Міністерство освіти і науки України
НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ «КИЄВО-МОГИЛЯНСЬКА АКАДЕМІЯ»
Кафедра мультимедійних систем факультету інформатики

РОЗРОБКА ГРИ З ВИКОРИСТАННЯМ ФРЕЙМВОРКУ UNITY

Текстова частина до курсової роботи
за спеціальністю „Інженерія програмного забезпечення” 121

Керівник курсової роботи
с.в. _Борозенний С.О.__
(прізвище та ініціали)
_____
(підпис)
“____” _____ 2022 р.

Виконав студент _____
___Перун Є.Т._____
(прізвище та ініціали)
“____” _____ 2022 р.

Київ, 2022

Міністерство освіти і науки України

НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ «КИЄВО-МОГИЛЯНСЬКА АКАДЕМІЯ»

Кафедра мультимедійних систем факультету інформатики

ЗАТВЕРДЖУЮ

Зав.кафедри мультимедійних систем,

доцент, к.ф-м.н.

_____О. П. Жежерун (підпис)

„____”_____2022 р.

ІНДИВІДУАЛЬНЕ ЗАВДАННЯ

на курсову роботу

студенту Перун Єлизаветі Тарасівні факультету інформатики 3-го курсу

ТЕМА Розробка гри з використанням фреймворку Unity

Зміст ТЧ до курсової роботи:

Індивідуальне завдання

Вступ

1 Історія розробки відео-ігор та огляд поширених інструментів для розробки відео-ігор

2 Діаграми Вороного

3 Розробка гри

Висновки

Список літератури

Додатки (за необхідністю)

Дата видачі „___” _____ 2022 р. Борозенний О.С. _____ (підпис)

Завдання отримав _____ (підпис)

Тема: Розробка гри з використанням фреймворку Unity

Календарний план виконання роботи:

№ п/п	Назва етапу дипломного проекту (роботи)	Термін виконання етапу	Примітка
1.	Отримання завдання на курсову роботу	01.11.2021	
2.	Аналіз матеріалів за темою	14.01.2022	
3.	Розробка та програмування алгоритму	14.02.2022	
4.	Написання текстової частини до курсової роботи	20.03.2022	
5.	Коригування виконаної роботи	07.04.2022	
6.	Створення слайдів для доповіді та написання доповіді.	09.04.2022	
7.	Остаточне оформлення роботи та слайдів	11.04.2022	
8.	Захист курсової роботи	13.06.2022	

Перун Є. Т. _____

Борозенний О. С. _____

“_____”_____

Зміст

Анотація

Робота присвячена розробці гри з використанням фреймворку Unity. Гра реалізована мовою C# з використанням діаграм Вороного для генерації ігрового поля.

Ключові слова: Unity, діаграма Вороного.

Вступ

Постановка завдання

Метою роботи було обрано створення гри від третього лиця на ігровому рушії Unity з базовими механіками пересування, атаки, дослідження та з використанням діаграм Вороного для генерації ігрового поля.

Ігри як вид інтерактивного аудіо-візуального мистецтва не втрачають своєї актуальності протягом десятків років, а такі інструменти, як ігровий рушій Unity та діаграми Вороного, є добре вивченими, поширеними та ефективними у вирішенні поставленого завдання.

Основна частина роботи складається з 3 розділів.

Перший розділ присвячено історії відео-ігор, їхньому місці у сучасному світі та огляду найбільш поширених ігрових рушіїв.

Другий розділ містить інформацію про діаграми Вороного та їхнє застосування у розробці відео-ігор.

У третьому розділі розглядається дослідження обраних методів та інструментів розробки гри та результати роботи.

Основна частина

1. Історія розробки відео-ігор та огляд поширених інструментів для розробки відео-ігор

1.1 Історія відео-ігор

Історія розробки відео-ігор почалася у 1950-х з розвитком технологій та появою перших міні-комп’ютерів і технологій відтворення зображення на екранах таких комп’ютерів. Першою грою з використанням дисплею вважають гру Tennis For Two, що було розроблено у 1958 році.

Рис. 1 Гра Tennis for Two

У 1962 році студентами Массачусетського технологічного інституту було розроблено гру Spacewar! [1], що підштовхнуло розвитку пристроїв призначених спеціально для відтворення відео-ігор.

Рис. 2 Гра Spacewar!

Важливим етапом розвитку ігор стало рішення компанією Id Software продавати ліцензії на використання ігрових рушіїв, що зробило розробку відео-ігор легшою та доступнішою незалежним розробникам. У 2000-х набули поширення онлайн-ігри, також розповсюдженою стала цифрова дистрибуція ігор, що дало дорогу незалежним «інді»-розробникам[2]. Сучасний стан розвитку технологій дозволяє використовувати технології Augmented Reality і Virtual Reality та прирівнює ігри за масштабом та якістю зображення до кінофільмів.

В сучасних реаліях відео-ігри використовуються не лише для розваги, а також можуть бути навчальними засобами чи витворами мистецтва.

1.2 Сучасні ігрові рушії для розробки відео-ігор

Хоча у світі існує безліч ігрових рушіїв, найбільш поширеними вважають рушії Unreal Engine та Unity[3].

Unreal Engine – рушій, що розроблюється компанією Epic Games з 1998 року. Рушій написано мовою C++, він підримує розробку ігор для більшості поширених операційних систем та платформ, серед яких Microsoft Windows, Linux, Mac OS і Mac OS X, консолі Xbox, Xbox 360, PlayStation 2, PlayStation Portable, PlayStation 3, Wii, Dreamcast і Nintendo GameCube.

Історія рушія Unity починається у 2002 році, спочатку рушій розроблявся лише для платформи Mac OS[3], хоча зараз, як і Unreal Engine, підтримує більшість популярних операційних систем та платформ разом з уже переліченими. Unity написано мовою C#.

Рушії пропонують розробнику широкий вибір функціоналу та засобів для розробки гри, серед яких:

- Механізм візуалізації для 2D та 3D-графіки
- Фізичний двигун для управління рухом
- Звук
- Написання сценаріїв
- Анімація
- Штучний Інтелект
- Мережа
- Управління пам'яттю
- Інструменти VR
- Підтримка як 2D-ігор, так і 3D-ігор

Оскільки Unity написано на C#, а UE на C++, перший вважається швидшим, оскільки C# більше підходить для розробки відео-ігор. Також Unity пропонує більше користувацьких матеріалів, в нього більша користувацька база і він більш доступний для початківців через існування більшої кількості відповідних курсів та уроків. В свою чергу UE пропонує кращу якість графіки та відкрите програмне забезпечення[4].

2. Діаграми Вороного

2.1 Визначення та способи побудови діаграми Вороного

Діаграма Вороного — вид розбиття метричного простору, що визначається відстанями до заданої множини точок цього простору.

У найпростішому випадку ми маємо множину точок площини S, які називаються вершинами діаграми Вороного. Кожній вершині s належить комірка Вороного V(s), утворена з усіх точок ближчих до s ніж до будь-якої іншої вершини. Границі на діаграмі Вороного є всіма точками на площині, які рівновіддалені від двох найближчих вершин. Вузли Вороного — точки рівновіддалені від трьох і більше вершин.

Дуальний граф діаграми Вороного має вигляд тріангуляції Делоне цієї ж множини точок.

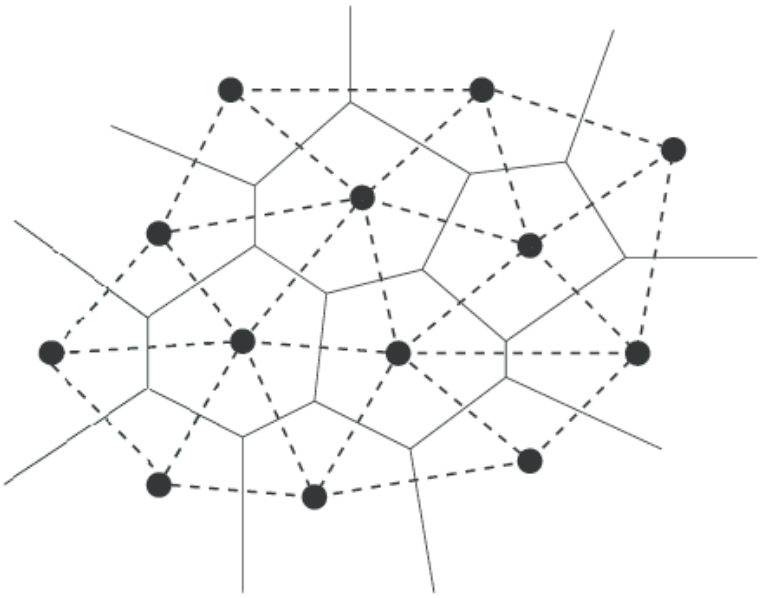

Рис. 3 Діаграма Вороного(суцільною) та відповідна їй тріангуляція Делоне(пунктиром)

Для $n \geq 3$ точок на площині кількість вершин Діаграми Вороного становить не більше $2n - 5$, а кількість ребер не більше $3n - 6$.

