;
249     } else if ( (move == 'A' || move=='a')  && player.y>1) {
250         player.y--;
251     } else if ( (move == 'S' || move=='s') && player.x<(ROWS-2)) {
252         player.x++;
253     } else if ( (move == 'D' || move=='d') && player.y<(COLS-2)) {
254         player.y++;
255     }
256     game_screen[currentX][currentY] = ' ';
257
258     game_screen[player.x][player.y] = 'P';
259     if(game_screen[currentX][currentY]==' ' && currentX==15 && currentY==14)
260     {
261         game_screen[currentX][currentY]='D';
262     }
263 }
264 void moveZombies( bool zombies_shot[] position zombies[] int level position &player int round position ammos[] position vaccines[]) {
```

Compiler Resources Compile Log Debug Find Results Console

Line: 392 Col: 1 Sel: 0 Lines: 1300 Length: 54719 Insert Done parsing in 6.109 seconds

Type here to search

ENG 4:12 PM 1/30/2024

- تابع کلیر اسکرین را از کاتالوگ مرجع پروژه با اندکی تغییراستفاده کردیم.
- تابع Move Player برای حرکت بازیکن با کلید های w a s d نوشته شده بود.
- Wحرکت رو به بالا
- A حرکت رو به چپ
- S حرکت رو به پایین
- D حرکت رو به راست
- سپس مختصات قبلی پلیر خالی شده و مختصات جدید ان با حرف P نمایش داده میشود.
- دو چالش در این بخش وجود داشت. اول اینکه اگر بازیکن به نقطه D می رسید اولویت نمایش به D داده می شد و ثانیا بعد از اینکه اولویت بندی تنظیم شد, بعد از اینکه بازیکن نقطه D را ترک می کرد آن نقطه محو می شد و عملا پایان مرحله غیر ممکن می شد پس در انتها یک شرط اضافه کردیم که نقطه D ثابت باقی بماند.

```
void moveZombies(  bool zombies_shot[],position zombies[], int level, position &player, int round,position ammos[],position vaccines[]) {
    for (int i = 0; i < level; i++) {
        if(zombies_shot[i]==false)
      {  int currentX = zombies[i].x;
        int currentY = zombies[i].y;

        if (player.y < zombies[i].y && round%2==0) {
        //move left
            zombies[i].y--;
        } else if (player.y > zombies[i].y && round%2==0 ) {
            // move right
            zombies[i].y++;
        } else {
            //vertical movement
            if (player.x < zombies[i].x && round%2==0 ) {
                // move up
                zombies[i].x--;
            } else if (player.x > zombies[i].x && round%2==0) {
                // move down
                zombies[i].x++;
            }
        }

        // Update game screen with changes
        game_screen[currentX][currentY] = ' ';
        game_screen[zombies[i].x][zombies[i].y] = 'Z';
        if(game_screen[currentX][currentY]==' ' && currentX==15 && currentY==14)
        {
            game_screen[currentX][currentY]='D';
        }
        else{
            for(int i=0;i<level;i++)
             {
                if(game_screen[currentX][currentY]==' ' && currentX==vaccines[i].x && currentY==vaccines[i].y)
                {
                    game_screen[currentX][currentY]='V';
                }
                else if(game_screen[currentX][currentY]==' ' && currentX==zombies[i].x && currentY==zombies[i].y)
                {
                    game_screen[currentX][currentY]='Z';
                }

            }
             for(int i=0;i<(level/2);i++)
             {
                 if(game_screen[currentX][currentY]==' ' && currentX==ammos[i].x && currentY==ammos[i].y)
                 {
                     game_screen[currentX][currentY]='A';
                 }
             }
        }

