**МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ**

**ДЕРЖАВНИЙ УНІВЕРСИТЕТ ЕКОНОМІКИ І ТЕХНОЛОГІЙ**

Кафедра інформатики

і прикладного програмного забезпечення

**КУРСОВА РОБОТА**

|  |  |
| --- | --- |
| з дисципліни | “Об’єктно - орієнтоване програмування” |
|  | (назва дисципліни) |

|  |  |
| --- | --- |
| на тему: | “Розробка гри морський бій” |
| (назва теми) | |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Студента | 2 | курсу | ФІТ | групи | ІПЗ-21-1 |
|  |  |  | (назва факультету) |  |  |

|  |
| --- |
| Носур Данило |
| (прізвище, ім’я та по батькові, підпис) |

|  |
| --- |
| Керівник курсової роботи |
| к.е.н., доцент Лисенко В.С. |
| (прізвище, ім’я, по батькові) |

Кривий Ріг 2023

**МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ**

**ДЕРЖАВНИЙ УНІВЕРСИТЕТ ЕКОНОМІКИ І ТЕХНОЛОГІЙ**

Кафедра інформатики

і прикладного програмного забезпечення

|  |
| --- |
| ЗАТВЕРДЖУЮ |
| Завідувач кафедри |
| д.т.н., професор Зеленський О.С. |
| “ \_\_\_\_\_“\_\_\_\_\_\_\_\_\_\_\_201\_\_\_\_\_\_р. |

**ЗАВДАННЯ**

|  |  |
| --- | --- |
| На курсову роботу з дисципліни | Об’єктно - орієнтоване програмування |
|  | (назва дисципліни) |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Студента | 2 | курсу | ФІТ | групи | ІПЗ-21-1 |
|  |  |  | (назва факультету) |  |  |

|  |
| --- |
| Носур Данило |
| (прізвище, ім’я та по батькові, підпис) |

|  |  |
| --- | --- |
| Тема курсової роботи | “Розробка гри морський бій” |
|  | (назва теми) |

|  |  |
| --- | --- |
| Керівник курсової роботи | к.е.н., доцент Лисенко В.С. |
|  | (прізвище, ім’я та по батькові) |

|  |  |
| --- | --- |
| Мета роботи | створення гри на запам’ятовування |
|  | |
| мова програмування Visual C++; | |
| (мета, вихідні дані, програмне забезпечення, середовища розробки, СУБД) | |

**РЕФЕРАТ**

-

**“Розробка гри морський бій”**

Курсова робота студента 2 курсу напряму підготовки "Програмна інженерія" з дисципліни: "Об’єктно-орієнтоване програмування". Державний університет економіки та технологій. Кривий Ріг, 2023.

**Обсяг роботи**\_\_ 52\_\_стор.\_\_3\_\_табл.\_\_6\_\_рис.\_\_7\_\_джерел.

**Предмет дослідження**: інформаційна підтримка розробки гри морський бій.

**Об’єкт дослідження**: гра морський бій.

**Мета роботи**: створення гри морський бій.

**Методи дослідження**: методи теорії алгоритмів, методи об’єктно-орієнтованого проектування, методика роботи з мапами.

**Програмне забезпечення:** мова програмування Visual C++.

**Результати та їх новизна**: створено гри морський бій.

**Висновки**: розроблено гру морський бій.

**Ключові слова**: ГРА МОРСЬКИЙ БІЙ, АЛГОРИТМИ, МАПИ, ПРОГРАМНЕ ЗАБЕЗПЕЧЕННЯ.

# ЗМІСТ

[ЗМІСТ 4](#_Toc134832117)

[ВСТУП 5](#_Toc134832118)

[РОЗДІЛ 1. ПОСТАНОВКА ЗАДАЧІ 6](#_Toc134832119)

[1.1.Характеристика задачі 6](#_Toc134832120)

[1.2.Вхідна інформація 6](#_Toc134832121)

[1.3 Вихідна інформація 8](#_Toc134832122)

[РОЗДІЛ 2. РОЗРОБКА АЛГОРИТМУ РОЗВ’ЯЗАННЯ ЗАДАЧІ 9](#_Toc134832123)

[2.1 Розробка методів 9](#_Toc134832124)

[РОЗДІЛ 3. ОРГАНІЗАЦІЯ ІНФОРМАЦІЙНОГО ЗАБЕЗПЕЧЕННЯ 10](#_Toc134832125)

[3.1 Структура класів 10](#_Toc134832126)

[РОЗДІЛ 4. РОЗРОБКА ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ ЗАДАЧІ. 12](#_Toc134832127)

[ВИСНОВКИ 16](#_Toc134832128)

[СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ 25](#_Toc134832129)

ВСТУП

Ігор в просторі Інтернету дуже багато, вони є різні але я вирішив створити гру, яка не лише зможе зробити цікавим відпочинок, а й принесе користь користувачеві. Гра морський бій – це класика серед тактичних та тренувальних ігор на логіку, та тактичне мисленя. Вона є дуже цікавою та різноманітною, із багатьма варіаціями розвитку подій на ігровому полу. Також, у цьому консольному додатку будуть застосовані кольори, та можливість керувати та обирати об’єкти на екрані консолі за допомогою правої кнопки миші та клавіатури, що внесе до гри легкість та приємність у експлуатації.

Мета курсової роботи – створення програмного забезпечення, котре уособлюватиме собою усім нам відому гру – морський бій.

Головне завдання роботи: повноцінне функціонування програмного забезпечення під час використання, та наявність штучного інтелекту у ворога на достатньому рівні.

Розроблена програма виступає моделлю програмного забезпечення гри морський бій.

Продукт розроблено інструментальними засобами мови програмування С++.

# РОЗДІЛ 1. ПОСТАНОВКА ЗАДАЧІ

## 1.1.Характеристика задачі

При створенні даного програмного забезпечення, було поставлено основну задачу – спроектувати гру морський бій, котра міститиму у собі кольори, зручне ведення даних, та ворога зі штучним інтелектом.

