FINAL FLIGHT

پروژه پایانی مبانی

نام و نام خانوادگی : مهرداد ورمزیار

شماره دانشجویی : 40212358045

نام استاد : حسن بشیری

زمستان 1402

فهرست

معرفی پروژه

شرح پروژه

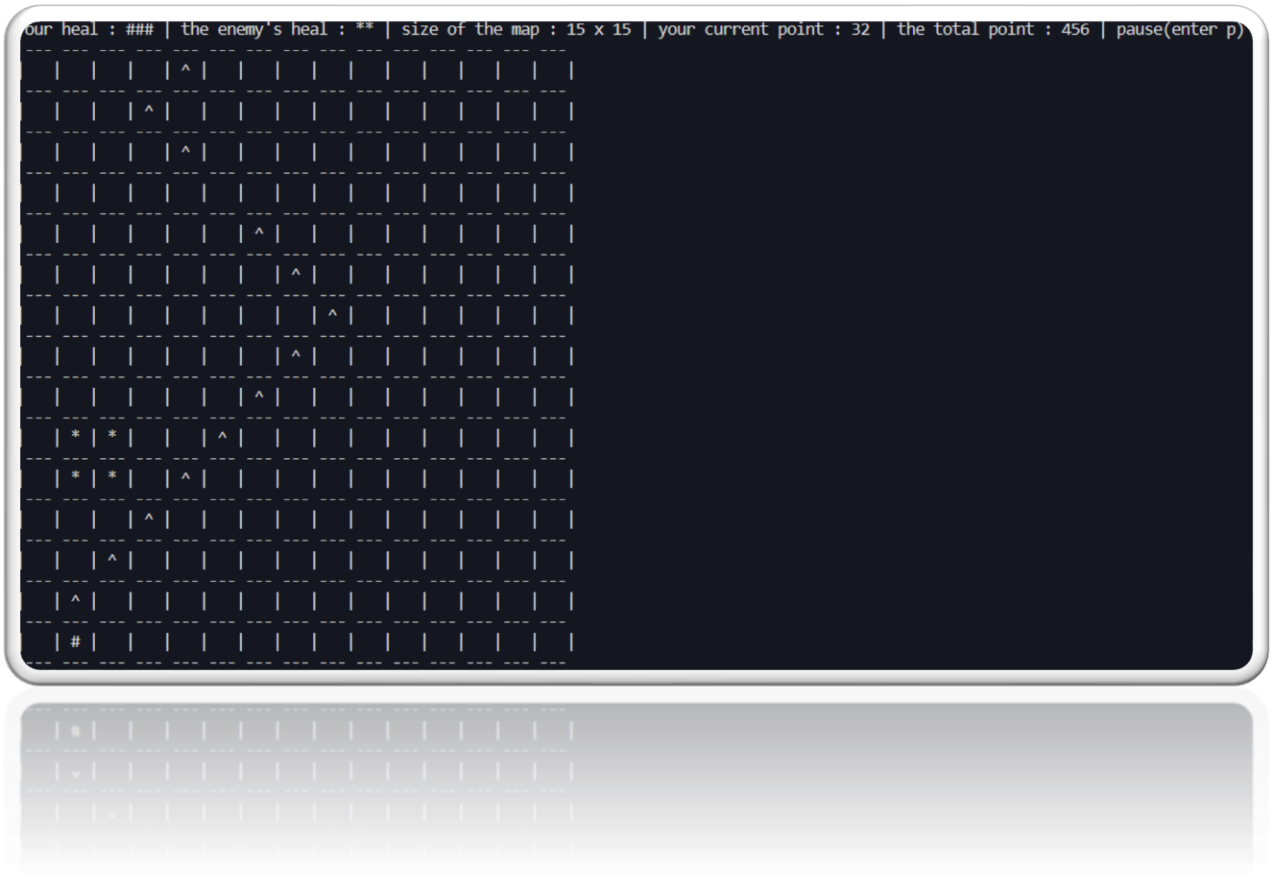
برآمد های آموزشی

مقدمه

با سلام

در این پروژه سعی بر پیاده سازی یک بازی با استایل مشابه مینی پروژه اما با یک تفاوت اساسی در گیم پلی بازی است که امتیاز اساسی این پروژه این است که یک پروژه کامل شامل منو و سیو و ... است .