        }
        else if(zombies_shot[i]==true)
        {
            game_screen[zombies[i].x][zombies[i].y]=' ';
        }
    }
}
```

MOVE ZOMBIES

- این بخش که یکی از پر چالش ترین بخش های پروژه بود که تقریبا با ورودی مجاز یا غیر مجاز اجرا می شود.
- در این تابع ابتدا الگوریتم حرکت زامبی تعریف شده که به سمت بازیکن حرکت می کند و حرکت افقی نسبت به حرکت عمودی ارجعیت دارد.
- اولین چالش این بخش رفتار زامبی ها بعد از کشته شدن بود. برای اینکه زامبی کشته شده از بازی خارج شود ابتدا یک تابع بولین به نام zombies_shot تعریف کردیم و الگوریتم حرکتی را فقط در زمانی برای یک زامبی پیاده سازی کردیم که آن زامبی زنده باشد. و درصورتی که کشته شده باشد جای آن خالی گردد که دیگر بازیکن وقتی به مکان قبلی زامبی می رود اولا دوباره از زامبی ضربه نخورد و ثانیا وقتی که بازیکن به همان مکان برود محو نشود.
- دومین چالش این بود که زامبی ها در صورتی که با یک آیتم دیگر در یک مکان قرار می گرفتند آن آیتم بعد از رفتن زامبی از آن مکان کلا ناپدید می شد. با توجه به ثابت بودن آیتم های دیگر از آرایه مکان های آنها استفاده کردیم و این شرط را تعریف کردیم که بعد از رفتن زامبی آن مکان دوباره با ماهیت قبلی خود پر شود.

C:\Users\Salam\Downloads\project final.cpp - Embarcadero Dev-C++ 6.3

File Edit Search View Project Execute Tools AStyle Window Help

TDM-GCC 9.2.0 64-bit Release

(globals)

Project Clas

project final.cpp

```
325
326 void collectItem(position player, position vaccines[], position ammos[], int &Vaccine, int &Credit, int &Healthnum, int &pocketammo,int level) {
327     // Check if the player is on the same position as any vaccine
328     for (int i = 0; i < 20; i++) {
329         if (player.x == vaccines[i].x && player.y == vaccines[i].y) {
330             // Player collected a vaccine
331             Vaccine++;
332             Credit += level + 1;
333             cout << "Vaccine collected!" << endl;
334             cout << level + 1 << " credit gained!" << endl;
335             // Remove the collected vaccine from the map
336             vaccines[i].x = 18;
337             vaccines[i].y = 18;
338             // Update Healthnum
339             Healthnum++; // Example update, adjust as needed
340             cout << "Health updated!" << endl;
341 
342             return; // Exit the function since the item is collected
343         }
344     }
345 
346     // Check if the player is on the same position as any ammo
347     for (int i = 0; i < 10; i++) {
348         if (player.x == ammos[i].x && player.y == ammos[i].y) {
349             // Player collected ammo
350             pocketammo++;
351             cout << "Ammo collected!" << endl;
352             // Remove the collected ammo from the map
353             ammos[i].x = 18;
354             ammos[i].y = 18;
355             return; // Exit the function since the item is collected
356         }
357     }
358 }
359 void zombie_attacks(position player, position zombies[], bool zombies_shot[], int &Healthnum, int level) {
360     for(int i = 0; i < level; i++) {
361         if((player.x == zombies[i].x && player.y == zombies[i].y && !zombies_shot[i] ) ||
```

Compiler Resources Compile Log Debug Find Results Console

Line: 392 Col: 1 Sel: 0 Lines: 1300 Length: 54719 Insert Done parsing in 6.109 seconds

Type here to search

ENG 4:24 PM 1/30/2024

- این تابع برای جمع کردن واکسن و تیر طراحی شده و همانطور که در ابتدای پی دی اشاره شد, از استراکت ایکس و وای برای طراحی شرط برخورد استفاده کردیم.
- چالشی که در اینجا وجود داشت این بود که بعد از جمع آوری واکسن و تیر, به چه شکلی آنها را از صفحه بازی پاک کنیم.راهی که با همفکری به آن رسیدیم این بود که پس از جمع اوری انها توسط بازیکن.مختصات آنرا به ۱۸ و ۱۸ منتقل کنیم(چون صفحه بازی در واقع ۱۶ در ۱۷ است و این کار یعنی آیکون تیر یا زامبی به بیرون از صفحه بازی انتقال داده شود.) و با توجه به اینکه خارج از صفحه بازی نمایش داده نمی شود در واقع آن آیتم حذف شده است.

C:\Users\Salam\Downloads\project final.cpp - Embarcadero Dev-C++ 6.3

File Edit Search View Project Execute Tools AStyle Window Help

TDM-GCC 9.2.0 64-bit Release

(globals)

Project Clas

project final.cpp