Одним зі способів побудувати діаграму Вороного є провести між кожною точкою на площині уявні прямі та серединні перпендикуляри до них. Точки перетину цих перпендикулярів і будуть вузлами діаграми Вороного, а кожна під площина обмежена цими перпендикулярами буде коміркою діаграми.

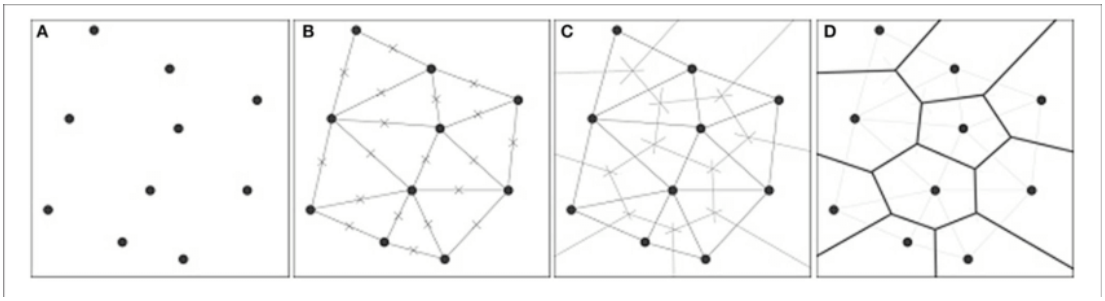

Рис. 4 Процес побудови через серединні перпендикуляри

2.2 Використання діаграм Вороного у розробці ігор

2.2.1 Представлення діаграми Вороного

Визначимо структуру діаграми Вороного у псевдокоді. Тут і далі будемо називати клітину діаграми регіоном.

Точка діаграми містить інформацію про її координати та посилання на регіон, в якому вона знаходиться:

```
class VoronoiPoint {
    float x
    float y
    VoronoiRegion* region
}
```

Кожний регіон, в свою чергу, містить посилання на свою точку та масив ребр регіону:

```
class VoronoiRegion {
    VoronoiPoint* point
    Edge *edges[]
}
```

Ребро діаграми містить інформацію про свої точки початку та кінця, їх координати та відстань між точками:

```
class VoronoiEdge {
    VoronoiPoint* pointA
    VoronoiPoint* pointB
    float distance
    float x1, z1, x2, z2
}
```

2.2.2 Пошук шляху до найближчого предмету

Один зі способів використання діаграми Вороного в відео-іграх – задача пошуку найближчого до гравця предмету. У заданому прикладі ми шукаємо найближчу аптечку. Кожна аптечка в такому випадку буде вершиною діаграми Вороного.

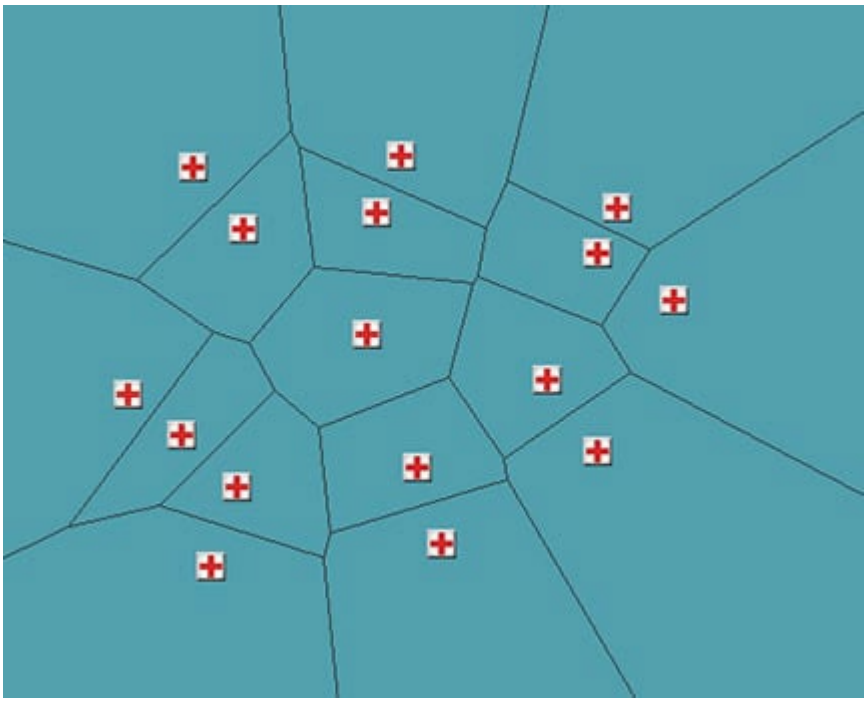

Рис. 5 Аптечки як вершини діаграми

Перш за все необхідно визначити в якому регіоні знаходиться гравець. Діаграма Вороного не надає ефективного способу з'ясувати це. Однак ви можете зберігати посилання на кожен регіон у квадродереві або R-дереві,

тоді маючи посилання на свій регіон, ми можемо знайти його сусідів та їхніх сусідів.

Якщо в регіоні гравця немає аптечки, потрібно знайти найближчий регіон з аптечкою. З нашої структури діаграми Вороного ми можемо дізнатися її ребра, а відповідно і найближчих сусідів, що також містять ці ребра. Саме в цих сусідніх регіонах і потрібно шукати аптечку.

Тут також можна використовувати тріангуляцію Делоне. Ребра тріангуляції будуть позначати шлях між парами аптечок. Оскільки діаграма Вороного це також і граф, можна застосувати алгоритм пошуку A*, щоб знайти шлях до найближчої з аптечок[5].

2.2.3 Пошук найбезпечнішого маршруту

Уявімо, що точки діаграми Вороного це місця розташування ворогів. Гравцю потрібно пройти найбезпечнішим маршрутом – тобто таким, що як проходить як можна далі від ворогів.

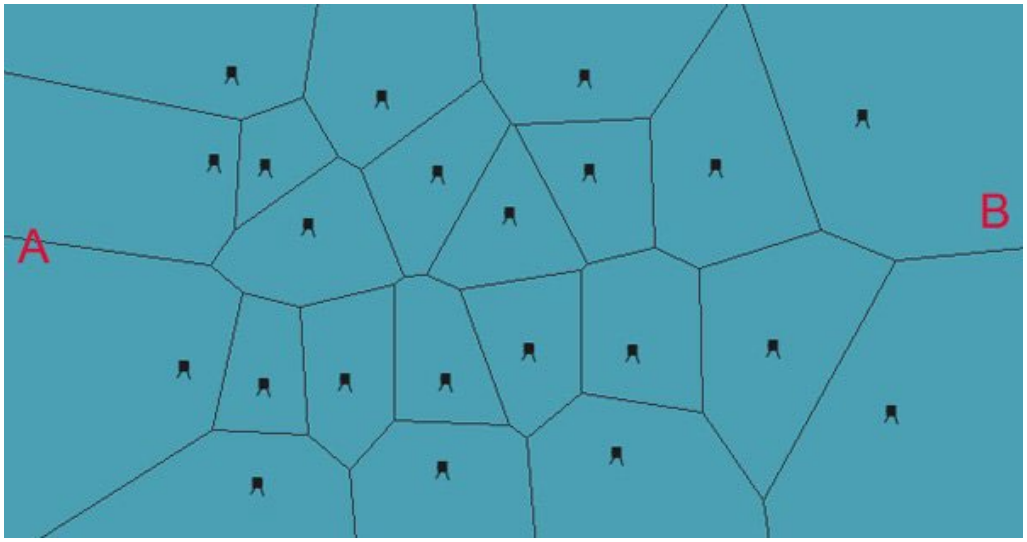

Рис. 6 Місця розташування ворогів

Можна знову застосувати алгоритм A*, щоб визначити вагу кожного ребра діаграми. В даному випадку ми шукаємо найважчі ділянки, тобто такі, що знаходяться найдалі від ворогів. В нашій структурі вже закладена інформація про відстань від ребр до точок.

