Міністерство освіти і науки України

Національний університет «Києво-Могилянська академія»

Факультет інформатики

Кафедра математики

Курсова робота

на тему: «РОЗРОБКА ГРИ З ВИКОРИСТАННЯМ МАТЕМАТИЧНИХ СТРАТЕГІЙ»

Виконав: студент 3-го року

навчання,

Освітньої програми «Прикладна

математика», 113

Верещак Олександр Олегович

Керівник , Борозенний С.О.

Кваліфікаційна робота захищена

з оцінкою

_____

Секретар ЕК

_____

«____» _____

20____ р.

Київ – 20___

Календарний план виконання роботи:

№ п/п	Назва етапу курсового проекту (роботи	Термін виконання етапу	Примітка
1.	Отримання завдання на курсову роботу.	01.10.2022	
2.	Огляд літератури за темою та аналіз матеріалів	01.11.2022	
3.	Опрацювання літератури	20.01.2023 – 10.02.2023	
4.	Розробка комп’ютерної гри	15.03.2023 – 29.03.2023	
5.	Написання текстової частини курсової роботи	12.04.2023 – 05.05.2023	
6.	Захист курсової роботи.	22.05.2023 – 23.05.2023	

Науковий керівник Борозенний Сергій Олександрович

Виконавець курсової роботи Верещак Олександр Олегович

ЗМІСТ

ВСТУП.....4

РОЗДІЛ 1. ДОСЛІДЖЕННЯ МАТЕМАТИЧНИХ СТРАТЕГІЙ В ІГРАХ.....5

1.1 Мінімакс.....5

1.2 Симетрія.....7

1.3 Ізоморфізм.....8

1.4 Ігри Шпрага-Гранді.....10

1.4.1 Нім.....10

1.4.2 Функція Шпрага-Гранді.....11

РОЗДІЛ 2. АНАЛІЗ АЛГОРИТМІВ.....15

2.1 Аналіз покрокових комп'ютерних ігор.....15

2.2 Комп'ютерний аналіз.....15

РОЗДІЛ 3. РОЗРОБКА ГРИ В UNITY......17

3.1 Постанова задачі.....17

3.2 Структура проєкту.....17

3.3 Інтеграція алгоритмів, розробка ботів.....22

ВИСНОВКИ.....23

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ.....24

ВСТУП

Наразі ігрова індустрія займає провідне місце у сфері розваг, люди завжди грали один з одним щоб просто скоротати час або відпочити. За останні роки цей напрям став приносити непоганий прибуток, в геймдизайні все ще використовують фундаментальні правила й стратегії минулих століть.

Мета моєї курсової роботи – об’єднати навички у програмуванні й знання математичних стратегій щоб ознайомити гравців з деякими з них. Провести аналіз ігор та зрозуміти, які правила слід встановити задля рівності гравців.

Зазначу, що в своїй роботі я розгляну лише ігри на скінченному горизонті, які позбавлені випадкових подій, оптимальна стратегія – деякий алгоритм, який будується на аналізі початкової та поточної позицій, можливих ходів супротивника й ставленню його у програшну позицію.

Об’єктом дослідження є ігри, 2 з яких я реалізую за допомогою Unity, даючи змогу грати вдвох чи з ботом.

Предметом дослідження є математичні стратегії в іграх та алгоритми в них.

РОЗДІЛ 1. ДОСЛІДЖЕННЯ МАТЕМАТИЧНИХ СТРАТЕГІЙ В ІГРАХ

1.1 Мінімакс

Назвемо гравців Максом і Міном, результатом гри буде сума яку Мін має виплатити Максу (у разі від'ємного результату - навпаки). Очевидно що Макс хоче максимізувати суму а Мін – мінімізувати. Таке представлення дає можливість аналізувати ігри з різними результатами.

Ціна гри n – число для якого мають справджуватись наступні умови:

1) Макс має стратегію, яка гарантує результат не менше n

2) Мін має стратегію, яка гарантує результат не більше n

Для прикладу розглянемо таку гру:

В клітинці a1 шахівниці стоїть пішак. Двоє гравців по черзі рухають його на одну клітинку вгору або вправо. Макс ходить першим. Коли пішак потрапляє на діагональ (після 7 ходів), гра закінчується і її результат визначається за рисунком 1.1:

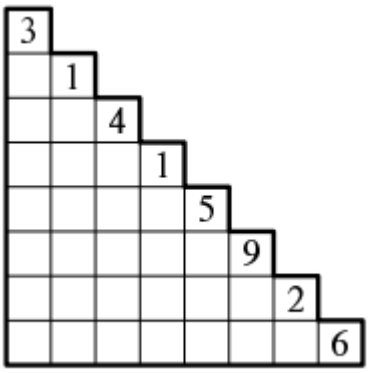

Рисунок 1.1 Початкова значення гри

Знаючи, що Макс ходить останнім, можемо продовжити цей рисунок 1.2:

3							
3	1						
	4	4					
		4	1				
			5	5			
				9	9		
					9	2	
						6	6

Рисунок 1.2 Ціна позицій після першої ітерації

Продовжуючи цей алгоритм чергуючи ходи Макса і Міна – розпишемо весь рисунок:

3							
3	1						
3	4	4					
4	4	4	1				
4	4	4	5	5			
4	4	5	5	9	9		
4	5	5	9	9	9	2	
5	5	9	9	9	6	6	6

Рисунок 1.3 Ціна всіх позицій після алгоритму

Чорні клітинки – ходи Макса, білі – Міна. Таким чином ми бачимо, що ціною цієї гри з заданими результатами буде число 5. Щоб гарантувати собі такий результат (5 чи більше), Макс має ходити в сусідню клітинку з більшою ціною. Аналогічно Мін має обирати меньшу ціну клітинки. Це і буде оптимальною стратегією для кожного з гравців.

Це був приклад гри з повною інформацією на скінченному горизонті, виходячи з цього можемо записати наступний алгоритм задання гри:

1) Вказати скінченну множину, елементами якої будуть позиціями гри.

2) Розділити позиції на 3 виключні класи: заключні (гра закінчена), ходи Макса, ходи Міна.

3) Для кожної заключної позиції вказати результат гри (задати суму)

4) Розрахувати ціну всіх попередніх ходів оперуючись ходами Макса й Міна.
5) Розрахувати ціну гри.

Цей алгоритм дасть нам оптимальну стратегію для кожного гравця в будь-якій позиції.

Також цю гру можно представити у вигляді одностороннього орієнтовного графа, де кожна вершина має свою ціну. В такому разі гравці також обирають ціну вершини за своїми стратегіями.

Проста зміна візуального відображення гри може змінити підхід гравця до неї, ця властивість називається ізоморфізмом, його я розпишу в пункті 1.4

1.2 Симетрія

Інколи вдається придумати виграшну стратегію не проводячи повного аналізу гри (що особливо важко при великій кількості позицій). В деяких іграх така стратегія основана на симетрії.

Приклад:
Двоє гравців кладуть однакові круглі монети на стіл. Вони не можуть перекривати одне одну й не мають падати зі стола. Гравець який не може положити монету - програв.

Виграшна стратегія для першого гравця – положити монету в центр стола й повторювати ходи другого гравця симетрично відносно центру стола. В цьому разі якщо існує деяке місце p для другого гравця, то також буде існувати симетричне йому місце q.

Цей факт здається очевидним, але якщо другий гравець спробує керуватись цією стратегією, то інсує 2 варіанти:
1) Перший гравець рано чи пізно покладе монету в центр й симетрія стане неможливою

2) Другий гравець покладе монету в центр й перший гравець рано чи пізно порушить її поклавши монету своїй попередній симетрично.

Проаналізувавши цю стратегію, можна дійти висновку, що для симетрії потрібно:

1) Зрівняти положення гравців (віддати ініціативу другому гравцю)

2) Відновлювати баланс гри своїм ходом, дублюючи ходи іншого гравця симетрично.

1.3 Ізоморфізм

Як я вже писав вище, ізоморфізм – властивість ігор, які здаються різними але використовують однаковий алгоритм виграшної стратегії. Як приклад:
На клітинці h8 стоїть “одностороння тура”, яка може рухатись лише вліво чи вниз. Двоє гравців ходят по черзі, рухая туру вниз або вліво на будь-яку кількість клітинок (але не менше 1). Гравець який не може зробити хід – програв.
Дізнавшись правила, можемо одразу записати результат деяких крайніх клітинок (рисунок 1.4):

В							
В							
В							
В							
В							
В							
В							
П	В	В	В	В	В	В	В

Рисунок 1.4 Результуючі позиції

Далі очевидно, що клітинка a2 також буде програшною, тож можемо записати наступний рівень (рисунок 1.5):

В	В						
В	В						
В	В						
В	В						
В	В						
В	В						
В	П	В	В	В	В	В	В
П	В	В	В	В	В	В	В

Рисунок 1.5 Позиції після першої ітерації

Продовжуючи данну думку, розпишемо всю шахівницю:

В	В	В	В	В	В	В	П
В	В	В	В	В	В	П	В
В	В	В	В	В	П	В	В
В	В	В	В	П	В	В	В
В	В	В	П	В	В	В	В
В	В	П	В	В	В	В	В
В	П	В	В	В	В	В	В
П	В	В	В	В	В	В	В

Рисунок 1.6 Остаточна оцінка позиції

То ж можемо дійти висновку, що в нашому випадку перемагає другий гравець. Для цього він має повертати туру на діагональ, коли її здвигає перший.