Підкреслимо основне завдання:

1. Повноцінне функціонування гри;
2. Ведення та створення ігрового поля за допомогою клавіатури та миші
3. Різноманітність кольорів на екрані.
4. Ворог зі штучним інтелектом

Таким чином необхідно створити гру, яка може змінюватися в залежності від введеної інформації користувачем .

## 1.2.Вхідна інформація

Вхідна інформація та її коректність є найважливішими параметрами для точної та автоматизованої гри. Всі основні етапи гри (Початок, бій та кінець), та спостерігання за станом подій на полі бою, відбуваються у класі «GameController». На початку роботи програми користувачеві необхідно розставити свої кораблі на полі бою (Зображення 1.1).

Рис. 1.1 Поле користувача

В залежності від корабля, котрий користувачеві необхідно розташувати на полі, буде змінено його назву (де 1 – однопалубний, 4 – чотирьох палубний) та його розмір на екрані.

Зображення, що містить знімок екрана, текст

Автоматично згенерований описЗа для того щоб розташувати корабель на обраному користувачем місці – необхідно натиснути клавішу “Enter”. Якщо корабель розташовано на місці, де його розташувати можна, він буде зеленого кольору, та при натисканні на “Enter” – корабель буде розташовано, в іншому випадку, корабель буде червоного кольору, що означатиме неможливість розташовування корабля на обраній позиції (Зображення 1.2).

Рис. 1.2 Корабель неможливо розташувати

За для обертання користувача користувачеві необхідно натиснути клавішу “R”. Також, у будь-який момент користувач може очистити своє поле, натиснувши на клавішу “C”, або скористатися авто – генеруванням поля, натиснувши клавішу “F”(Зображення 1.3).

Зображення, що містить знімок екрана

Автоматично згенерований опис

Рис. 1.3 Автоматично згенероване поле користувача

При успішному розташуванні усіх кораблів на полі, за для початку гри користувачеві необхідно натиснути клавішу “Space”, коли та має зелений колір обарвлення. Згодом, почнеться етап бою (Зображення 1.4).

Зображення, що містить знімок екрана, текст, дизайн

Автоматично згенерований опис

Рис. 1.4 Початок бою

Як можна помітити – поле ворога має трохи інше обарвлення, так як ми не маємо розуміння про позиції ворожих кораблів – вони знаходяться для нас у тумані.

За для пострілу по ворожому полю, користувачеві необхідно натиснути лівою кнопкою миші по комірці поля ворога(Зображення 1.5).

Зображення, що містить знімок екрана

Автоматично згенерований опис

Рис. 1.5 Постріл по ворожому полю

Таким чином, при промаху, на комірці, по якій був проведений постріл, буде відображено “o”, у результаті влучання – “X”.

1.3 Кінцевий результат роботи програми

Кінцевим результатом роботи є виведення результатів гри за котрі відповідає метод «End». При негативній перемозі ворога у кінці гри користувачеві буде виведене наступне повідомлення (Зображення 1.6).

Зображення, що містить текст, Шрифт, знімок екрана, дизайн

Автоматично згенерований опис

Рис. 1.6 Повідомлення про перемогу

Тут користувачеві запропоновано зіграти повторно. За для цього користувачеві потрібно натиснути на клавішу “R”.

При поразці гравця повідомлення матиме негативний зміст, та червоний колір забарвлення повідомлення.

РОЗДІЛ 2. РОЗРОБКА АЛГОРИТМУ РОЗВ’ЯЗАННЯ ЗАДАЧІ

2.1 Розробка методів

Програма працює за наступним алгоритмом:

1. Як було сказано раніше, головним класом програми є клас “GameContoller”, котрий має три основні методи, котрі відповідають періоду, у котрім перебуває гра. Це методи “Start”, “Battle” та “End”.

void GameController::Start()

{

clear();

end = false;

win = false;

userField.ShowField(false, 10, 10);

map<int, int> ships;

int maxShipSize = 4;

int selectedShip = 1;

bool horizontal = true;

auto initShipsMap = [](map<int, int>& ships, int& maxShipSize) {

for (int i = 0; i < maxShipSize; i++) {

ships[i + 1] = maxShipSize - i;

}

};

auto setBottomMenu = [](Field& userField, Color& color) {

gotoxy(10, 25);

color.SetColor(color.LightCyan);

cout << "Enter - Set ship ";

color.SetColor(color.LightMagenta);

cout << "R - Reverse ship ";

color.SetColor(color.Brown);

cout << "C - Clear field ";

color.SetColor(color.Cyan);

cout << "F - Auto set ";

if (userField.GetShipsCount() == 10) color.SetColor(color.Green);

else color.SetColor(color.Red);

cout << "Space - Continue";

};

initShipsMap(ships, maxShipSize);

COORD currPos = {5, 9};

COORD endPos = {-1, -1};

setBottomMenu(userField, color);

bool exit = false;

int press = 0;

while (!exit) {

switch (press)

{

case 13: { //Enter

if (userField.GetShipsCount() < 10 && endPos.X >= 0 && endPos.Y >=0 && selectedShip && ships[selectedShip] > 0 && userField.CanSetShip(currPos, endPos)) {

userField.SetShip(currPos, endPos, selectedShip);

ships[selectedShip] -= 1;

if (ships[selectedShip] <= 0) {

ships.erase(selectedShip);

selectedShip++;

if (selectedShip == 5) {

setBottomMenu(userField, color);

}

}

}

break;

}

case 32: { //Space

if (userField.GetShipsCount() >= 10) exit = true;

break;

}

case 72: { //Up

if (currPos.Y <= 0) break;

currPos.Y--;

userField.ClearColors();

endPos = { (SHORT)((horizontal) ? currPos.X + selectedShip - 1 : currPos.X), (SHORT)((horizontal) ? currPos.Y : currPos.Y + selectedShip - 1) };

userField.Draw(currPos, endPos, selectedShip);

break;

}

case 75: { //Left

if (currPos.X <= 0) break;

currPos.X--;

userField.ClearColors();

endPos = { (SHORT)((horizontal) ? currPos.X + selectedShip - 1 : currPos.X), (SHORT)((horizontal) ? currPos.Y : currPos.Y + selectedShip - 1) };

userField.Draw(currPos, endPos, selectedShip);

break;