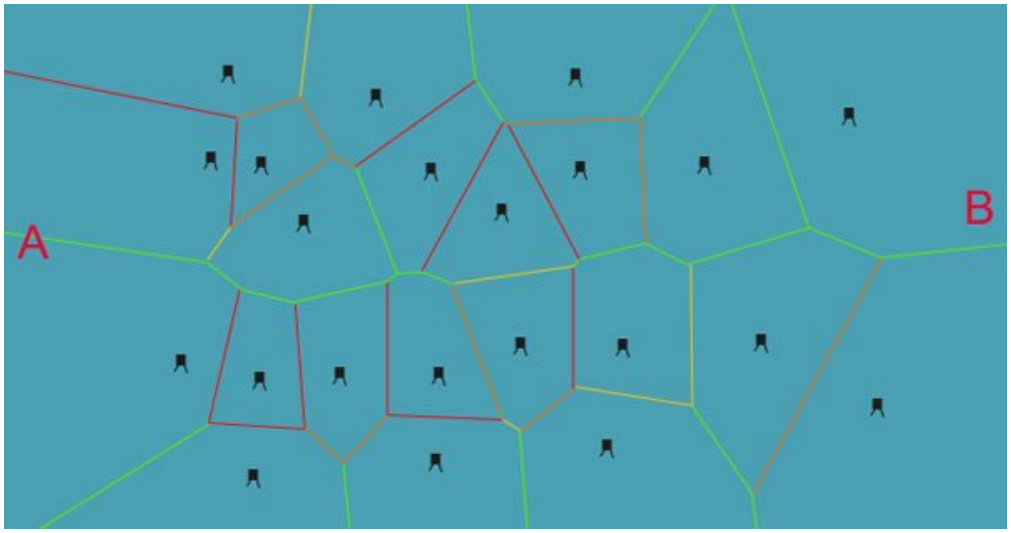

Рис. 7 Діаграма з вагою кожного ребра

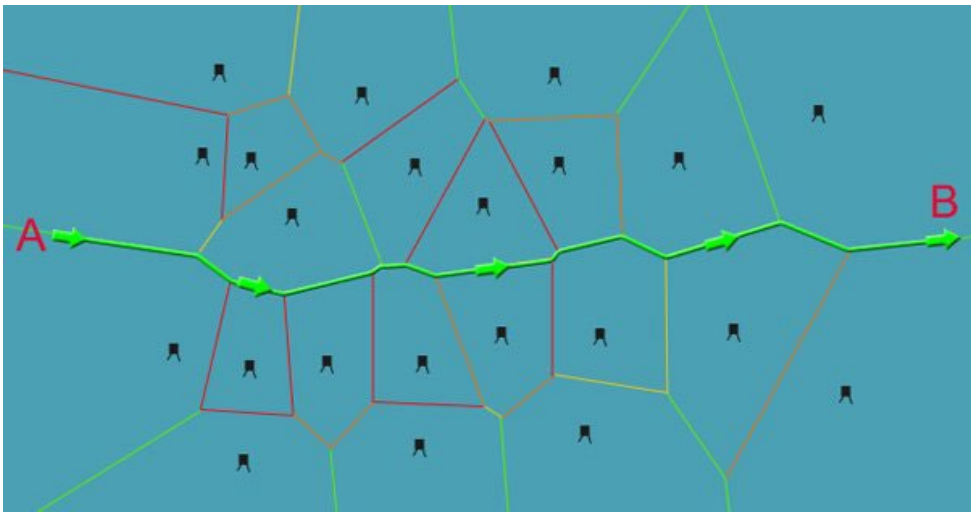

Рис. 8 Фінальний оптимальний шлях

2.2.4 Згладжування кутів рельєфу

Інше застосування діаграм Вороного – згладжування гострих кутів на рельєфі ігрового світу. Спочатку рельєф грубо формують за допомогою квадратів, що представляють траву, землю тощо.

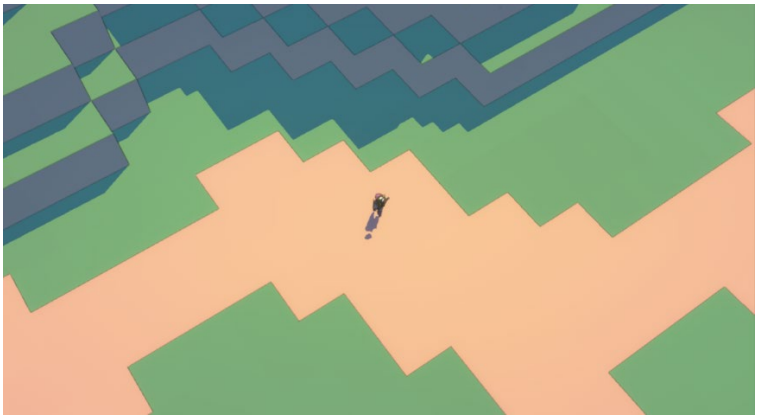

Рис. 9 Початковий рельєф

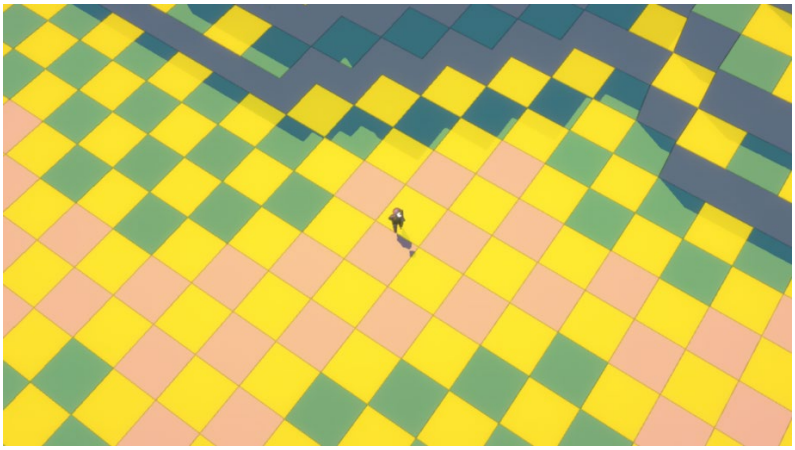

Рис. 10 Початковий рельєф з виділеними квадратами

Далі утворюють діаграму Вороного, кількість точок якої відповідає кількості квадратів рельєфу.

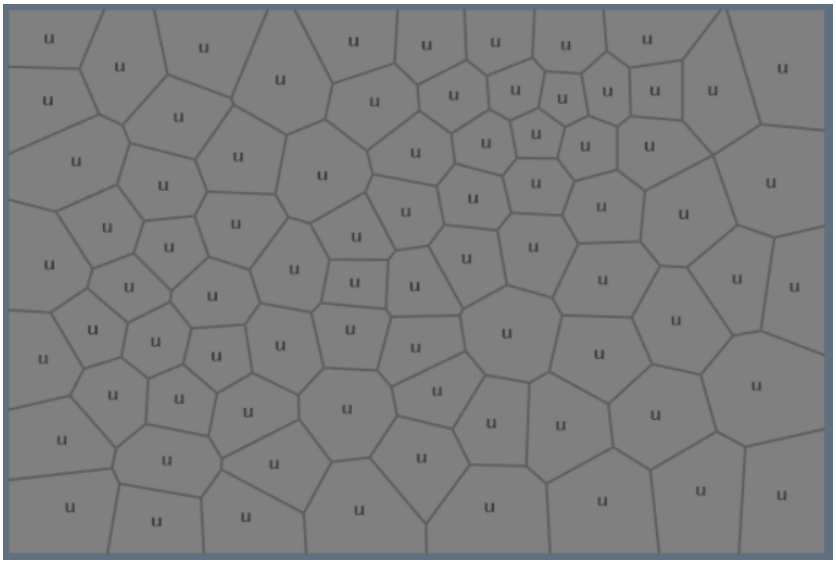

Рис. 11 Діаграма Вороного

Після цього рельєф співставляють з діаграмою Вороного так, щоб кожному квадрату поставити у відповідність одну клітину діаграми Вороного. Квадрати на рельєфі змінять форму на форму відповідної їм клітини діаграми Вороного.