Ізоморфізм полягає в тому, що цю гру можна перефразувати як гру Нім, з двума купками по сім сірників. Гравці так само беруть будь-яку кількість

елементів з однієї купки (рухають туру по одній осі). В координаті (0;0) сірників не залишилось.

(0, 4)	(1, 4)	(2, 4)	(3, 4)	(4, 4)
(0, 3)	(1, 3)	(2, 3)	(3, 3)	(4, 3)
(0, 2)	(1, 2)	(2, 2)	(3, 2)	(4, 2)
(0, 1)	(1, 1)	(2, 1)	(3, 1)	(4, 1)
(0, 0)	(1, 0)	(2, 0)	(3, 0)	(4, 0)

Рисунок 1.7 Ізоморфічне представлення позицій

Отже, в цьому випадку, гравець який знає виграшну стратегію німа, одночасно знає й виграшну стратегію поточної гри й навпаки.

1.4 Ігри Шпрага-Гранді

1.4.1 Нім

Ще одним прикладом ізоморфізму можна вважати тип ігр “Нім”.
Нім – гра в якій 2 гравці по черзі черзі беруть предмети, розкладені на кілька купок. За один хід можна брати будь-яку кількість предметів (більше нуля) з одної купки. Виграє той, хто забрав останній предмет.

При одній кучці, очевидно, виграє перший гравець.

При двух кучках ми можемо керуватись стратегією симетрії. Якщо вони різні – перший гравець урівнює кількість елементів й симетрично повторює за другим. У разі рівності елементів – виграє другий гравець за аналогічною стратегією.

Для загальної кількості кучок можна скористатись наступним алгоритмом:
1) Запишемо кількість елементів в кожній кучці в двоїчній системі один під одним в стовпчик

2) Порахуємо відповідну кількість одиниць в кожному стовпчику

3) Якщо сума всіх стовпців парна – позиція програшна

4) Якщо існує хоча б один стовпчик з непарною сумою – позиція виграшна

Нехай позиція буде називатись парною, якщо сума одиниць у всіх стовпцях парна, і непарною, якщо існує хоча б 1 стовпчик з непарною сумою.

Тоді аналізуючи цей алгоритм, гравець у непарній позиції має керуватись наступною стратегією:
1) Зробити парну позицію

2) Не дати іншому гравцю залишитись в парній позиції

Треба зауважити, що в непарній позиції завжди можна зробити хід, який приведе у парну позицію. В непарній позиції існує один чи декілька непарних стовпців. Розглянемо самий лівий з них. Він має непарну кількість одиниць, то ж має хоча б одну одиницю. Виберемо одну з таких одиниць. Заміним її на нуль, зробив стовпчик парним, якщо справа все ще залишились непарні стовпці, замінимо 0 на 1 чи навпаки.
Також треба сказати, що зробивши будь-який хід у парній позиції, ми отримаєм непарну. Тому як хід змінить одне з чисел, залишаючи інші числа незмінними. В зміненому числі є зміненний розряд і якщо раніше в стовпці була парна кількість одиниць, то тепер – непарна.

1.4.2 Функція Шпрага – Гранді

Попередня гра “Нім” відноситься до класу ігр Шпрага-Гранді, цей тип ігор представляє собою орієнтовний граф без циклів. Гра полягає в тому, що гравці рухають фішку по вершинам графа поки це можливо, хто не може зробити хід – програє.

Класифікація позицій якими ми керувались наступна: позиція виграшна, якщо з нього існує хід у програшну позицію, програшна якщо всі ходи ведуть у виграшну позицію.

Тепер змінимо це правило. Замість виграшної/програшної позиції, ми будемо задавати їх оцінку у вигляді цілого невід'ємного числа, тоді правило буде наступним: В позицію ставиться найменше ціле невід'ємне число, яке відступне на тих вершинах, в які можна зробити хід.