```
358 }
359 void zombie_attacks(position player, position zombies[], bool zombies_shot[], int &Healthnum, int level) {
360     for(int i = 0; i < level; i++) {
361         if((player.x == zombies[i].x && player.y == zombies[i].y && !zombies_shot[i] ) ||
362            (player.x == zombies[i].x && player.y == zombies[i].y + 1 && !zombies_shot[i]) ||
363            (player.x == zombies[i].x && player.y == zombies[i].y - 1 && !zombies_shot[i]) ||
364            (player.x == zombies[i].x + 1 && player.y == zombies[i].y && !zombies_shot[i]) ||
365            (player.x == zombies[i].x - 1 && player.y == zombies[i].y && !zombies_shot[i]) ||
366            (player.x == zombies[i].x + 1 && player.y == zombies[i].y + 1 && !zombies_shot[i]) ||
367            (player.x == zombies[i].x + 1 && player.y == zombies[i].y - 1 && !zombies_shot[i]) ||
368            (player.x == zombies[i].x - 1 && player.y == zombies[i].y - 1 && !zombies_shot[i]) ||
369            (player.x == zombies[i].x - 1 && player.y == zombies[i].y + 1 && !zombies_shot[i]) ) {
370             Healthnum--;
371             cout << "The Zombie is eating you! You lost one of your healths!" << endl;
372 
373         }
374         else if(player.x == zombies[i].x && player.y == zombies[i].y && zombies_shot[i])
375         {
376             game_screen[player.x][player.y]='P';
377         }
378     }
379 }
380 
381 int main()
382 {
383     //enable ansi codes on windows consoles
384     HANDLE hConsole = GetStdHandle(STD_OUTPUT_HANDLE);
385 
386     DWORD mode;
387     GetConsoleMode(hConsole, &mode);
388 
389     mode |= ENABLE_VIRTUAL_TERMINAL_PROCESSING;
390 
391     SetConsoleMode(hConsole, mode);
392 
393     string RESET="\033[0m";
394     string RED="\u001b[31m";
```

Compiler Resources Compile Log Debug Find Results Console

Line: 392 Col: 1 Sel: 0 Lines: 1300 Length: 54719 Insert Done parsing in 6.109 seconds

Type here to search

4:35 PM
1/30/2024

- در این تابع حالت های برخورد یا هم جواری زامبی ها داخل شرط نوشته شده و اگر یکی از انها رخ دهد, از جان شخص کم میشود. این حالت ها شامل ۱۰ حالت هستند: مکان زامبی و بازیکن یکی شود که دو حالت دارد: اگر زامبی مرده باشد با توجه به اینکه در قبل گفتیم بازیکن در حالت محو می شود آن مکان برابر P می شود.در غیر این صورت برخورد رخ داده و از جان بازیکن کم می شود. در ۹ حالت دیگر, مکان های هستند که زامبی و بازیکن یک واحد در جهت افقی, عمودی و مایل با هم فاصله دارند.

- در ادامه در تابع main منوی اصلی بازی چاپ شده و از کاربر ورودی انتخابش را میگیریم.برای این قسمت صلاح دیده شد که حالت های مختلف داخل سوییچ کیس نوشته شود و منطق بازی هم داخل کیس ۱ (start new game) طراحی شود.کلیت بازی شامل دستور های کلیدی شلیک,حرکت بازیکن,حرکت زامبی ,جمع اوری واکسن و تیر, ارتقای تجهیزات,ریلود و منطق برد و باخت میباشد.

- چالشی که در بحث تیر اندازی داشتیم این بود که اگر چند زامبی در یک ردیف بودند و تیر میزدیم,به جای یک تیر,چند تیر کم میکرد. در واقع با توجه به اینکه زامبی ها به طور تصادفی قرار گرفته اند اگر ما از هر نوع حلقه ای استفاده کنیم ممکن است ابتدا زامبی را هدف قرار دهد که دورتر است. در قسمت منطق تیر اندازی در این باره توضیح داده خواهد شد.
- همچنین چالش دیگری که وجود داشت بحث شروع مجدد بازی بعد از مرگ بازیکن یا با استفاده از منوی وسط بازی بود که برای این کار از مفهوم start و goto استفاده کردیم. در ادامه خلاصه ای از عملکرد عملگر های حین بازی آورده شده است.

Game Logics

- در حرکت بازیکن, توابعی که مربوط به حرکت ها بودند(move player,move zombies,zombies attack) فراخوانی می شوند. همچنین تابع جمع آوری آیتم ها هم مورد بررسی قرار می گیرد و صفحه بازی با تابع show_updated_game_screen به روز رسانی می شود.
- در ادامه بخشی از کد را بررسی می کنیم که مربوط به اتفاقاتی است که بعد از game over شدن اتفاق می افتد.