}

case 77: { //Right

if (currPos.X + 1 >= userField.GetFieldWidth()) break;

currPos.X++;

userField.ClearColors();

endPos = { (SHORT)((horizontal) ? currPos.X + selectedShip - 1 : currPos.X), (SHORT)((horizontal) ? currPos.Y : currPos.Y + selectedShip - 1) };

userField.Draw(currPos, endPos, selectedShip);

break;

}

case 80: { //Down

if (currPos.Y + 1 >= userField.GetFieldHeight()) break;

currPos.Y++;

userField.ClearColors();

endPos = { (SHORT)((horizontal) ? currPos.X + selectedShip - 1 : currPos.X), (SHORT)((horizontal) ? currPos.Y : currPos.Y + selectedShip - 1) };

userField.Draw(currPos, endPos, selectedShip);

break;

}

case 99: { //C - clear

userField.ClearField();

userField.ClearColors();

selectedShip = 1;

setBottomMenu(userField, color);

initShipsMap(ships, maxShipSize);

break;

}

case 102: { //F - Generate

userField.Generate();

userField.ClearColors();

setBottomMenu(userField, color);

ships.clear();

break;

}

case 114: { //R - reverse

horizontal = !horizontal;

userField.ClearColors();

endPos = { (SHORT)((horizontal) ? currPos.X + selectedShip - 1 : currPos.X), (SHORT)((horizontal) ? currPos.Y : currPos.Y + selectedShip - 1) };

userField.Draw(currPos, endPos, selectedShip);

break;

}

default:

break;

}

press = \_getch();

//userField.ClearColors();

}

userField.ClearColors();

Battle();

}

void GameController::Battle()

{

HANDLE hin = GetStdHandle(STD\_INPUT\_HANDLE);

INPUT\_RECORD InputRecord;

DWORD Events;

COORD coord;

DWORD prev\_mode;

GetConsoleMode(hin, &prev\_mode);

SetConsoleMode(GetStdHandle(STD\_INPUT\_HANDLE), ENABLE\_EXTENDED\_FLAGS | (prev\_mode & ~ENABLE\_QUICK\_EDIT\_MODE));

SetConsoleMode(hin, ENABLE\_MOUSE\_INPUT);

enemy.SetUserField(userField);

enemy.Start();

auto userShoot = [](Enemy& enemy, COORD& coord, vector<COORD>& currentShipCoords, bool& isKill) {

COORD fieldCoord = enemy.GetEnemyField().GetFieldCoord(coord);

int field\_num = enemy.GetEnemyField().GetField(fieldCoord.X, fieldCoord.Y);

if (field\_num == -2 || field\_num == -1) return false;

bool isHit = enemy.GetEnemyField().ShootTo(fieldCoord.X, fieldCoord.Y);

int killedField = 1;

currentShipCoords.clear();

currentShipCoords.push\_back(fieldCoord);

//Kill check

if (field\_num > 0 && field\_num == killedField) {

isKill = true;

return !isHit;

}

//Right

for (SHORT i = fieldCoord.X + 1; i < enemy.GetEnemyField().GetFieldWidth(); i++) {

if (enemy.GetEnemyField().GetField(i, fieldCoord.Y) == -2) {

killedField++;

currentShipCoords.push\_back({ i, fieldCoord.Y });

if (field\_num > 0 && field\_num <= killedField) {

isKill = true;

return !isHit;

}

}

if (enemy.GetEnemyField().GetField(i, fieldCoord.Y) > 0) return !isHit;

if (enemy.GetEnemyField().GetField(i, fieldCoord.Y) == -1 ||

enemy.GetEnemyField().GetField(i, fieldCoord.Y) == -3) break;

}

//Left

for (SHORT i = fieldCoord.X - 1; i >= 0; i--) {

if (enemy.GetEnemyField().GetField(i, fieldCoord.Y) == -2) {

killedField++;

currentShipCoords.push\_back({ i, fieldCoord.Y });

if (field\_num > 0 && field\_num <= killedField) {

isKill = true;

return !isHit;

}

}

if (enemy.GetEnemyField().GetField(i, fieldCoord.Y) > 0) return !isHit;

if (enemy.GetEnemyField().GetField(i, fieldCoord.Y) == -1 ||

enemy.GetEnemyField().GetField(i, fieldCoord.Y) == -3) break;

}

//Up

for (SHORT i = fieldCoord.Y - 1; i >= 0; i--) {

if (enemy.GetEnemyField().GetField(fieldCoord.X, i) == -2) {

killedField++;

currentShipCoords.push\_back({ fieldCoord.X, i });

if (field\_num > 0 && field\_num <= killedField) {

isKill = true;

return !isHit;

}

}

if (enemy.GetEnemyField().GetField(fieldCoord.X, i) > 0) return !isHit;

if (enemy.GetEnemyField().GetField(fieldCoord.X, i) == -1 ||

enemy.GetEnemyField().GetField(fieldCoord.X, i) == -3) break;

}

//Down

for (SHORT i = fieldCoord.Y + 1; i < enemy.GetEnemyField().GetFieldHeight(); i++) {

if (enemy.GetEnemyField().GetField(fieldCoord.X, i) == -2) {

killedField++;

currentShipCoords.push\_back({ fieldCoord.X, i });

if (field\_num > 0 && field\_num == killedField) {

isKill = true;

return !isHit;

}

}

if (enemy.GetEnemyField().GetField(fieldCoord.X, i) > 0) return !isHit;

if (enemy.GetEnemyField().GetField(fieldCoord.X, i) == -1 ||

enemy.GetEnemyField().GetField(fieldCoord.X, i) == -3) break;

}

return !isHit;

};

bool move = true;

vector<COORD> currentShipCoords;

while (!end) {

gotoxy(0, 0);

if (move) {

ReadConsoleInput(hin, &InputRecord, 1, &Events);

bool isKill = false;

if (InputRecord.Event.MouseEvent.dwButtonState == 1)