Можна зрозуміти, що 0 буде стояти в кінцевих ходах й коли в сусідніх вершинах не стоїть 0, схожим правилом ми керувались коли розбивали позиції на програшні/виграшні: позиція програшна, коли з неї не можна зробити хід в програшну позицію. Тим самим ми довели лему: позиція має оцінку 0 тоді й тільки тоді, коли вона є програшною.

Тепер ми задамо наступну функцію V(m, n) | $m, n \in N \cup \{0\}$.

V(m, n) поверне найменше ціле число відмінне від V(m, n`) при n`< n і всіх V(m`, n) при всіх m`< m. Це правило можна переформулювати у вигляді трьох властивостей функції V:

1) Якщо $m \neq m$` , то V(m, n) ≠ V(m`, n) для будь-якого n.

2) Якщо $n \neq n$` , то V(m, n) ≠ V(m, n`) для будь-якого m.

3) Якщо k < V(m, n), то k = V(m, n`) при деякому n`< n або k = V(m`, n) при деякому m`< m.

Виявляється, що для обчислення

$v(m, n)$ треба записати m і n у двійковій системі числення, а потім скласти в стовпчик, але порозрядно (без переносів у наступний розряд). Іншими словами, в i-му розряді числа $v(m, n)$ стоїть одиниця в тому і тільки тому випадку, коли в i-му розряді чисел m і n стоять різні цифри (нуль і одиниця). Позначаючи таке додавання знаком $\oplus$, отримуємо, що $v(m, n) = m \oplus n$. Щоб довести це, треба перевірити такі властивості операції $\oplus$:

1) Якщо $m' \neq m$, то $m \oplus n \neq m' \oplus n$ для будь-якого n.

2) Якщо $n' \neq n$, то $m \oplus n \neq m \oplus n'$ за будь-якого m.

(в) Якщо $k < m \oplus n$, то $k = m' \oplus n$ для деякого $m' < m$ або $k = m \oplus n'$ для деякого $n' < n$.

Доведення:
Якщо в сумі $m \oplus n$ змінити один доданок, то він зміниться в деякому розряді - порозрядна сума зміниться в тому самому розряді. Звідси випливають пункти 1 і 2.

Доведемо тепер 3. Нехай $k < m \oplus n$. Розглянемо найстарший розряд, у якому k відрізняється від $m \oplus n$. У цьому розряді в k стоїть нуль, а в $m \oplus n$ стоїть одиниця. Отже, (рівно) в одному з чисел m і n у цьому розряді стоїть одиниця. При заміні цієї одиниці на нуль число зменшиться, навіть якщо ми змінимо молодші розряди так, щоб підігнати $m \oplus n$ до k.

Тепер визначимо поняття суми двох ігор Шпрага - Гранді. При цьому сумою двох ігор з однією купкою сірників виявиться гра з двома купками сірників. Неформально кажучи, дві гри відбуваються паралельно (кожна на своїй дошці), причому гравець може вибрати, на якій із двох дощок він хоче ходити в даний момент. Отже, позиція в грі-сумі являє собою пару (P, Q), де P - позиція в першій грі, а Q - у другій. З позиції (P, Q) можна піти в позицію (P', Q) для будь-якої позиції $P`$, у яку можна піти з $P$, а також у позицію$(P, Q`)$ для будь-якої позиції $Q`$, у яку можна піти з Q.

Теорема Шпрага – Гранді: оцінка позиції (P, Q) у сумі двох ігор дорівнює побітовій сумі оцінок позицій P і Q. Іншими словами, якщо позиція P у першій грі має оцінку p, а позиція Q у другій грі має оцінку q, то позиція (P, Q) у сумі ігор має оцінку $p \oplus q$.
Доведення: проводимо індукцію за максимальним числом ходів, що залишилися в позиції (P, Q). Позначаючи оцінку буквою φ, рівність, що доводиться, можна записати як $\varphi(P, Q) = \varphi(P) \oplus \varphi(Q)$.

За визначенням ми повинні перевірити, що:

1) $\varphi(P) \oplus \varphi(Q)$ відрізняється від $\varphi(P', Q')$ для будь-якої позиції (P', Q'), у яку можна піти з (P, Q).

2) $\varphi(P) \oplus \varphi(Q)$ є найменшим числом з такою властивістю.

За визначенням суми ігор один із членів пари (P', Q') залишився таким самим, як у парі (P, Q), а інший вийшов ходом у грі-доданку. Тому величина $\varphi(P', Q')$, за припущенням індукції рівна $\varphi(P') \oplus \varphi(Q')$, відрізняється від $\varphi(P) \oplus \varphi(Q)$ рівно в одному доданку, і залишається скористатися першими двома властивостями операції $\oplus$. Першу частину доведено.