```
if(Healthnum<=0)
{
cout<< "You Died!"<<endl;
cout<<"You Lose! Would you like to try again?(y/n)"<<endl;
char pchoice;
cin>>pchoice;
if(pchoice=='y')
{
   cout<<"The game will restart in 5 seconds"<<endl;
   level=1;
   Vaccine=0;
   Credit=0;
   Round=0;
   Kill=0;
   Gunammo=3;
   pocketammo=1;
   maxgunammo=3;
   gunRange=5;
   Healthnum=3;
   for(int i=0;i<20;i++)
   zombies_shot[i]=false;
  show_game_screen(level, Vaccine, Credit, Round, Healthnum, Gunammo, pocketammo, Kill, player,zombies,ammos,vaccines);
  game_goes_on=true;
  goto start2;
}
else if(pchoice=='n')
    game_goes_on = false;
    level=1;
    Vaccine=0;
    Credit=0;
    Round=0;
    Kill=0;
    Gunammo=3;
    pocketammo=1;
    maxgunammo=3;
    gunRange=5;
    Healthnum=3;
    for(int i=0;i<20;i++)
    zombies_shot[i]={false};
    displayMenu=true;
     goto start1;
}
```

- این بخش از کد در زمان حرکت بازیکن, هنگام تیر اندازی و هنگام ریلود بررسی می شود و بیان می کند که بازی به اتمام رسیده است و به بازیکن حق انتخاب می دهد. اگر بازیکن بخواهد دوباره از اول شروع کند, تمامی متغیر ها به مقدار پیش فرض بر می گردند و همچنین کد مجدد از زمانی اجرا می شود که در ابتدا بازی جدید شروع می شد که با استفاده از goto اجرا می شود. همچنین اگر بازیکن نخواهد مجدد شروع کند منوی اصلی مجددا با دستور goto نمایش داده می شود.

- با توجه به اینکه حلقه های متعددی در فرایند اجرای کد اجرا می شوند, لذا بازگشت به عقب با دستور goto در هنگام restart بهترین راه حل ممکن بود. همانطور که قبل تر اشاره شد, یکی از چالش های بازی restart کردن بود که با این روش برطرف شد.
- همچنین برای بخش حرکت زامبی چالشی که داشتیم و قبل تر اشاره نشد این بود که قبل از اینکه ان را به صورت تابع بنویسیم,منطق حرکت زامبی را فقط برای حرکت هایwasd تعریف کرده بودیم ولی بعد تر متوجه شدیم که با تیر زدن,ریلود کردن و حتی ورودی اشتباه دادن هم تعداد راند افزایش پیدا میکند. پس ترجیح دادیم حرکت زامبی را تابع کنیم و در حالت های ذکر شده ان را call کنیم تا کد بهینه تری داشته باشیم.

Game level up

- در انتهای کد های مربوط به حرکت بازیکن با کلید های w a s d, این شرط بررسی شد که آیا بازیکن تمامی واکسن ها را جمع آوری کرده یا خیر. اگر بله که وارد مرحله بعدی می شود. در غیر این صورت بازی از بازیکن درخواست می کند که تمامی واکسن ها را جمع آوری کند. همچنین اگر بازیکن به مرحله آخر رسید بازی را تمام کند و به او تبریک بگوید.

```
if(player.x == 15 && player.y == 14) {
    bool allvaccinescollected = true;
    for(int i = 0; i < level; i++) {
        if( vaccines[i].x != 18 ||  vaccines[i].y != 18) {
            allvaccinescollected = false;
            break;
        }
    }

