МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ВСП «ОДЕСЬКИЙ ТЕХНІЧНИЙ ФАХОВИЙ КОЛЕДЖ ОНТУ»

Спеціальність: 121 «Інженерія програмного забезпечення»

Освітня програма: «Розробка програмного забезпечення»

Група: 4РП-06

Дипломний проект

здобувача освіти денної форми навчання
РП.06.16.000.ДП

МІРОШКІНА
АНДРІЯ
ІГОРОВИЧА

м. Одеса
2023 р.

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ВСП «ОДЕСЬКИЙ ТЕХНІЧНИЙ ФАХОВИЙ КОЛЕДЖ ОНТУ»

Спеціальність: **121 «Інженерія програмного забезпечення»**

Освітня програма: «**Розробка програмного забезпечення»**

Група: **4РП-06**

ПОЯСНЮВАЛЬНА ЗАПИСКА

до дипломного проекту (роботи) на тему:

Реалізація поведінки штучного інтелекту та алгоритмів взаємодії з ігровим оточенням для персонажів 2D-гри у жанрі top down shooter

Проектний матеріал складається з пояснювальної записки на 71 сторінках та графічного (презентаційного) матеріалу на 8 аркушах (слайдах).

Дипломник _____ (Мірошкін А. І.)

Керівник _____ (Джабраілов Д.В.)

Консультанти:

з економічної частини _____ (Копайгородська Т.Г.)

з охорони праці _____ (Чорновол Н.І.)

з дотримання вимог ЄСКД _____ (Петрашова В.І.)

старший консультант _____ (Кунуп Т.В.)

До захисту допущений

Голова циклової комісії _____ (Кривченко Ю.В.)

Завідувач відділення _____ (Скорнякова О.В.)

Захист «22» 06 2023 р. Протокол ДКК № 1

Оцінка ДКК 5 (п'ять)

Секретар ДКК _____

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

ВСП «ОДЕСЬКИЙ ТЕХНІЧНИЙ ФАХОВИЙ КОЛЕДЖ ОНТУ»

Відділення **комп'ютерних систем** Комісія **КТ та ПІ**

Спеціальність **121 «Інженерія програмного забезпечення»**

Освітня програма **«Розробка програмного забезпечення»**

ЗАТВЕРДЖУЮ:

Заст. дир. з НВР _____

Беркань І.В.

"_____" _____2023 р.

ЗАВДАННЯ

на дипломний проект (роботу)

Здобувачеві (здобувачці) освіти *Мірошкін Андрій Ігорович*

(прізвище, ім'я, по батькові)

1. Тема проекту (роботи) *Реалізація поведінки штучного інтелекту та алгоритмів взаємодії з ігровим оточенням для персонажів 2D-гри у жанрі top down shooter*

затверджена наказом по коледжу від "17" жовтня 2023р. № 235-А2-ОД

2. Термін здачі закінченого проекту (роботи) 09.06.2023

3. Вихідні данні до проекту (роботи)

1. Використання ігрового двигуна Godot Engine;

2. Використання принципів модульної розробки ігор;

3. Використання спеціалізованої мови програмування GDScript для розробки гри;

4. Реалізація поведінки штучного інтелекту та алгоритмів взаємодії з ігровим оточенням для персонажів у 2D-грі в жанрі top down shooter;

4. Зміст розрахунково-пояснювальної записки (перелік питань, які необхідно розробити)

Опис предметної області. Огляд існуючих рішень у ігровій індустрії. Загальний опис вирішення задачі дипломного проекту. Огляд інструментів та програмних рішень для виконання дипломного проекту. Проектування роботи поведінки штучного інтелекту та алгоритмів взаємодії з ігровим оточенням для персонажів. Реалізація поведінки штучного інтелекту та алгоритмів взаємодії з ігровим оточенням для персонажів 2D-гри.

5. Перелік графічного (презентаційного) матеріалу (з точним зазначенням обов'язкових креслень, кількості слайдів)

Особливості ігрового жанру top down shooter; Особливості Godot Engine – приклади ігор що базуються на ньому; Етапи створення макету 2D-гри в жанрі top down shooter; Опис реалізованих елементів поведінки штучного інтелекту; Опис взаємодії між ігровими персонажами; Скріншоти реалізованої гри;

6. Консультанти по проекту (роботі), із зазначенням розділів проекту, що їх стосується

Розділ	Консультант	Підпис, дата	
		Завдання видав	Завдання прийняв
1.Технологічний	*Джабраілов Д.В.*		
розділ			
2.Екон. частина	*Копайгородська Т.Г.*		
3.Охорона праці	*Чорновол Н.І.*		
Нормоконтроль	*Петрашова В.І.*		

7. Дата видачі завдання 24.04.2023

Керівник *Джабраілов Д.В.* _____
(підпис)

Завдання прийняв до виконання _____
(підпис)

КАЛЕНДАРНИЙ ПЛАН

№ з/р	Назва етапів дипломного проекту (роботи)	Термін виконання етапів дипломного проекту (роботи)	Відмітка про виконання
1.	*Вступ. Постановка мети та задач проектування*	*17.05.2023*	виконав
2.	*Опис предметної області*	*18.05.2023*	виконав
3.	*Огляд існуючих рішень у ігровій індустрії*	*19.05.2023*	виконав
4.	*Загальний технічний опис вирішення задачі*	*20.05.2023*	виконав
	дипломного проекту		
5.	*Огляд інструментів та програмних рішень*	*22.05.2023*	виконав
6.	*Проектування роботи поведінки штучного інтелекту*	*23.05.2023*	виконав
7.	*Проектування алгоритмів взаємодії з ігровим*	*24.05.2023*	виконав
	оточенням для персонажів		
8.	*Реалізація поведінки штучного інтелекту*	*25.05.2023*	виконав
9.	*Реалізація алгоритмів взаємодії з ігровим оточенням*	*28.05.2023*	виконав
	для персонажів		
10.	*Тестування елементів на коректну роботу*	*31.05.2023*	виконав
11.	*Виконання Економічної частини*	*02.06.2023*	виконав
12.	*Виконання Охорони праці*	*04.06.2023*	виконав
13.	*Підготовка матеріалів до захисту*	*06.06.2023*	виконав
14.	*Попередній малий захист*	*12.06.2023*	виконав
15.	*Проведення захисту дипломного проекту*	*19.06.2023*	виконав

Дипломник _____
(підпис)

Керівник _____
(підпис)

Формат	Зона	Поз.	Позначення	Назва	Кіл.	Примітка
				Документація		
			РП 06. 16 001. 00 ДП	Дипломний проект		
А4			РП 06. 16 001. 00 ДП ПЗ	Пояснювальна записка	1	

Зм.	Арк.	№ докум.	Підпис	Дата	РП 06.16 000.00 ДП		
Розробив		Мірошкін А. І.			Реалізація поведінки штучного інтелекту та алгоритмів взаємодії з ігровим оточенням 2D-гри у жанрі top down shooter	Літ.	Аркуш 5 / Аркушів 71
Перевірив		Джабраілов Д.В.				Н Д П	
Н. Контр.		Петрашова В.І.				ВСП "ОТФК ОНТУ" гр.4РП-06	
Затверд.		Кривченко Ю.В.					

ЗМІСТ

ВСТУП

Штучний інтелект у комп'ютерних іграх пройшли великий шлях, від NPC здатних лише донести сюжетну інформацію, до персонажів - які самостійно переміщуються по карті і можу вести діалог з гравцем на основі нейромереж. І в наш час цифрові ігри мають велику жанрову, структурну, апаратну та цільову диференціацію.

Темою даного дипломного проектування є «Реалізація поведінки штучного інтелекту та алгоритмів взаємодії з ігровим оточенням для персонажів 2D-гри у жанрі top down shooter» і саме реалізація цієї складової має важливий вклад у створенні якісного програмного продукту.

Крім того, актуальність теми дипломного проектування зумовлена і тим, що процес створення штучного інтелекту важливий для ігор різних жанрів, отже робота по його реалізації – це вклад у розвинення будь-якого жанру ігр. Реалізація зазначених в темі складових, дасть змогу краще зрозуміти процес розробки цифрових ігор, а також створити алгоритми та шаблони для подальшого використання їх в інших проектах.

Також, виконання теми дипломного проекту допоможе більш детально дослідити принципи реалізації NPC в іграх за допомогою актуальних програмних рішень та технологій. Такий підхід дозволить у подальшому створювати рішення на сучасній технологічній основі.

Є реалізація штучного інтелекту та основних складових будь якої гри.

1 ТЕХНОЛОГІЧНИЙ РОЗДІЛ

1.1 Опис предметної області

Тема дипломного проектування включає в собі такі складові:

- Штучний інтелект як фундаментальна складова ігрової індустрії;
- Визначення поведінки штучного інтелекту в іграх;
- Взаємодія штучного інтелекту із оточуючим середовищем та гравцем;
- Визначення з алгоритмом поведінки штучного інтелекту.

Індустрія комп'ютерних ігор – одна з найбільших швидко розвиваються комп'ютерних галузей технологій на сьогодні. І однією із фундаментальних складових успіху гри є система штучного інтелекту. У комп'ютерних іграх штучний інтелект застосовується для управління об'єктами, які утворюють ігрове середовище та уособлюють супротивників та союзників гравця. Існують такі методології практичної реалізації засобів штучного інтелекту:

- штучні нейронні мережі;
- еволюційні алгоритми;
- спеціалізовані архітектури побудови системи штучного інтелекту;
- підхід до реалізації ШІ у формі втіленого анімату.

ШІ отримує інформацію про гравця і події, що відбуваються приблизно так само, як і в реальному світі - у нього є спеціальні сенсори, за допомогою яких він досліджує оточення і стежить за тим, що відбувається. Сенсори бувають зовсім різними. Це може бути традиційний конус зору, "вуха", які вловлюють гучні звуки, або навіть нюхові рецептори. Звичайно, такі сенсори - лише імітація реальних органів почуттів, яка дозволяє зробити ігрові ситуації більш правдоподібними та цікавими. Наявність та реалізація сенсорів залежить від геймплею. Багато активних шутерах не потрібні комплексні

рецептори — достатньо конуса зору, щоб реагувати на появу гравця. А в стелс-екшенах весь геймплей ґрунтується на тому, щоб ховатися від супротивників, тому віртуальні органи почуттів улаштовані складніше.