نمای کلی :



معرفی پروژه

در این پروژه یک سفینه خودی داریم و یک سفینه دشمن که می تواند یکی از چهار نوع سفینه دارت ، استرایکر ، ریث یا بنشی باشد و اگر کاربر موفق به نابود کردن سفینه دشمن شود سفینه رندوم جدیدی وارد مپ می شود .

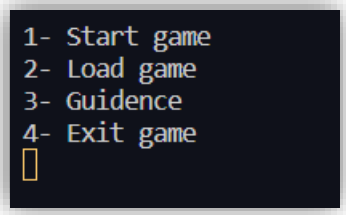
امتیاز کلی و سایز مپ از کاربر پرسیده می شود و در صورتی که کاربر به امتیاز مورد نظر برسد برنده می شود و در صورتی که جون سفینه خودی تمام شود می بازد .

بازی قابلیت سیو شدن در فایل و لود شدن از آن را دارد .

در این پروژه دو استایل تیر زدن پیاده سازی شده که یک حالت اتومات و حالت دیگر پله پله حرکت می می کند .

کاربر می تواند در صورت برد به بازی ادامه دهد یا به بازی پایان دهد و همچنین بازی یک منو کاملا تعاملی دارد .

نمایی از منو بازی :



نقاط ضعف و قوت

نقاط ضعف :

1 . ممنوعیت استفاده از متغیر گلوبال سبب پاسکاری غیر منطقی و افزایش پارامتر فانکشن ها شد .

2 . محدودیت ترمینال در ایجاد مپ های بزرگتر از سایز 40 .

3 . دشوار بودن کار با سفینه های دشمن با ابعاد بزرگ .

4 . مکانیزم پر باگ از بین بردن گلوله ها .

نقاط قوت :

1 . ذخیره بازی بر روی فایل ایده پر کاربرد است .

2 . خواندن از فایل و لود کردن گیم .

3 . گیم پلی مناسب و منطقی و پیشرفت نسبت به مینی پروژه .

4 . منو و ظاهر مناسب .

5 . در بر گرفتن کلیه مطالب تدریس شده در طول ترم .

چالش ها

پروژه کلا چالش بر انگیز بود از پیاده سازی رندوم تا سیو و لود کردن ...

برخی از چالش های مهم :

1 . استفاده از استراکت برای بخش های مختلف اعم از تیر و سفینه و مپ و .... .

2 . استفاده از فایل برای ذخیره و لود کردن .

3 . ایجاد دشمن رندوم .

4 . نابود کردن انمی با توجه به اینکه انواع متفاوت هایی دارند .

5 . ایجاد منو استارت و پاپ و آپ برای بازی .

6 . پاسکاری متغیر ها .

7 . استفاده از رفرنس ها .

نوع طراحی

اساس کلی طراحی این پروژه اساس طراحی ماژوال بود که در آن از استراکت و رفرنس و توابع استفاده شد .

در این پروژه از 27 تابع و 4 استراکت و 8 کتابخانه استفاده شده است .