    if(allvaccinescollected && level < 20) {
        level++;
        Round=0;
        for(int i=0;i<20;i++)
        zombies_shot[i]={false};
        cout << "You Won! You have reached level " << level << endl;
                show_game_screen(level, Vaccine, Credit, Round, Healthnum, Gunammo, pocketammo, Kill, player,zombies,ammos,vaccines);
    } else if(allvaccinescollected && level >= 20) {
        cout << "Congratulations! You have successfully collected all the vaccines! Now it is time to return to the Earth. Goodbye!" << endl;
        level=1;
        Vaccine=0;
        Credit=0;
        Round=1;
         Kill=0;
        Gunammo=3;
        pocketammo=1;
        maxgunammo=3;
        gunRange=5;
        Health="[ | | | ]";
        Healthnum=3;
        for(int i=0;i<20;i++)
       zombies_shot[i]=false;
        printMenu();
    } else {
        cout<<"First get all the vaccines."<<endl;
        show_updated_game_screen(level, Vaccine, Credit, Round, Healthnum, Gunammo, pocketammo, Kill);

    }
}
```

Shooting logic

- در این بخش که برای هر چهار حالت شلیک به بالا پایین چپ یا راست تقریبا مشابه است, روند کار به این صورت است که ابتدا بررسی شود که آیا تعداد تیر ها کافی هستند یا خیر. اگه تعداد تیر های درون خشاب صفر باشد بازی به بازیکن می گوید که تیر ندارد. همچنین زامبی ها هم در این حالت حرکت خواهد کرد. اگر تیر موجود بود, با متغیر های تعریف شده نزدیک ترین زامبی که در محدوده برد تفنگ است مورد هدف قرار می گیرد و همچنین همان طور که قبلا گفته شد در هر حالت منطق حرکت زامبی ها و ریستارت کردن بازی تعریف شده است.

```
    case 'T':
    case 't':
    {
        //shooting up
        if (Gunammo == 0) {
            cout << "No charged ammo!" << endl;
            Round++;
            show_updated_game_screen(level, Vaccine, Credit, Round, Healthnum, Gunammo, pocketammo, Kill);
            zombie_attacks(player, zombies, zombies_shot, Healthnum, level);
        }
        else {
            bool shot = false;
            int closest_zombie_index = -1;
            int closest_zombie_distance = gunRange + 1;
            bool outrange = false;
            for (int i = 0; i < level; i++) {
                if (zombies[i].y == player.y && zombies[i].x < player.x && zombies_shot[i] == false) {
                    int distance_to_zombie = player.x - zombies[i].x;
                    if (distance_to_zombie <= gunRange && distance_to_zombie < closest_zombie_distance) {
                        closest_zombie_index = i;
                        closest_zombie_distance = distance_to_zombie;
                    }
                    else if (distance_to_zombie > gunRange) {
                        Gunammo--;
                        cout << "Zombie is not in range! Come closer." << endl;
                        show_updated_game_screen(level, Vaccine, Credit, Round, Healthnum, Gunammo, pocketammo, Kill);
                        outrange = true;
                        break;
                    }

                }
            if (closest_zombie_index != -1) {
                Round++;
                zombies_shot[closest_zombie_index] = true;
                Kill++;
                Credit = Credit + level + 1;
                cout << level + 1 << " credit gained!" << endl;
                shot = true;
            }
            if (shot == true) {
                Gunammo--;
                killer_bar(Round, Kill);
                moveZombies(zombies_shot, zombies, level, player, Round, ammos, vaccines);
                show_updated_game_screen(level, Vaccine, Credit, Round, Healthnum, Gunammo, pocketammo, Kill);
            }
            else if (shot == false && outrange == false) {
                Round++;
                if (shot == false)
                    Gunammo--;
                show_updated_game_screen(level, Vaccine, Credit, Round, Healthnum, Gunammo, pocketammo, Kill);
            }
        }
```

- بزرگترین و اصلی ترین چالش این بخش این بود که کد در ابتدا اولویت بندی را به نزدیک ترین زامبی نمی داد و ممکن بود با یک بار زدن دکمه شلیک, زمانی که چند زامبی که برخی از آنها در محدوده تیر بودند و برخی دیگر خیر, چند بار به طور خودکار شلیک کند زیرا با توجه به مقادیر تصادفی مکان های زامبی ها نمی شد با یک حلقه معمولی این شرط را انجام داد.

Realoding

- ریلود کردن در این بازی از قاعده ساده ای پیروی می کند. اگر بازیکن تیر اضافه نداشته باشد به او اخطار No ammo نشان داده می شود. همچنین اگر خشاب به حداکثر ظرفیت خود رسیده باشد عملیات ریلود را انجام نمی دهد. در غیر این صورت به تعداد تیر های در دسترس تا زمانی که تیر های خشاب از حداکثر تعداد مجاز بیشتر نشود تعداد تیر ها افزایش می یابد. همچنین در هر حالتی حرکت زامبی و ریستارت بعد از مرگ برقرار است.

Updating

- در این بخش که با زدن دکمه **u** در حین بازی در دسترس است, با توجه به امتیازی که بازیکن دریافت کرده می تواند تعداد تیر های مجاز خشاب, برد اسلحه و تعداد جان خود را افزایش دهد. امتیاز لازم برای هر کدام از این موارد از این فرمول ها محاسبه می شود:
- ۱- افزایش خشاب: تعداد تیر در خشاب * شماره مرحله
- ۲- افزایش برد: برد کنونی * شماره مرحله
- ۳- افزایش جان: تعداد جان های کنونی * (شماره مرحله + ۱)