Основним принципом, що лежить в основі роботи ШІ, є алгоритм прийняття рішень. Для вибору після прийняття рішень система має проводити об'єкти з допомогою ШИ. При цьому така дія може бути організована у вигляді «мовлення ШІ» або «навернення об'єктів». У системах із «мовленням ШІ» система ШІ зазвичай ізольована у вигляді окремого елемента ігрової архітектури. Така стратегія найчастіше набуває форми окремого потоку або кількох потоків, у яких ШІ обчислює найкраще рішення для заданих параметрів гри. Коли ШІ приймає рішення, воно передається всім об'єктам, що беруть участь. Такий підхід найкраще працює у стратегіях реального часу, де ШІ аналізує загальний перебіг подій у всій грі.

1.2 Огляд існуючих рішень у ігровій індустрії

Для гри у жанрі 2D top down shooter можна створити як занадто примітивний ШІ - котрий буде лише переслідувати гравця, так і прописати справжнього вартового, котрий буде мати область видимості, зможе чути пересування гравця і навіть закликати на допомогу інших NPC.

Наприклад, у грі Alien Shooter 2 штучний інтелект досить простий. Всі противники мають область видимості і як тільки гравець входить до неї вони починають рухатися у його напрямі кожен зі своєю швидкістю. І загалом противники діляться на 2 типи: перший, який просто прагне підбігти до гравця щоб завдати йому шкоди рукопашною атакою або підірвати його разом із собою; другий, який підходить на певну відстань і починає обстрілювати гравця. І деякі юніти можу мати дві стаді. У першій вони стріляють у другій ідуть до гравця. тобто по суті один юніт вміщує можливості двох типів представлених раніше.

Рисунок 1.1. Приклад реалізації штучного інтелекту у Alien Shooter 2 №1

Рисунок 1.2. Приклад реалізації штучного інтелекту у Alien Shooter 2 №2

А в Shadow Tactics: Blades of the Shogun використовується просунутий конус для огляду. Конус поділений на зони видимості, які залежать від особливостей ландшафту. Якщо гравець потрапить у однорідну зелену зону, його майже відразу побачать. У смугастій зоні видимість утруднена, тому гравець буде непомітний, якщо рухатиметься, пригнувшись. А в зоні, покритій крапками, герой повністю прихований. Завдяки цій механіці симулюється реалістичний огляд, що дозволяє гравцеві краще поринути в гру, адже чим реалістичніша тим більш захоплююча гра.

Логіка поведінка різних видів противників:

Звичайні воїни - солдати звичайні, озброєні списами чи примітивним, але дієвим вогнепальним стрільцем. Поодинці не небезпечні, тому часто ходять групами. Ведуться на будь-які відволікаючі маневри: дивляться у бік каменю, що впав, біжать за пляшкою саке, перевіряють, хто свистить у кущах.

Солом'яні капелюхи - зовні від звичайних солдатів відрізняються тільки самими капелюхами, але поводяться куди як обережніше. На провокації ці бійці не піддаються, у бік підозрілого шуму дивляться недовго, тож вивести їх із ладу вже складніше.

Офіцери - відрізняються прапорами на спинах, в іншому - ті ж солом'яні капелюхи з усіма плюсами та мінусами.

Самураї - ходять у важкій броні та озброєні двома мечами. Провокації ігнорують, легко розкривають шпигуна, спробу вбивства зі спини припиняють швидко і безжально, і навіть куля в спину лише ненадовго приголомшує їх. Найнебезпечніші вороги, і якщо з самотнім самураєм впоратися ще можна, то двоє разом без проблем знищать весь загін.

Рисунок 1.3. Приклад реалізації штучного інтелекту у Shadow Tactics

Пересування ШІ у просторі:

Дії ШІ безпосередньо пов'язані з його завданням. Незалежно від завдання він має дістатися місця виконання дії — для цього йому потрібна інформація про навколишнє середовище. Зазвичай такі дані містяться в навігаційній сітці — це особлива карта, де зазначено, де NPC можуть пересуватися.

Наприклад, у Assassin's Creed Origins у всіх NPC є свій розпорядок дня, якого вони дотримуються навіть без участі гравця. Щоб персонажі наслідували сценарій і не застрягали в оточенні, розробники зробили спеціальну мережу вузлів — точки на площині, де NPC виконують своє персональне завдання. Така механіка дозволяє чітко розмежовувати область патрулювання юнітів та підвищує візуальну реалістичність їх пересування що призводить до підвищення інтересу гравця.

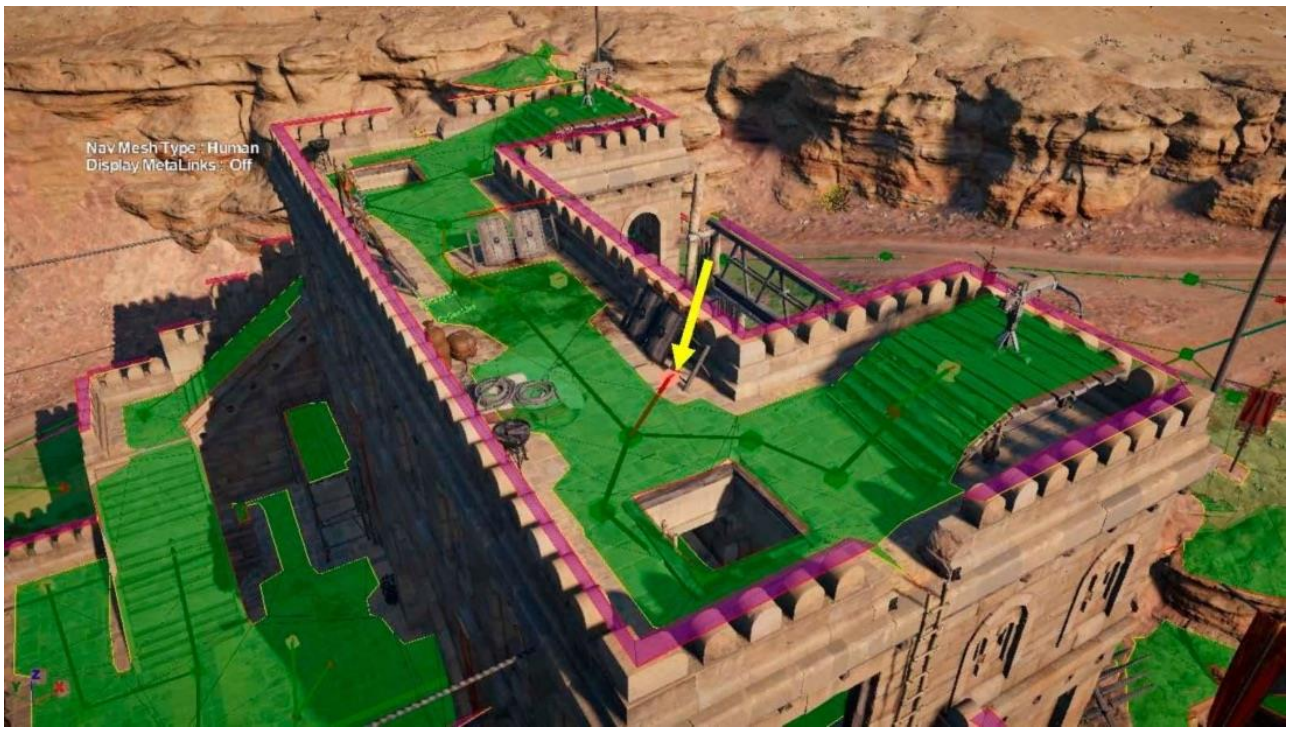


Рисунок 1.4. Приклад навігаційної сітки, а поверх неї системи вузлів у грі Assassin's Creed Origins

А у Just Cause 3 існує система, що вибирає пріоритетні позиції, на яких NPC зможе найефективніше виконати своє завдання. Вибір тієї чи іншої позиції залежить від конкретних критеріїв: дальності від мети, наявності перешкод між юнітом та метою, віддаленістю від союзників тощо.

Рисунок 1.5. Приклад системи вузлів заснований на точках пріоритету

1.3 Загальній опис вирішення задачі дипломного проекту

Для якісної реалізації гри та штучного інтелекту в ньому, я почав планування чітких етапів розробки:

Крок 1. Ідея

Спочатку потрібно було придумати будь-що, щоб затягнути майбутнього гравця — викликати радість, сміх, інтерес, страх, мотивацію досягти чогось у грі. Незважаючи на те, що робота над ідеєю – творчий процес, у ньому є й технічна частина. Створення ідеї складається з кількох складових:

- Опис - собор всіх напрацювань, схем, скетчів та формування з них цілісної концепції.

- Мета проекту – розробка програми на своє задоволення та з метою навчання. Хоча в основному я поставив за мету навчитися процесу розробки, зробити щось цікаве та отримати досвід.

- Сеттинг - почав проектувати гру, після вибору жанру та основ сеттингу я прописав дійові особи, особливо значущі події та місця.

Крок 2. Вибір середовища розробки

Вибір середовища розробки сильно впливає ринку, жанр і управління грою. І як середовище розробки підходящий під потреби студента я вибрав Godot тому, що він поширюється під вільною ліцензією і володіє широким інструментом. А підтримка здійснюється ком'юніті.

Крок 3. Жанр

Як я вже писав вище, вибір платформи впливає на те, на якому жанрі слід зупинитися. Кожна платформа має свою аудиторію та технічні обмеження. А про жанри ігор можна написати окрему статтю, але в розрізі першого досвіду варто розробити щось справді просте. Один із таких жанрів, може підійти:

- Топдаун шутер (Topdown Shooter) - спрощений варіант шутера, вигляд зверху, звідси і назва. Основна механіка - це стрілянина та бій.

Приклади: Helldivers, Alien Shooter, Hotline Miami.