نمایی از توابع :

```
oid SatrtMenue() ; //to make start menue
oid Puase(MapInfo& mapInfo , Spaceship& spaceship , Enemy& enemy , int& CurrentPoint ,std::vector<std::vector<char>>& space ,  std::vector<Bullet>& bullet) ; //to s
oid Guidence() ; //to guide user to play
oid ExitGame() ; //to exit game
apInfo MapSize() ; //to take map size and point from user
oid GenerateGame(int choice) ; //to generate game
oid RunGame(MapInfo& mapInfo , Spaceship& spaceship , Enemy& enemy , int& CurrentPoint ,std::vector<std::vector<char>>& space , std::vector<Bullet>& bullet) ; //to
oid Space (int& size , std::vector<std::vector<char>>& space ,Spaceship& spaceship ,Enemy& enemy , std::vector<Bullet>& bullet) ; //to give the first position to sp
oid LoadSpace (int& size , std::vector<std::vector<char>>& space ,Spaceship& spaceship ,Enemy& enemy, std::vector<Bullet>& bullet) ; //to load information on map
oid Map(MapInfo mapInfo, std::vector<std::vector<char>>& space, int& heal , int& CurrentPoint , Enemy enemy) ; //to generate map
oid RandomEnemy(int& y , int& size) ; //to give random place to enemy
oid Dart(int& size, std::vector<std::vector<char>>& space , Enemy& enemy, std::vector<Bullet>& bullet) ; //to make Dart spaceship and return its y
oid Striker(int& size, std::vector<std::vector<char>>& space, Enemy& enemy, std::vector<Bullet>& bullet) ; //to make Striker spaceship and return its y
oid Wraith(int& size, std::vector<std::vector<char>>& space, Enemy& enemy, std::vector<Bullet>& bullet) ; //to make Wraith spaceship and return its y
oid Banshee(int& size, std::vector<std::vector<char>>& space, Enemy& enemy, std::vector<Bullet>& bullet) ; //to make Banshee spaceship and return its y
oid Mover(Spaceship& spaceship , Enemy& enemy ,std::vector<std::vector<char>>& space , MapInfo& mapInfo , int& CurrentPoint ,  std::vector<Bullet>& bullet) ; //to m
oid MoveEnemy (Spaceship& spaceship , Enemy& enemy ,std::vector<std::vector<char>>& space , MapInfo& mapInfo , std::vector<Bullet>& bullet) ; //to move enemy
oid Attack (Spaceship& spaceship , Enemy& enemy ,std::vector<std::vector<char>>& space , MapInfo& mapInfo , int& CurrentPoint ,  std::vector<Bullet>& bullet) ; //to
oid Right (Spaceship& spaceship , Enemy& enemy ,std::vector<std::vector<char>>& space , MapInfo& mapInfo, int& CurrentPoint,  std::vector<Bullet>& bullet) ; //to de
oid Left (Spaceship& spaceship , Enemy& enemy ,std::vector<std::vector<char>>& space , MapInfo& mapInfo, int& CurrentPoint,  std::vector<Bullet>& bullet) ; //to dp
oid DestroyEnemy(std::vector<std::vector<char>>& space , MapInfo& mapInfo ) ; // to destroy enemy
oid InsertEnemy(int& size , std::vector<std::vector<char>>& space ,Spaceship& spaceship ,Enemy& enemy , std::vector<Bullet>& bullet) ; // to insert enemy in map
oid Damage(Spaceship& spaceship , std::vector<std::vector<char>>& space , Enemy& enemy , MapInfo& mapInfo ) ; //to decrease our heal when the spaceships are in coll
oid SaveGame(std::vector<std::vector<char>>& space , Spaceship& spaceship , Enemy& enemy , int& CurrentPoint , MapInfo& mapInfo, std::vector<Bullet>& bullet) ; //to
oid LoadGame(Spaceship& spaceship , Enemy& enemy , int& CurrentPoint , MapInfo& mapInfo, std::vector<Bullet>& bullet) ; //to load game from aa textfile
nt stringToInt(std::string txt) ; //to convert string to int
oid Win(std::vector<std::vector<char>>& space , Spaceship& spaceship , Enemy& enemy , int& CurrentPoint , MapInfo& mapInfo, std::vector<Bullet>& bullet) ; //to show
oid Lose() ; //to show lose popup
```

LIBRARIES

```
#include<iostream>
#include<vector>
#include<stdlib.h>
#include<time.h>
#include <thread>
#include <chrono>
#include<fstream>
#include<sstream>
```

از کتابخانه وکتور جهت ایجاد وکتور و کتابخانه Stdlib.h جهت ایجاد تابع رندوم و رنگ ها و تایم برای استفاده از زمان به عنوان ورودی تابع رندوم و کرونو و ترد برای افزودن وقفه زمانی و فایل استریم برای استفاده از فایل و استرینگ استریم برای کلمه به کلمه کردن یک رشته استرینگی کاربرد دارد .

