Опис програми, алгоритмів та стратегій гри “Морський бій”

Виконав Єгор Демченко

---

1. Загальний опис програми

Дана програма реалізує адаптовану версію класичної гри "Морський бій"на мові програмування C++. Для створення графічного інтерфейсу використовується WinAPI, а бібліотека GDI+ забезпечує покращену візуалізацію елементів гри. Основна особливість цієї реалізації — кожен гравець (включно з ботом) має лише один однопалубний корабель, який може переміщуватися після кожного пострілу.

Гра підтримує такі режими:

- Режим для одного гравця — змагання з ботом на рівнях складності "Легкий"та "Складний".
- Режим для двох гравців — дозволяє двом користувачам змагатися один з одним.

Основна мета гри — знайти та знищити однопалубний корабель супротивника, одночасно захищаючи власний.

2. Структура програми

2.1. Підключені бібліотеки

1. `<Windows.h>` і `<windowsx.h>`:
   - Для створення віконного інтерфейсу програми за допомогою WinAPI.
   - Забезпечують обробку подій, пов’язаних із взаємодією користувача з вікном (наприклад, кліки миші, натискання кнопок).
2. `<vector>`:
   - Використовується для створення ігрових полів у вигляді двовимірних векторів (`std::vector<std::vector<char»`).
3. `<cstdlib>` і `<ctime>`:
   - `cstdlib` — для роботи з випадковими числами, використовується при визначенні координат корабля бота.
   - `ctime` — для ініціалізації генератора випадкових чисел.
4. `<iostream>`:
   - Використовується для виведення діагностичної інформації в консоль під час розробки (наприклад, для налагодження логіки).
5. `<set>` і `<queue>`:
   - `std::set` — для зберігання можливих позицій для переміщення чи стрільби бота.

- `std::queue` — використовується для обробки послідовних подій.

6. `<gdiplus.h>`:
   - Бібліотека GDI+ для рендерингу графіки, наприклад, завантаження фонового зображення чи візуалізації ігрового процесу.
7. `<climits>`:
   - Для роботи з граничними значеннями чисел, наприклад, `INT_MAX` у складному режимі бота.

2.2. Опис функцій програми

2.2.1. Ініціалізація гри

- `void initializeBoard(std::vector<std::vector<char>>& board)`
  Ініціалізує ігрове поле перед початком гри. Заповнює двовимірний вектор `board` символами `EMPTY ('-')`, що позначає порожні клітинки.
- `void startNewGame()`
  Виконує повне скидання стану гри для нового початку:
  - Очищує поля гравця і бота.
  - Розміщує корабель бота на випадковій позиції.
  - Скидає всі ігрові змінні (наприклад, `playerShipPlaced`, `isPlayerHit`).
- `void initializeTwoPlayersGame()`
  Підготовка до гри в режимі двох гравців:
  - Очищує обидва поля.
  - Встановлює черговість ходів, дозволяючи кожному гравцеві розмістити свій корабель.