- Роуглайк (Roguelike) - «рогаліки», покрокова гра, зазвичай має процедурно-генеровані рівні, смерть персонажа необоротна, потрібно розпочинати гру заново. Приклади: Angband, Dungeon Crawl Stone Soup.

- Платформер (Platformer) - динамічна гра з основною механікою у вигляді різноманітних платформ та переміщення персонажа по них, також є безліч допоміжних інструментів. Приклади: Super Mario Bros., Hollow Knight, Ori та Blind Forest.

- Пазл (Puzzle) - такі ігри мають механіки, пов'язані з вирішенням головоломок та завдань, можуть стосуватися різних типів головоломок, конструювання, візуальних пазлів. Приклади: Harry Potter: Puzzles & Spells, 2048, World of Goo.

- Гоночний симулятор (Racing game) – жанр пов'язаний із гонками на швидкість, необов'язково автомобілів. Приклади: Need For Speed, GRID, Real Racing.

Крок 4. Вивчення координатної системи 2D

У 2D просторі координати визначаються за допомогою горизонтальної осі (x) і вертикальної осі (y). Позиція у просторі записується парою значень, наприклад(4, 3). Таким чином, задавши дві точки в координатній площині і з'єднавши їх, можна отримати вектор. В Godot цими векторами можна маніпулювати, наприклад, переміщати в просторі або змінювати їх напрями, перезадаючи значення точок. Що дозволяє з легкістю реалізовувати функціонал багатьох об'єктів, наприклад, переміщення персонажа: ходу, біг, пересування лежачи і стрибки вдовж (рисунок 1.6.).

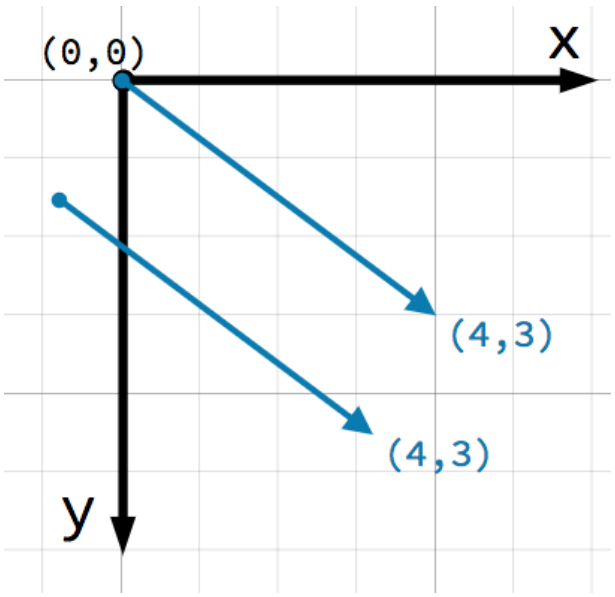


Рисунок 1.6. Приклад координати у просторі

У Godot координати реалізовані за допомогою Vector2 для 2D. До окремих компонентів вектора можна звертатися на ім'я. Вектор можна складати або обчислювати, складати відповідні компоненти. Після вивчення можливостей вектора можна зрозуміти сфери його застосування:

Рух - вектор може представляти напрямок, наприклад: позиція, швидкість, прискорення та сила. Наприклад космічний корабель на етапі 1 має вектор позиції (1,3) а вектор швидкості (2,1). Вектор швидкості представляє, як далеко піде корабель на кожному кроці.

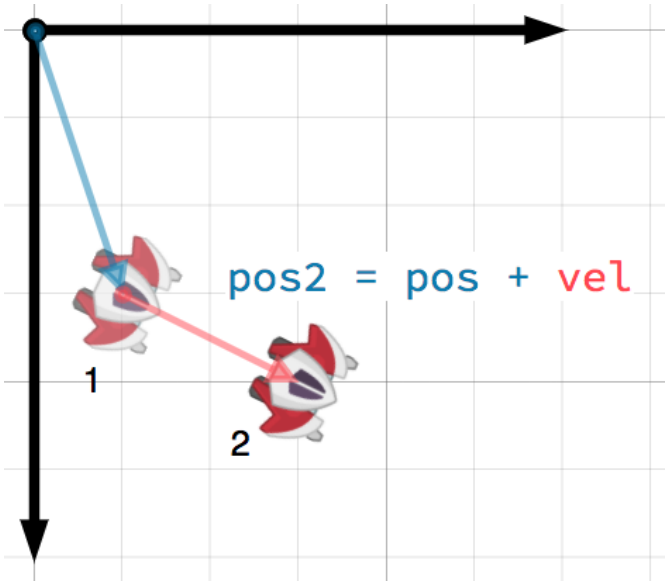


Рисунок 1.7. Приклад переміщення об'єкта у координатному просторі

Напрямок у бік об'єкта - наприклад, при керуванні танком, гравець захоче направити дуло на робота. Віднімання позиції танка з позиції робота дасть вектор, спрямований від танка до роботи.

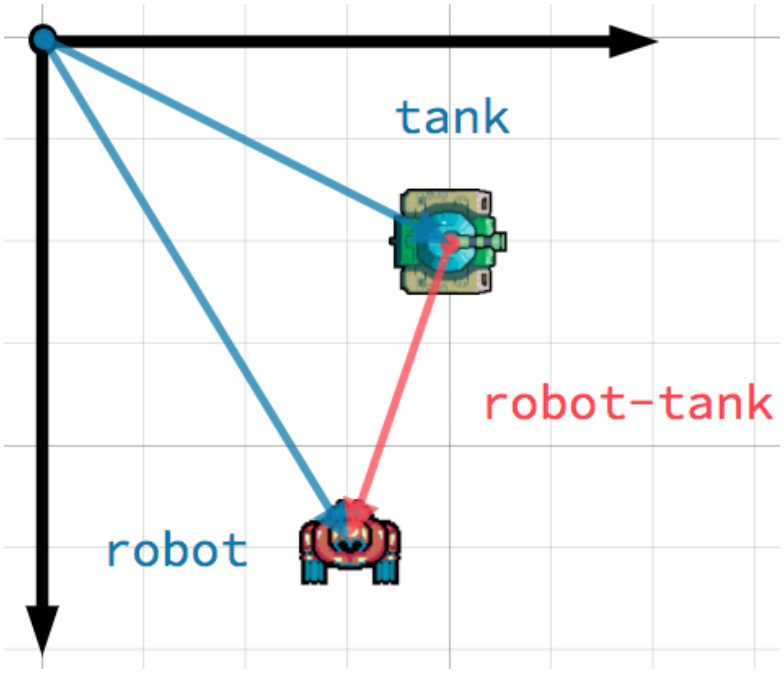


Рисунок 1.8. Приклад направлення погляду одного юніта на іншого

Після того, як я пройшов усі ці кроки, я розпочав розробку проекту. Для розробки дипломного проекту, необхідно створити персонажа гравця та пару NPC із різним штучним інтелектом. Керуючись принципами розробки розглянутими раніше, реалізую одного противника, котрий буде бігати за гравцем, а другий буде патрулювати по карті і побачивши гравця він почне по ньому стріляти.

Оскільки проект знаходиться на початковій стадії розробки, візуальна реалізація персонажів буде виконана за допомогою примітивних не анімованих спрайтів намальовані під вид зверху з дотриманням загального стилю. Для стилю юнітів я обрав stickman, оскільки вони не вимагають особливих навичок у малюванні. У результаті я отримав такі спрайти:

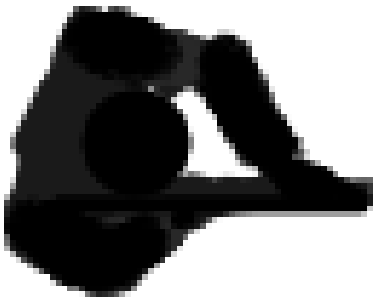


Рисунок 1.9. Супротивник зі стрілецькою зброєю

Рисунок 1.10. Персонаж гравця

Рисунок 1.11. Супротивник зі зброєю ближнього бою

Програмна частина буде реалізована за допомогою скриптів, мовою програмування GDScript. Взаємодія між персонажами буде реалізована за допомогою областей видимості, в яких персонажі будуть обробляти дані один про одного і робити певні алгоритми дій залежно від вхідних даних.

Розробка гри буде виконуватися із використанням модульного принципу, завдяки чому у майбутньому проект буде легше модернізувати, а несправності торкнуться тільки певних модулів. Також наявність глобальних модулів, дасть можливість створювати нових NPC не прописуючи для них весь код заново. Ось так виглядатиме файлова система проекту, за наявності розроблених об'єктів та модулів зі всіма глобальними скриптами та спрайтами:

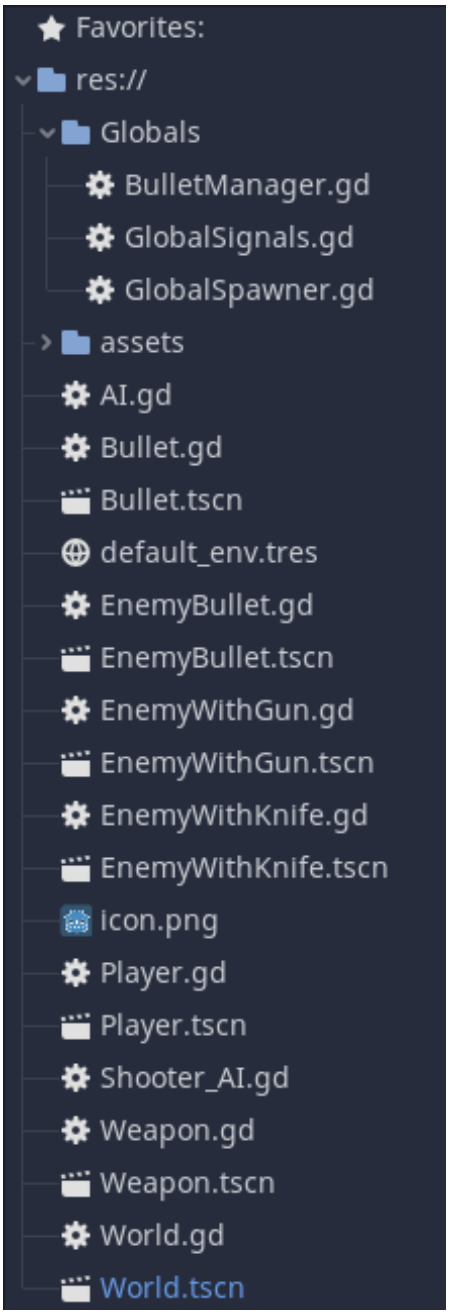


Рисунок 1.12. Файлова система проекту

У файловій системі знаходяться всі сцени, що є в проекті:

1) Сцена кулі є 2D об'єктом, що піддається законам фізики і має вагу, текстурою, колайдером. Завдяки чому з ним можна взаємодіяти. так само є таймер, який обмежує час її існування.

