Міністерство освіти і науки України

НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ «КИЄВО-МОГИЛЯНСЬКА АКАДЕМІЯ»

Факультет інформатики

Кафедра мультимедійних систем

Кваліфікаційна робота

освітній ступінь – бакалавр

на тему: **«РОЗРОБКА ГРИ НА UNREAL ENGINE З ВИКОРИСТАННЯМ ШТУЧНОГО ІНТЕЛЕКТУ**

Виконав: студент 4-го року навчання,

Освітньої програми «Прикладна математика», 113

Гусар Кирило Юрійович

Керівник

Борозенний С.О.

Рецензент

_____

Кваліфікаційна робота захищена

з оцінкою

_____

Секретар ЕК

_____

«____» _____

20____ р.

Київ 2023

Календарний план виконання роботи

№ п/п	Назва етапу курсового проекту	Термін виконання	Примітка
1.	Отримання завдання на кваліфікаційну роботу	02.11.2022	
2.	Аналіз актуальності, дослідження аналогів	15.01.2023	
3.	Вивчення технологій для розробки системи	30.01.2023	
4.	Створення технічного завдання	10.02.2023	
5.	Написання практичної частини	10.04.2023	
6.	Користування застосунку	24.04.2023	
7.	Написання теоретичної частини	10.05.2023	
8.	Коригування теоретичної частини	18.05.2023	
9.	Захист кваліфікаційної роботи	28.05.2023	

Зміст

Вступ

Актуальність теми

Тема використання ШІ в іграх на зараз є дуже актуальною. Якісні ШІ дають можливість гравцям глибше зануритись, більше повірити у світ гри. Саме цікаві моделі поведінки юнітів створюють імерсивність, дають можливість значно покращити досвід гравців, а також забезпечити глибоку інтерактивність, реалістичність, та кинути певний виклик гравцям. Окрім цього, ШІ – один з основних джерел навантажень на процесор в сінглплеєрних іграх. Тому, розвиток в цьому напрямку, а саме, покращення(ускладнення) поведінки та оптимізація(зменшення навантаження на обчислювальні компоненти) є перспективним напрямком.

Мета роботи

Метою даної дипломної роботи є розробка гри в середовищі Unreal Engine з використанням штучного інтелекту, а також аналіз алгоритмів для процедурної генерації ландшафту. В якості штучного інтелекту буде використовуватись простий ШІ написаний на blueprint, від використання

машинного навчання довелось відмовитись в процесі розробки, через нестачу аргументів на користь таких моделей в порівнянні з простими ШІ. В тому числі, через складність реалізації моделей машинного навчання. На мою думку, в межах цього конкретного проекту немає змісту в застосуванні машинного навчання – реалізувати ШІ за допомогою традиційних методів значно простіше, а різниця між простим ШІ та цілою моделлю скоріше за все не буде відчутна.

Конкретно, за мету було взято створення гри в стилі сурвайвл гри. Карта буде генеруватись процедурно – за допомогою шумів Перліна, а ШІ буде реалізований для ворогів з якими гравець буде битись.

Завдання роботи

1. Огляд літератури щодо розробки ігор в середовищі Unreal Engine та використання штучного інтелекту в ігровому середовищі.
2. Аналіз вимог до гри та визначення основних функцій та фіч, які потрібно реалізувати.
3. Розробка архітектури гри та системи штучного інтелекту.
4. Розробка системи процедурної генерації ландшафту
5. Реалізація графічної складової гри в середовищі Unreal Engine.
6. Реалізація поведінки ворогів з використанням штучного інтелекту.

Об’єкт та предмет дослідження

Об'єктом дослідження є розробка гри в середовищі Unreal Engine з використанням штучного інтелекту. Включаються такі аспекти, як процедурна генерація світу, реалістична поведінка юнітів та інтерактивність геймплею.

Предметом дослідження є використання штучного інтелекту для створення інтелектуальних агентів у грі в середовищі Unreal Engine. Досліджуються алгоритми та методи, що дозволяють моделювати реалістичну поведінку тварин та забезпечувати взаємодію з гравцем.

Теоретичні основи розробки

Огляд ігрових рушіїв

Ігрові рушії є важливою складовою ігрової індустрії, що дозволяють розробникам створювати ігри, без необхідності розробки власного програмного забезпечення з нуля. Існує багато ігрових рушіїв, проте ми розглянемо найбільш популярні та актуальні на сьогоднішній день.

1. Unity: Цей рушій відомий своєю гнучкістю та широкими можливостями. Unity підтримує ряд платформ, включаючи PC, консолі, мобільні пристрої та VR/AR. Unity використовує мову програмування C# для створення скриптів і має велику кількість ресурсів для навчання.
2. Unreal Engine: Розроблений компанією Epic Games, Unreal Engine є одним з найпотужніших ігрових рушіїв на ринку. Він пропонує високоякісну графіку, реалістичне освітлення та інтегровану систему штучного інтелекту. Unreal Engine використовує мову програмування C++ і має візуальну систему скриптів, що називається Blueprints. Даний рушій є добрим вибором для розробки гри в жанрі survival з використанням ШІ.
3. Godot Engine: Це вільний та відкритий рушій з легкою та гнучкою архітектурою. Godot має власну мову програмування GDScript, але також підтримує C# і C++. Godot пропонує інтуїтивний інтерфейс та широкі можливості для розробки 2D та 3D ігор.
4. CryEngine: Розроблений компанією Crytek, CryEngine відрізняється відмінною графікою та реалістичними фізичними моделями. Цей рушій підходить для розробки вис оків графіки та фізики ігор, проте може мати більш високі системні вимоги для розробки та запуску ігор. CryEngine використовує мову програмування C++ та Lua для реалізації геймплею і скриптів.
5. Lumberyard: Розроблений компанією Amazon, Lumberyard базується на CryEngine, але має додаткові інтеграції з хмарними сервісами Amazon та іншими технологіями. Lumberyard пропонує візуальний редактор та підтримку мов програмування C++ та Lua. Цей движок також надає можливість використовувати Twitch API для розробки ігор, що співпрацюють зі стрімінговими платформами.

6. GameMaker Studio: Цей рушій спеціалізується на створенні 2D ігор та має простий візуальний інтерфейс. GameMaker Studio використовує власну мову програмування GML, яка проста для розуміння та навчання. Цей рушій підходить для інді-розробників, які хочуть швидко створити прототипи ігор або розробляти невеликі проекти.