Рис. 12 Змінений рельєф

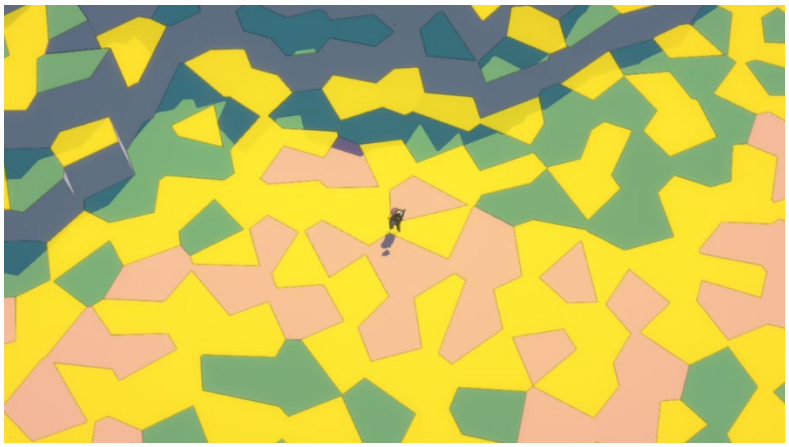

Рис. 13 Змінений рельєф з виділеними клітинами

Таке застосування діаграм Вороного є проміжним етапом в задачі надання рельєфу більш природнього вигляду. Далі можна застосувати так звану «релаксацію» діаграми. Для цього центри клітин зміщують певним чином навколо закруглених кутів в діаграмі (див. Рис 14).

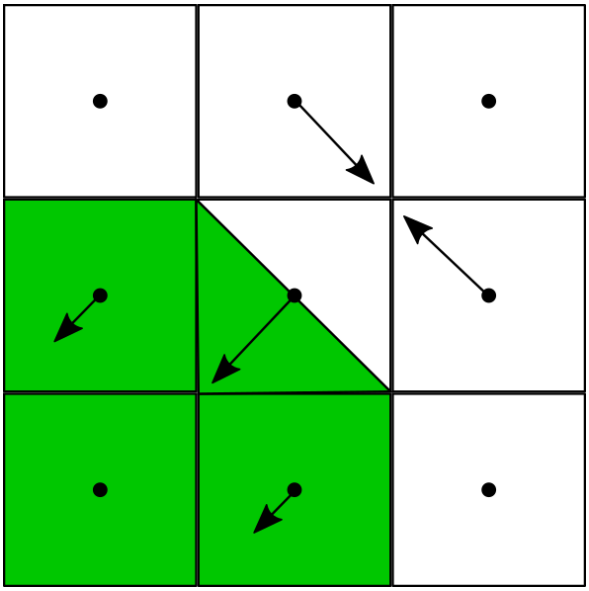

Рис. 14 Принцип зміщення вершин діаграми

Після зміщення вершин результат виглядає ще більш природнім.

Рис. 15 Результат після зміщення вершин

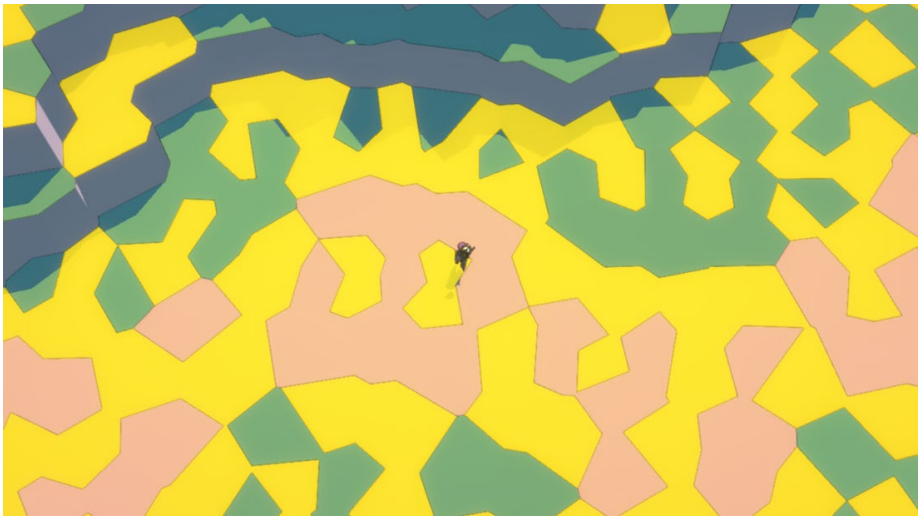

Рис. 16 Результат після зміщення вершин з виділеними клітинами

Наступним кроком є округлення гострих кутів діаграми[6].

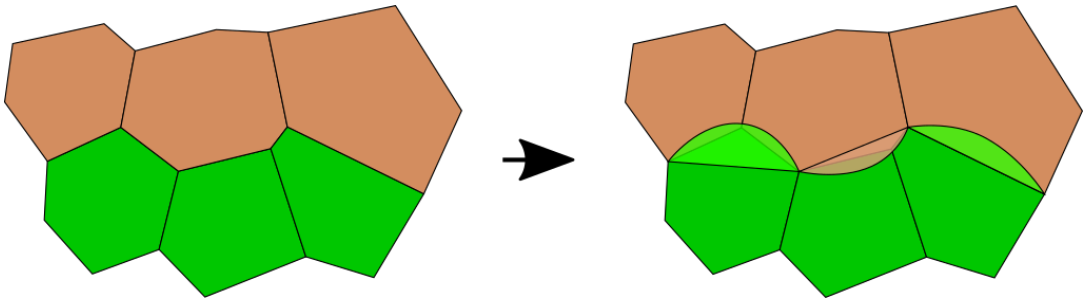

Рис. 17 Приклад округлення кутів

Рис. 18 Остаточний варіант

2.2.5 Побудова реалістичної мапи

Процес побудови ігрової мапи починається з побудови випадкової діаграми Вороного.

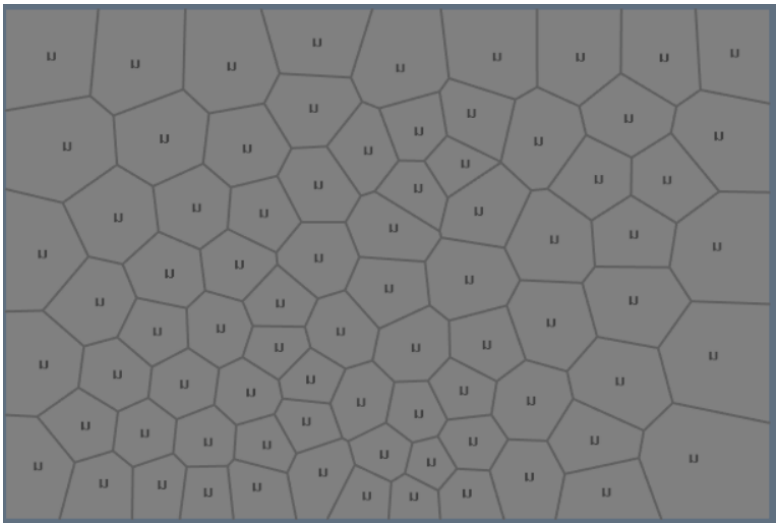

Рис. 19 Діаграма Вороного

Клітини діаграми Вороного спочатку вручну замальовують: синім колір позначають океан, сірим – гори. На карті також є умовний початок А та умовний кінець Б – вони потрібні для того, щоб далі будувати карту за певними правилами, а також позначають початок історії героя та його кінцеву мету.