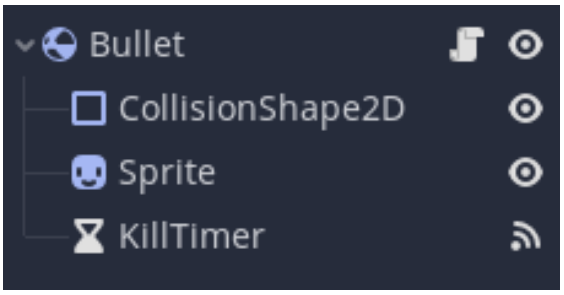


Рисунок 1.13. Сцена Bullet

2) Сцена ворожої кулі є 2D об'єктом, який не піддається законам фізики, має лише колайдером і таймер, який обмежує час її існування.

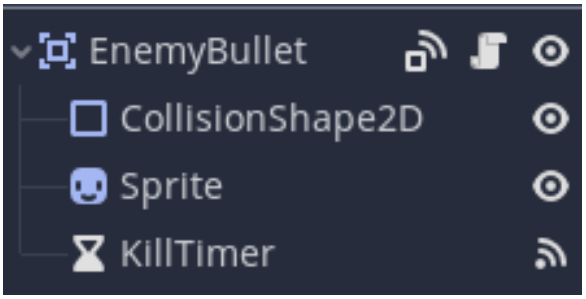


Рисунок 1.14. Сцена EnemyBullet

3) Сцена противника з вогнепальною зброєю досить велика, вона являє собою 2D об'єкт, який може рухатися та змінювати свій напрямок руху за власним бажанням, також не піддається законам фізики, має кілька колайдерів та зон потрібних для отримання інформації, котра буде впливати на поведінку цього юніта. Крім цього, має також 2 вузли, 1 з яких є глобальною сценою що дозволить створювати нових персонажів, використовуючи функціонал цього.

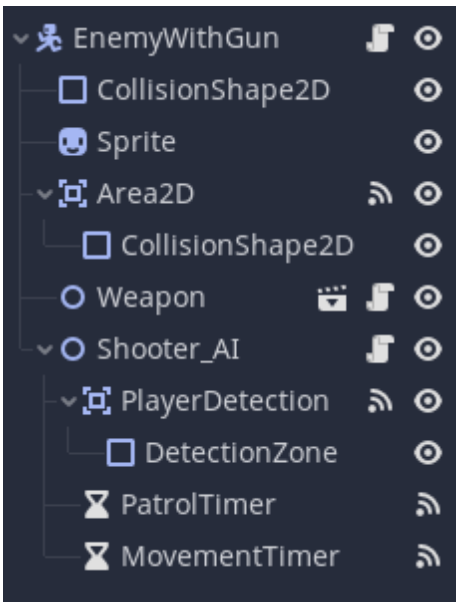


Рисунок 1.15. Сцена EnemyWithGun

4) Сцена супротивника зі зброєю ближнього бою менша за попередню, оскільки вона не потребує такого широкого функціоналу. Вона являє собою 2D об'єкт, який може рухатися, але не піддається законам фізики, має кілька колайдерів.

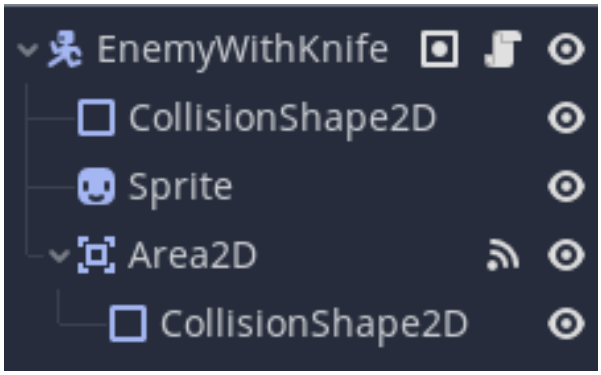


Рисунок 1.16. Сцена EnemyWithKnife

5) Сцена гравець одна з найбільш функціональних. Вона являє собою 2D об'єкт, який може рухатися, але не піддається законам фізики, має кілька колайдерів та 3 таймери. Кожен із яких відповідає за час, який необхідно почекати перед повторним використанням здібності персонажа гравця.

Рисунок 1.17. Сцена Player

6) Сцена world є основною, тому що на ній розміщуються всі юніти та об'єкти. Вона вміщує в собі сцену юніта, камеру, таймери, які служать для паузи між появою ворогів, глобальний вузол для відстеження ворожих куль, а також об'єкти - стіни та підлога.

Рисунок 1.18. Сцена World

1.4 Огляд інструментів та програмних рішень

Для виконання дипломного проекту я обрав ігровий двигун GODOT. Godot Engine - відкритий кросплатформовий 2D і 3D ігровий двигун під ліцензією MIT, який розробляється спільнотою Godot Engine Community.

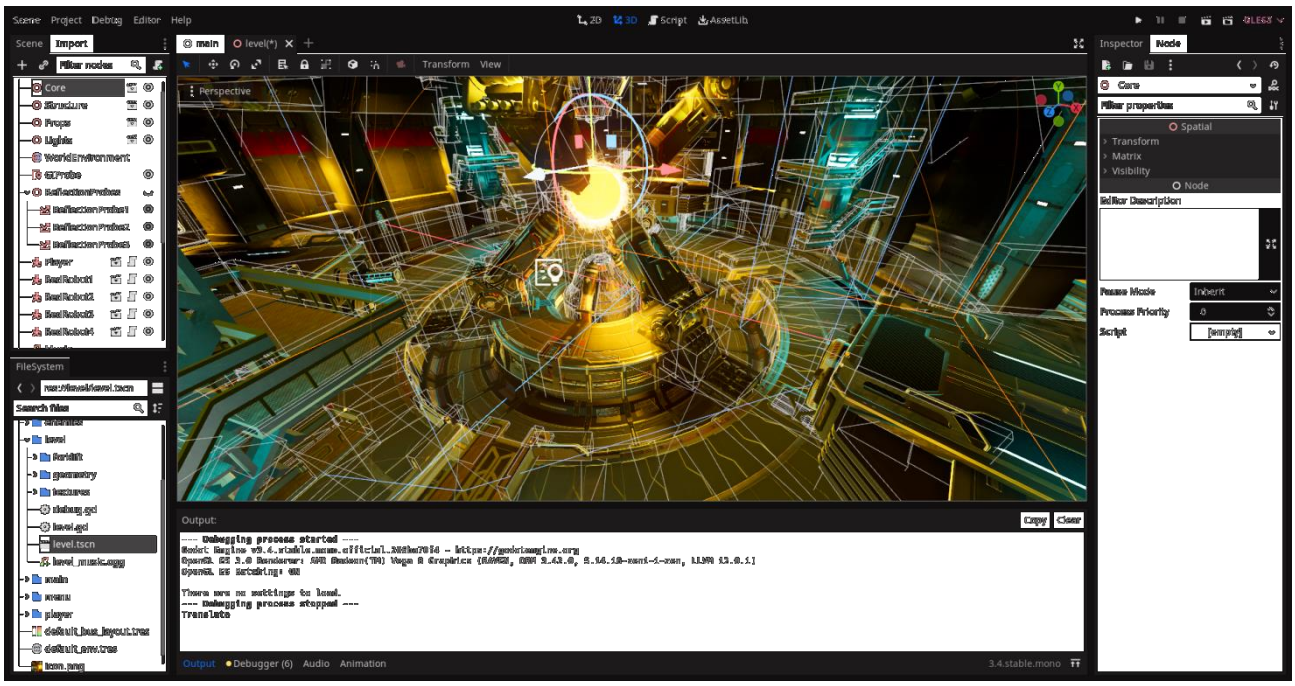


Рисунок 1.19. Робоче середовище ігрового двигуна GODOT

Godot надає великий набір інструментів, тому ви можете просто зосередитися на створенні гри, не винаходячи велосипеда та не прописуючи нові паттерни. Godot є повністю безкоштовним і відкритим вихідним кодом під дуже дозвільною ліцензією MIT. Ніяких умов, жодних роялті, нічого. Ваша гра належить вам, аж до останнього рядка коду двигуна. Завдання Godot – бути максимально інтегрованим та самодостатнім середовищем для розробки ігор. Середовище дозволяє програмістам створювати ігри з нуля, користуючись лише необхідними інструментами для створення ігрового контенту наприклад: елементи графіки, музичними треками і так далі. Проте роцес програмування все ще вимагає зовнішніх інструментів, хоча за необхідності використовувати зовнішній редактор, це можна зробити відносно легко.

Рисунок 1.20. Назва мови

Гра створюється за допомогою власної високорівневої динамічно типізованої скриптової мови програмування під назвою

Синтаксис GDScript нагадує мову Python. Але має відмінності від Python - насамперед обов'язкове визначення області видимості змінної через ключове слово var та оптимізація мови під потреби системи сцен та вузлів движка. Графічна система для всіх платформ, що підтримуються, побудована на OpenGL ES 3.0. Рендеринг включає технології order-independent transparency, normal mapping, specularity, повноекранні постефекти типу FXAA, bloom, DOF, HDR, гамма-корекції, distance fog, динамічні тіні на основі shadow maps та інші.