نکته : می توانستیم از تابع کرونو و ترد استفاده نکنیم و از اسلیپ استفاده کنیم اما این روش بهینه تر بود .

STRUCTS

```
// to hold our spaceship's location and it's he
struct Spaceship
{
    int x;
    int y;
    int heal;
};

// to hold map's information
struct MapInfo
{
    int size;
    int point;
    int level ;
    int DestroyedEnemy[4]{0,0,0,0} ;
};

// to save enemy's informations
struct Enemy
{
    std::string name;
    int heal;
    int point;
    int x;
    int y;
    bool ltr;
};

// to save bullets location
struct Bullet
{

    int x;
    int y;
};
```

4 تا استراکت تعریف شده که به ترتیب اطلاعات سفینه خودی و نقشه بازی و سفینه دشمن و گلوله را در بر می گیرند و ما می توانم متغیر هایی از تایپ آنها هر جا که لازم بود تعریف کنیم .

FUNCTIONS

بخش اصلی و بدنه کلی پروژه بر اساس ۲۷ فانکشن تعریف شده که در ادامه توضیح مختصری از عملکرد هر فانکشن می دهم .

```
void SatrtMenue() ; //to make start menue
void Puase(MapInfo& mapInfo , Spaceship& spaceship , Enemy& enemy , int& CurrentPoint ,std::vector<std::vector<char>>& space ,  std::vector<Bullet>& bullet) ; //to sho
void Guidence() ; //to guide user to play
void ExitGame() ; //to exit game
MapInfo MapSize() ; //to take map size and point from user
void GenerateGame(int choice) ; //to generate game
void RunGame(MapInfo& mapInfo , Spaceship& spaceship , Enemy& enemy , int& CurrentPoint ,std::vector<std::vector<char>>& space , std::vector<Bullet>& bullet) ; //to ru
void Space (int& size , std::vector<std::vector<char>>& space ,Spaceship& spaceship ,Enemy& enemy , std::vector<Bullet>& bullet) ; //to give the first position to spac
void LoadSpace (int& size , std::vector<std::vector<char>>& space ,Spaceship& spaceship ,Enemy& enemy, std::vector<Bullet>& bullet) ; //to load information on map
void Map(MapInfo mapInfo, std::vector<std::vector<char>>& space, int& heal , int& CurrentPoint , Enemy enemy) ; //to generate map
void RandomEnemy(int& y , int& size) ; //to give random place to enemy
void Dart(int& size, std::vector<std::vector<char>>& space , Enemy& enemy, std::vector<Bullet>& bullet) ; //to make Dart spaceship and return its y
void Striker(int& size, std::vector<std::vector<char>>& space, Enemy& enemy, std::vector<Bullet>& bullet) ; //to make Striker spaceship and return its y
void Wraith(int& size, std::vector<std::vector<char>>& space, Enemy& enemy, std::vector<Bullet>& bullet) ; //to make Wraith spaceship and return its y
void Banshee(int& size, std::vector<std::vector<char>>& space, Enemy& enemy, std::vector<Bullet>& bullet) ; //to make Banshee spaceship and return its y
void Mover(Spaceship& spaceship , Enemy& enemy ,std::vector<std::vector<char>>& space , MapInfo& mapInfo , int& CurrentPoint ,  std::vector<Bullet>& bullet) ; //to mov
void MoveEnemy (Spaceship& spaceship , Enemy& enemy ,std::vector<std::vector<char>>& space , MapInfo& mapInfo , std::vector<Bullet>& bullet) ; //to move enemy
void Attack (Spaceship& spaceship , Enemy& enemy ,std::vector<std::vector<char>>& space , MapInfo& mapInfo , int& CurrentPoint ,  std::vector<Bullet>& bullet) ; //to d
void Right (Spaceship& spaceship , Enemy& enemy ,std::vector<std::vector<char>>& space , MapInfo& mapInfo, int& CurrentPoint,  std::vector<Bullet>& bullet) ; //to do r
void Left (Spaceship& spaceship , Enemy& enemy ,std::vector<std::vector<char>>& space , MapInfo& mapInfo, int& CurrentPoint,  std::vector<Bullet>& bullet) ; //to dp le
void DestroyEnemy(std::vector<std::vector<char>>& space , MapInfo& mapInfo ) ; // to destroy enemy
void InsertEnemy(int& size , std::vector<std::vector<char>>& space ,Spaceship& spaceship ,Enemy& enemy , std::vector<Bullet>& bullet) ; // to insert enemy in map
void Damage(Spaceship& spaceship , std::vector<std::vector<char>>& space , Enemy& enemy , MapInfo& mapInfo ) ; //to decrease our heal when the spaceships are in collis
void SaveGame(std::vector<std::vector<char>>& space , Spaceship& spaceship , Enemy& enemy , int& CurrentPoint , MapInfo& mapInfo, std::vector<Bullet>& bullet) ; //to s
void LoadGame(Spaceship& spaceship , Enemy& enemy , int& CurrentPoint , MapInfo& mapInfo, std::vector<Bullet>& bullet) ; //to load game from aa textfile
int stringToInt(std::string txt) ; //to convert string to int
void Win(std::vector<std::vector<char>>& space , Spaceship& spaceship , Enemy& enemy , int& CurrentPoint , MapInfo& mapInfo, std::vector<Bullet>& bullet) ; //to show w
void Lose() ; //to show lose popup
```