```
    case 'u':
        case 'U':
              {  int uchoice;
            do{

            int Health_needed_credit=(level+1) * Healthnum;
            int  Range_needed_credit=gunRange+level;
            int  Mag_needed_credit=Gunammo *level;
                      printupgrademenu(maxgunammo, Mag_needed_credit, gunRange, Range_needed_credit, Healthnum, Health_needed_credit);
  cin>>uchoice;
  if (uchoice==0)
  {
   show_updated_game_screen(level, Vaccine, Credit, Round, Healthnum, Gunammo, pocketammo, Kill);

  }
lse    if (uchoice==1)
  {

    if (Credit>=Mag_needed_credit &&maxgunammo!=7)
    {
    maxgunammo++;
    Credit=Credit-Mag_needed_credit;
    cout<<"Upgrade done successfully.Your magazine capacity is now: "<<maxgunammo<<endl;
    }
    else if (Credit<Mag_needed_credit &&maxgunammo!=7)
    {

    cout<<"Unfortunately, your credit is not enough to get this item. Please gain"<<Mag_needed_credit-Credit<< "more credit by playing."<<endl;
  }
    else
    {
    cout<<"The selected item is maximum."<<endl;
    }

  }
  else if (uchoice==2)
  {
    if (Credit>=Range_needed_credit && gunRange!=10)
  {
    gunRange++;
    Credit=Credit-Range_needed_credit;
    cout<<"Upgrade done successfully.Your shotgun range is now: "<<gunRange<<endl;
```

```
    else if(Credit<Range_needed_credit &&gunRange!=10){

        cout<<"Unfortunately,your credit is not enough to get this item. Please gain"<<Range_needed_credit-Credit<< "more credit by playing."<<endl;
     }
     else
     {
         cout<<"The selected item is maximum."<<endl;;

     }

     }
     else if(uchoice==3)
     {
     if (Credit>=Health_needed_credit &&Healthnum!=5)
     {

     Healthnum++;
     Credit=Credit-Health_needed_credit;
cout<<"Additional health received successfully.Your health is now "<<Healthnum<<endl;
  }
     else if(Credit<Health_needed_credit &&Healthnum!=5)
         cout<<"Unfortunately, your credit is not enough to get this item. Please gain"<<Health_needed_credit-Credit<< "more credit by playing."<<endl;
         else {
         cout<<" The selected item is maximum."<<endl;

         }

     }
     else
     cout<<"Please just enter the numbers in the menu:"<<endl;
         }
         while (uchoice != 0);
     break;
         }
         default:
          Round++;
         show_updated_game_screen(level, Vaccine, Credit, Round, Healthnum, Gunammo, pocketammo, Kill);
         moveZombies(zombies_shot,zombies,level,player,Round,ammos,vaccines);
         zombie_attacks(player,zombies,zombies_shot,Healthnum,level);
         if(Healthnum<=0)
         {
```

- برای بخش منوی اپگرید,چالشی که داشتیم این بود منو باید بعد از هر انتخاب پرینت میشد.تا جایی که پلیر عدد ۰ (return to the game) را وارد میکرد.در ابتدا سعی شد تا با حلقه while پیاده سازی شود ولی درست عمل نمی کرد. در انتها به این نتیجه رسیدیم که از مفهوم do-while که در زمان تدریس حلقه ها معرفی شد استفاده کنیم و در نهایت چالش را بر طرف کنیم.

پایان