Фізичний двигун для 2D та 3D розроблений з нуля, що допомогло досягти необхідного рівня оптимізації фізичної підсистеми. Реалізовано можливості рейкастингу, виявлення зіткнень, динаміки твердих тіл та з'єднань між ними. Також є власна реалізація кінематичного контролера персонажа та 3D-контролер автотранспортних засобів зі спрощеною системою підвіски.

Наприкінці розробки проект може бути експортований на різні цільові платформи, які можна поділити на ПК, мобільні, Інтернет та консолі. Для різних платформ можна задавати різні параметри, такі як спосіб зберігання даних (і їх захист за потреби), компресія текстур, роздільна здатність, а також деякі унікальні параметри, характерні лише для якоїсь конкретної платформи (наприклад, дозволи для Android). Поточна підтримка платформ включає Windows (та UWP OS), MacOS, X11 (Linux, BSD), Android OS, iOS, HTML5.

Також можна експортувати інші платформи вручну через компілювання движка для SDK цільової платформи. Використання Godot незначної кількості зовнішніх бібліотек полегшує цей процес. На Godot у 2022 році очікується чимало ігор таких як:

1. TailQuest Defense – стратегія в стилі Tower Defense

2. Lumencraft - 2d Екшен про будівництво бази, видобуток ресурсів та битви з жуками

3. The Garden Path - пригодницька інди гра

4. The Zone: Stalker Stories - рольова екшен гра з елементами ККІ

5. Seedlings – незвичайна казуальна гра в атмосфері природи

Філософія архітектури Godot.

Godot базується на об'єктно-орієнтованому проектуванні з ієрархією вузлів та гнучкою системою сцен. Він намагається обійтися без суворих шаблонів програмування, щоб запропонувати інтуїтивний спосіб структурування вашої гри для спрощення порогу входження.

Godot дозволяє збирати або розбирати сцени. Це нагадує префаби, наприклад ви можете створити мерехтливе світло і сцену зі зламаним ліхтарем, який використовує цю несправну лампочку. Потім створити місто наповнене такими ліхтарями. Після чогопоміняти колір мерехтіння, зберегти об'єкт, і всі ліхтарі тут же оновляться. Більш того, так можна успадковуватися від будь якої сцени.

Сцена Godot може бути персонажем, зброєю, предметом, столом, стільцем, рівнем, частиною рівня – будь чим. Вона працює як клас у коді, за винятком того, що можна вільно спроектувати її лише за допомогою редактора, використовуючи лише код, або змішуючи та поєднуючи обидва способи.

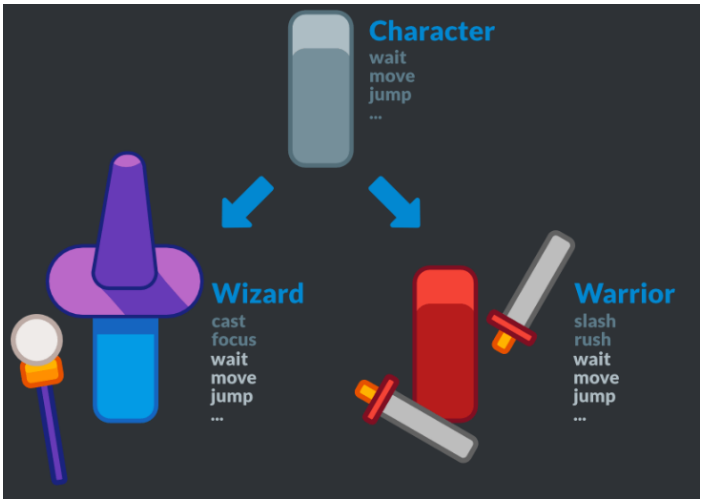


Рисунок 1.21. Успадкування функціоналу
від базового класу

Це відрізняється від збірних даних, які можна знайти в деяких тривимірних редакторах, тому що ви можете успадкувати та розширити ці сцени. Також можна створити сцену Wizard, яка буде успадкована та розширена зі сцени Character. Змінити Персонаж можна у редакторі, і Wizard також оновиться. Це допомагає будувати свої проекти так, щоб їх структура відповідала архітектурі гри, що набагато скоротить час внесення правок.

Також Godot пропонує безліч різних типів об'єктів, які називаються вузлами, кожен з певним призначенням. Вузли є частиною дерева і завжди успадковуються від батьків аж до класу Node. Хоча редактор має такі компоненти, як форми зіткнення, які використовують батьківське фізичне тіло, більшість вузлів працюють незалежно один від одного.

Іншими словами, вузли Godot не працюють як компоненти в деяких інших ігрових рушіях.

Sprite — це Node2D, CanvasItem і Node. Він володіє всіма властивостями та функціями трьох батьківських класів, як-от перетворення або можливість малювати користувацькі фігури та відображати за допомогою спеціального шейдера.

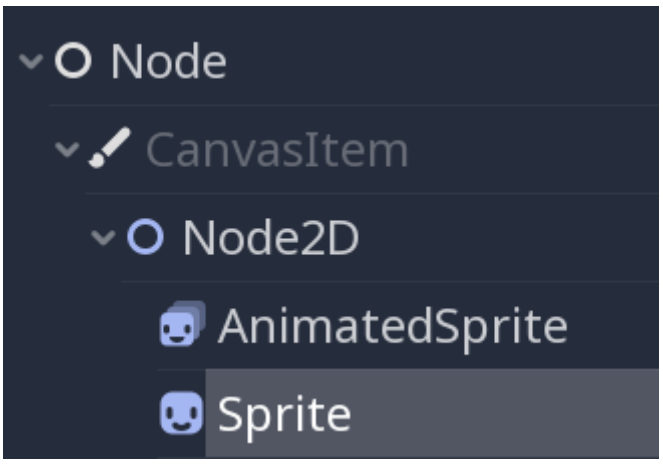


Рисунок 1.22. Панель сцен та об'єктів

Godot надає свої власні інструменти для задоволення найбільш поширених потреб. Він має вбудований редактор коду, редактор анімації, редактор tilemap, редактор шейдерів, зневаджувач, профілювальник, можливість перезавантаження на ходу локально і на віддалених пристроях і багато іншого.

Мета полягає в тому, щоб запропонувати повний пакет для створення ігор і безперервного навчання. Ви можете працювати з зовнішніми програмами, якщо для них є плагін імпорту. Або ви можете створити його, наприклад, Tiled Map Importer.

Рисунок 1.23. Карта для 2D гри
побудована в GODOT

Частково тому Godot пропонує власні мови програмування GDscript та VisualScript разом із C#. Вони розроблені для потреб розробників ігор та дизайнерів ігор, і вони тісно інтегровані в рушій та редактор.

GDScript дозволяє писати код із використанням синтаксису, заснованого на відступах, але він виявляє типи та пропонує якість автоматичного завершення статичної мови. Він також оптимізований для ігрового коду з вбудованими типами, такими як вектори та кольори.

Зауважте, що за допомогою GDNative ви можете писати високоефективний код, використовуючи компільовані мови, такі як C, C++, Rust, або Python (використовуючи компілятор Cython), без перекомпіляції рушія.

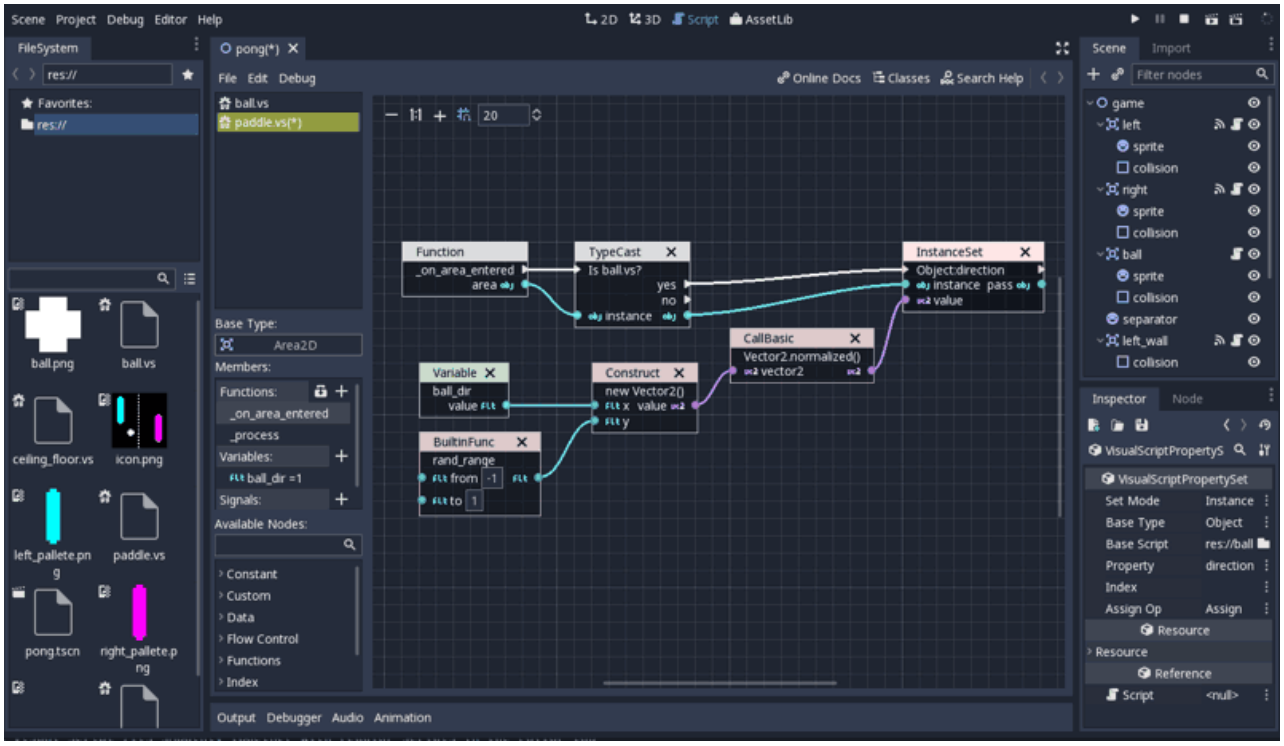


Рисунок 1.24. Приклад використання VisualScript №1

VisualScript - мова програмування на основі вузлів, яка добре інтегрується в редактор. Ви можете перетягувати вузли, або ресурси, в граф для створення нових блоків коду. Треба зауважити, що в робочій області 3D немає такої кількості інструментів, як у робочій області 2D. Знадобляться зовнішні програми, або додатки, для редагування місцевості, анімації складних персонажів тощо. Godot пропонує повний API для розширення функціональності редактора за допомогою ігрового коду.