Кожен з перерахованих ігрових рушіїв має свої переваги та недоліки. Вибір залежить від потреб розробника, жанру гри, платформи, на якій планується розгортання, та наявності ресурсів для навчання. В даній дипломній роботі було використано Unreal Engine.

Загальні відомості про розробку ігор на Unreal Engine

Unreal Engine є одним з найпопулярніших ігрових рушіїв на сьогоднішній день, що дозволяє розробникам створювати вражаючі ігри з високою якістю графіки та ефективною роботою на різних платформах. Створений компанією Epic Games, Unreal Engine почав свій шлях у 1998 році з виходом першої версії, яка використовувалась у грі Unreal. З тих пір рушій пройшов кілька ітерацій та суттєвих поліпшень, а вже четверта версія (Unreal Engine 4) стала однією з найбільш відомих та використовуваних у світі ігрових рушіїв.

Одним з ключових переваг Unreal Engine є його візуальна якість. Рушій володіє потужними засобами для рендерінгу графіки, що дозволяє створювати вражаючі візуальні ефекти та реалістичні сцени. Завдяки використанню передових технологій, таких як фізично-коректне освітлення, PBR (Physically-Based Rendering), та підтримка різних API для рендерінгу, Unreal Engine пропонує розробникам можливість створювати якісні та візуально привабливі ігри.

Ще одним важливим аспектом Unreal Engine є його гнучкість та масштабованість. Движок пропонує розширений набір інструментів для розробки ігор, які можуть задовольнити потреби як початківців, так і досвідчених розробників. Зокрема, він містить візуальний сценарій мову програмування Blueprint, яка дозволяє розробляти логіку гри безпосередньо в редакторі та без необхідності написання коду на C++. Водночас, професіоналам надається можливість програмувати на мові C++, що дає більше контролю та оптимізації над різними аспектами гри.

Unreal Engine має широку підтримку платформ, дозволяючи розробникам легко портувати свої ігри на різні операційні системи та пристрої. Це включає в себе настільні комп'ютери, консолі, мобільні пристрої та навіть пристрої віртуальної (VR), змішаної (MR) реальності.

Крім того, Unreal Engine має вбудовану підтримку фізики, що дозволяє розробникам легко інтегрувати реалістичну фізику в свої проекти. Рушій містить системи фізичних матеріалів, колізій та розподіл сил, що дає можливість створювати ігри з реалістичними рухами об'єктів та взаємодією з навколишнім середовищем.

Unreal Engine також має велику спільноту розробників, яка постійно розвивається та розширюється. Це означає, що ви можете знайти величезну кількість ресурсів, таких як документація, відеоуроки, форуми та статті, що допоможуть вам вивчити движок та розвиватися як розробник. Також, спільнота створює численні плагіни, розширення та готові активи, які можуть бути використані в вашому проекті, значно прискорюючи процес розробки.

Загалом, Unreal Engine є потужним та гнучким ігровим рушієм, який надає розробникам величезні можливості для створення вражаючих ігрових проектів. Від візуальної якості графіки до широкої підтримки платформ, рушій забезпечує все необхідне для успішної реалізації ігрових ідей. Не має значення, створюєте ви інді-проект або працюєте над великою комерційною грою, Unreal Engine пропонує набір інструментів та ресурсів, які допоможуть вам досягти ваших цілей.

Одним з основних компонентів успіху гри є її геймплей, а Unreal Engine пропонує ряд інструментів для його реалізації. Зокрема, система AI та навігації, яка включає в себе різні алгоритми та техніки для створення інтелектуальної поведінки персонажів, роботи з патрулюванням, переслідуванням та багато іншого.

Система анімації в Unreal Engine також є дуже потужною, надаючи розробникам можливість створювати рухи персонажів та об'єктів з високою точністю. Використовуючи систему анімації, можна створювати складні послідовності рухів, комбінувати анімації та навіть створювати процедурні анімації на основі фізики або AI.

Крім основних інструментів розробки, Unreal Engine також пропонує ряд допоміжних сервісів та ресурсів. Наприклад, ринок активів Unreal Engine (Unreal Marketplace) пропонує тисячі готових 3D-моделей, текстур, анімацій, звуків та інших ресурсів, які можуть бути використані в вашому

проекті. Це дозволяє значно економити час та ресурси, особливо для невеликих команд та інді-розробників.

Висновок: Unreal Engine є потужним ігровим рушієм, який надає широкий спектр можливостей для розробки ігор різних жанрів та рівнів складності. З його допомогою розробники можуть створювати вражаючі візуальні ефекти, реалістичні анімації, інтелектуальну поведінку персонажів та багато іншого. Доступність ресурсів, документації та активної спільноти розробників робить процес навчання та розвитку на Unreal Engine зручним та ефективним для різних категорій розробників.

Окрім стандартних інструментів розробки, Unreal Engine має відкриту архітектуру, що дозволяє інтегрувати сторонні бібліотеки та розширення. Це означає, що розробники можуть легко адаптувати движок під свої специфічні потреби та вимоги.

Завдяки своїй гнучкості та масштабованості, Unreal Engine використовується не тільки для створення ігор, але й для різноманітних застосунків, таких як віртуальна реальність, змішана реальність, архітектурна візуалізація, симуляції, реклама, навчальні проекти та навіть для створення сцен в фільмах.

У міру того, як ігрова індустрія продовжує розвиватися, розробники продовжують пошук нових технік та методів для створення ще більш захоплюючих ігрових досвідів. Використання передових технологій, таких як штучний інтелект, процедурна генерація контенту, реалістична фізика та інші, дозволяє створювати ігри, які перетворюють галузь та забезпечують незабутні враження для гравців. У цьому контексті Unreal Engine продовжує бути одним з найбільш впливових та важливих інструментів для розробки ігор та інших інтерактивних проектів.

Майбутнє Unreal Engine також виглядає міцним та інноваційним. Epic Games неухильно працює над поліпшенням та впровадженням нових технологій. Останнім глобальним релізом став Unreal Engine 5 де було введено безліч нових технологій, чимало зручних для розробників фіч а також відбулось явне покращення графічної складової (хоч і графіка Unreal Engine 4 ще недавно була еталоном та рахувалась однією з найбільш передових).