{

coord.X = InputRecord.Event.MouseEvent.dwMousePosition.X;

coord.Y = InputRecord.Event.MouseEvent.dwMousePosition.Y;

move = !userShoot(enemy, coord, currentShipCoords, isKill);

}

if (isKill) {

for (COORD coord : currentShipCoords) {

if (coord.Y - 1 >= 0 && enemy.GetEnemyField().GetField(coord.X, coord.Y - 1) == -3)

enemy.GetEnemyField().SetField(coord.X, coord.Y - 1, -1); //Up

if (coord.Y + 1 < enemy.GetEnemyField().GetFieldHeight() && enemy.GetEnemyField().GetField(coord.X, coord.Y + 1) == -3)

enemy.GetEnemyField().SetField(coord.X, coord.Y + 1, -1); //Down

if (coord.X - 1 >= 0 && enemy.GetEnemyField().GetField(coord.X - 1, coord.Y) == -3)

enemy.GetEnemyField().SetField(coord.X - 1, coord.Y, -1); //Left

if (coord.X + 1 < enemy.GetEnemyField().GetFieldWidth() && enemy.GetEnemyField().GetField(coord.X + 1, coord.Y) == -3)

enemy.GetEnemyField().SetField(coord.X + 1, coord.Y, -1); //Right

if (coord.X - 1 >= 0 && coord.Y - 1 >= 0 && enemy.GetEnemyField().GetField(coord.X - 1, coord.Y - 1) == -3)

enemy.GetEnemyField().SetField(coord.X - 1, coord.Y - 1, -1); //Left up

if (coord.X + 1 < enemy.GetEnemyField().GetFieldWidth() && coord.X + 1 < enemy.GetEnemyField().GetFieldHeight() && enemy.GetEnemyField().GetField(coord.X + 1, coord.Y + 1) == -3)

enemy.GetEnemyField().SetField(coord.X + 1, coord.Y + 1, -1); //Right down

if (coord.X - 1 >= 0 && coord.Y + 1 < enemy.GetEnemyField().GetFieldHeight() && enemy.GetEnemyField().GetField(coord.X - 1, coord.Y + 1) == -3)

enemy.GetEnemyField().SetField(coord.X - 1, coord.Y + 1, -1); //Left donw

if (coord.X + 1 < enemy.GetEnemyField().GetFieldWidth() && coord.Y - 1 >= 0 && enemy.GetEnemyField().GetField(coord.X + 1, coord.Y - 1) == -3)

enemy.GetEnemyField().SetField(coord.X + 1, coord.Y - 1, -1); //Right up

}

enemy.GetEnemyField().SetShipsCount(enemy.GetEnemyField().GetShipsCount() - 1);

if (enemy.GetEnemyField().GetShipsCount() <= 0) {

end = true;

win = true;

continue;

}

enemy.GetEnemyField().ClearColors(true);

continue;

}

if (move) continue;

}

else {

move = enemy.Shoot();

if (enemy.IsWin()) {

end = true;

win = false;

continue;

}

}

Sleep(1000);

}

End();

}

void GameController::End()

{

userField.ClearField();

enemy.GetEnemyField().ClearField();

clear();

color.SetColor((win) ? color.Green : color.Red);

cout << "\n\t\t\tGAME OVER!\n";

color.SetColor(color.DarkGray);

cout << "\t\t\tYou " << ((win) ? "win" : "lose") << "\n";

cout << "\t\t\tPress 'r' to restart";

int press = 0;

press = \_getch();

clear();

if (press == 114) Start();

}

1. За для генерації поля, та управлінням усього, що стосується взаємодії з ігровим полем, відповідає однойменний клас “Field”.

void Field::Set(SHORT& y, SHORT& x, int& num)

{

field[y][x] = num;

if(y - 1 >= 0 && (field[y - 1][x] == 0 || field[y - 1][x] < -3))

field[y - 1][x] = -3; //Up

if(y + 1 < field\_height && (field[y + 1][x] == 0 || field[y + 1][x] < -3))

field[y + 1][x] = -3; //Down

if (x - 1 >= 0 && (field[y][x - 1] == 0 || field[y][x - 1] < -3))

field[y][x - 1] = -3; //Left

if (x + 1 < field\_width && (field[y][x + 1] <= 0 || field[y][x + 1] < -3))

field[y][x + 1] = -3; //Right

if (x - 1 >= 0 && y - 1 >= 0 && (field[y - 1][x - 1] == 0 || field[y - 1][x - 1] < -3))

field[y - 1][x - 1] = -3; //Left up

if (x + 1 < field\_width && y + 1 < field\_height && (field[y + 1][x + 1] == 0 || field[y + 1][x + 1] < -3))

field[y + 1][x + 1] = -3; //Right down

if (x - 1 >= 0 && y + 1 < field\_height && (field[y + 1][x - 1] == 0 || field[y + 1][x - 1] < -3))

field[y + 1][x - 1] = -3; //Left donw

if (x + 1 < field\_width && y - 1 >=0 && (field[y - 1][x + 1] <= 0 || field[y - 1][x + 1] < -3))

field[y - 1][x + 1] = -3; //Right up

}

Field::Field()

{

ClearField();

}

void Field::ClearField()

{

shipsCount = 0;

for (int i = 0; i < field\_height; i++) {

for (int j = 0; j < field\_width; j++) {

field[i][j] = 0;

}

}

}

void Field::ClearColors(bool fog)

{

int mainWaterColor = (!fog) ? color.LightBlue : color.DarkGray;

int shipsColor = (!fog) ? color.Yellow : color.DarkGray;

color.SetColor(mainWaterColor);

for (map<COORD, COORD>::const\_iterator i = coords.begin(); i != coords.end(); i++) {

gotoxy(i->first.X, i->first.Y);

if (!field[i->second.Y][i->second.X] || field[i->second.Y][i->second.X] == -3) cout << ((!fog) ? "~ " : "\* ");

if (field[i->second.Y][i->second.X] > 0) {

color.SetColor(shipsColor);

cout << ((!fog) ? "^ " : "\* ");

color.SetColor(mainWaterColor);