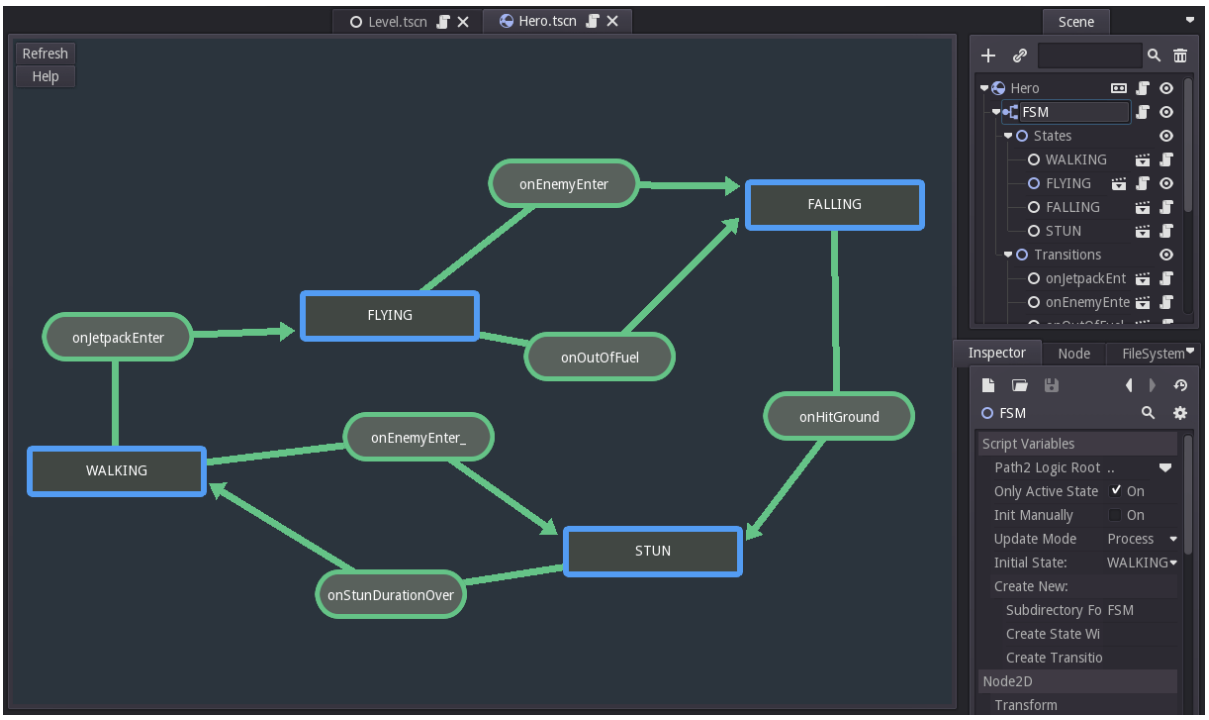


Рисунок 1.25. Приклад використання VisualScript №2

Керований спільнотою.

Godot створений його спільнотою, для спільноти та для всіх творців ігор. Саме основні потреби користувачів та відкриті дискусії визначають основні оновлення. Нові функції основних розробників часто зосереджуються на тому, що в першу чергу принесе користь користувачам.

Хоча повний робочий день над ним працюють лише декілька основних розробників, проект нараховує понад 600 учасників. Доброзичливі програмісти працюють над можливостями, які можуть знадобитися їм самим, тому ви побачите покращення в усіх аспектах рушія в кожному великому випуску.

Редактор Godot працює на ігровому рушії. Він використовує власну систему інтерфейсу рушія, може перезавантажувати код і сцени під час тестування проектів, або запускати ігровий код у редакторі. Це означає, що ви можете використовувати той самий код і сцени для своїх ігор, або створювати плагіни та розширювати редактор.

Це призводить до надійної та гнучкої системи користувацького інтерфейсу, оскільки вона приводить в дію сам редактор. За допомогою

ключового слова tool ви можете запустити будь-який ігровий код у редакторі.

Хоча Godot і має власний редактор для вирізування спрайтів під гру, проте створити спрайт з нуля він не дозволяє, тому для створення спрайтів (2D текстур) для об'єктів та персонажів я використав Paint.net.

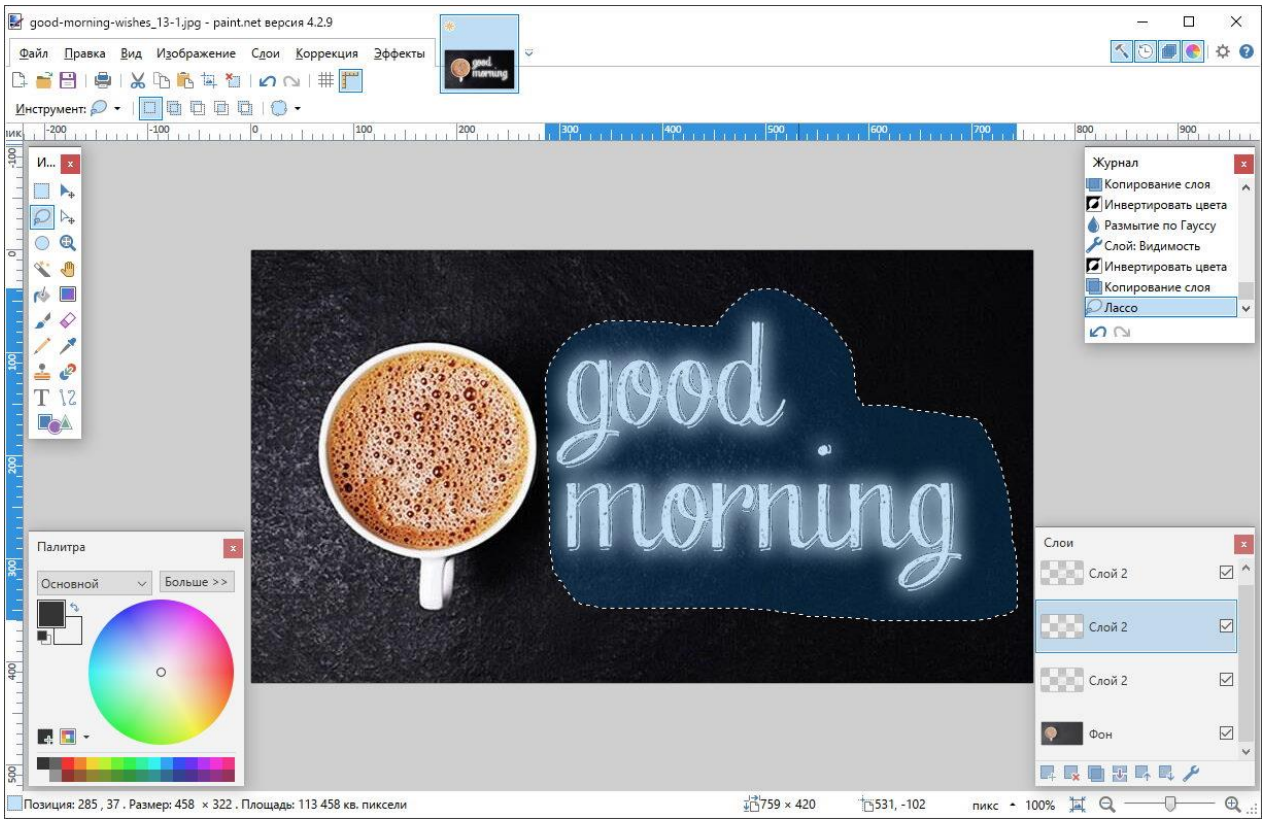


Рисунок 1.26. Робоче середовище програми Paint.net

Paint.NET – безкоштовний растровий графічний редактор для Windows, розроблений на платформі .NET Framework. Paint.NET є чудовою заміною редактору графічних зображень, що входить до складу стандартних програм операційних систем Windows. Саме я використовував для малювання персонажів для гравця та NPC і на це була низка причин: peint.net дозволяє працювати з кількома документами одночасно, підтримує роботу із шарами, та простий у використанні.

1.5 Проектування роботи поведінки штучного інтелекту та алгоритмів взаємодії з ігровим оточенням для персонажів

Створюю ігрову карту. Для цього додаю на сцену 2 об'єкти TileMap, перший для стінок другій для полу. Після чого задаю ієрархію так, щоб стіни відображались поверх полу, далі обираю потрібний набір спрайтів і заповнюю карту. Потім задаю колайдери для стін, щоб персонажі не могли проходити через неї. Для того, щоб зібрати повноцінну квадратну стіну рівня, мені знадобилося заготовити десять спрайтів, в основному у них квадратні колайдери, але деякі мають складний, налаштований вручну, колайдер (рисунок 1.27.).