Для доведення другої, розглянемо довільне ціле невід'ємне k, менше $\varphi(P) \oplus \varphi(Q)$. За третьою властивістю операції $\oplus$ число k можна подати у вигляді $p' \oplus \varphi(Q)$ з $p' < \varphi(P)$ або у вигляді $\varphi(P) \oplus q'$ з $q' < \varphi(Q)$. Нехай правильне, скажімо, перше. Тоді (за визначенням оцінки) p' дорівнює $\varphi(P')$ для деякої позиції P', у яку можна піти з P, і тому $k = p' \oplus \varphi(Q) = \varphi(P') \oplus \varphi(Q) = \varphi(P', Q)$ (остання рівність правильна за припущенням індукції), так що k зустрічається серед оцінок позицій, у які можна піти з (P, Q).Теорему доведено.

РОЗДІЛ 2. АНАЛІЗ СТРАТЕГІЙ

2.1 Аналіз покрокових комп’ютерних ігор

Опираючись на розглянуті стратегії ігор на скінченному горизонті з повною інформацією, ми можемо провести аналіз сучасних комп’ютерних покрокових стратегій.

Більшість з них являють собою ігри з неповною інформацією (гравець не має точних данних про положення інших гравців та їх можливих ходів). Через те що вищепредставлені приклади представляють доволі елементарні правила і стратегії – вони не були б достатньо цікавими й складними. Натомість комп’ютерні ігри з повною інформацією містят настільки велику кількість позицій, що на даний момент навіть комп’ютерний аналіз не може дати гарантований результат оптимальної стратегії.

2.2 Компьютерний аналіз

Комп’ютерний аналіз зазвичай використовує алгоритм мінімаксу та його модифікацію Альфа-Бета відсічення, яка дозволяє зменшити кількість перебраних ходів, не втрачаючи оптимальності рішення.
Основна ідея полягає в тому, щоб не розглядати всі можливі ходи в кожному стані гри, а зупинитися на тих, які не можуть дати кращий результат. Для цього використовуються два параметри - альфа і бета, які позначають найкращий результат, який гарантовано можна отримати для максимізуючого гравця та для мінімізуючого гравця відповідно.

Алгоритм починається з максимізуючого гравця, який розглядає всі можливі ходи та визначає їх оцінки. Після цього переходиться до мінімізуючого гравця, який також розглядає всі можливі ходи та визначає їх оцінки. При цьому, якщо оцінка ходу мінімізуючого гравця менша за альфа, то цей хід можна відкинути, оскільки максимізуючий гравець вже може отримати не гірше за альфа. Аналогічно, якщо оцінка ходу максимізуючого гравця більша

за бета, то цей хід можна відкинути, оскільки мінімізуючий гравець вже може запобігти гіршому результату за бета.

При належному використанні, алгоритм Альфа-Бета відсічення може дозволити зменшити кількість перебраних ходів вдвічі.

РОЗДІЛ 3. РОЗРОБКА ГРИ В UNITY

3.1 Постанова задачі

Моя мета – створити гру, яка допоможе гравцю ознайомитись зі стратегіями двух описаних вище ігор (мінімакс алгоритм пішака на шахівниці, симетрія/нім з сірниками) й дасть змогу на практиці зрозуміти чому вони саме такі. Гравець має мати змогу грати з другом чи ботом, який у свою чергу буде керуватись виграшною стратегією в кожній грі.

Керування буде реалізовано за допомогою миші, що дасть змогу грати як на телефонах так і на персональних комп'ютерах.

Також треба знайти спрайти шахівниці з фігурами та музику зі звуками.

3.2 Структура проєкту

Проєкт складається з трьох сцен: головного меню, ігор в шахівницю та сірники. Загалом проєкт містить 15 класів.