نکته : تمامی این توابع باهم در ارتباط اند و دائما دسته ای از متغیر ها را پاس می دهند و عملیات هایی را روی آنها اعمال می کنند .

START MENU

منو اولیه و شروع بازی را با گزینه های شروع و لود کردن و راهنما و خروج را به کاربر نمایش می دهد .

```
//to show the menue to user and  ask how he wants to play
void SatrtMenu(){
system ("CLS") ;
std::cout<<"1- Start game"<<'\n' ;
std::cout<<"2- Load game"<<'\n' ;
std::cout<<"3- Guidence"<<'\n' ;
std::cout<<"4- Exit game"<<'\n' ;
int choice ;
do{
std::cin>>choice ;
switch (choice)
{
case 1 :
    GenerateGame(1) ;
    break;
case 2 :
    GenerateGame(2) ;
    break ;
case 3 :
    Guidence() ;
    break;
case 4 :
    ExitGame() ;
    break ;

default:
std::cout<<"invalid choice"<<'\n' ;
    break;
}
}while(choice != 1 && choice != 2 && choice != 3 && choice != 4 ) ;

}
```

PAUSE

منو پاپ آپ را به کاربر در صورتی که کاربر قصد توقف بازی را داشته باشد نمایش می دهد که شامل گزینه های ادامه بازی و سیو و خروج است .

GUIDANCE

صفحه راهنما بازی را در صورت درخواست کاربر (اشاره به آن در منو استارت) نمایش می دهد .

EXITGAME

در صورت درخواست کاربر به خروج در منو اولیه اجرا می شود و از کاربر ابتدا می پرسد که آیا می خواهد خارج شود و اگر کاربر تایید کرد برنامه را می بندد .

MapSize

سایز مپ و امتیاز کل را از کاربر به به عنوان ورودی می گیرد و در قالب استراکتی از اطلاعات مپ بر می گرداند .

GenerateGame

این تابع و تابع ران اساسی ترین توابع اند به این دلیل که این تابع بنا به ورودی ای که می گیرد یک بازی جدید ایجاد یا یک بازی سیو شده را لود می کند سپس با فراخوانی تابع ران بازی را به اجرا در می آورد .

