Національний університет «Києво-Могилянська академія»

ЗВІТ  
ДО ПРАКТИЧНОГО ЗАВДАННЯ  
З ДИСЦИПЛІНИ  
«НАВЧАЛЬНА ПРАКТИКА»

Роботу виконав:  
Леськів Олександр Андрійович,  
учень І курсу НаУКМА.  
Факультет Інформатики.  
Спеціальність: інженерія програмного забезпечення

Викладач:  
Тригуб Олександр Семенович

*Зміст*

[Постановка задачі 3](#_Toc485031083)

[Обраний алгоритм та структури даних, необхідні для його реалізації 4](#_Toc485031084)

[Програмні засоби 7](#_Toc485031085)

[Тестування 8](#_Toc485031086)

[Особливості реалізації 9](#_Toc485031087)

[Висновки 11](#_Toc485031088)

# Постановка задачі

Написати програму, що реалізує гру «Морський бій».

Гра розрахована на двох гравців: користувача та комп’ютера. Правила гри:

1. Кожен з гравців на полі 10х10 має розмістити деякі кораблі на своєму полі таким чином, аби вони не доторкались один одного ні сторонами, ні кутами (орієнтація довільна);
2. Розміри і кількість кораблів незмінні: 1х – розміром 4:1, 2х – 3:1,  
   3х – 2:1, 4х – 1:1;
3. Гравці по черзі ходять, стріляючи в довільному порядку в комірки поля супротивника. Відповідь опонента може бути лише однією з трьох: «поцілив», «убив» або «мимо».
4. Якщо гравець «поцілив» або «убив», то йому надається право ходити позачергово.
5. Корабель вважається «убитим», якщо супротивник поцілив у всі комірки, у яких він знаходиться.
6. Ціль гри – знищити всі кораблі супротивника.

Програма повинна мати інтуїтивно-зрозумілий графічний інтерфейс та надавати користувачу змогу зручно підготовуватись до гри, розміщуючи свої кораблі на полі, та, безпосередньо, грати за правилами. Окрім цього, повинна бути додана можливість за необхідності подивитись розташування кораблів супротивника.

# Обраний алгоритм та структури даних, необхідні для його реалізації

Морський бій – гра, у якій немалу роль відіграє удача. Але все ж є декілька стратегій, які дозволяють у середньому значно зменшувати кількість ходів, тим самим збільшуючи свої шанси на перемогу. Враховуючи нескладність гри, алгоритми придумувались самостійно, без використання жодних зовнішніх джерел.

Стратегія штучного інтелекту

Ми знаємо, що кораблі не можуть доторкатись одне до одного. З цього випливає, що навколо знищеного корабля стріляти немає сенсу. Тому для наступних пострілів залишається менше варіантів, а отже збільшуються шанси влучання. Це й породжує стратегію, яка полягає в тому, аби намагатись знищити більші кораблі спочатку, адже їх, по-перше, легше знайти на полі, а по-друге, вони утворюють більші ділянки, у які точно не варто стріляти потім.

Для цього мій штучний інтелект (далі – AI (artificial intelligence)) розпочинає стратегію серією ходів з кодовою назвою Skew Grid 4 (рис. 1). Це дозволяє знайти корабель розміром 4 за найменшу кількість ходів. «Сітка заповнюється» зліва направо. Вертикальний напрямок, а також горизонтальний зсув (від 0 до 3) визначаються випадковим чином. Ця серія ходів використовується, доки найбільший корабель не буде знайдено.

Коли Skew Grid 4 завершується, черга доходить до Skew Grid 2 (рис. 2). Працює цей алгоритм аналогічно і завершується, коли на полі не залишаються лише кораблі розміром 1, після чого контроль переходить до Random Iterator, який буде стріляти по всіх ще не перевірених комірках до кінця гри.

Зазначимо, що ці три підалгоритми лише описують серію ходів, і майже не підлаштовуються під їх результати – єдина поправка є лише в алгоритму Skew Grid: комірки, у яких не можуть вміститись кораблі з розміром більшим або рівним розміру сітки, будуть проігноровані (такі позиції можуть утворитись внаслідок утворення зон, де точно нічого немає, утворених навколо знищених кораблів).

Коли ж результатом ходу AI є поранений корабель (у якого влучили, але не ще знищили), то в стек алгоритмів потрапляє більш просунутий алгоритм – Boat Search, який завершиться після знищення цього корабля.