}

if (field[i->second.Y][i->second.X] == -1) {

color.SetColor(color.LightGray);

cout << "o ";

color.SetColor(mainWaterColor);

}

if (field[i->second.Y][i->second.X] == -2) {

color.SetColor(color.Red);

cout << "X ";

color.SetColor(mainWaterColor);

}

}

}

void Field::Generate()

{

ClearField();

int shipSize = 1;

vector<COORD> freeCoords;

auto clearBusy = [](vector<COORD>& freeCoords, int (&field)[10][10]) {

vector<COORD> temp;

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

if (field[freeCoords[i].Y][freeCoords[i].X] > 0 || field[freeCoords[i].Y][freeCoords[i].X] == -3) {

continue;

}

temp.push\_back(freeCoords[i]);

}

freeCoords.clear();

freeCoords.swap(temp);

};

for (SHORT i = 0; i < field\_height; i++) {

for (SHORT j = 0; j < field\_width; j++) {

freeCoords.push\_back({ j, i });

}

}

COORD currPos;

COORD endPos;

while (shipsCount < 10) {

for (SHORT i = 0; i < 5 - shipSize;) {

int it = rand() % freeCoords.size();

currPos = freeCoords[it];

endPos = currPos;

endPos.X = currPos.X + (shipSize - 1);

if (CanSetShip(currPos, endPos)) { //Right

SetShip(currPos, endPos, shipSize);

i++;

clearBusy(freeCoords, field);

continue;

}

endPos = currPos;

endPos.X = currPos.X - (shipSize - 1);

if (CanSetShip(currPos, endPos)) { //Left

SetShip(currPos, endPos, shipSize);

i++;

clearBusy(freeCoords, field);

continue;

}

endPos = currPos;

endPos.Y = currPos.Y + (shipSize - 1);

if (CanSetShip(currPos, endPos)) { //Down

SetShip(currPos, endPos, shipSize);

i++;

clearBusy(freeCoords, field);

continue;

}

endPos = currPos;

endPos.Y = currPos.Y - (shipSize - 1);

if (CanSetShip(currPos, endPos)) { //Up

SetShip(currPos, endPos, shipSize);

i++;

clearBusy(freeCoords, field);

continue;

}

}

shipSize++;

}

}

void Field::SetField(int x, int y, int num)

{

if (x >= 0 && x < field\_width && y >= 0 && y < field\_height) {

field[y][x] = num;

}

}

int Field::GetField(int x, int y)

{

if (x >= 0 && x < field\_width && y >= 0 && y < field\_height) return field[y][x];

}

int Field::GetField(COORD mouseCoords)

{

if (coords.size()) {

COORD fieldCoord = coords[mouseCoords];

return field[fieldCoord.Y][fieldCoord.X];

}

return 0;

}

bool Field::ShootTo(int x, int y)

{

if (x >= 0 && x < field\_width && y >= 0 && y < field\_height && field[y][x] != -1 && field[y][x] != -2 && coords.size()) {

field[y][x] = ((field[y][x] > 0) ? -2 : -1);

COORD coord = { x, y };

coord = GetCoordBy(coord);

gotoxy(coord.X, coord.Y);

if (field[y][x] == -1 || field[y][x] == -3) {

color.SetColor(color.LightGray);

cout << "o ";

color.ResetColor();

return false;

}

if (field[y][x] == -2) {

color.SetColor(color.Red);

cout << "X ";

color.ResetColor();

return true;

}

return false;

}

return false;

}

map<COORD, COORD> Field::GetCoords()

{

return coords;

}

COORD Field::GetCoordBy(COORD fieldCoord)

{

for (map<COORD, COORD>::const\_iterator i = coords.begin(); i != coords.end(); i++) {

if (i->second.X == fieldCoord.X && i->second.Y == fieldCoord.Y) {

return i->first;

}

}

}

COORD Field::GetFieldCoord(COORD mouseCoord)

{

return coords[mouseCoord];

}

bool Field::ContainsMouseCoord(COORD mouseCoord)

{

for (map<COORD, COORD>::const\_iterator it = coords.begin(); it != coords.end(); it++) {

if (it->first.X == mouseCoord.X && it->first.Y == mouseCoord.Y) return true;

}

return false;

}

int Field::GetFieldWidth()

{

return field\_width;

}

int Field::GetFieldHeight()

{

return field\_height;

}

void Field::SetShipsCount(int shipsCount)

{

if(shipsCount <= 10 && shipsCount >= 0) this->shipsCount = shipsCount;

}

int Field::GetShipsCount()

{

return shipsCount;

}

void Field::Draw(COORD StartPos, COORD EndPos, int num)

{

bool canSet = CanSetShip(StartPos, EndPos);

color.SetColor(((canSet) ? color.Green : color.Red));

COORD coord;

if (StartPos.X > EndPos.X) { //Left

for (int i = StartPos.X; i >= EndPos.X; i--) {

if (i < 0) break;

coord = GetCoordBy({ (SHORT)i, StartPos.Y });

gotoxy(coord.X, coord.Y);

cout << num;

}

}

if (StartPos.X < EndPos.X) { //Right

for (int i = StartPos.X; i <= EndPos.X; i++) {

if (i >= field\_width) break;

coord = GetCoordBy({ (SHORT)i, StartPos.Y });

gotoxy(coord.X, coord.Y);

cout << num;

}

}

if (StartPos.Y < EndPos.Y) { //Down

for (int i = StartPos.Y; i <= EndPos.Y; i++) {

if (i >= field\_height) break;

coord = GetCoordBy({ StartPos.X, (SHORT)i });

gotoxy(coord.X, coord.Y);

cout << num;

}

}

if (StartPos.Y > EndPos.Y) { //Up

for (int i = StartPos.Y; i >= EndPos.Y; i--) {

if (i < field\_height) break;

coord = GetCoordBy({ StartPos.X, (SHORT)i });

gotoxy(coord.X, coord.Y);

cout << num;

}

}

if (StartPos.X && StartPos.Y && StartPos.X == EndPos.X && StartPos.Y == EndPos.Y) {

coord = GetCoordBy({ StartPos.X, StartPos.Y });

gotoxy(coord.X, coord.Y);

cout << num;