RunGame

این تابع اساسی ترین تابع است و اگر نباشد عملا برنامه به طور کلی اجرا نمی شود که در این تابع یک حلقه داریم که دائما توابع دیگر را فرا می خواند و در صورت برد یا باخت متوقف می شود .

```
//to run game
void RunGame(MapInfo& mapInfo , Spaceship& spaceship , Enemy& enemy , int& CurrentPoint ,std::vector<std::vector<char>>& space , std::vector<Bullet>& bullet){


while(CurrentPoint<mapInfo.point){

    if(spaceship.heal <1){
        break;
    }

Mover(spaceship , enemy , space , mapInfo , CurrentPoint , bullet) ;

if(enemy.heal == 0){
    CurrentPoint += enemy.point ;
    DestroyEnemy(space , mapInfo) ;
    InsertEnemy(mapInfo.size , space , spaceship , enemy , bullet) ;
    SaveGame (space , spaceship , enemy , CurrentPoint , mapInfo , bullet) ;
    Map(mapInfo , space, spaceship.heal , CurrentPoint , enemy);

}
}

if(spaceship.heal<1){
Lose() ;
}
else{
Win(space , spaceship ,  enemy , CurrentPoint , mapInfo , bullet) ;
}
}
```

Space

محل اولیه سفینه خودی را مشخص می کند و با فراخوانی تابع اینزرت یک دشمن رندوم وارد مپ می کند و توجه کنید که محل اولیه سفینه خودی وسط ردیف پایینی فضا است .

LoadSpace

این تابع تفاوت اساسی با تابع بالا دارد و این تقاوت این است که این تابع بر اساس اطلاعات لود شده از فایل متنی محل سفینه هارا مشخص می کند .

Map

این تابع همانطور که از اسمش مشخص است وظیفه ایجاد مپ را به صورت گرافیکی بر عهده دارد .

RandomEnemy

وظیفه دارد که یک موقعیت رندوم به انمی بدهد تا از آن برای ایجاد انمی رندوم استفاده کنیم.

Dart , Striker , Wraith , Banshee

همانظور که از نام ها مشخص است هر کدام وظیفه ایجاد یک نوع سفینه را بر اساس یه موقعیت عرضی را دارند که به عنوان ورودی به آنها می دهیم .

Mover

این تابع ورودی ای برای حرکت از کاربر می گیرد و بر اساس آن سفینه خودی را به حرکت در می آرد و شلیک می کند .

MoveEnemy

این تابع بعد از هر حرکت سفینه خودی فراخوانی می شود که وظیفه دارد سفینه دشمن یک واحد به پایین بیاورد .

Attack

این تابع وظیفه شلیک کردن را بر عهده دارد و در تابع حرکت فراخوانی می شود و با هر حرکت انجام شده یک شلیک انجام می دهد .

Right , Left

این دو تابع که در تابع حرکت فراخوانده می شود وظیفه حرکت دادن سفینه به چپ و راست را بر عهده دارند بعد در خود تابع اتک را فراخوانی می کنند .(زیرا بعد هر حرکت اتک فراخوانی می شود .

DestroyEnemy

وظیفه نابودی انمی و پاک کردن آن از مپ را بر عهده دارد .

InsertEnemy

وظیفه ایجاد یک سفینه رندوم در مپ را بر عهده دارد که هر بار یک انمی نابود می شود فراخوانی می شود تا یک سفینه دیگر وارد مپ شود .

Damage

وظیفه کم کردن هیل سفینه خودی را در صورت برخورد سفینه دشمن با آن را دارد .

SaveGame

وظیفه ذخیره اطلاعات بازی را بر روی تکست فایل بر عهده دارد .

LoadGame

وظیفه خواندن از فایل و انتقال مقادیر به متغیر هارا بر عهده دارد .

StringzToInt

تبدیل استرینگ به اینت . (این تالع به صورت آماده وجود داشت اما برای جلوگیری از ارور دستی نوشته شد .

Win

ایجاد پاپ آپ برد بازی و سوال از کاربر که آیا قصد ادامه دارد .

Lose

ایجاد پاپ آپ باخت بازی .

Level

بالا بردن لول بازی و برسی آن.

HISTORY

ذخیره کردن تاریخچه بازی در فایل جداگانه پس از برد یا باخت هر مرحله بازی.

Type

دریافت تایپ سفینه خودی به عنوان ورودی از کاربر .

