فاطمه شکری -حسنا شیرزادی

۴۰۲۱۲۳۵۸۰۲۲

۴۰۲۱۲۳۵۸۰۲۵

مبانی کامپیوتر و برنامه سازی

استاد حسن بشیری

پاییز ۱۴۰۲-دانشگاه بوعلی سینا

فهرست

مقدمه

هممون بازی های آتاری رو یادمونه...حالا یا فقط دیده بودیم یا بازی هم کرده بودیم ....اون موقع که هنوز پلی استیشن و کنسول و این جور چیزا مد نبود دست همه بچه ها یه چیزی بود شبیه دسته بازی هایی که الان داریم و توش یه سری بازی داشت که حتی رنگی هم نبود و سیاه و سفید بود...اون موقع داشتن آتاری دیگه ته پولدار بودن بود..البته میدونید آتاری بیشتر نوستالژی دهه شصتی ها است تا ما (دهه هشتادی و دهه هفتادی ها)ولی به هر حال...این بازی تقریبا یه بازی شبیه بازی هواپیما آتاریه....یعنی کلیت بازی همون بازی های آتاریه ولیییی با تکنولوژی های جدید.....

پروژه در یک نگاه

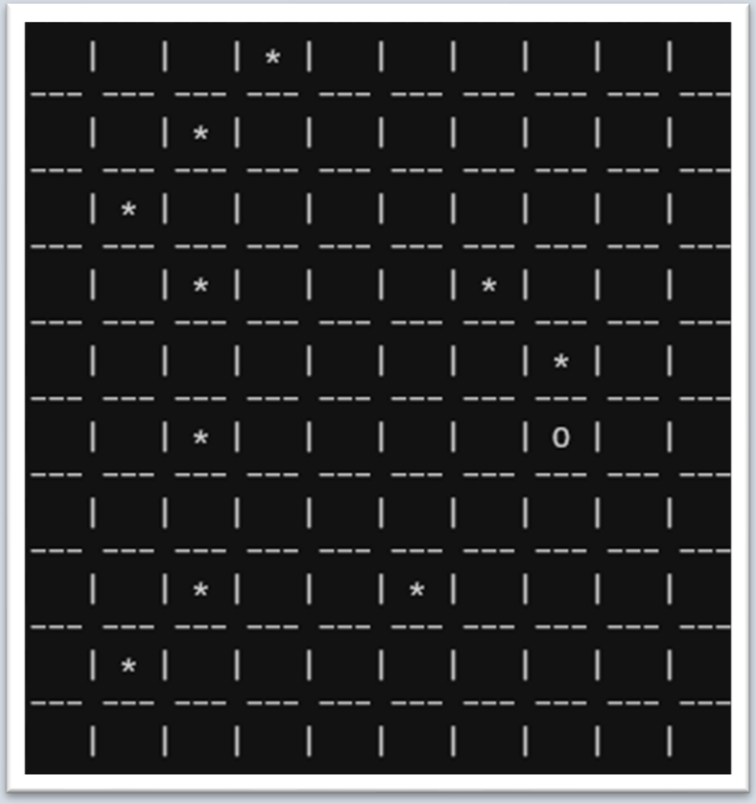
بازی در فضای دو بعدی که کاربر با حرکت دادن سفینه خود و رساندن آن به سفینه های دشمن و قرار گرفتن در موقعیت مناسب نسبت به نابودی آنها اقدام و کسب امتیاز میکند.

هدف بازی :

در این بازی شما یک سفینه فضایی هستید که با حرف انگلیسی (O) در صفحه ی طراحی شده نمایش داده می شوید و هدف شما در این بازی از بین بردن تمام دشمنانی است که در صفحه با علامت (*) نشان داده می شوند .

ساختار این بازی :

- شما در یک صفحه گرافیکی ۱۰ در ۱۰ قرار دارید (map size=10) , و با حرکت در این فضا به دنبال دشمنان هستید.



- کاربر می تواند با وارد کردن دستورات در صفحه کلید (دستور U : حرکت به بالا , دستور D : حرکت به پایین , دستور R : حرکت به راست , دستور L : حرکت به چپ , دستور S : شلیک کردن) در چهار جهت در صفحه ی طراحی شده حرکت کند و در دو جهت چپ و راست به سمت دشمن شلیک کند.
- در این بازی برای سفینه شما سه جان تعریف شده و اگر در طول بازی سفینه ی شما به دشمن برخورد کند , شما یک جان خود را از دست میدهید.

!!!!یادتان باشد که در دو صورت بازی به پایان میرسد!!!