Одним з ключових нововведень Unreal Engine 5 є система Nanite Virtualized Geometry, яка дозволяє створювати деталізовані сцени з мільйонами полігонів без значного впливу на продуктивність. Це відкриває нові горизонти для розробки ігор, де розробники можуть

працювати з високодеталізованими об'єктами та сценами без страху втрати продуктивності.

Ще однією важливою інновацією є система Lumen, яка пропонує динамічне глобальне освітлення, що робить процес налаштування освітлення сцен значно простішим та більш реалістичним. Завдяки Lumen, розробники можуть створювати освітлення в реальному часі без необхідності перерахунку та зниження продуктивності.

З урахуванням нових можливостей, які пропонує Unreal Engine 5, та активної роботи Epic Games над покращенням існуючого движка, можна з упевненістю стверджувати, що Unreal Engine залишатиметься ключовим інструментом для розробки ігор та інтерактивних проектів у майбутньому. Завдяки своїй потужності, гнучкості та доступності, Unreal Engine продовжує приваблювати розробників по всьому світу і стимулює їх до створення незабутніх ігрових та інтерактивних вражень для гравців та користувачів.

Ураховуючи зростання віртуальної реальності (VR), доповненої реальності (AR) та змішаної реальності (MR) у різних сферах розваг, освіти, медицини та інших галузях, Unreal Engine також активно розвиває та впроваджує інструменти та підтримку цих технологій. Це забезпечує розробникам можливість створювати захоплюючі проекти з використанням передових технологій, розширюючи область ігрового досвіду за межі звичайних екранів.

Інтеграція з хмарними технологіями, допоміжними сервісами та платформами, такими як онлайн-магазини, соціальні мережі та ігрові сервіси, також є важливою частиною роботи над Unreal Engine. Це дозволяє розробникам легко розгортати, просувати та монетизувати свої ігри та проекти, забезпечуючи зручний доступ для кінцевих користувачів.

У підсумку, Unreal Engine є потужним, гнучким та доступним ігровим рушієм, який відкриває розробникам неймовірні можливості для створення вражаючих ігор та інтерактивних проектів. Зі своєю активною спільнотою, ресурсами та постійним розвитком, рушій продовжує задавати стандарти якості та інновацій у галузі розробки ігор. І хоча майбутнє ігрової індустрії непередбачуване, Unreal Engine забезпечує солідну основу для розробників, щоб продовжувати створювати захоплюючі ігрові досвіди та впливати на напрямок розвитку галузі.

Порівняння Unreal Engine з найближчими конкурентами

На даний момент Unreal Engine – один з провідних рушіїв в індустрії. Більша частина AAA проектів на ПК ринку розробляється за допомогою саме цього рушія. Зазвичай його порівнюють з найближчим конкурентом – Unity. Інші рушії не дуже підходять для цього порівняння: або використовуються в конкретних компаніях та не дуже популярні за їх межами, або якість та масштабність розроблених на них проектів не може навіть близько змагатись з Unreal. Звичайно, вибір рушія залежить від задач, але саме на Unity та Unreal Engine можна створювати будь-які проекти. Якщо ці рушії не дають достатню кількість інструментів для розробки гри, інші точно не зможуть задовольнити цю потребу.

З плюсів Unreal Engine варто зазначити:

- Швидка реалізація нових технологій. Unreal Engine 5 на зараз є мабуть найбільш технологічним рушієм. Студії масово відмовляються від власних рушіїв на користь Unreal Engine 5. Такі речі як глобальне динамічне освітлення (з використанням RT),нативне сприйняття та конвертація моделей з різних форматів (в тому числі сканованих об’єктів з реального світу), алгоритми для зменшення кількості полігонів нативно підтримуються рушієм.
- Швидкий та надійний фізичний рушій. Фізичний рушій Unity працює порівняно повільно, з ним виникають проблеми у великих проектах.
- Передові графічні можливості. Через це має позитивну репутацію серед гравців. Якщо гра розроблена на UE5 – вважається, що там має бути дуже красива та передова графіка (бо рушій це дозволяє).

Звісно, є і мінуси, а саме:

- Проблеми з документацією. Через швидкий ввід нових фіч та технологій документація та гайдлайни не встигають оновлюватись. Тому нерідко буває таке, що документації для нових компонентів не має і доводиться розбиратись самому. Особливо сильно це помітно в контрасті з Unity, де дуже розвинена система навчання, купа форумів, докладна документація та навіть онлайн курси.
- Використання c++ як основної мови в межах рушія. Як відомо c++ - мова низького/середнього рівня. Так, звісно, вона швидка і цілком може бути використана в даному контексті, але це сповільнює та

ускладнює розробку. З іншого боку, c++ дає можливість прямого втручання в роботу рушія, хоча фіча дуже специфічна.
- Мультиплатформенність. Unreal Engine підтримує ПК та нові консолі, в той час як Unity окрім консолей та ПК підтримує також мобільні пристрої.

Unreal Engine є потужним інструментом для розробки відеоігор, який надає широкі можливості для створення ігор різного жанру та складності. Використання Unreal Engine дозволяє розробникам ефективно втілювати свої ідеї та створювати захоплюючі ігрові досвіди для гравців.

Жанр Survival і його особливості

Жанр Survival (виживання) є одним з піджанрів екшн-ігор, де гравець опиняється в складних умовах та повинен використовувати доступні ресурси та уміння для виживання. Особливістю цього жанру є те, що гравці зазвичай стикаються з різними викликами та загрозами, такими як харчування, спрага, погода, хвороби, дика природа, а також інші гравці або комп'ютерні персонажі.

Основні характеристики ігор у жанрі Survival:

- Збір ресурсів: Гравці мають збирати ресурси з навколишнього середовища, такі як дерево, камінь, їжа та вода, для підтримки своєї життєдіяльності та створення засобів захисту.

- Крафтинг: Гравці мають використовувати зібрані ресурси для створення інструментів, зброї, одягу та інших предметів, які допоможуть їм вижити.

- Будівництво та укріплення: Гравці можуть будувати тимчасові або постійні споруди, щоб захиститися від небезпек, відпочити та зберігати зібрані ресурси.