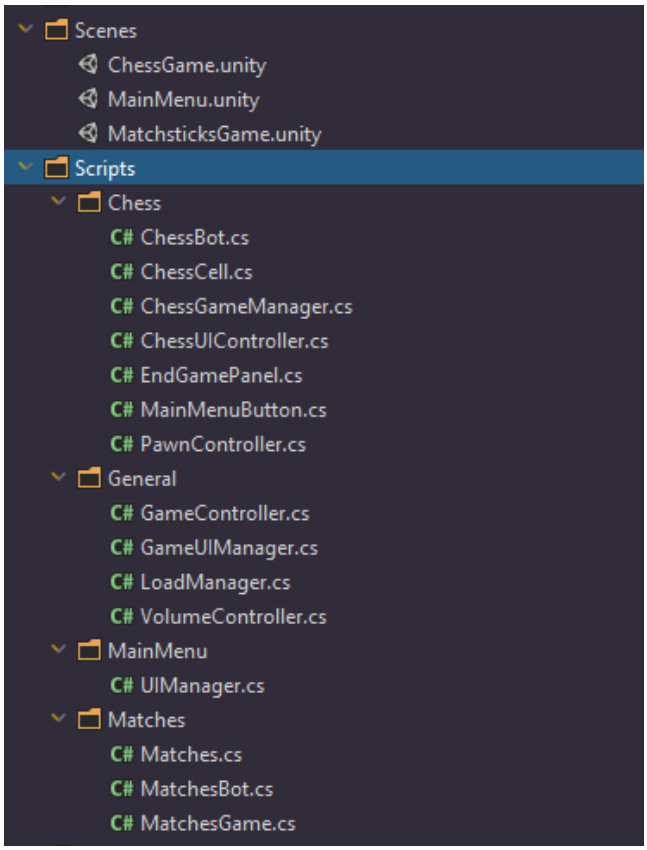

Рисунок 3.1 Структура проєкту

Головне меню я вирішив організувати наступним чином:

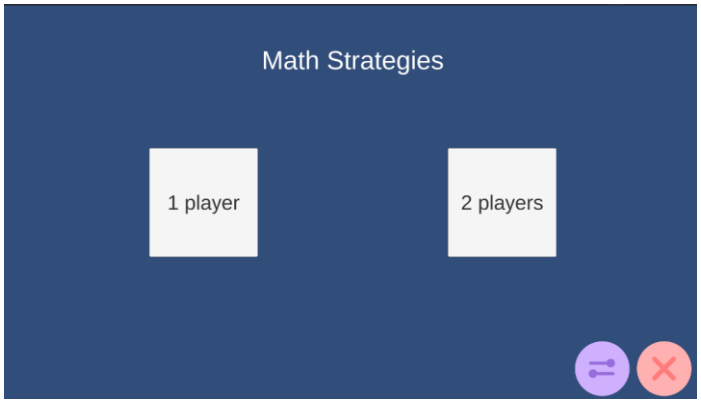

Рисунок 3.2 Вигляд головного меню

Маємо 2 кнопки режиму гри (під час натискання на які в статичну змінну numberOfPlayers записується кількість гравців) для одного гравця проти бота, чи для двох гравців.

Також внизу маємо кнопки налаштувань та виходу. В налаштуваннях я вирішив додати слайдери гучності звуку й музики.

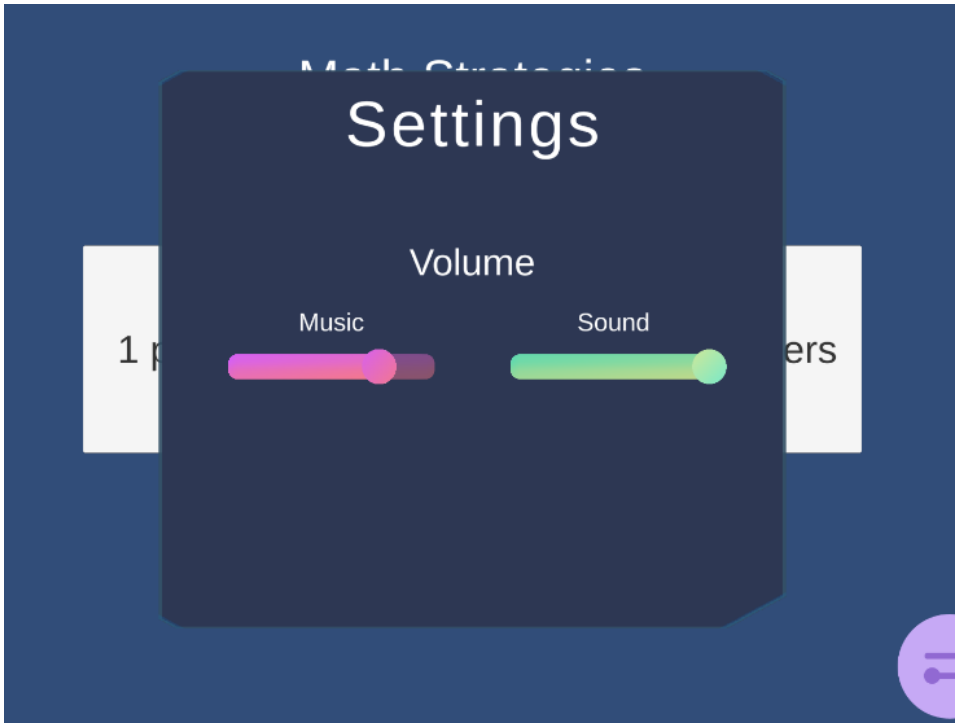

Рисунок 3.3 Вигляд налаштувань

При натисканні на будь-який режим, ми переходимо в меню вибору гри, наразі їх 2, над ними описаний алгоритм, за яким будується виграшна стратегія. Також я додав кнопку Back щоб гравці мали змогу повернутись до головного меню з режиму вибору.