}

color.ResetColor();

}

void Field::SetShip(COORD StartPos, COORD EndPos, int ship\_size)

{

if (CanSetShip(StartPos, EndPos)) {

if (StartPos.X > EndPos.X) { //Left

for (SHORT i = StartPos.X; i >= EndPos.X; i--) {

Set(StartPos.Y, i, ship\_size);

}

}

if (StartPos.X < EndPos.X) { //Right

for (SHORT i = StartPos.X; i <= EndPos.X; i++) {

Set(StartPos.Y, i, ship\_size);

}

}

if (StartPos.Y < EndPos.Y) { //Down

for (SHORT i = StartPos.Y; i <= EndPos.Y; i++) {

Set(i, StartPos.X, ship\_size);

}

}

if (StartPos.Y > EndPos.Y) { //Up

for (SHORT i = StartPos.Y; i >= EndPos.Y; i--) {

Set(i, StartPos.X, ship\_size);

}

}

if (StartPos.X && StartPos.Y && StartPos.X == EndPos.X && StartPos.Y == EndPos.Y) {

Set(StartPos.Y, StartPos.X, ship\_size);

}

shipsCount++;

}

}

bool Field::CanSetShip(COORD StartPos, COORD EndPos)

{

if (StartPos.X > EndPos.X) { //Left

if (EndPos.X < 0) return false;

for (int i = StartPos.X; i >= EndPos.X; i--) {

if (field[StartPos.Y][i] > 0 || field[StartPos.Y][i] == -3) return false;

}

return true;

}

if (StartPos.X < EndPos.X) { //Right

if (EndPos.X >= field\_width) return false;

for (int i = StartPos.X; i <= EndPos.X; i++) {

if (field[StartPos.Y][i] > 0 || field[StartPos.Y][i] == -3) return false;

}

return true;

}

if (StartPos.Y < EndPos.Y) { //Down

if (EndPos.Y >= field\_width) return false;

for (int i = StartPos.Y; i <= EndPos.Y; i++) {

if (field[i][StartPos.X] > 0 || field[i][StartPos.X] == -3) return false;

}

return true;

}

if (StartPos.Y > EndPos.Y) { //Up

if (EndPos.Y < 0) return false;

for (int i = StartPos.Y; i >= EndPos.Y; i--) {

if (field[i][StartPos.X] > 0 || field[i][StartPos.X] == -3) return false;

}

return true;

}

if (StartPos.X && StartPos.Y && StartPos.X == EndPos.X && StartPos.Y == EndPos.Y && field[StartPos.Y][StartPos.X] != -3)

return true;

return false;

}

void Field::ShowField(bool fog, int startX, int startY)

{

if (startX && startY) gotoxy(startX, startY);

else cout << "\n\n\t\t";

int mainWaterColor = (!fog) ? color.LightBlue : color.DarkGray;

int shipsColor = (!fog) ? color.Yellow : color.DarkGray;

color.SetColor(mainWaterColor);

for (int i = 0; i < field\_height; i++) {

for (int j = 0; j < field\_width; j++) {

COORD coord = { j, i };

coords[getxy()] = coord;

if(!field[i][j] || field[i][j] == -3)

cout << ((!fog) ? "~ " : "\* ");

if (field[i][j] > 0) {

color.SetColor(shipsColor);

cout << ((!fog) ? "^ " : "\* ");

color.SetColor(mainWaterColor);

}

if (field[i][j] == -1) {

color.SetColor(color.LightGray);

cout << "o ";

color.SetColor(mainWaterColor);

}

if (field[i][j] == -2) {

color.SetColor(color.Red);

cout << "X ";

color.SetColor(mainWaterColor);

}

}

cout << endl;

if (startX && startY) gotoxy(startX, getxy().Y);

}

color.ResetColor();

}

void Field::operator=(Field& field)

{

for (int i = 0; i < field\_height; i++) {

for (int j = 0; j < field\_width; j++) {

SetField(j, i, field.GetField(j, i));

}

}

coords = field.GetCoords();

shipsCount = field.GetShipsCount();

}

bool operator<(const COORD& left, const COORD& right)

{

if (left.Y < right.Y)

{

return true;

}

else if (right.Y < left.Y)

{

return false;

}

else if (left.X < right.X)

{

return true;

}

else /\* (right.x < left.x) \*/

{

return false;

}

}

1. За для ворога, його функціонування, та поведінки, було створено клас “Enemy”, код якого представлено нижче:

void Enemy::UnshootedRemoveAt(int id)

{

vector<COORD> temp;

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

if (i != id) temp.push\_back(unshooted[i]);

}

unshooted.swap(temp);

}

void Enemy::UnshootedRemoveAt(COORD coord)

{

vector<COORD> temp;

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

if (unshooted[i].X != coord.X || unshooted[i].Y != coord.Y) temp.push\_back(unshooted[i]);

}

unshooted.swap(temp);

}

void Enemy::UnshootedRemoveAt(int x, int y)

{

vector<COORD> temp;

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

if (unshooted[i].X != x || unshooted[i].Y != y) temp.push\_back(unshooted[i]);

}

unshooted.swap(temp);

}

void Enemy::ShootAI()

{

if (aiShoot[0].X + 1 < userField.GetFieldWidth()) {

for (SHORT i = aiShoot[0].X + 1; i < userField.GetFieldWidth(); i++) {

aiShoot.push\_back({ i, aiShoot[0].Y });

if (userField.GetField(i, aiShoot[0].Y) > 0) {

currentShipSize--;

if (!currentShipSize) return;

continue;

}

break;

}

}

if (aiShoot[0].X - 1 >= 0) {

for (SHORT i = aiShoot[0].X - 1; i >= 0; i--) {

aiShoot.push\_back({ i, aiShoot[0].Y });

if (userField.GetField(i, aiShoot[0].Y) > 0) {

currentShipSize--;

if (!currentShipSize) return;

continue;

}

break;