- Боротьба з ворогами: Гравці змушені захищатися від агресивних комп'ютерних персонажів або інших гравців.

- Елементи випадковості та динамічні події: Ігри в жанрі Survival часто використовують випадкову генерацію карт або подій, що забезпечують унікальний досвід для кожного гравця.

- Прогресія персонажа: Гравці можуть розвивати своїх персонажів, вдосконалюючи їхні здібності і навички, що допомагають їм вижити у складних умовах.
- Система здоров'я та стаміни: Гравці повинні стежити за своїм здоров'ям та стаміною, оскільки вони впливають на можливості персонажа та його здатність вижити.

- Мультиплеєр: Багато ігор у жанрі Survival пропонують мультиплеєр, що дозволяє гравцям співпрацювати або змагатися один з одним.

Алгоритми і методи штучного інтелекту в грі

Основи штучного інтелекту

Штучний інтелект (ШІ) є галуззю комп'ютерних наук, яка займається створенням систем, здатних виконувати завдання, які зазвичай вимагають людського розуму. У контексті гри на Unreal Engine, ШІ може включати різні аспекти, такі як поведінка персонажів, рух, прийняття рішень та інтеракція з навколишнім середовищем. В даному розділі ми розглядаємо основні поняття штучного інтелекту та досліджуємо, як ШІ можна реалізувати в грі на Unreal Engine за допомогою Blueprint та C++.

Штучний інтелект може бути розділений на дві основні категорії: символічний ШІ та підхід на основі нейромереж. Символічний ШІ базується на використанні правил та експертних систем для реалізації інтелектуального поведінки, тоді як нейромережі - це математичні моделі, які намагаються імітувати роботу мозку. У нашому проекті ми зосередимося на символічному ШІ та реалізації його на Blueprint та C++.

Blueprint - це візуальна система програмування, яка дозволяє розробникам легко створювати ШІ, поведінку персонажів та інтерактивні елементи безпосередньо в Unreal Engine. Завдяки Blueprint, розробники можуть створювати скрипти для різних аспектів ШІ, таких як рух, прийняття рішень, комунікація між персонажами та адаптація до змін у навколишньому середовищі.

Математичні принципи роботи Шумів Перліна

Шум Перліна є фрактальним шумом, що створює неперервні, органічні текстури та переходи. Він був розроблений Кеном Перліном у 1983 році. Це один з найбільш вживаних шумів на сьогодні. Широко застосовується для процедурної генерації. Зазвичай, для початку генерується карта висот, а потім ця карта застосовується на потрібну місцевість.

Шуми не обов’язково використовуються поодинці. Нерідко кілька шумів накладаються один на одного створюючи більш складні і специфічні ландшафти. Це дозволяє зробити генерацію цікавішою та більш унікальною.

Основний принцип шуму Перліна полягає в генерації псевдовипадкових градієнтів в заданих точках, а потім інтерполяції значень між цими точками. Це досягається шляхом використання лінійної інтерполяції між сусідніми значеннями градієнтів, а також використання згладжуючої функції для усунення різких переходів між значеннями.

Загалом згладжуюча функція може бути будь-якою функцією. Проте, є певні вимоги до цієї функції. Для згладжування ми беремо тільки частину функції, а основною вимогою є нульова похідна в мінімальних та максимальних значеннях функції.

Для отримання значень шуму в проміжних точках застосовується інтерполяція між векторами градієнту. Зазвичай використовуються кубічна або косинусна інтерполяція. Це дозволяє плавно переходити від одного вектора до іншого і забезпечує плавність шуму.

Для створення шуму Перліна ми можемо використовувати наступну формулу:

$$P(x) = Lerp(G(a) * (x - a), G(b) * (b - x), Smooth((x - a) / (b - a)))$$

де:

P(x) - значення шуму Перліна в точці x;

Lerp(A, B, t) - лінійна інтерполяція між значеннями A та B з параметром t;

G(a) та G(b) - значення градієнтів в точках a та b відповідно;

Smooth(t) - згладжуюча функція для параметра t, зазвичай використовується функція, що враховує кривізну переходу, така як S-крива або косинусоїдальна інтерполяція.

Математичні методи роботи Diamond Square

Алгоритм Diamond-Square — це метод генерації випадкового ландшафту. Він використовує концепцію рекурсивного поділу для створення випадкового фрактального ландшафту.

Алгоритм поділяється на наступні етапи:

1. **Ініціалізація**: Спочатку ми створюємо двовимірний масив. Чотири кутові точки масиву задаються випадковими значеннями висоти.
2. **Етап "Diamond"**: Беремо квадрат, визначений чотирма кутовими точками, і обчислюємо середнє значення цих чотирьох точок, плюс додавання випадкового шуму. Це дає нам нову точку посередині квадрата, створюючи "алмаз".

   Формула для цього етапу:

   M = (TL + TR + BL + BR) / 4 + Random(-R, R)

   де:

   - **M** — середня точка (висота) квадрата.
   - **TL**, **TR**, **BL**, **BR** — верхній лівий, верхній правий, нижній лівий та нижній правий кутові точки квадрата відповідно.
   - **Random(-R, R)** — випадкове число в діапазоні [-R, R].
3. **Етап "Square"**: Для кожного "алмазу" з попереднього кроку, беремо кутові точки алмазу і середню точку, обчислену на попередньому етапі, щоб створити чотири нових квадрата. Для кожного квадрата обчислюємо середню точку точно так само, як ми це зробили на етапі "алмаз".

   Формула для цього етапу:

   M = (T + L + B + R + C) / 5 + Random(-R, R)

   де:

   - **M** — середня точка (висота) нового квадрата.
   - **T**, **L**, **B**, **R** — верхня, ліва, нижня, права точки відповідно.
   - **C** — центральна точка, що була обчислена на попередньому етапі.
   - **Random(-R, R)** — випадкове число в діапазоні [-R, R].

Ці етапи повторюються рекурсивно, доки всі точки масиву не будуть обчислені. Параметр **R** (шум) зазвичай зменшується з кожним новим кроком рекурсії, що забезпечує гладкий перехід висот.

Порівняння Diamond Square та шуму Перліна

Шум Перліна і Diamond-Square є двома популярними методами генерації процедурного контенту, особливо для синтезу ландшафту в комп'ютерних грах. Однак вони представляють два різних підходи до вирішення цієї задачі.