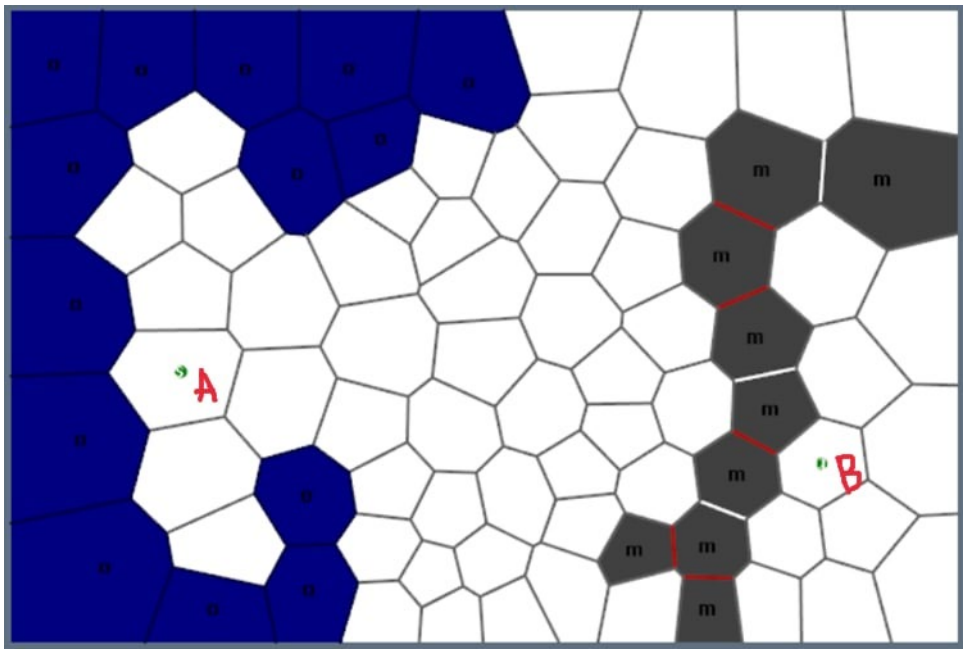

Рис. 20 Зафарбована діаграма Вороного з умовним стартом А та фінішем Б

При цьому для клітин вводять правила. Так, кожна «пуста» клітина біля гірської перетворюється на гірську, якщо вона відповідає певним умовам. В даному випадку така клітина має бути більш як за 5 клітин від старту, менш

як за 4 клітини від фінішу, має мати не більше однієї сусідської гірської клітини та відстань до старту менша ніж загальна відстань від старту до фінішу.

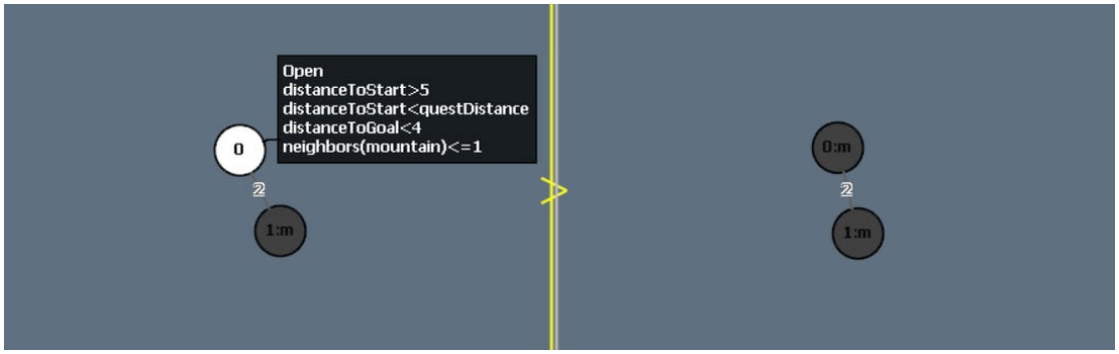

Рис. 21 Правило за яким клітина перетворюється на гірську

Також вручну зафарбовують інші регіони, що складають інтерес для гравця.

Наступним кроком на мапу добавляють річки. Це також описується правилами: наприклад, річка буде починатися з океану, проходити переважно між границями регіонів,