Рисунок 3.4 Вигляд режимів гри

Керування в шахівниці я реалізував за допомогою Drag-and-drop системи, також я вирішив додати кнопку ShowNextStep, яка показує ціну клітинки на наступну ітерацію мінімакса. Таким чином гравець завжди зможе подивитись ціну клітинки й зрозуміти чому алгоритм працює саме таким чином. Після сьомого ходу при потраплянні на діагональ, з'являється панель з результатом

гри й кнопкою виходу в головне меню.

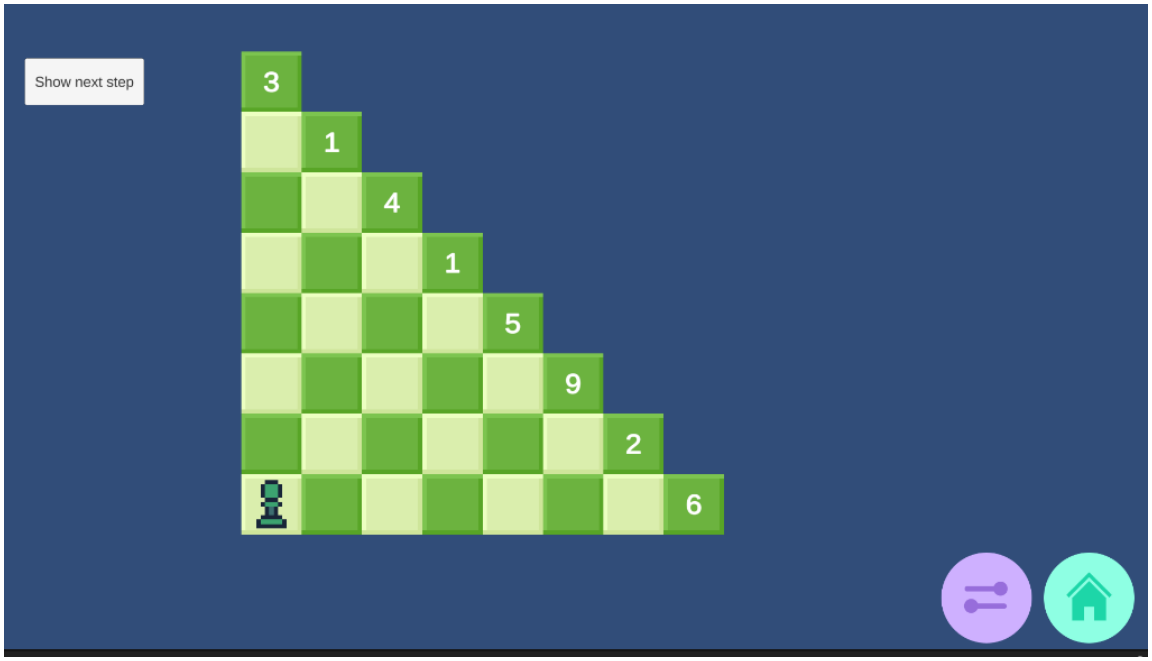

Рисунок 3.5 Вигляд гри у шахівницю

Управління в сірниках реалізовано через натискання на сірники з наступним натисканням кнопки EndTurn. Слід пам'ятати, що гравець може брати елементи лише з одної групи за хід, для реалізації цього я створив Enum поточної купки гравця, якщо він не взяв сірник – може взяти з будь-якої

купки, після чого ця група запишеться у поточну й гравець зможе брати сірники лише з неї.