Boat search – алгоритм, який дозволяє за мінімальну кількість ходів знайти всі комірки корабля, коли відомі координати лише однієї з них. Boat Search поділений на дві частини – Hitting Around та Hitting in Direction. Маючи координату лише однієї знайденої частинки – працює перша частина, а після знаходження другої, відповідно, – друга.

Hitting Around по черзі повертає координати комірок навколо знайденої частинки у випадковому порядку. Робиться також поправка на те, що в тій комірці разом з поточною має вміститись хоча б один ще живий корабель розміром більше 1. Наприклад, якщо кораблів з розміром 2 більше не залишилось, і виникає ситуація як на рис. 3, і AI спробує влучити в нижню комірку (і не влучить, у даному випадку), то верхня буде проігнорована.

Маючи дві сусідні частинки корабля, Hitting in Direction може точно визначити орієнтацію корабля. Тоді кожний наступний хід алгоритм по черзі перевіряє один з двох можливих напрямків, де наступна частина корабля може розташовуватись. Порядок перевірки визначається випадково з кожною новою знайденою частинкою. На рис. 4 у корабель спершу влучили, поціливши в центральну комірку і з наступним ходом корабель розміром 4 буде гарантовано знищений, адже ліва комірка (відносно трьох вже знайдених частин) пуста.

Усі частини загального алгоритму на програмному рівні представлені об’єктами із станом. Коли комп’ютер має здійснити хід, керування передається поточному підалгоритму, який залежно від свого стану визначає наступний хід. Результат ходу повертається одночасно і коду, що визначає наступного гравця, який робитиме хід, і алгоритму, що здійснив цей хід, аби той міг, за необхідності, зредагувати свої дії, або завершитись. Створені підалгоритми розміщуються у стек, тобто останній доданий виконується найпершим.

# Програмні засоби

У якості основної мови я вирішив використовувати JavaScript. Вибір обумовлений високою популярністю цією мови, легкою інтеграцією в Html сторінку та можливістю запускати гру на всіх платформах, де є браузер.

Для написання коду я вирішив використовувати програмне середовище від JetBrains – WebStorm. У цьому програмному середовищі, як і в багатьох інших продуктах на основі Intellij, є такі зручні та корисні функції як code auto completion, code templates, а також можливість запуску локального сервера зі своїм сайтом для тестування продукту.

Також при написанні використовувався фреймворк “p5.js”, який спрощує взаємодію з графічним контекстом html сторінки, а також створення DOM-елементів та їх маніпулювання. Він знаходиться у вільному доступі за адресою <https://p5js.org/>.

Для запуску програми використовувалися браузери Chrome (v58) та Microsoft Edge (v40).

# Тестування

Аби протестувати алгоритми, я запускав гру з деякими початковими умовами унікальними для кожного з них, варіюючи кількість живих кораблів, їхню позицію та стан комірок.

Для перевірки роботи Skew Grid, алгоритм запускався на полях як без кораблів, так і з кораблями – аби проаналізувати його середньостатистичну ефективність. У ході тестування я вирішив, що найбільш вдало алгоритм працює, якщо прибрати випадковий порядок пострілів у позиції сітки, зробивши випадковим лише вертикальний напрямок уздовж цієї сітки.

Random Iterator запускався лише на полях без кораблів.