Шум Перліна:

1. *Неперервність*: Шум Перліна створює неперервні, гладкі шумові патерни, що робить його ідеальним для синтезу натуральних текстур, таких як хмари, вогонь, або ландшафт.
2. *Фрактальність*: Шум Перліна може використовуватися для створення фрактального шуму, шляхом шарування шуму на різних масштабах. Це дозволяє створювати більш складні та деталізовані текстури.
3. *Висока обчислювальна складність*: Генерація шуму Перліна може бути досить вимогливою до ресурсів, особливо для великих масштабів та високої деталізації.

Diamond-Square:

1. *Простота і швидкість*: Diamond-Square - це відносно простий алгоритм, який може швидко генерувати шум на великій області. Це робить його ідеальним для ситуацій, коли швидкість важливіша за детальність.
2. *Геометричний шум*: Diamond-Square створює шум, що базується на геометрії, що може призвести до зрозумілої геометрічної структури в результуючому ландшафту. Це може бути корисно для створення певних типів ландшафту, але також може виглядати нереалістично для інших сцен.
3. *Артефакти*: Алгоритм Diamond-Square може призвести до артефактів, таких як видимі квадрати або ромби, особливо при використанні низьких значень параметрів.

1. **Складність генерації шуму Перліна**:

Шум Перліна створюється шляхом інтерполяції багаточисельних шумових функцій, що визначаються градієнтами на регулярній сітці в n-вимірному просторі. Використовуючи n-вимірний простір, кожна операція обчислення шуму Перліна має складність O(n), де n - кількість вимірів. Однак в контексті генерації ландшафту, яка зазвичай використовує двовимірний або тривимірний шум Перліна, складність можна вважати константною (O(1)) для кожної операції генерації шуму.

2. **Складність алгоритму Diamond-Square**:

Diamond-Square є рекурсивним алгоритмом, який обробляє все більше і більше точок з кожним кроком рекурсії. Припустимо, що наша початкова сітка є квадратом зі стороною 2^n + 1. За кожний крок рекурсії алгоритм обробляє кілька точок, пропорційних кількості квадратів в сітці, що є пропорційною 4^(n-1). Тому, складність алгоритму Diamond-Square становить O(4^n) для квадратного поля розміром 2^n + 1.

Проте варто зауважити, що алгоритм Diamond-Square є стохастичним, тому його точна складність може варіюватися в залежності від випадкових значень, які генеруються під час виконання.

Отже, виходячи з цих даних, складність Diamond-Square є вищою ніж складність шуму Перліна. Однак, так як Diamond-Square є стохастичним, приймати рішення на основі тільки цих даних досить безглуздо. Також варто розуміти, що в той час як шум Перліна завжди генерує неперервну функцію, в випадку з Diamond-Square можна регулювати рівень деталізації, що дозволяє прискорити алгоритм.

Застосування алгоритмів у грі

У нашій грі про полювання, ми використовуємо шуми Перліна та Diamond-Square для реалізації генератора ландшафту.

Простими словами, шуми Перліна це двовимірна функція, яка генерує псевдо рандомні значення. В ігровій індустрії шуми Перліна використовуються майже для всього що генерується процедурно: ландшафт, хмари, рослинність… Вони дозволяють створювати більш природні та органічні форми, ніж традиційні генератори ландшафтів.

Для перетворення шуму в ландшафт ми робимо інтерполяцію значень (щоб не було різких перепадів висот) та підставляємо отримане значення в координатну вісь, в якості координати Z. Таким чином ми отримуємо плавний, унікальний ландшафт.

Також, нерідко використовується кілька шумів, причому вони накладаються один на один. Фактично, кожен наступний шум просто зміщує координату Z в певний бік. Так можна досягнути більшої деталізації, або отримати більш специфічний ландшафт, наприклад, гори або кратери.

Diamond-Square – більш складний для розуміння алгоритм. Замість функції все починається з 4 вершин-кутів квадрата. Потім, обчислюється середнє значення цих вершин з додаванням випадкового шуму, це висота точки по середині.

Далі все повторюється для кожного отриманого квадрату, стільки раз скільки це потрібно.

Розробка моделі поведінки ворогів за допомогою ШІ

Архітектура системи ШІ

В ході аналізу задачі було прийняте рішення написання ШІ з використанням Blueprints, а також мови програмування c++ як інструмент для розробки ШІ.

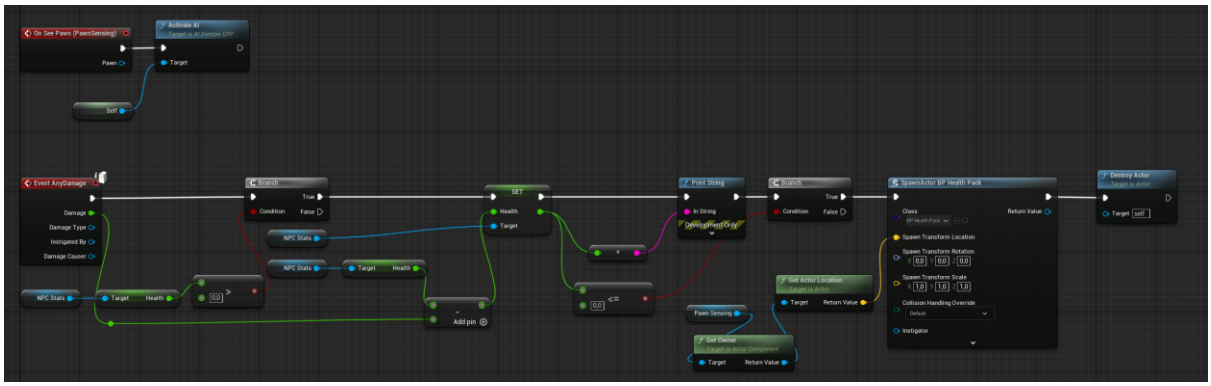
Unreal Engine надає інструмент під назвою Blueprint, що дозволяє розробникам створювати логіку гри без необхідності писати код на мові програмування. Blueprint використовує графічний підхід до програмування, де розробники можуть створювати логічні блоки (ноди) та з'єднувати їх між собою для визначення поведінки об'єктів у грі.