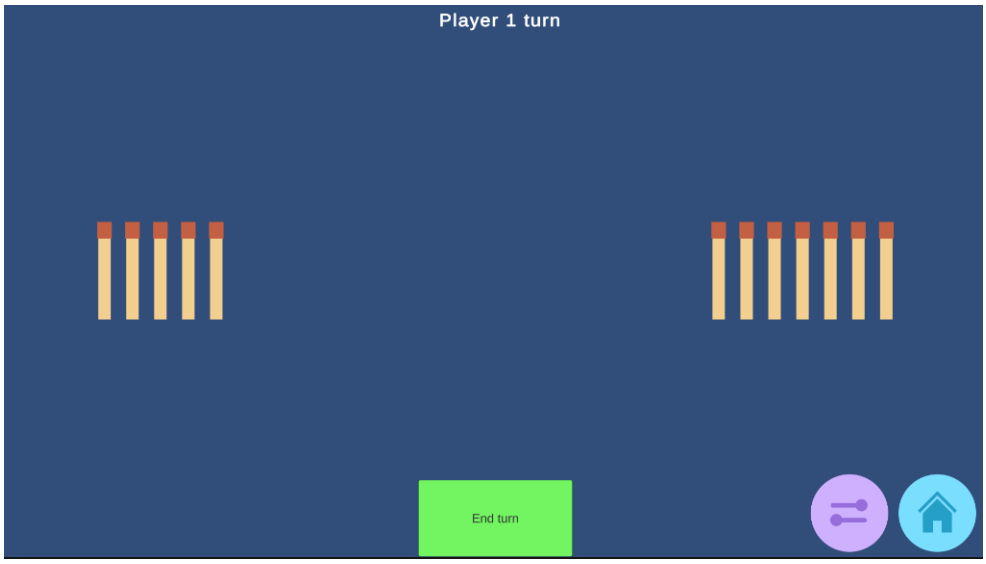

Рисунок 3.6 Вигляд гри у сірники

Після закінчення сірників, з'являється панель з відображенням переможця.

Рисунок 3.7 Панель закічнення гри

3.3 Інтеграція алгоритмів, розробка ботів

Першим алгоритмом я реалізував Мінімакс (шахівницю). Гравець ходить першим, то ж алгоритм ми маємо реалізувати для Міна.

У класі ChessCell я створив вектор розташування клітинки відносно a1 й змінну ціни. Розмістив скрипт на кожну клітинку й задав відповідні значення.

У класі PawnController міститься інформація про поточну позицію пішака, то ж алгоритм мінімаксу у класі ChessBot просто зчитує ціну сусідніх клітинок й рухається на Min(price(a), price(b)).
Додати цього бота й реалізувати його алгоритм було нескладно, так як все що потрібно – порівнювати ціну сусідніх клітинок.
Ціна цієї гри – 5, то ж граючи з ботом, ви можете гарантувати собі виграш у 5 очок, але не більше.

Другий алгоритм для реалізації – симетрія. Основний принцип – порівнюємо кількість елементів в купках за допомогою класа MatchesGame й беремо елементи з більшої поки купки не зрівняються. Тут теж не виникало особливих труднощів, єдине що цей алгоритм вів себе некоректно при симетричних купках. Я задав логіку, щоб у цьому разі він брав 1 сірник з випадкової кучки.
Цей бот так само керується оптимальною стратегією, то ж ви можете виграти лише при урівнянні купок на першому кроці.

ВИСНОВКИ

Під час написання курсової роботи було проведено дослідження типових математичних стратегій у іграх без випадкових подій, 2 з яких були реалізовані за допомогою Unity та C#. Також було проаналізовано стратегії сучасних покрокових комп'ютерних ігор.

Можна сказати, що ігрова індустрія зробила величезний крок у порівнянні з фундаментальними стратегіями і її слід аналізувати керуючись теорією ігор з випадковими подіями. Хоча низка сучасних ігор все ще будуються на великій купі інформації, яку неможливо повністю проаналізувати за допомогою алгоритму мінімакса чи альфа-бета відсіченням. Натомість комп'ютерний аналіз покрокових ігор з повною інформацією все ще керується цими алгоритмами.

Я надав можливість гравцям ознайомитись з елементарними математичними стратегіями і зрозуміти їх алгоритм на практиці.

ДЖЕРЕЛА

1. Шень О. Х. Игры и стратегии с точки зрения математики / Олександр Х. Шень. – Москва: МЦНМО, 2018. – 56 с.

2. Vanderhoek Z. GRE Math Strategies / Zeke Vanderhoek., 2018. – 792 с.

3. Engel A. Problem-Solving Strategies / Arthur Engel. – Berlin: Springer, 1997. – 413 с.

4. Itch.io. (n.d). Sprite Asset https://brysiaa.itch.io/pixel-chess-assets-pack

5. Unity.com. (n.d). UI Asset https://assetstore.unity.com/packages/2d/gui/icons/sleek-essential-ui-pack-170650