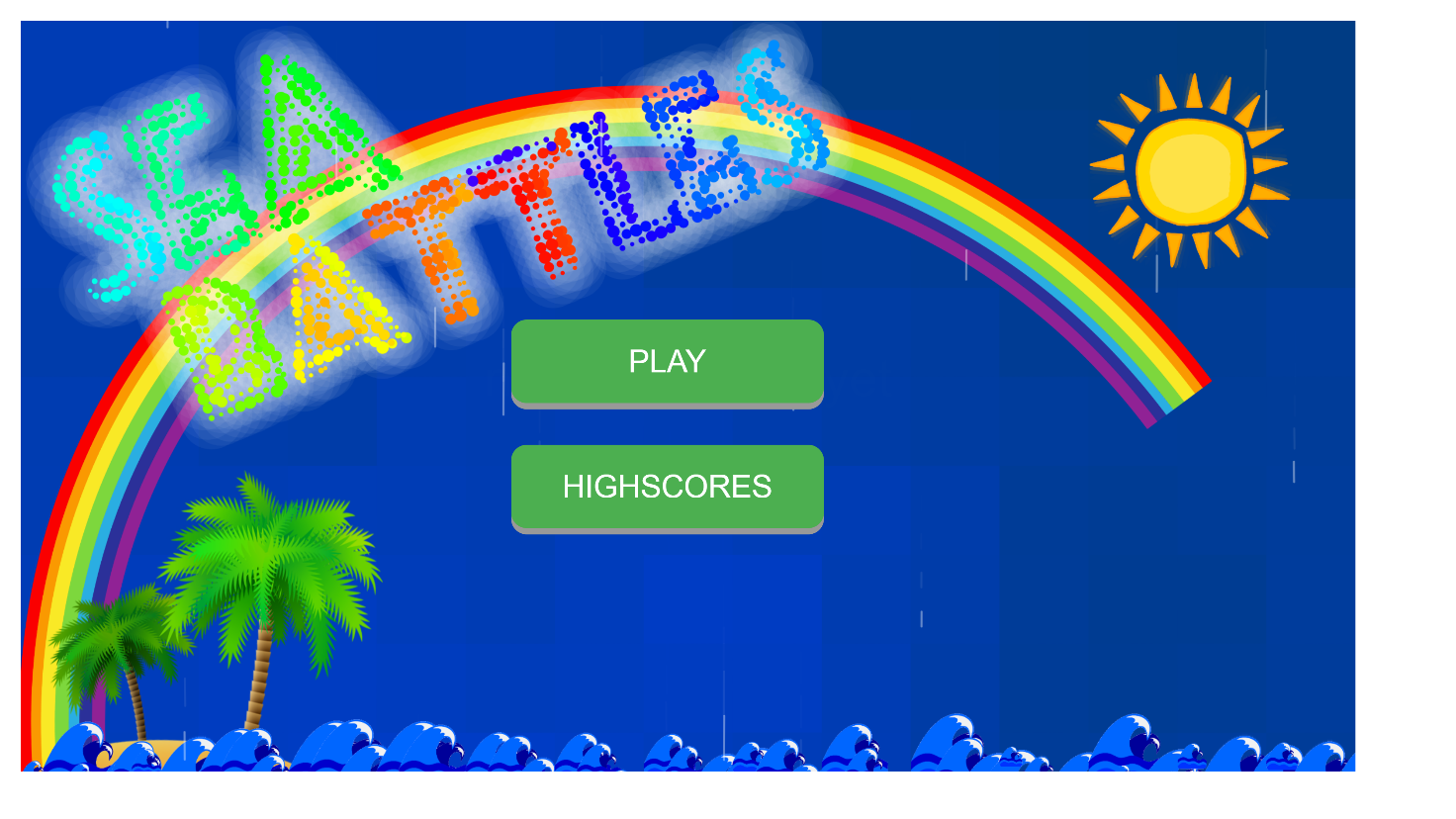
Найбільше перевірок потребував саме Boat Search, початковими умовами для якого були вже «поранені» в одну клітинку кораблі в різних місцях поля серед інших кораблів. Алгоритм не одразу працював злагоджено, і довелось переписувати його декілька разів, доки він не запрацював описаним вище чином.

Та все ж найкращим і фінальним тестуванням стала безпосередньо серія ігор зі своїм AI. Останні версії алгоритму виявились дійсно успішними, і відсоток його перемог досяг приблизно 40-60%.