У контексті нашої гри, ми можемо використати Blueprint для реалізації простої логіки поведінки тварин, які бігають по карті. Наприклад, можна створити Blueprint для тварини, який містить логіку вибору цілі (гравця або інших об'єктів), руху по карті та реагування на зовнішні події. Використовуючи вбудовані ноди та функціональність Blueprint, ми можемо програмувати поведінку тваринок без необхідності вдаватись до складного коду.

Також, в Unreal Engine алгоритми на Blueprint нативно конвертуються в код C++ що дозволяє створити приблизну реалізацію логіки на Blueprint, а потім, там де потрібно, корегувати логіку вже на C++. Саме такий метод розробки і був використаний.

Опис поведінкових моделей ворогів

Блупрінт зомбі виглядає наступним чином:



Загалом, вороги чекають поки не побачать гравця, починають переслідування, при наближенні наносять шкоду гравцю. Коли гравець атакує ворога в нього зменшується кількість здоров’я, коли здоров’я

добігає 0 відбувається анімація смерті, модель знищується, а на місці смерті з’являється аптечка.

Розробка гри на Unreal Engine

Реалізація методів генерації ландшафту

В першому методі генерації Basic Generation використовується блюпринт BP_BasicProceduralGeneration і метод GenerateLandscape із нього.

В GenerateLandscape генеруються вертекси методом Generating Vertex, в який передається параметри кількості вертиксів і максимальної висоти для них.

В другому методі генерації Simple Perlin як і в методі Complex Perlin використовується метод GenerateLand, який приймає параметр useSimpleAlgo, від значення якого залежить яку реалізацію шума перліна ігра буде використовувати, просту чи більш складнішу.

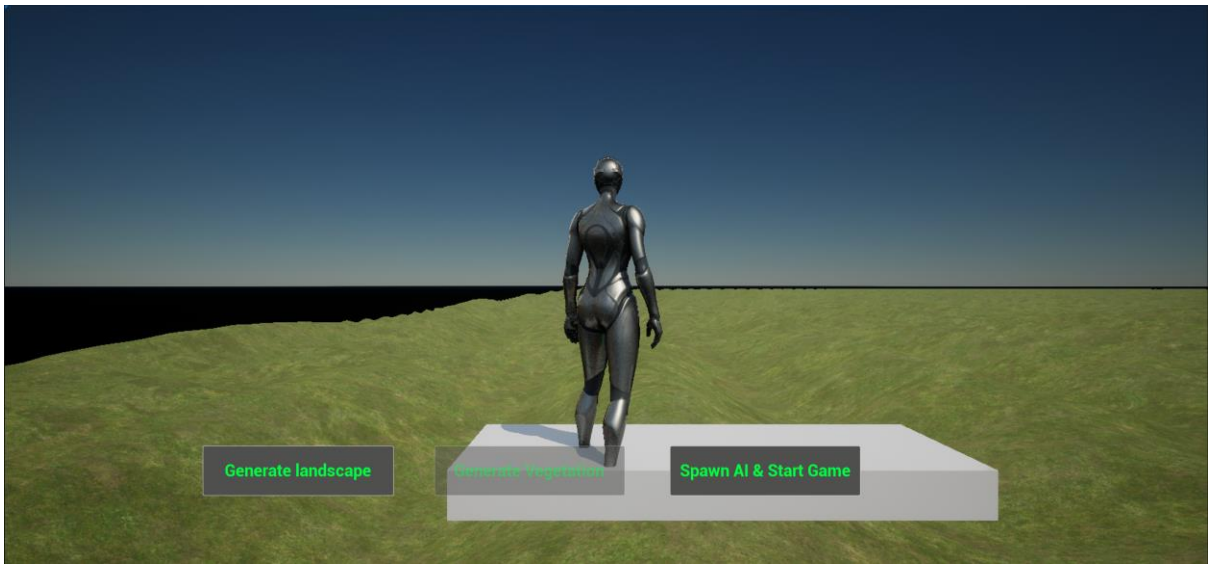
Спочатку очищаються 3 масива - вертексів, трикутників, які формуються на основі вертексів і UV мапи для накладання текстури на згенеровану 3D модель.

Використовується метод CreateVertices, в якому використовується метод генерації шума Перліна залежно від вибраної складності алгоритма PerlinNoise2D, або TwoD_Perlin_Noise, генерується масив вертексів і масив UV для накладання текстури.

Після генерації шума Перліна використовується метод CreateTriangles, який будує трикутники на основі створених вертексів.

Розробка графічного інтерфейсу

Готовний інтерфейс гри виглядає наступним чином:



Generate landscape – згенерувати ландшафт. Кожен згенерований ландшафт буде унікальним, так як використовується випадкове зміщення.

Generate Vegetation – згенерувати рослинність.

Spawn AI & Start Game – кнопка для початку гри.

На карті для тестування генерації ландшафту реалізований наступний інтерфейс:

Кожна кнопка відповідає за генерацію своїм методом. А значення параметрів відповідають значенням відповідних параметрів генераторів. Серед них:

- XSize, YSize, Polygon Size – відповідають за розміри ландшафту.
- Z Multipliter, Landscape Height – за висоту ландшафту
- Noise scale – розмір шуму
- Scale, UV scale – розмір сітки
- Vertex Number – кількість вершин.

Створення моделей, текстур та анімацій

Як вже було описано, Unreal Engine має власний магазин асетів де можна купити текстури, моделі, матеріали ітд. Всі асети використані в цій роботі я брав саме там.

Реалізація геймплею та логіки гри

Весь геймплей реалізований в сцені «NewWorld». Гравець генерує карту за допомогою спеціальної кнопки, запускає гру та йде битись з зомбі. В гравця є наступні параметри: голод, спрага, витривалість та здоров'я. Від ударів ворогів здоров'я зменшується.

Є шанс випадіння з ворогів аптечки при підбиранні якої здоров'я відновлюється.