Рисунок 1.27. Редактор спрайтів

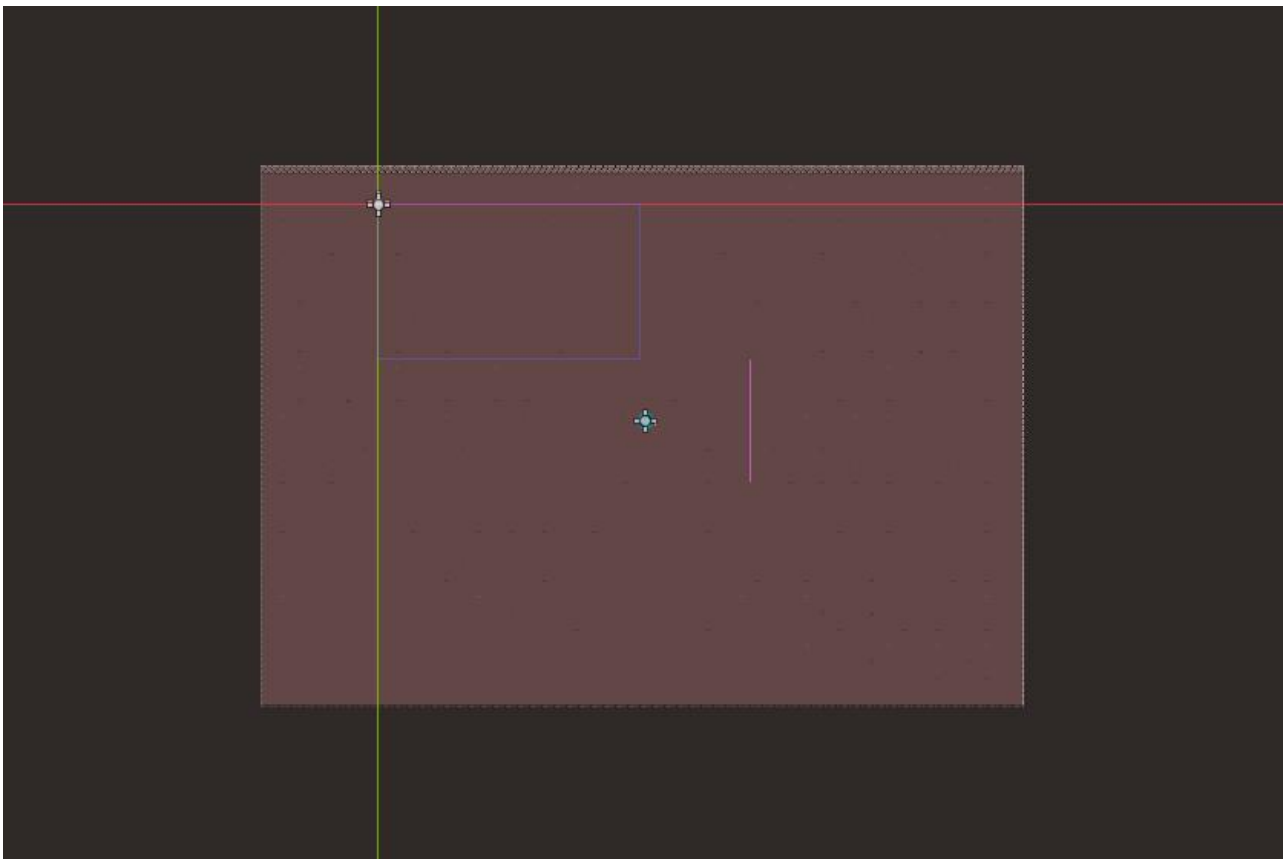


Рисунок 1.28. Ігрова карта

Потім створюю персонажа гравця та реалізую логіку його переміщення, ривків та стрільби. Для цього користуюся зміною його координат у просторі за допомогою Vector2. Переміщення відбувається за допомогою натискання на певні клавіші на клавіатурі, і при натисканні на них подається сигнал, котрий визиває спрацювання функції для передання персонажу гравця швидкості у певному напрямку. Для ривків теж призначаю клавішу - "Space" при натисканні на яку короткочасно швидкість гравця буде збільшена в 3 рази, після чого вона повернеться до початкового значення. Далі реалізую стрільбу з налаштуванням темпу стрільби та підв'язую її до ЛКМ. Приклад роботи зображено на алгоритмі (рисунок 1.29.)

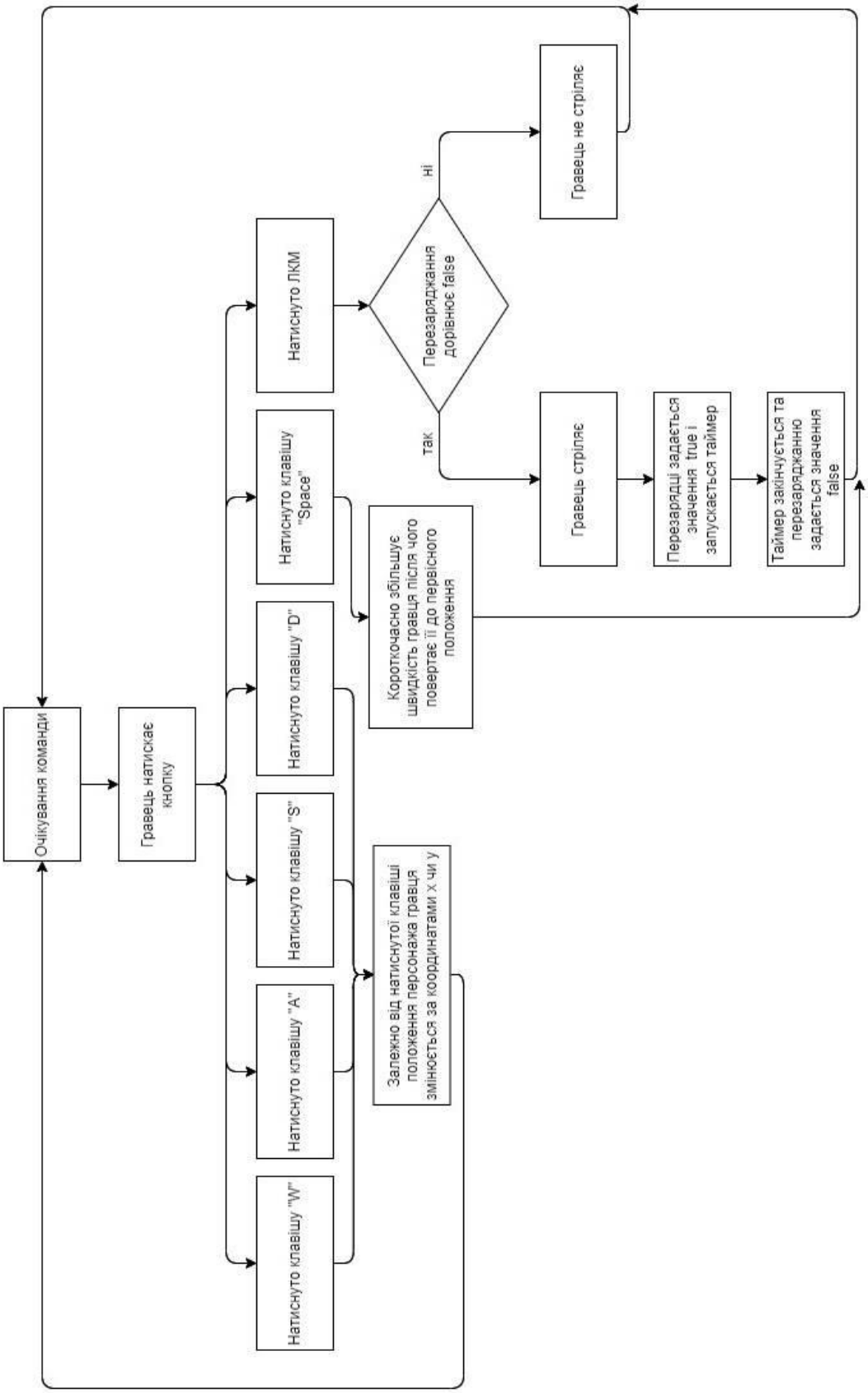


Рисунок 1.29. Функціональна схема логіки роботи персонажа гравця

Після реалізації карти та персонажа гравця, можна приступати написання алгоритму штучного інтелекту. Для реалізації логіки штучного інтелекту використовуватиму систему на основі правил. Це проста форма штучного інтелекту. Така система являє собою набір заздалегідь заданих алгоритмів, що визначає поведінку ігрових об'єктів. З урахуванням різноманітності дій кінцевий результат може бути неявною поведінковою системою.

Почну зі створення самого юніту. Задам йому спрайт, колайдери, кілька класів, які описуватимуть його: зброю, поведінку, сприйняття. Розміри регіонів та колайдера налаштовую вручну, так само ставлю для них події на входження об'єктів для найдовших маніпуляцій у скриптах.