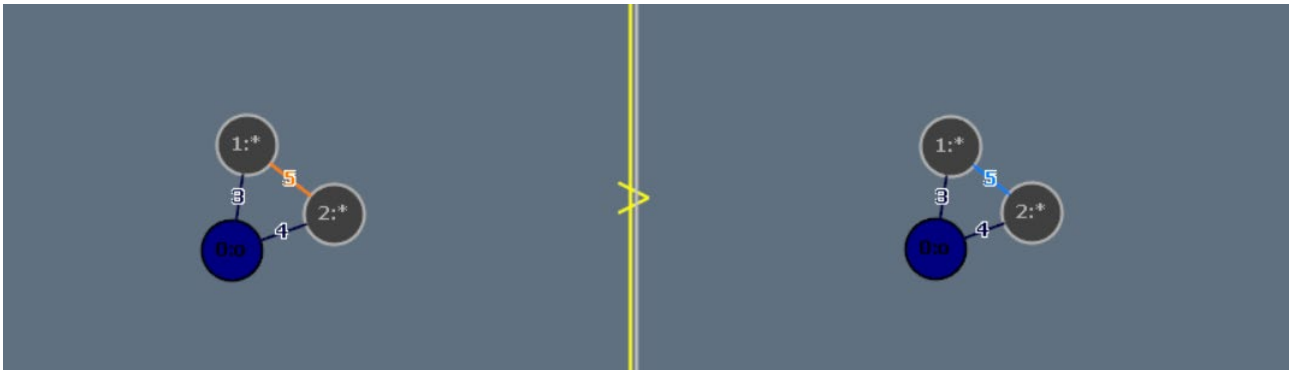

Рис. 22 Правило за яким річка починається з океану

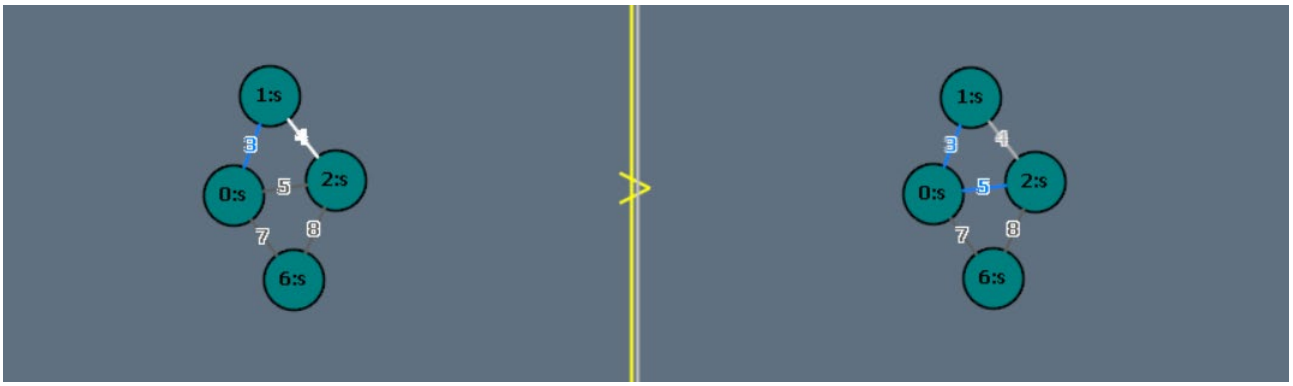

Рис. 23 Приклад правила формування річки в регіоні s

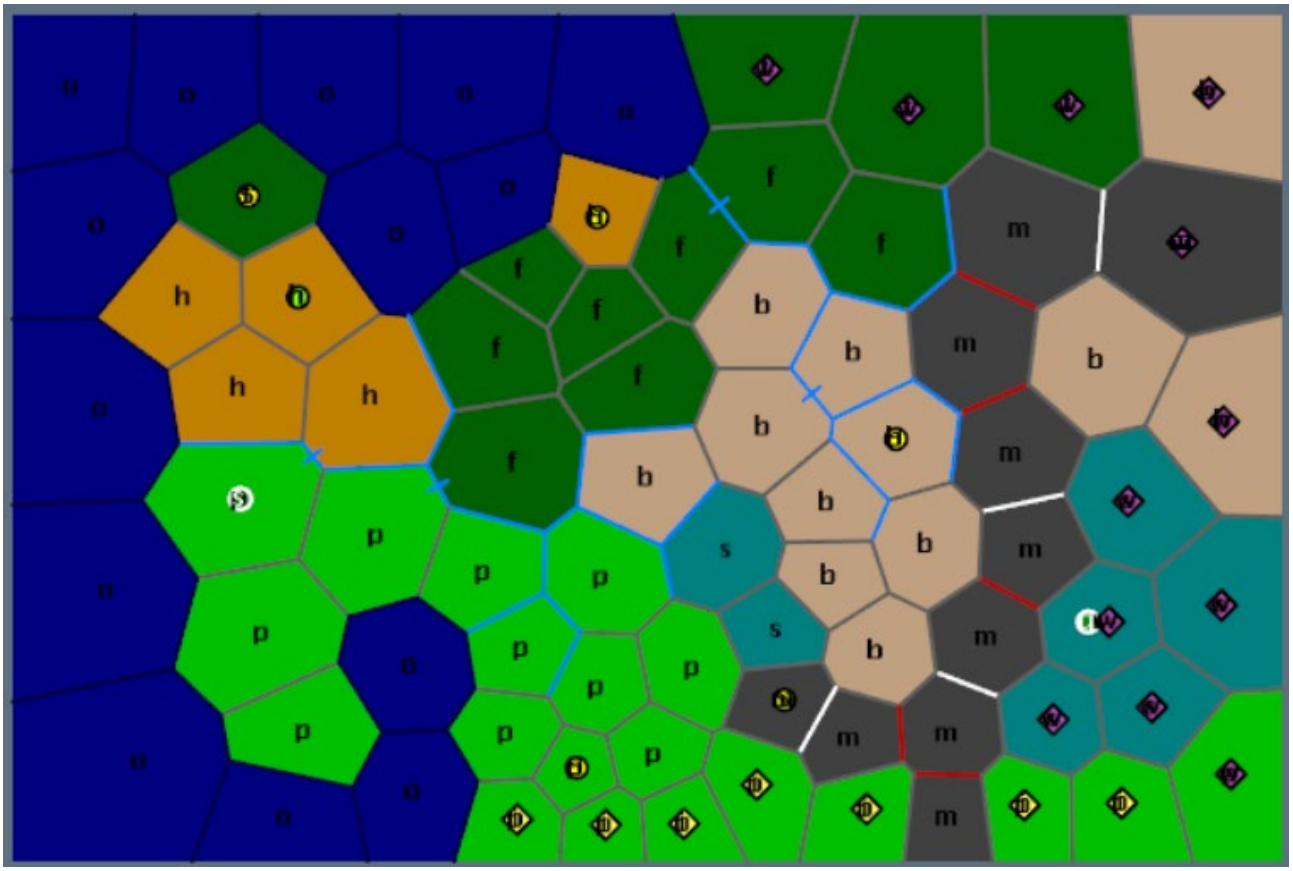

Рис. 24 Карта з зафарбованими регіонами, побудованими річками та гірськими масивами

Діаграма Вороного використовується в цьому прикладі, тому що має природній вигляд та вся мапа може бути представлена у вигляді графу, де кожен вузол графу це певний регіон. Це полегшує подальшу роботу з мапою та дозволяє визначити складність певних регіонів, які вороги на ньому зустрічатимуться тощо, порахувати оптимальні шляхи між клітинами тощо[7].

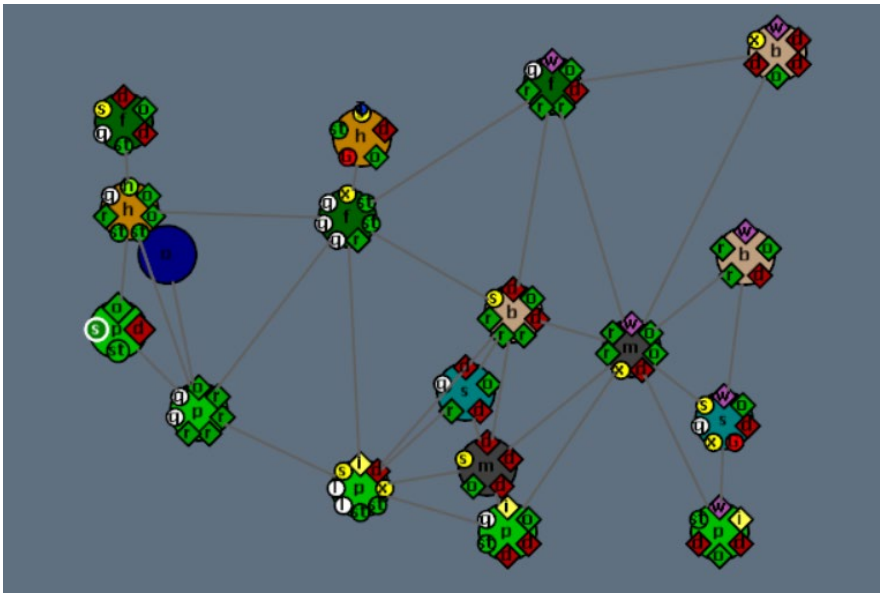

Рис. 25 Мапа, представлена у вигляді графу

Після роботи художників мапа може мати такий цілком реалістичний вигляд:

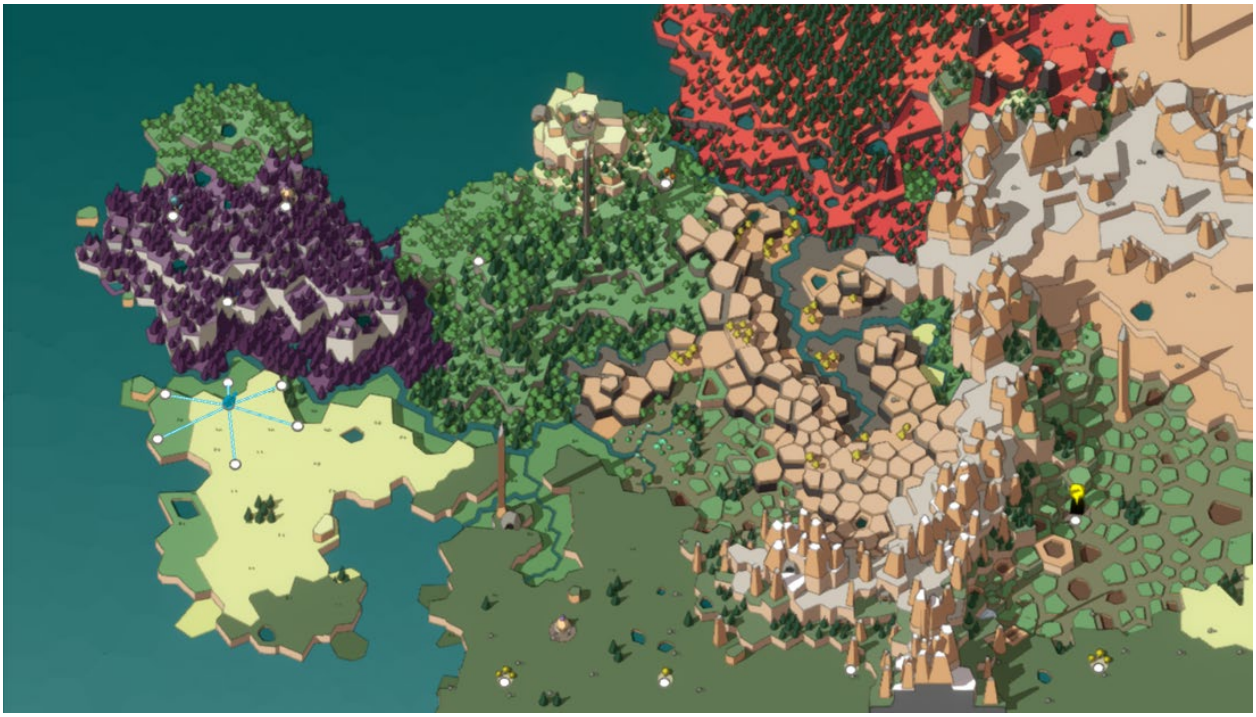

Рис. 26 Остаточний вигляд мапи

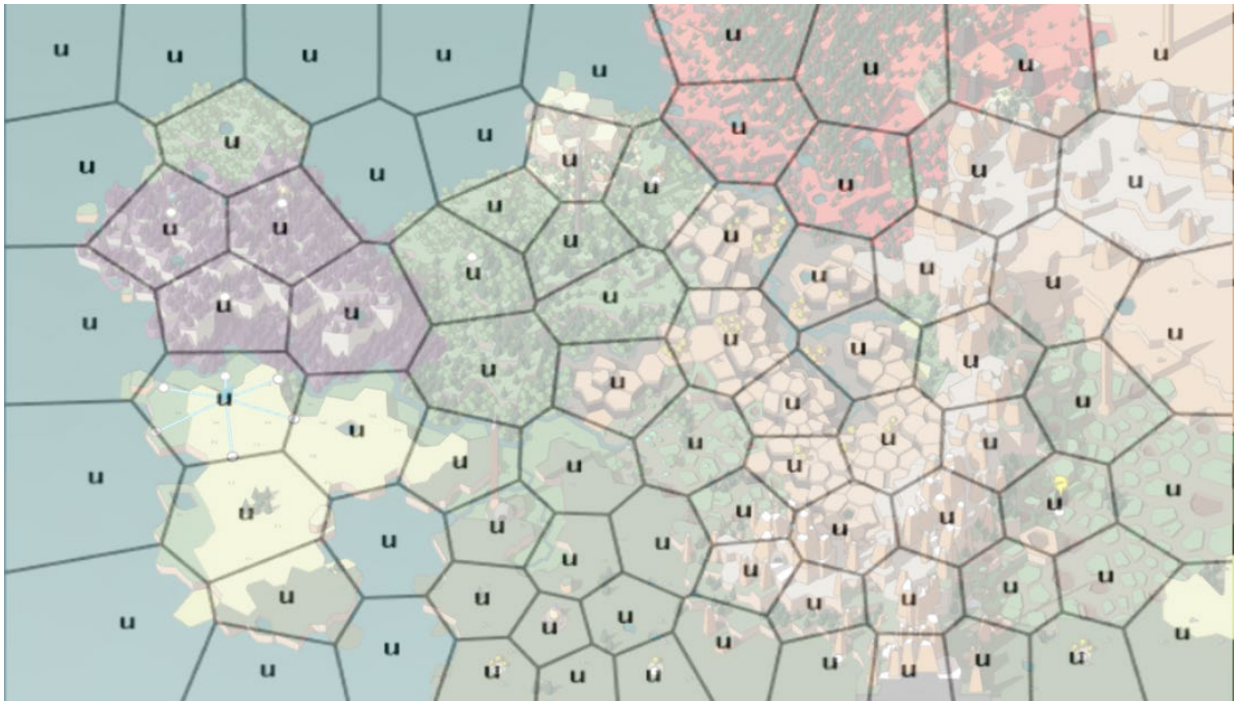

Рис. 27 Остаточний вигляд мапи з накладеною на неї вихідною діаграмою

3. Розробка гри

3.1 Обрані інструменти та алгоритми

3.1.1 Особливості Unity

Для реалізації роботи було обрано ігровий двигун Unity на мові C#.

В Unity простір, де відбувається гра, представлений у вигляді сцени (Scene), а всі об’єкти в ній – у вигляді екземплярів класу GameObject. Всі ігрові об’єкти мають стандартні властивості: розмір, позиція в світі, текстури, колайдери тощо. Також є можливість додавати додаткові властивості, наприклад, Rigidbody, що дозволяє надавати об’єкту фізичні властивості (масу, швидкість, тертя тощо); скріпти, що визначають поведінку об’єкта. Інтерфейс дозволяє користувачу безпосередньо взаємодіяти з об’єктами: переміщувати їх по ігровому простору, змінювати їх розміри тощо.

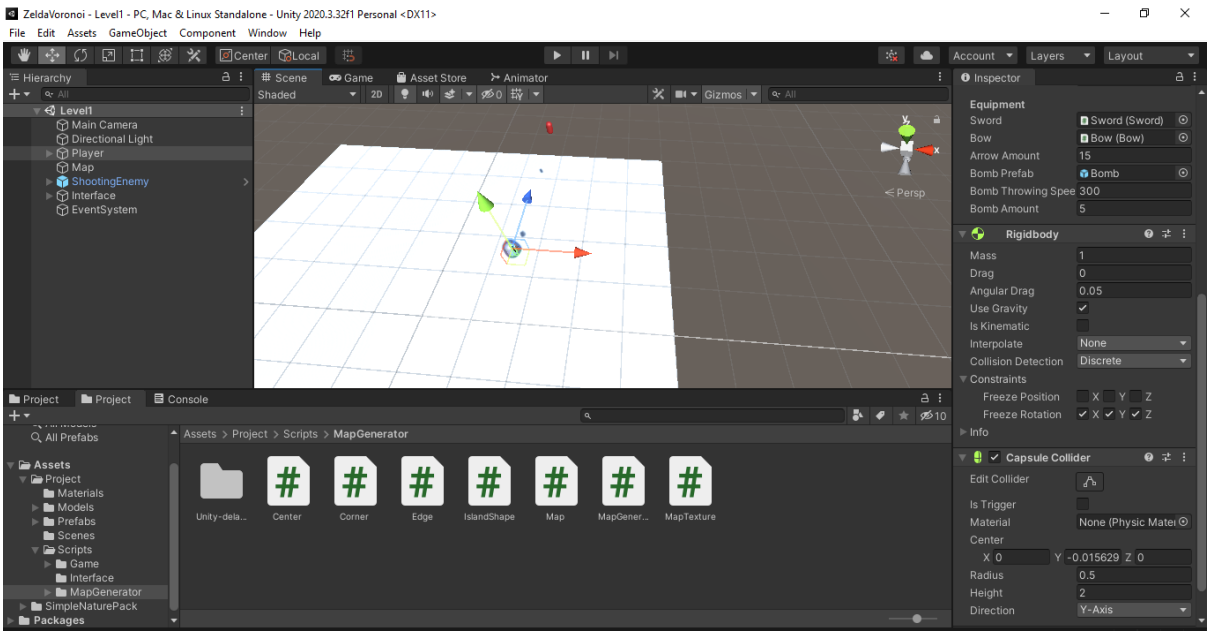

Рис. 28 Інтерфейс Unity

Ще однією особливістю рушія є наявність так званих префабів (Prefab) – батьківських ігрових об’єктів, які можна копіювати у дочірні ігрові об’єкти, що наслідуватимуть всі властивості батьківської копії. Таким чином при наявності великої кількості однакових об’єктів

(наприклад монет, які гравець повинен збирати) та потребі змінити їхні властивості достатньо змінити необхідну властивість лише у батьківському об’єкті і внесені зміни застосуються до всіх дочірніх об’єктів.

3.1.2 Алгоритм побудови світу для гри

Першим кроком алгоритму[8] побудови мапи на основі діаграми Вороного є власне створення самої діаграми Вороного. Для цього обирається задана кількість довільно розташованих точок, на основі яких і будується діаграма. Наступним кроком є застосування так званого алгоритму релаксації діаграми – алгоритм Ллойда[9]. Він полягає у переміщенні кожної точки діаграми Вороного до центроїди її клітини, що може повторюватися потрібну кількість разів.

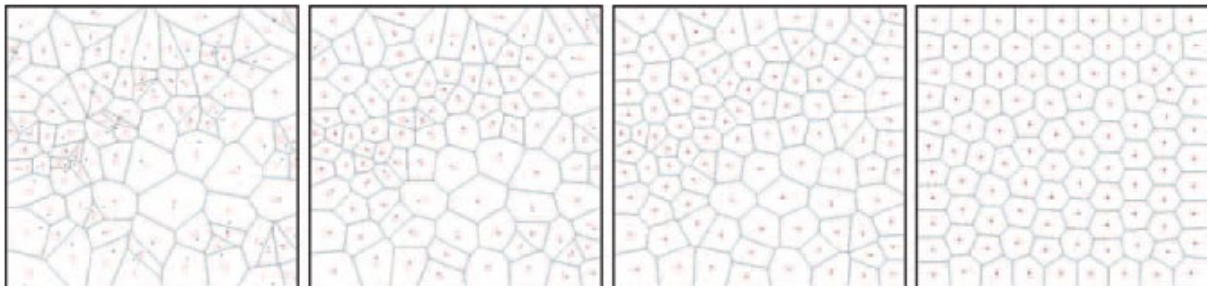

Рис. 29 Приклад діаграми Вороного до та після алгоритму Ллойда

Для створення даіграми обрано бібліотеку Unity-delaunay [10], що реалізовує тріангуляцію Делоне та, власне, діаграми Вороного.

Наступним кроком є визначення «висоти» кожної клітини діаграми [11] – так, клітини, що знаходяться вище будуть представляти землю чи гори, нижчі клітини будуть водоймами. По краям карти висота має значення 0, далі до центру висота збільшується. Це дозволяє карті мати вигляд острову, також можливе утворення водойм, оточених землею - озер. Під кінець цього етапу алгоритму ми маємо діаграму, де клітини, що мають висоту 0 є водоймами, решта клітин не є водоймами.

Наступним кроком є визначення типу водойми. Кожна клітина води, що пов’язана з краєм карти та не є грунтом, стає океаном. Клітини води, що оточені грунтом стають озерами.

Далі алгоритм визначає клітини берегу – ті, що мають хоча б одного сусіда клітину вода та хоча б одного сусіда клітину землі.

Наступним кроком є перерозподіл висоти клітин, щоб нижча висота була переважною – це зменшує кількість гір на мапі.

За значеннями висоти алгоритм рахує, де на мапі є схили. В нижчих точках між схилами створюються річки, що прямують в сторону клітин океану.

Ще однією властивістю клітин мапи є вологість, по ній в подальшому визначатиметься тип біому. Вологість підвищують джерела прісної води – річки та озера.