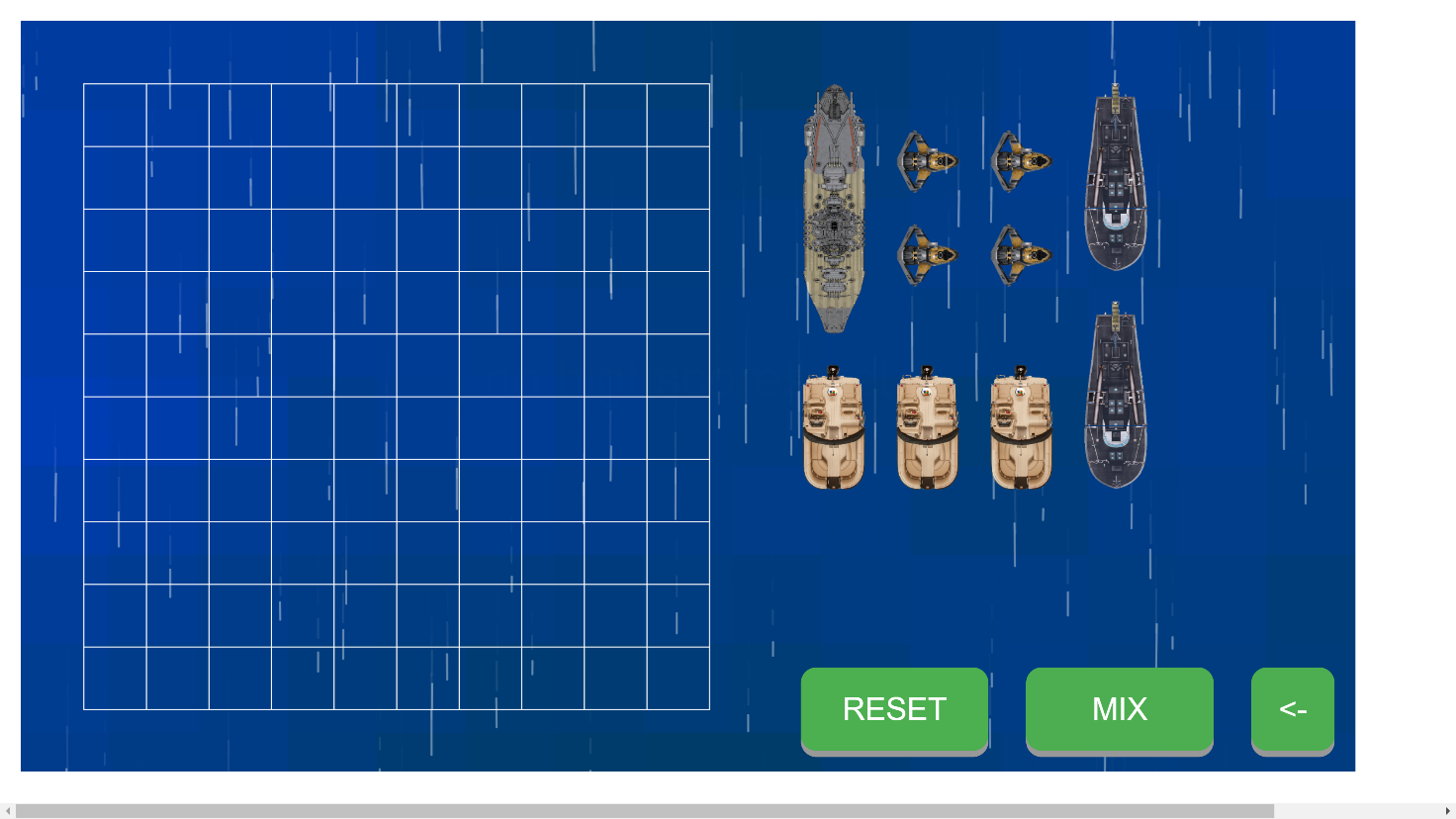
# Особливості реалізації

Програма має чотири основних стани: «меню», «найкращі результати», «підготовка до бою» та, безпосередньо, «бій». Під час гри користувач відчуває занурення в атмосферу відпочинку під час перебування в перших двох станах та, відповідно, атмосферу напруженого бою в останніх двох. Це досягається за рахунок імітації погоди, графічних спецефектів та аудіосупроводу.

У «меню» можна спостерігати дрібний дощ, веселку, яка неквапливо сходить на задньому плані, повільні та гармонійні коливання хвиль, острів, обертання сонця, а також переливання кольорів і розмірів множини дрібних кіл, що разом утворюють назву гри “Sea Battles”.



За кожне влучання в корабель користувач отримує деяку кількість очок. Окрім цього, за комбінацію послідовних потраплянь а також знищення корабля можна заробити бонусні бали. 5 найкращих результатів сеансу гри можна переглянути в розділі «найкращі результати».



Аби розташувати кораблі перед грою, можна просто перетягнути їх мишею. Щоб повернути корабель, достатньо покрутити колесо миші або натиснути на букву ‘R’.