}

}

if (aiShoot[0].Y + 1 < userField.GetFieldHeight()) {

for (SHORT i = aiShoot[0].Y + 1; i < userField.GetFieldHeight(); i++) {

aiShoot.push\_back({ aiShoot[0].X, i });

if (userField.GetField(aiShoot[0].X, i) > 0) {

currentShipSize--;

if (!currentShipSize) return;

continue;

}

break;

}

}

if (aiShoot[0].Y - 1 >= 0) {

for (SHORT i = aiShoot[0].Y - 1; i >= 0; i--) {

aiShoot.push\_back({ aiShoot[0].X, i });

if (userField.GetField(aiShoot[0].X, i) > 0) {

currentShipSize--;

if (!currentShipSize) return;

continue;

}

break;

}

}

}

void Enemy::CyrcleAI()

{

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

if (userField.GetField(aiShoot[i].X, aiShoot[i].Y) == -2) Cyrcle(aiShoot[i].X, aiShoot[i].Y);

}

aiShoot.clear();

aiIterator = 0;

}

void Enemy::Cyrcle(int x, int y)

{

if (y - 1 >= 0 && userField.GetField(x, y - 1) <= 0 && userField.GetField(x, y - 1) != -2)

UnshootedRemoveAt(x, y - 1); //Up

if (y + 1 < userField.GetFieldHeight() && userField.GetField(x, y + 1) <= 0 && userField.GetField(x, y + 1) != -2)

UnshootedRemoveAt(x, y + 1); //Down

if (x - 1 >= 0 && userField.GetField(x - 1, y) <= 0 && userField.GetField(x - 1, y) != -2)

UnshootedRemoveAt(x - 1, y); //Left

if (x + 1 < userField.GetFieldWidth() && userField.GetField(x + 1, y) <= 0 && userField.GetField(x + 1, y) != -2)

UnshootedRemoveAt(x+1, y); //Right

if (x - 1 >= 0 && y - 1 >= 0 && userField.GetField(x - 1, y - 1) <= 0 && userField.GetField(x - 1, y - 1) != -2)

UnshootedRemoveAt(x - 1, y - 1);

if (x + 1 < userField.GetFieldWidth() && y + 1 < userField.GetFieldHeight() && userField.GetField(x + 1, y + 1) <= 0 && userField.GetField(x + 1, y + 1) != -2)

UnshootedRemoveAt(x + 1, y + 1); //Right down

if (x - 1 >= 0 && y + 1 < userField.GetFieldHeight() && userField.GetField(x + 1, y - 1) <= 0 && userField.GetField(x + 1, y - 1) != -2)

UnshootedRemoveAt(x + 1, y - 1); //Left donw

if (x + 1 < userField.GetFieldWidth() && y - 1 >= 0 && userField.GetField(x - 1, y + 1) <= 0 && userField.GetField(x - 1, y + 1) != -2)

UnshootedRemoveAt(x - 1, y + 1); //Right up

}

Enemy::Enemy()

{

win = false;

}

Enemy::Enemy(Field& userField)

{

win = false;

SetUserField(userField);

}

void Enemy::SetUserField(Field& userField)

{

this->userField = userField;

}

Field& Enemy::GetEnemyField()

{

return field;

}

bool Enemy::IsWin()

{

return win;

}

bool Enemy::Shoot()

{

if (!aiShoot.size()) {

int field\_id = rand() % unshooted.size();

int fieldNum = userField.GetField(unshooted[field\_id].X, unshooted[field\_id].Y);

bool isShooted = userField.ShootTo(unshooted[field\_id].X, unshooted[field\_id].Y);

if (isShooted && fieldNum - 1 > 0) {

aiShoot.clear();

aiShoot.push\_back(unshooted[field\_id]);

currentShipSize = fieldNum - 1;

aiIterator = 1;

ShootAI();

}

if (fieldNum == 1) {

userField.SetShipsCount(userField.GetShipsCount() - 1);

win = !userField.GetShipsCount();

Cyrcle(unshooted[field\_id].X, unshooted[field\_id].Y);

}

UnshootedRemoveAt(field\_id);

return !isShooted;

}

else {

bool isShooted = userField.ShootTo(aiShoot[aiIterator].X, aiShoot[aiIterator].Y);

UnshootedRemoveAt(aiShoot[aiIterator]);

aiIterator++;

if (aiIterator >= aiShoot.size()) {

userField.SetShipsCount(userField.GetShipsCount() - 1);

win = !userField.GetShipsCount();

CyrcleAI();

}

return !isShooted;

}

return true;

}

void Enemy::Start()

{

field.Generate();

field.ShowField(true, 50, 10);

win = false;

unshooted.clear();

for (SHORT i = 0; i < field.GetFieldHeight(); i++) {

for (SHORT j = 0; j < field.GetFieldWidth(); j++) {

unshooted.push\_back({ j, i });

}

}

}

1. Для внесення барв у програму, за для зручної маніпуляції із кольорами було додано клас “Color”:

Color::Color(){

hStdOut = GetStdHandle(STD\_OUTPUT\_HANDLE);

}

void Color::SetColor(int text, int background) {

SetConsoleTextAttribute(hStdOut, (WORD)((background << 4) | text));

}

void Color::ResetColor() {

SetConsoleTextAttribute(hStdOut, (WORD)((ConsoleColor::Black << 4) | ConsoleColor::White));

}

РОЗДІЛ 3. ОРГАНІЗАЦІЯ ІНФОРМАЦІЙНОГО ЗАБЕЗПЕЧЕННЯ

## Загальна характеристика інформаційного забезпечення

Всі вхідні дані вводяться у класі «GameController» у методі «Start». Згодом, діючий метод класу “GameController” змінюється в залежності від стану подій у грі. Поле, та все, що його стосуєтсья, генерується та оброблюється у класі “Field”. За штучний інтелект, котрий є ворогом гравця, відповідає клас “Enemy”.