Останнім кроком алгоритму є визначення типу біому за його вологістю та висотою: менш вологі ділянки будуть пустелями, більш вологі – тропічними лісами, вищі ділянки будуть горами і так далі.

```
if (p.ocean) return Biome.Ocean;
if (p.water)
{
    if (p.elevation < 0.1) return Biome.Marsh;
    if (p.elevation > 0.8) return Biome.Ice;
    return Biome.Lake;
}

if (p.coast) return Biome.Beach;
if (p.elevation > 0.8)
{
    if (p.moisture > 0.50) return Biome.Snow;
    if (p.moisture > 0.33) return Biome.Tundra;
    if (p.moisture > 0.16) return Biome.Bare;
    return Biome.Scorched;
}
```

Рис. 30 Принцип визначення типу біому

Залишається лише заповнити всі клітини кольором під тип біому [12]. В результаті отримуємо мапу поділену на кольорові сектори.

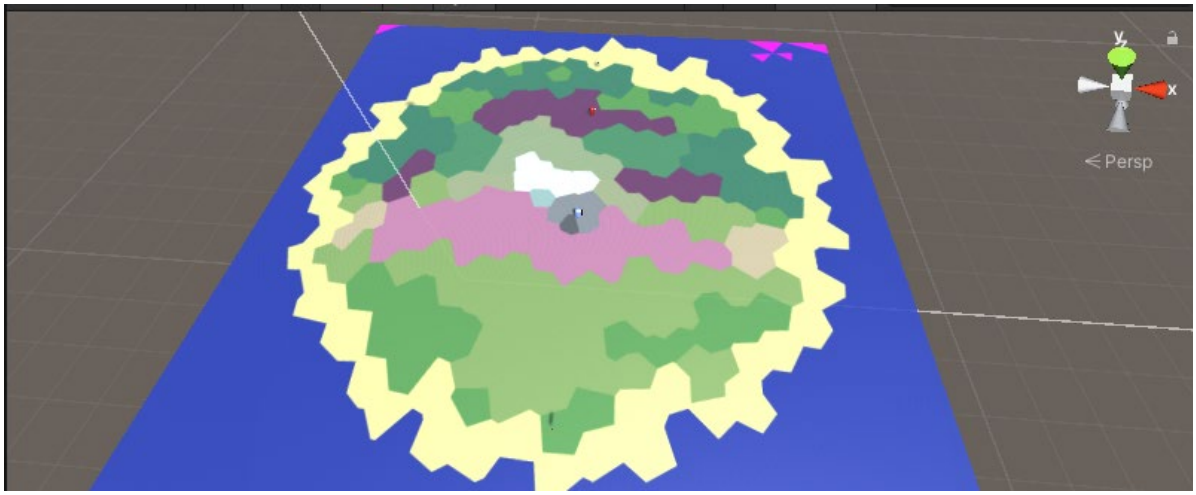

Рис. 31 Результат роботи алгоритму.

Алгоритм можна доповнювати під потреби розробника, наприклад реалізувати автоматичне розставлення у відповідних біомах об’єктів по типу дерев, трави, каміння тощо.

3.1.3 Створення основних ігрових об’єктів

В грі є об’єкт гравця – екземпляр класу Player, об’єкти ворогів – екземпляри класу Enemy.

Класи містять методи, необхідні для керування об’єктами; також властивості, такі як: кількість очків здоров’я, кількість зброї, швидкість тощо.

```
private void Jump()
{
    _playerRigidbody.velocity = new Vector3(
        _playerRigidbody.velocity.x,
        y: jumpVelocity,
        _playerRigidbody.velocity.z
    );
}
```

Рис. 32 Приклад методу, що змінює положення об’єкту в ігровому світі

```
public class Player : MonoBehaviour
{
    [Header("Properties")] public int health = 10;  Unchanged
    [Header("Visual")] public GameObject model;  Model
    [Header("Movement")] public float rotationSpeed = 1f;  "4"
    public float movementVelocity = 5f;  Unchanged
    public float jumpVelocity = 5f;  "4"
    public float knockbackForce = 5f;  "100"
    [Header("Equipment")] public Sword sword;  Changed in 1 asset
    public Bow bow;  Changed in 1 asset
    public int arrowAmount = 15;  Unchanged
    public GameObject bombPrefab;  Bomb
    public float bombThrowingSpeed = 5f;  "300"
    public int bombAmount = 5;  Unchanged

    private float _knockbackTimer;

    private Rigidbody _playerRigidbody;

    private Quaternion _targetModelRotation;

    private bool _canJump = false;
```

Рис. 33 Властивості гравця

Гравець та вороги можуть використовувати зброю та наносити шкоду один одному.

3.2 Результати

В результаті створено гру з базовими можливостями для керування об’єктом гравця та ворогами, що можуть наносити гравцю шкоду. Гра закінчується, коли у гравця не залишається здоров’я. Реалізовано простий інтерфейс, що повідомляє усю основну інформацію про статус гравця. Мапа генерується за допомогою алгоритму описаного вище.

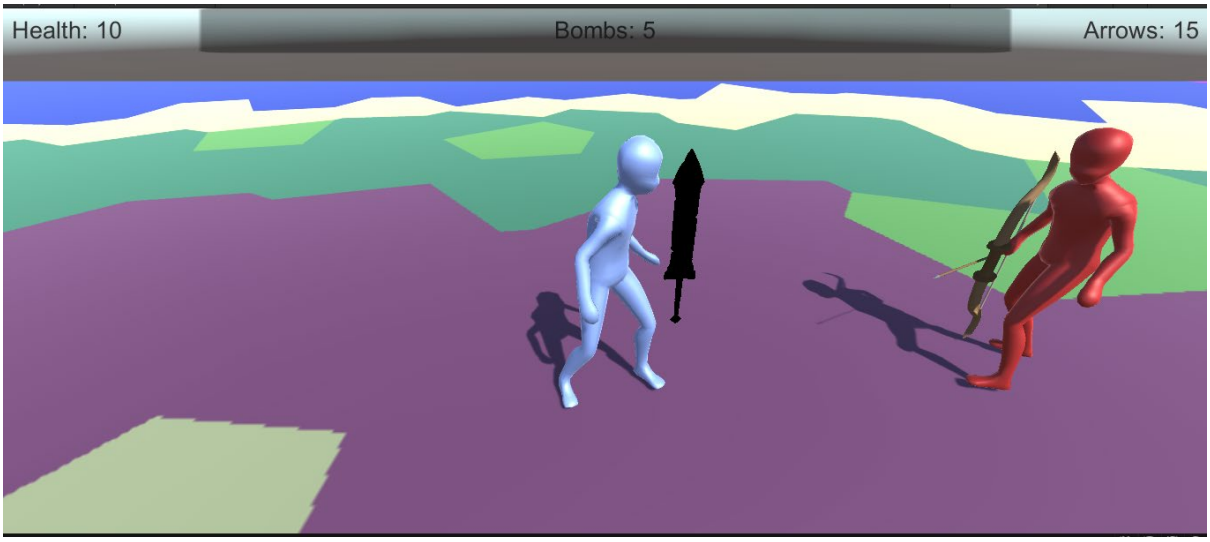

Рис. 34 Приклад рівня гри

Висновок

Реалізовано гру на ігровому рушію Unity з використанням діаграм Вороного для побудови мапи ігрового світу.

Список прийнятих скорочень

AR – Augmented Reality

VR – Virtual Reality

UE – Unreal Engine

Список використаних джерел

[1] https://www.wheels.org/spacewar/creative/SpacewarOrigin.html

[2] https://www.museumofplay.org/video_games/

[3] https://techcrunch.com/2019/10/17/how-unity-built-the-worlds-most-popular-game-engine/

[4] https://www.perforce.com/blog/vcs/most-popular-game-engines

[5] https://gamedevelopment.tutsplus.com/tutorials/how-to-use-voronoi-diagrams-to-control-ai--gamedev-11778

[6] https://www.gamedev.net/articles/programming/general-and-gameplay-programming/tiles-to-curves-fun-with-voronoi-graphs-part-1-r5150/

[7] https://www.gamedev.net/articles/programming/general-and-gameplay-programming/generating-world-maps-fun-with-voronoi-graphs-part-2-r5155/

[8] http://www-cs-students.stanford.edu/~amitp/game-programming/polygon-map-generation/

[9] https://www.jasondavies.com/lloyd/

[10] https://github.com/jceipek/Unity-delaunay

[11] https://github.com/staff0rd/polygon-map-unity

[12] http://alienryderflex.com/polygon_fill/