2.2.2. Графічна візуалізація

- ```
  void drawGrid(HDC hdc,
  const std::vector<std::vector<char>& board,
  int startX, int startY, bool hideShips)
  ```
  - Малює сітку поля на основі двовимірного вектора `board`.
  - Відображає клітинки відповідно до їх стану (порожня, корабель, попадання, промах).
  - Для режиму з ботом може приховувати кораблі, якщо встановлено `hideShips = true`.
  ```

2.2.3. Взаємодія з користувачем

- `void handlePlayerPlacement(int x, int y)`
  - Обробляє вибір клітинки для розміщення корабля гравцем.
  - Перевіряє, чи координати `x, y` відповідають ігровому полю.
  - Якщо клітинка порожня, розташовує корабель у вибраній точці.
- `void handlePlayerMove(int x, int y)`
  - Дозволяє гравцеві перемістити свій корабель на сусідню клітинку.
  - Перевіряє, чи переміщення можливе (сусідня порожня клітинка).
  - Змінює положення корабля у векторі поля.
- `void handlePlayerShot(HWND hwnd, int x, int y)`
  - Реалізує постріл гравця.
  - Визначає, чи є попадання, промах або повторна спроба по вже обстріляній клітинці.
  - Змінює стан клітинки на `HIT` або `MISS`.

2.2.4. Логіка бота

- `void placeBotShip(std::vector<std::vector<char>& board)`
  - Розміщує корабель бота у випадковій порожній клітинці.
- `void moveBotShip()`
  - Переміщує корабель бота на сусідню клітинку.
  - У легкому режимі вибирає випадкову клітинку.
  - У складному режимі аналізує ризики (обстріляні клітинки) та обирає безпечну позицію.
- `void smartBotShot(HWND hwnd)`
  - Реалізує стрільбу бота в складному режимі.
  - Використовує Манхеттенську відстань для визначення найбільш ймовірної позиції корабля гравця.
  - Зберігає можливі позиції для наступних пострілів.
- `void botShot(HWND hwnd)`
  - Виконує стрільбу бота.
  - У легкому режимі обирає випадкову клітинку.
  - У складному режимі викликає `smartBotShot()`.

2.2.5. Обслуговування гри

- `int calculateManhattanDistance(int x1, int y1, int x2, int y2)`
  - Розраховує Манхеттенську відстань між двома точками `(x1, y1)` та `(x2, y2)`.
  - Використовується ботом для оцінки ймовірності попадання.
- `void showDistance(int distance, int startX, int startY)`
  - Виводить текстову інформацію про відстань до корабля противника.
- `void updatePossiblePositions(int distance, int shotX, int shotY)`
  - Оновлює список можливих позицій корабля гравця після пострілу бота.
  - Враховує Манхеттенську відстань для уточнення позицій.
- `void StartEndGameTimer(HWND hwnd, bool playerWon, bool isTwoPlayersMode)`
  - Запускає таймер для завершення гри.
  - Виводить відповідне повідомлення про переможця.

2.2.6. Управління інтерфейсом

- `void createStartButton(HWND hwnd)`
  - Створює кнопки для вибору режиму гри, початку нової гри або виходу з програми.
- `void hideButtons()`
  - Видаляє всі кнопки з екрана після початку гри.
- `LRESULT CALLBACK WindowProc(HWND hwnd, UINT uMsg, WPARAM wParam, LPARAM lParam)`
  - Головна функція обробки подій.
  - Реагує на кліки миші, натискання кнопок та завершення гри.
  - Відповідає за відображення вікна і перерисовку графіки.

2.2.7. Точка входу

- `int WINAPI WinMain(HINSTANCE hInstance, HINSTANCE hPrevInstance, LPSTR lpCmdLine, int nCmdShow)`
  - Головна функція програми.
  - Ініціалізує середовище `GDI+`, запускає вікно та основний цикл повідомлень Windows.
  - Викликає функції для запуску нової гри, відображення елементів інтерфейсу та завершення програми.

3. Механіка гри та алгоритми

3.1. Розміщення кораблів

- Гравці розміщують свій корабель вручну, натискаючи на клітинки поля.
- Бот розташовує свій корабель випадковим чином на порожній клітинці.

3.2. Стрільба

- Гравець:
  - Натискає на клітинку поля супротивника, щоб здійснити постріл.
  - Попадання: Клітинка позначається як `HIT`.
  - Промах: Клітинка позначається як `MISS`.
- Бот:
  - Легкий рівень: Бот стріляє у випадкову клітинку, яка ще не була обстріляна.
  - Складний рівень: Бот аналізує ймовірні позиції корабля гравця за допомогою Манхеттенської відстані та стріляє в оптимальну клітинку.

3.3. Переміщення кораблів

Після кожного невлучного пострілу кораблі можуть переміститися:

- Гравець: Вибирає сусідню клітинку або залишається на місці.
- Бот:
  - Легкий рівень: Випадковий вибір сусідньої клітинки.
  - Складний рівень: Аналіз ризиків обстрілу з боку гравця. Бот вибирає клітинку з мінімальним ризиком.

3.4. Завершення гри

Гра завершується, коли корабель одного з гравців знищено. Програма виводить повідомлення про переможця та пропонує розпочати нову гру.

4. Стратегії гри

4.1. Стратегії для гравців

1. **Розташування корабля:**
   - Вибирайте позиції ближче до центру поля для більшої свободи руху.
   - Уникайте розміщення поблизу країв, щоб зменшити ймовірність блокування руху.

2. **Атака:**

   - Починайте стрільбу з центральних або кутових зон, щоб швидше звузити область пошуку.

3. **Переміщення:**

   - Уникайте передбачуваних маршрутів переміщення.
   - Переміщуйте корабель у зони, де супротивнику складніше спрогнозувати наступний постріл.

5. Особливості гри

1. **Ігрова динаміка:**
   Кораблі можуть переміщуватися після кожного пострілу, що додає варіативності та ускладнює тактичну гру.

2. **Манхеттенська відстань:**
   У складному режимі бот використовує цей показник для аналізу та прогнозування позицій корабля гравця.

3. **Симетрія можливостей:**
   І гравець, і бот мають однакові можливості для руху та стрільби, забезпечуючи баланс.

6. Висновок

Реалізація гри "Морський бій"з однопалубними рухомими кораблями є цікавим експериментом, який надає динамічності та унікальності класичній грі. Завдяки простій графіці та адаптованій механіці, програма є зручною для гравців будь-якого рівня підготовки.