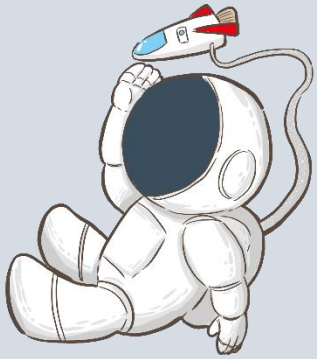
- ۱) اگر با شلیک به سمت دشمن همه ی سفینه های دشمن را نابود کرده باشید (امتیازات شما برابر تعداد سفینه های دشمن باشد) و برنده شوید.

```
Congratulations!MISSION ACCOMPLISHED!
Your score: 11
```

- ۲) اگر شما در حین بازی تمام جان خود را از دست داده باشید و **«Game over»** شوید .

```
MISSION FAILED!
Your score: 0
```

توابع functions



در این بازی , از چند تابع مختلف برای اجرای عملیات های متنوعی نظیر حرکت کاربر , برخورد با دشمنان , اعلام پیام ها و مدیریت وضعیت بازی استفاده شده است . در ادامه به توضیح توابع اصلی این برنامه می پردازیم :

۱-پاک کردن صفحه"Clear Screen"

این تابع برای پاک کردن صفحه ترمینال یا کنسول استفاده می شود . البته این قسمت برای سیستم عامل های مختلف ممکن است نیاز به تنظیمات خاصی داشته باشد . عملکرد این تابع به این صورت است که بعد از اتمام بازی (در صورت برد یا باخت) ابتدا صفحه بازی پاک شده سپس پیام مناسب نشان داده میشود.

```
#ifdef _WIN32
void clearScreen() {
    system("cls");
}
#else
void clearScreen() {
    system("clear");
}
#endif
```

۲-مقدار دهی اولیه بازی "Initialize Game"

این تابع برای شروع بازی و مقدار دهی اولیه , نقش و ویژگی بازیکن ها کاربرد دارد .بعنوان مثال جایگذاری سفینه های دشمن و سفینه خودی که بصورت رندوم است در این تابع تعریف میشود.

```
void initializeGame() {
    // مقداردهی اولیه بازی
}
```

۳-کلاس سفینه"Class Ship"

```
class Ship {
public:
    int x, y;
    int health;

    Ship(int startX, int startY, int startHealth) : x(startX), y(start

    void move(int d, int mapsize) {
        // حرکت سفینه به سمت مشخص شده
    }
};
```

تابع این کلاس برای نمایش و مدیریت سفینه کاربر در بازی است . سفینه دارای موقعیت (x , y) متفاوت است . برای حرکت سفینه به سمت های مختلف از move استفاده می شود .

۴-بروزرسانی بازی"Update Game"

```
}

void updateGame() {
    // Update the game logic
}
```

این تابع برای بروزرسانی منطق بازی استفاده می شود .

۴-کلاس بازی"Class Game"

```
class Game {
private:
    const int mapsize = 10;
    int mapIndices[10][10] = {0};
    Ship player{0, 0, 3};
    int score;

public:
    // توابع مربوط به مدیریت بازی
    void initializeGame();
    void updateGame();
    void renderGame();
    bool isGameOver();
    bool isGameWin();
    void gameOver();
    void gameWin();

    // توابع مربوط به ورودی و رفتار بازیکن
    void handleInput(char input);
    void move(int direction);
    void shoot();
};
```

این کلاس بطور کلی مسئول بازی است . از این کلاس و توابع برای مدیریت اطلاعات صفحه ی بازی , اطلاعات بازیکن , امتیاز و تعیین وضعیت بازی استفاده می شود .

۶-وضعیت فعلی بازی"Render Game"

```
void renderGame() {
    // نمایش وضعیت فعلی بازی به کاربر
}
```

این تابع برای نمایش وضعیت جاری بازی , اطلاعات صفحه نمایش , مکان بازیکن و دشمنان کاربرد دارد .

۷-باخت"Is Game Over"

```
bool isGameOver() {
    return player.health <= 0;
}
```

این تابع برای برسی وضعیت پایان بازی (در صورت باختن) استفاده می شود .

۸-برد"Is Game Win"

```
bool isGameWin() {
    // بررسی شرط پیروزی بازی
}
```

این تابع برای برسی وضعیت پایان بازی (در صورت بردن) استفاده می شود .

۹-مدیریت ورودی ها"Handle Input"

```
void handleInput(char input) {
    // Handle user input
```

این تابع برای پردازش ورودی کاربر و اعمال تغییرات مرتبط به حرکت یا شلیک کاربرد دارد.