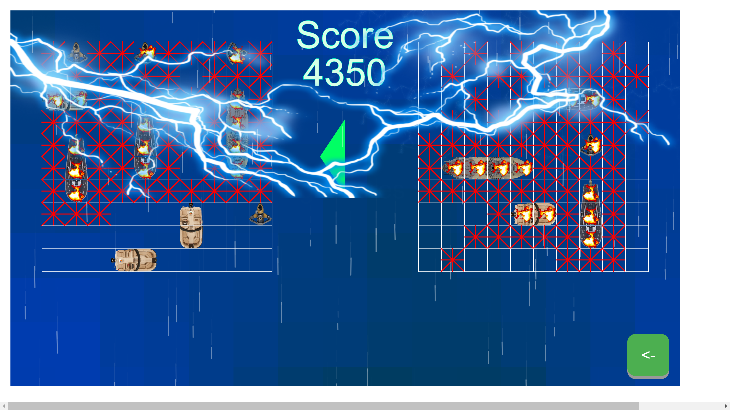
Якщо ж спробувати поставити корабель у позицію, яка за правилами заборонена, то фон корабля стане червоним, а при відтисканні клавіші миші корабель розміститься на останній коректній позиції або десь поза полем, якщо корабель «на шляху перетягування» не міг розташуватись ніде. Після розміщення корабля на полі, навколо нього позначається зона, де не можуть розташовуватись інші кораблі.

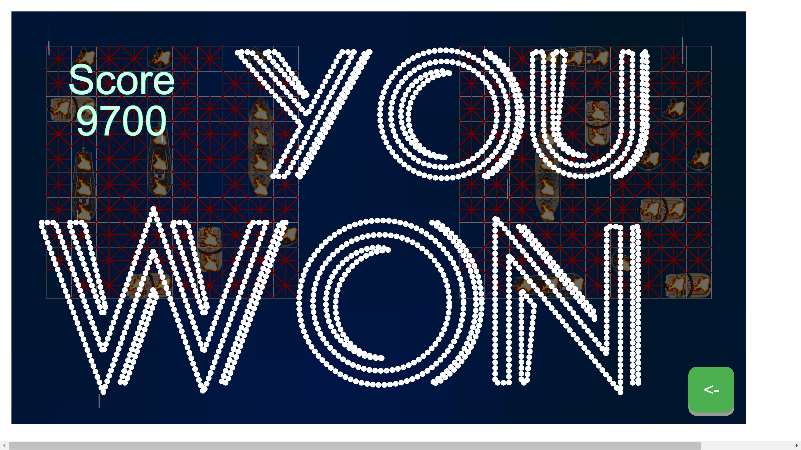
За бажання, користувач може використати випадково генеровані позиції, натискаючи “MIX”, доки не обере собі розташування кораблів до вподоби.

Коли всі кораблі розташовуються за правилами, з'являється кнопка “START” для початку гри.

Поле користувача розташовується зліва. Вказівник посередині змінює колір та орієнтацію, вказуючи на поле гравця, що має зробити хід. Для створення ефекту «AI, що “думає”», комп’ютер ходить після деякої неоднакової затримки після користувача. Потрапляння в пусту клітинку та клітинку з кораблем супроводжуються різними звуковими та візуальними ефектами.



Гра супроводжується динамічною музикою. На полі бою час від часу посилюється дощ, і з деякими інтервалами з’являється навіть грім та блискавка, реалізована за принципом фрактального дерева з випадковими для кожного кадру параметрами, що надає їй реалістичності.

Зрозуміло, що гра завершується виграшем комп’ютера чи користувача. Відповідно утворюються написи «GAME OVER» та «YOU WON». Написи саме “утворюються”, а не ”з’являються” завдяки анімації на основі steering behavior: певна кількість кружечків, необхідних для утворення напису, розташовуються рівномірно по всьому екрану та робляться невидимими. Кожний кружечок «знає», куди йому треба рухатись залежно від результату гри. Коли та закінчується, кружечки поступово стають непрозорими та рухаються на своє місце, складаючи напис. Партія завершується, і користувач може зіграти ще раз.

Після закінчення партії, можна також повернутись до перегляду полів, натиснувши клавішу “g” аби, можливо, подивитись розташування кораблів супротивника в разі програшу, які автоматично стають видимі після закінчення партії. Враховуючи вимоги практичного завдання, також була надана можливість у будь-який час відобразити позиції кораблів комп’ютера натиском клавіші “s”.

# Висновки

Завдання виявилось не лише гарним тренуванням для мозку, а також і можливістю проявити креативний підхід до вирішення задач такого роду. У ході виконання я удосконалив свої навички програмування на Javascript, динамічного створення DOM-елементів на Html сторінці (та маніпулювання ними), а також застосування CSS стилів.

Гра вийшла цікава та динамічна завдяки анімаціям і непоганій графіці. Музикальний супровід за мотивами «Піратів Карибського моря» та дощ із блискавками та громом не дає відірватись від гри, а просунутий штучний інтелект змушує користувача пробувати свої сили знову й знову.

Повний лістинг коду можна переглянути в додатку А (28ст).