3.2 Структура класів

Поля класу подані детальніше у таблицях 3.1:

Табл.3.1

Опис структури класу <GameController>

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Найменування | Поле | Тип даних поля | Довжина у байтах | Кількість знаків після десятинної точки |
| Поле | userField | Field | 428 | - |
| Ворог | enemy | Enemy | 900 | - |
| Кінець гри | end | Bool | 1 | - |
| перемога | win | bool | 1 | - |
| Контролер кольорів | color | Color | 4 | - |

Табл.3.2

Опис структури класу <Field>

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Найменування | Поле | Тип даних поля | Довжина у байтах | Кількість знаків після десятинної точки |
| Поле | field | int[10][10] | 400 | - |
| Довжина поля | field\_width | const int | 4 | - |
| Ширина поля | field\_height | const int | 4 | - |
| Координати миші та комірки поля | coords | map<COORD, COORD> | 12 | - |
| Контролер кольорів | color | Color | 4 | - |
| Кількість кораблів на полі | shipsCount | int | 4 | - |

Табл.3.3

Опис структури класу <Enemy>

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Найменування | Поле | Тип даних поля | Довжина у байтах | Кількість знаків після десятинної точки |
| Поле ворога | field | Field | 428 | - |
| Поле гравця | userField | Field | 428 | - |
| Доступні поля для пострілу | unshooted | vector<COORD> | 16 | - |
| Поля для пострілу розраховані штучним інтелектом | aiShoot | vector<COORD> | 16 | - |
| Ітератор пострілу штучним інтелектом | aiIterator | int | 4 | - |
| Розмір корабля, по котрому відпрацьовує ворог | currentShipSize | int | 4 | - |
| Перемога ворога | win | bool | 1 | - |

# РОЗДІЛ 4. РОЗРОБКА ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ ЗАДАЧІ.

Дана програма є консольний додаток, що створений засобами мови програмування Visual C++. Всі вхідні та вихідні дані не зберігаються.

Тепер розглянемо принципи роботи гри. Після запуску нашої гри створюється головне вікно, а точніше кажучи вся гра починається з нього. Першим чином, у головній функції програми “main” створюється об’єкт класу “GameController”, від якого викликається метод “Start”.

На початку роботи методу, від поля userField класу “Field”, котре знаходиться у класі “GameControlelr”, викликається метод “ShowField”, котрий відображає поле на екрані. Згодом, поле гравця перемальовуватися повністю не буде. Буде лише оновлюватися потрібна частина поля за допомогою методу “Draw” класу “Field”, за для відображення нової інформації.

Тут же, у методі “Start” відбуватиметься опрацювання на натискання клавіш. Процес позиціювання кораблів на полі відбуватиметься доки користувач не розташує на полі потрібну кількість кораблів, та не натисне клавішу “Space”. При натисканні на “Space” відбудеться вихід із циклу “while”, та відбудеться переадресація користувача до методу “Battle”, де відбуватиметься процес поєдинку гравця із ворогом, котрий представляє собою поле “enemy” класу “Enemy” у класі “GameController”.

У процесі поєдинку, за для введення інформації про постріл за допомогою кліку лівої кнопки миші, використовується дескриптор введення, котрий допомагає нам зчитувати буфер введення у консолі, та повертати координати кліку.

Провести постріл по координатам поля, куди було натиснуто мишею, допомагає створений лямбда-вираз “userShoot”, котрий приймає посилання на координати пострілу, та поля, проводе постріл, та повертає “true” або “false”, якщо не було влучання, та змінну “isKill”, котра вказує чи вбив гравець корабель ворога.

Коли доходе черга ворога зробити постріл, виконується метод “Shoot”, класу “Enemy”.

Спочатку, якщо буфер пострілів “aiShoot” є порожнім, ворог стріляє по випадковим клітинкам, котрі доступні для пострілу. Якщо було завіксовано влучання, метод “ShootAI” генерує послідовність логічних пострілів, котрий у кінцевому результати призведе до потоплення корабля гравця.

У кінцевому результаті, якщо корабель гравця був вбитий, клітинки навколо нього стають недоступними для влучання, чому сприяє метод “CyrcleAI”.

Згодом, коли бій завершено із будь-яким результатом, викликається метод “End” класу “GameController”, та гравця інформує про стан речей.

Як ми бачимо, що всі коди стосовно нашої гри оформлено правильно та зрозуміло, що не може не задовольняти. Гра створена логічно, лаконічно, а головне зрозуміло, як і для користувача так і для розробника.

# ВИСНОВКИ

У результаті роботи над курсовим проектом, було створено програмне забезпечення гри у морський бій.

На початку роботи було сформульовано основні задачі та цілі роботи. Був намічений план вирішення поставленої задачі. Були визначені структура та зміст вхідної та вихідної інформації.

В процесі розробки було обрано методи вирішення задачі: методи теорії алгоритмів, методи об’єктно-орієнтованого програмування.

Для реалізації задачі за допомогою ЕОМ були створені класи для повноцінного функціонування гри. Для реалізації задачі було обрано мову програмування Microsoft С++.

Основною метою було створити працездатний додаток. Серед переваг програми слід відмітити можливість вводу інформації за допомогою клавіатури та миші, та ворога із штучним інтелектом, котрий працює на достатньому рівні.

В цілому, мета роботи була виконана. Було створено програму, що повноцінно функціонує. В майбутньому проєкт можна розширити, додаючи більше нових можливостей, що дозволить створити якісний продукт

# СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Зеленський О.С., Лисенко В.С. Методичні вказівки для самостійного вивчення основ класичного програмування на мові С++, з дисциплін "Інструментальні засоби прикладного програмування","Об’єктно-орієнтоване програмування", "Прикладне програмне забезпечення", Кривий Ріг, КЕІ КНЕУ, 2008. – 94с.
2. Грегори К. Использование Visual C++ 6. Учебное пособие – СПб: К.: Издательский дом "Вильямс", 1999. – 864 с.
3. Эккель Брюс. Философия C++
4. Офіційниційний довідник [Microsoft MSDN](https://visualstudio.microsoft.com/ru/msdn-platforms/)
5. Джесс Либерти, Дэвид Б. Хорват Освой самостоятельно C++ за 24 часа. – М: Вильямс, 2007г. – 448 с.
6. Бьярне Стауструп. Программирование: принципы и практика использования C++
7. Программирование на C++ в примерах и задачах. Васильев А.Н.