۱۰-جابجایی"Move"

```
void move(int direction) {
    // حرکت بازیکن به سمت مشخص شده
}
```

این تابع برای حرکت سفینه به سمت های مختلف کاربرد دارد .

۱۱-حمله یا شات کردن"Shoot"

```
void shoot() {
    // شلیک بازیکن به دشمنان
}
```

این تابع برای انجام عمل شلیک بازیکن به دشمن کاربرد دارد .

۱۲-باخت بازی"Game Over"

این تابع برای نمایش پیام پایان بازی (در صورت باخت) استفاده می شود .

```
void gameOver() {
    // پایان بازی با شرایط باخت
}
```

۱۳-برد بازی"Game Win

این تابع برای نمایش پیام پایان بازی (در صورت برد)است .

```
void gameWin() {
    // پایان بازی با شرایط برد
}
```

*این توابع با همکاری یکدیگر به منظور اجرای اصول بازی و ارتباط با کاربر در طول فرایند بازی مشغول به کار هستند *

توضیحات قطعه کد ها :

در این قطعه کد حرف 'd' مخفف جهت حرکت"direction" است .

```
void move(int d, int mapsize) {
    // 0: Left 1: Right 2: Up 3: Down

    // Check the direction and update coordinates accordingly
    if (d == 0 && y > 0) {
        y--;
    } else if (d == 1 && y < mapsize - 1) {
        y++;
    } else if (d == 2 && x > 0) {
        x--;
    } else if (d == 3 && x < mapsize - 1) {
        x++;
    }
}
```

در اینجا مقادیر ممکن آن عبارت اند از :

- "0": حرکت به چپ
- "1": حرکت به راست
- "2": حرکت به بالا
- "3" : حرکت به چپ

برای مثال در این تابع با استفاده از بلوک شرطی (if-else if) جهت حرکت را اعمال میکند :

- اگر y بزرگتر یا مساوی با صفر باشد"d" یک واحد کاهش میابد و حرکت به سمت چپ را نمایش می دهد .
- اگر y کوچک تر یا مساوی با صفر باشد "d" یک واحد افزایش میابد و حرکت به سمت راست را نمایش می دهد.
- و همینطور جهت های دیگر...

```
// Check for collision with an enemy after each move
if (mapIndices[player.x][player.y] == 2) {
    std::cout << "Collision with enemy! Lose one health." << std::endl;
    player.health--;

    // Check if player health is zero
    if (player.health <= 0) {
        gameOver();
        return;
    }
```

این قسمت از کد بررسی می کند که آیا سفینه ما به سفینه دشمن برخورد کرده است یا خیر!

در "if" اول اگر شرط برقرار باشد اعلام می کند که بازیکن با سفینه دشمن برخورد کرده و یک واحد از سلامتی اش کاسته می شود .

در " if "دوم بررسی میشود که وضعیت جانهایمان چگونه است.اگر جانهایمان به صفر رسیده باشد (شرط برقرار باشد) به سراغ تابع **"game Over"** رفته و بازی به اتمام میرسد.

این بخش از کد مربوط به یک شمارنده در بازی است و بررسی می کند که اگر تعداد حرکات به یک حد مشخص برسد , به وضعیت پایان بازی منتقل می شود .

```
static int moveCount = 0;
if (++moveCount >= 1000) {
    isGameOver = true;
}
```

این ماکزیمم به عنوان یک محدودیتی برای انجام حرکات در بازی به کار می رود که جلوی اجرای بی نهایت بازی را بگیرد .

دراین قسمت از کد , متغیر های مربوط به محیط بازی و اطلاعات بازیکن تعریف میشود .

بعنوان مثال این کد مشخص میکند که اندازه صفحه بازی 10 در 10 است و از آرایه دو بعدی برای رسم آن استفاده میشود که هر خانه از این آرایه ها می تواند یک وضعیت خاص را نمایش دهد .

```
class Game {
private:
    const int mapsize = 10;
    int mapIndices[10][10] = {0}; // Initialize mapIndices with zeros
    Ship player{0, 0, 3}; // Initialize player position and health
    int score;
```

علاوه بر آن موقعیت اولیه بازیکن و سلامتی اولیه را نیز میتوان در این کد دید که تعریف شده اند.

در این کد مقدار امتیاز هر بازیکن نمایش داده می شود و با هر شلیک سفینه به سمت دشمن امتیاز بازیکن افزایش میابد.