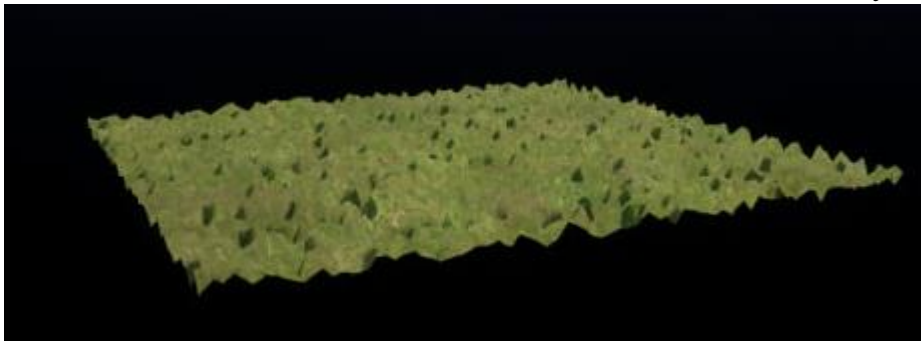
Аналіз отриманих результатів

Для генерації ландшафту було використано 4 методи. В цьому розділі буде розглянута специфіка та результати виконання цих методів.

Зразу варто зазначити що зробити вибір на користь якогось одного методу буде складно: більшість методів генерації мають свій спектр застосування, власні переваги та недоліки.

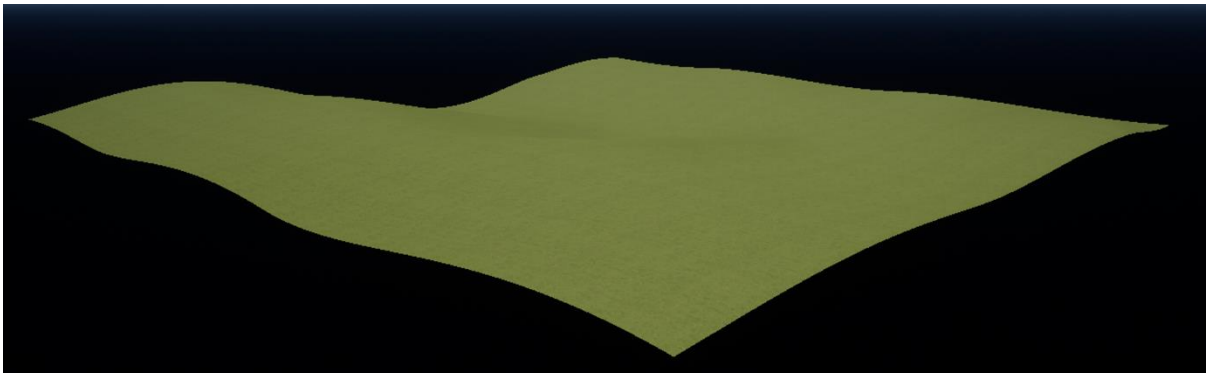
В ході виконання роботи було реалізовано 4 способи генерації ландшафту:

1. Basic procedural generation – простий рандом. Був реалізований для наглядної демонстрації чому ми використовуємо спеціальні методи замість того щоб покласти випадкові величини на сітку.

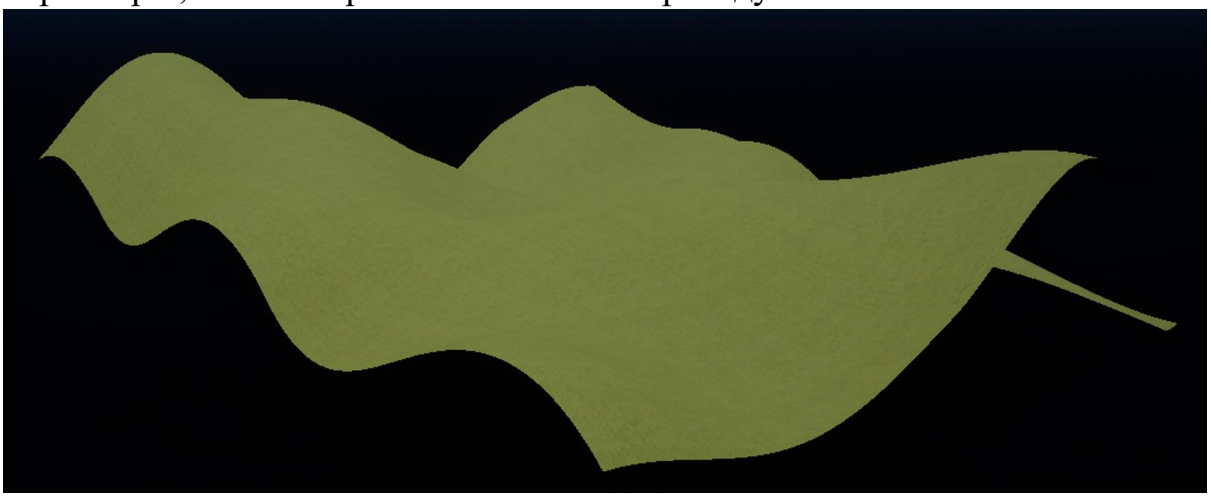


Зі скріншоту видно, що результат виконання алгоритму незадовільний. На ландшафт це аж ніяк не схоже, перепади дуже різкі, а кути гострі.

2. Simpe Perlin procedural generation – реалізація шуму Перліна

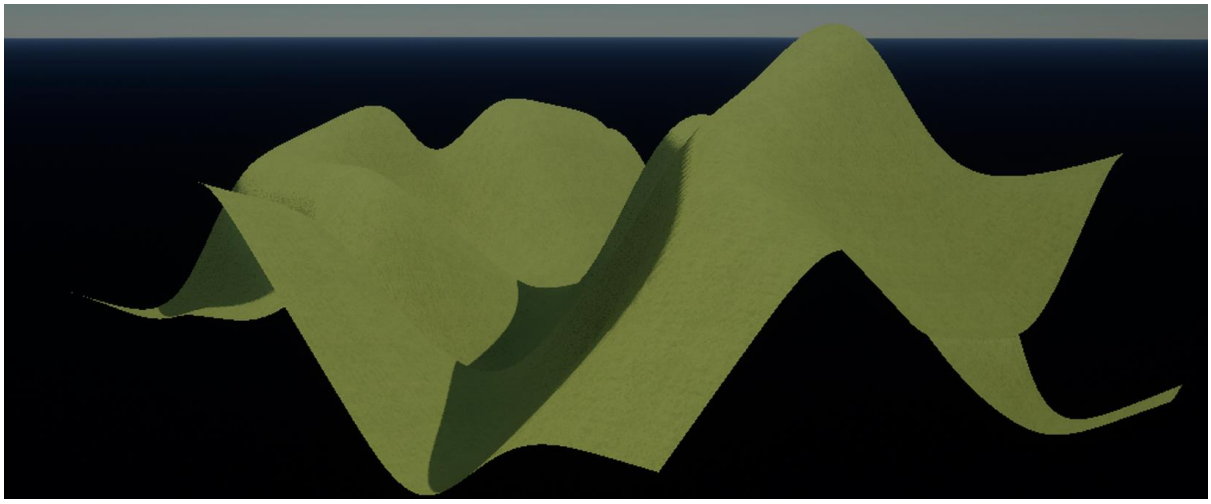


Ландшафт плавний, візуально повторів патерну немає. При зміні параметрів, можна отримати більшого перепаду висот:

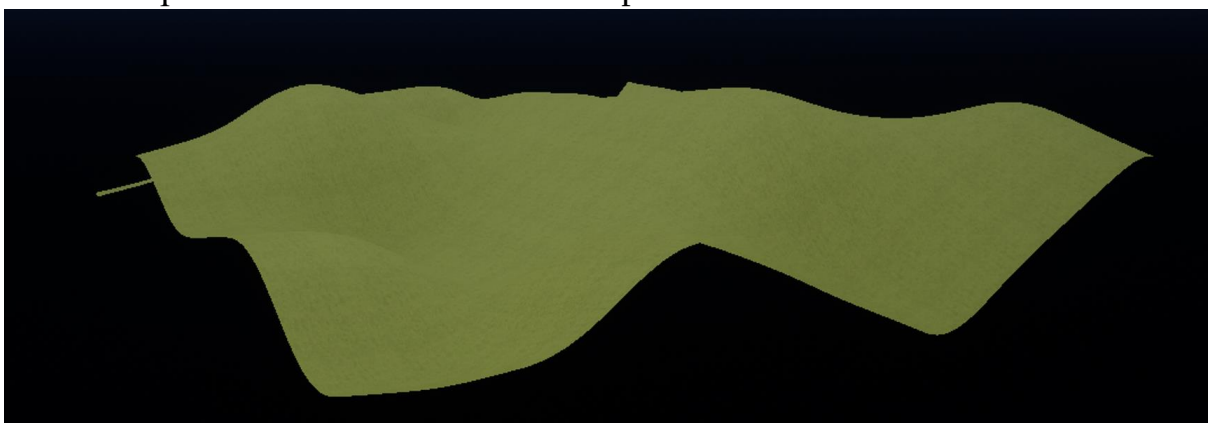


Але нерівності все ще будуть дуже плавними. Одним з серйозних мінусів алгоритму буде те що його складно оптимізувати. В будь-якому випадку, при використанні шуму Перліна ми будемо отримувати неперервну площину, при цьому використовувати тільки частину точок для генерації ландшафту. Не залежно від рівня деталізації навантаження буде те саме.

3. Complex Perlin procedural generation – той самий шум Перліна, тільки підхід до обчислень значень інший: в цій реалізації використовуються векторні градієнти які обчислюються на основі випадкових кутів, а не дотичних градієнтів як в першому випадку.



На виході більш цікаві форми ніж в простому Перліні. Також це помітно при нижчих значеннях z-multiplier.



Мінуси ті самі – відсутність оптимізації.

4. Diamond-Square procedural generation – алгоритм Diamond Square

Результат роботи досить впізнаваний: гострі вершини та наявність гострих кутів. Отримати щось схоже за допомогою одного лише шуму Перліна не вийде через інтерполяцію. Всі гострі вершини згладжуються. Ще одним плюсом алгоритму буде можливість оптимізації. Так як алгоритм є рекурсивним, зменшивши кількість ітерацій ми отримуємо меншу швидкість виконання.

Висновки

1. У ході цієї роботи було проведено детальний аналіз та дослідження використання математики в галузі комп'ютерних ігор. В тому числі шум Перліна та алгоритм Diamond-Square.
2. Розроблено систему процедурної генерації ландшафту в середовищі Unreal Engine, яка використовує різноманітні алгоритми для створення реалістичних і динамічних ландшафтів.
3. Було продемонстровано, що процедурна генерація може збільшити різноманітність і повторюваність ігрового досвіду, знижуючи при цьому обсяг ручної роботи для геймдизайнерів.
4. Також, було визначено, що Unreal Engine, попри свою складність та місцями не інтуїтивність добре підходить для подібних проектів

Ця робота підтверджує, що процедурна генерація є потужним інструментом в галузі розробки ігор, який може бути використаний для створення більш різноманітних і динамічних ігрових середовищ.

Список використаних джерел

https://docs.unrealengine.com/5.1/en-US/

https://www.youtube.com/watch?v=iY1jnFvHgbE

https://www.youtube.com/watch?v=N6D1_GHNwCc&list=PL3hM7_RDaHMz0umAPdszT172uhZTFuP

https://www.youtube.com/watch?v=H5K8Qc0vTYo

https://techguided.com/best-game-engine/

https://medium.com/@oganga-mangiti/the-pros-and-cons-of-artificial-intelligence-what-you-need-to-know-77afad650565

https://medium.com/@DylanCa_/evolution-of-ai-in-video-games-from-1980-to-today-e3344acaed4c

https://medium.com/@bawagupta69/use-of-artificial-intelligence-in-video-games-54ab54911838

https://www.engati.com/blog/ai-in-gaming

https://docs.unrealengine.com/5.1/en-US/artificial-intelligence-in-unreal-engine/

Adams, E. (2014). Fundamentals of Game Design (3rd Edition). New Riders.

Goodfellow, I., Bengio, Y., & Courville, A. (2016). Deep Learning. MIT Press.

Russell, S. J., & Norvig, P. (2016). Artificial Intelligence: A Modern Approach (3rd Edition). Pearson.

Филоненко, А. (2018). Искусственный интеллект в играх на примере Unreal Engine. БХВ-Петербург.

Rouse, R. (2019). Unreal Engine 4 Game Development Quick Start Guide. Packt Publishing.

https://docs.unrealengine.com/5.1/en-US/blueprints-visual-scripting-in-unreal-engine/

Додатки

Вихідний код розробленої гри

Файли проекту: https://github.com/opadfnezig/grad_project