از این پروژه چه چیز هایی یاد گرفتید ؟

پیاده سازی این پروژه به طور کلی در بخش گیم پلی نسبت به مینی پروژه فانکشن های بیشتری را طلب می کرد از سویی ارتباط تنگاتنگ این فانکشن ها باهم سبب تبادل متوالی متغیر ها شد پس می توان گفت این پروژه در بحث تقویت مهارت پیاده سازی ماژوال آموزش های مفیدی داشت.

از سویی دیگر این پروژه از وکتور و استراکت ها استفاده بالایی داشت و مهارت کار با متود های وکتور را تقویت کرد .

اما بخش اساسی تر از نظر آموزشی پیاده سازی توابع سیو و لود بود که نیاز به خواندن و نوشتن از فایل در یک پروژه بزرگ را داشت .

همچنین برای ذخیره ایده های نوعی وجود .

در صفحه بعد نمایی از تابع ذخیره را می بینید .

```
//to save game in a text file
void SaveGame(std::vector<std::vector<char>>& space , Spaceship& spaceship , Enemy& enemy , int& CurrentPoint , MapInfo&

std::ofstream Save ;

Save.open("GameInfo.txt" ,  std::ios::out) ;

if(Save.is_open()){
Save<<spaceship.heal<<' '<<spaceship.x<<' '<<spaceship.y<<'\n' ;

Save<<enemy.name<<' '<<enemy.point<<' '<<enemy.heal<<' '<<enemy.ltr<<' '<<enemy.x<<' '<<enemy.y<<'\n' ;

Save<<mapInfo.size<<' '<<mapInfo.point<<'\n' ;

Save<<CurrentPoint<<'\n' ;

for(int i = 0 ; i<bullet.size() ; i++){

Save<<bullet[i].x<<' '<<bullet[i].y<<'\n';

}

}
else{
    std::cerr<<"cant open the file\n" ;

    lose() ;
```

برای ذخیره فقط موقعیت ها نیاز به ذخیره داشت که این خود به نوعی ایده پردازی و آشایی با روش ذخیره سازی بازی ها بود .

آیا راه دیگری پیدا کردید ؟

در ساختار کلی پروژه راه کلی تقریبا ساختار کلی اجرا همین است اما اگر در بخش های کوچک تر مثل تابع نابودی انمی متمرکز تر شویم :

```
//to destroy enemy
void DestroyEnemy(std::vector<std::vector<char>>& space , MapInfo& mapInfo){

for(int i = 0 ; i<mapInfo.size ; i++){
    for(int j = 0 ; j<mapInfo.size ; j++){
        if(space[i][j] == '*')
            space[i][j] = ' ' ;
    }
}

}
```

مثلا می توانستیم نابودی انمی را بر اساس موقعیت ذخیره شده اعمال کنیم و در تابع حرکت انمی هم به همین صورت اما این راه کد کمتری نیاز داشت .

یا اگر بخوایم راه دیگری برای ساختار کلی اعمال کنیم استفاده از متغیر گلوبال بود زیرا هم کد کوتاه تر می شد هم به این شدت پاسکاری رفرنس ها و اشاره گر ها صورت نمی گرفت .

در خصوص توابع دیگربرای تابع سیو می توانستیم مپ را هم ذخیره کنیم اما کار جالب و بهینه ای نبود .

در بخش دمیج به سفینه خودی از برسی مپ کمک گرفتیم که می شد با کمک موقعیت انمی نیز اعمال شد .

در ساختار این پروژه از رفرنس ها استفاده شد اما می توانستیم که مقادیر را از توابع بر گردانیم و در متغیر ها بریزیم .

امتیازی ها

1 . تیر اتومات (کامنت شده)

۲ . لول

۳ . اطلاعات سفینه های نابود شده

۴ . ذخیره تاریخچه بازی در فایل جداگانه

۵ . تایپ برای سفینه خودی

۶ . سطح بندی اینزرت شدن انمی ها

GIT AND GITHUB

پروژه بر روی گیت نیز ذخیره شده که می توانید از لینک زیر مشاهده کنید :

منابع

تمامی توابع دستی نوشته شده و فقط برای برخی نکات مانند اضافه کردن اسلیپ از سایت استک اور فلو کمک گرفته شده .