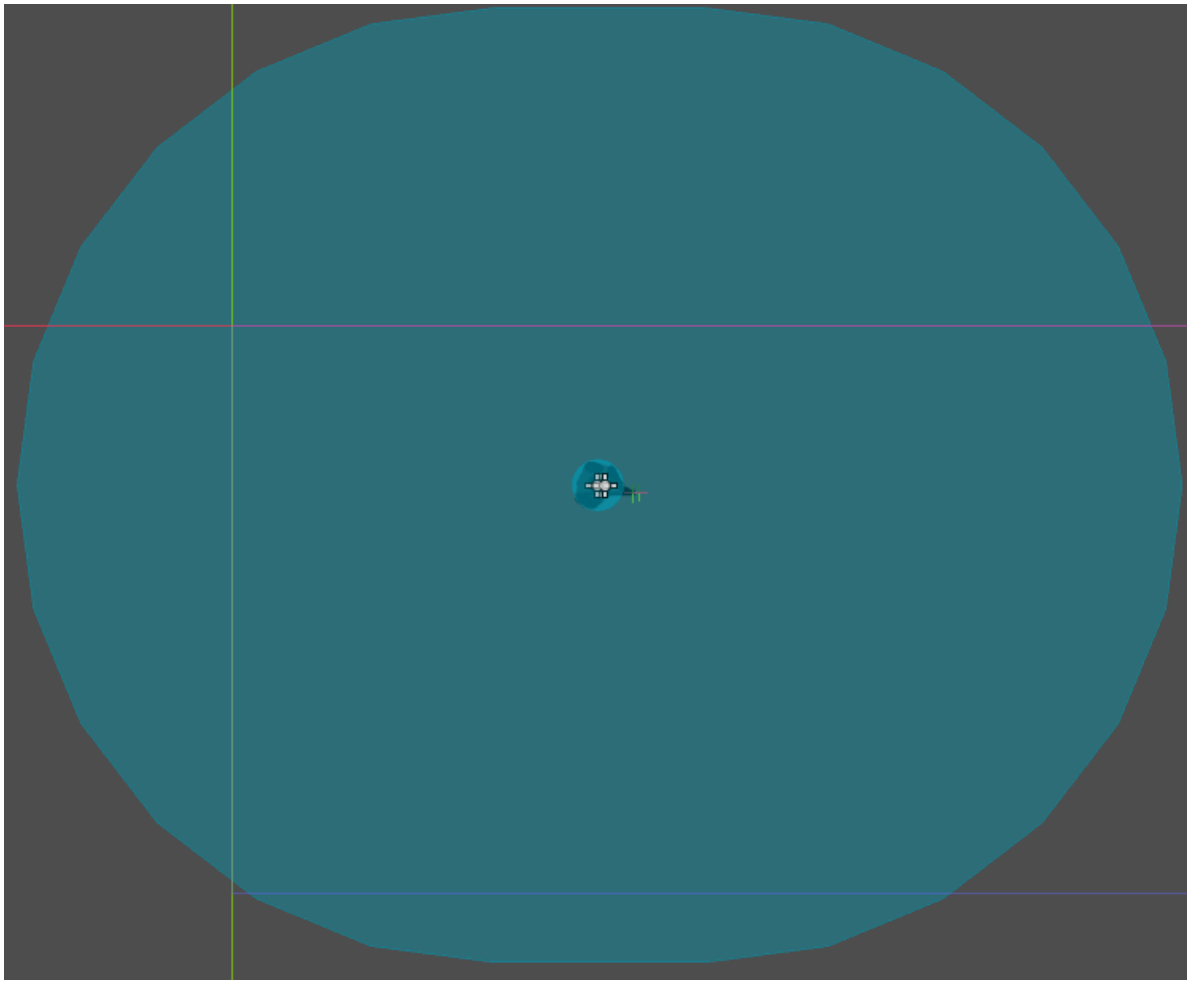


Рисунок 1.30. Загальний вигляд юніта з усіма зонами сприйняття

Після цього створюю скрипти в яких прописую:

- Зброя: в якій налаштовую ворожий патрон так, щоб він летів з певною незмінною швидкістю і не піддавався законам фізики, при цьому пролітав крізь NPC, але знищувався при зіткненні з об'єктами. При попаданні в колайдер гравця викликав метод знищення гравця та перезапуск гри.

Після цього налаштовую сам модуль зброї, в якому підключаю раніше прописаний патрон і реалізую метод стрільби для того, щоб у дулі автомата з'являвся патрон, повернутий у той бік, куди дивиться юніт. Після чого викликається функціонал кулі і вона починає рух із заданою швидкістю.

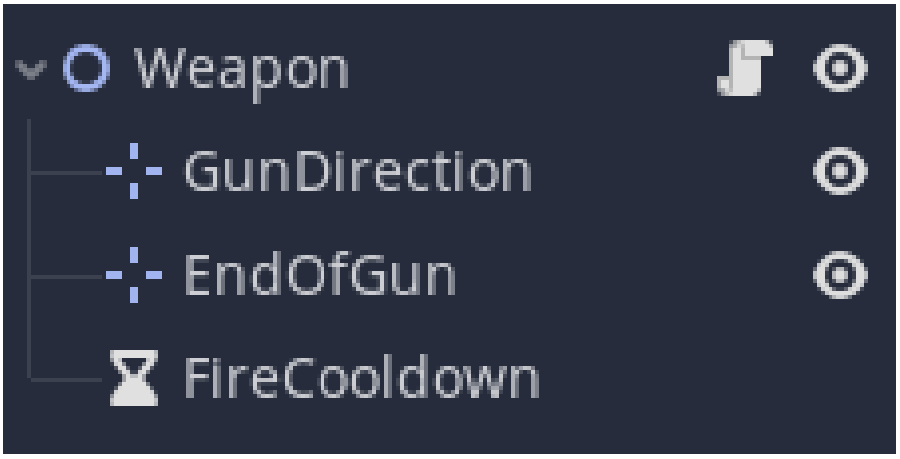


Рисунок 1.31. Структура сцени зброя

- Зони сприйняття:

1) колайдер юніта, по суті його тіло, яке буде спиратися на інші об'єкти та персонажів, тим самим не даючи проходити крізь текстури;

2) наступна зона трохи більше ніж колайдер і вона служить для того, щоб відстежувати входження шляху гравця. При контакті з кулею гравця вбиває юніта. Дана зона необхідна для того, щоб не було зависань або багів при дотику клайдера юніту з фізичним об'єктом - кулею;

3) зона огляду - зона завдяки якій юніт може відстежувати пересування гравця та зчитувати інформацію про оточення. Саме в ній прописується найбільше сигналів, і вона впливає на поведінку NPC.

- Штучний інтелект: найважливіша складова персонажа. Саме він конкретизує його як унікального унікального юніта. Цей скрипт відповідає за орієнтацію у просторі, пересування, зміну стану з патрулювання на наслідування та навпаки. У цьому скрипті пов'язуються дані наведення прицілу на гравця, постріл, точка появи патрона, швидкість стрільби та відстань стрільби. Ведеться розподіл вхідних юнітів на групи "ворог" та "союзник". Прописую швидкість пересування та зміну напрямку руху.

```
hostileVelocity = hostile.global_position.direction_to(patrolLocation) * 100

hostile.move_and_slide(hostileVelocity)

hostile.rotation=lerp(hostile.rotation,hostile.global_position.direction_to(patro
lLocation).angle(), 0.1)
```

Цей спосіб реалізації переміщення юніта є специфічним, тому що при статичній швидкості пересування гравець міг би звикнути до того, як рухається противник і це було б логічніше. Однак хоч цей спосіб реалізації пересування і є нестандартним, він має низку плюсів:

- дозволяє юніту долати різні відстані за однаковий час:

- юніт рухатиметься з різною швидкістю, що ускладнить попадання по ньому;

- за такої реалізації юніт швидше переміщатиметься по карті, на відміну від статичної швидкості.

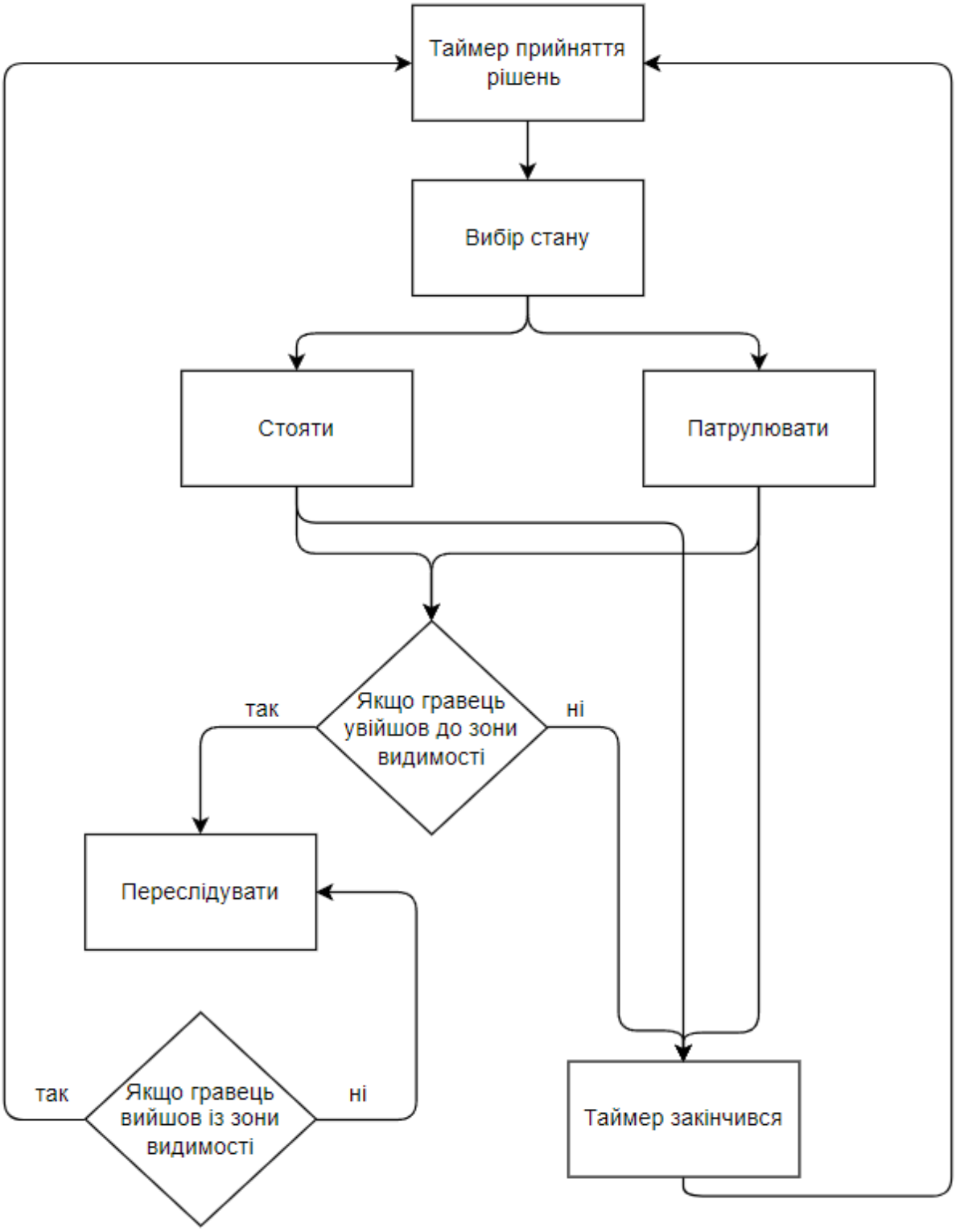


Рисунок 1.32. Функціональна схема зміни стану юніта

При обіграванні ігрових ситуацій даний штучний інтелект показав себе позитивно, так як він часто пересувається по карті, але через періодичні зупинки його пересування виглядають реалістично.

А під час бою він може як стоячи, так і пересуватися і при спостереженні за боєм гравця з декількома ворогами бій виглядає динамічно, тому що деякі юніти стоять, а деякі рухаються, при чому рухаються вони або постійно або уривчасто, що ускладнює попадання по них.

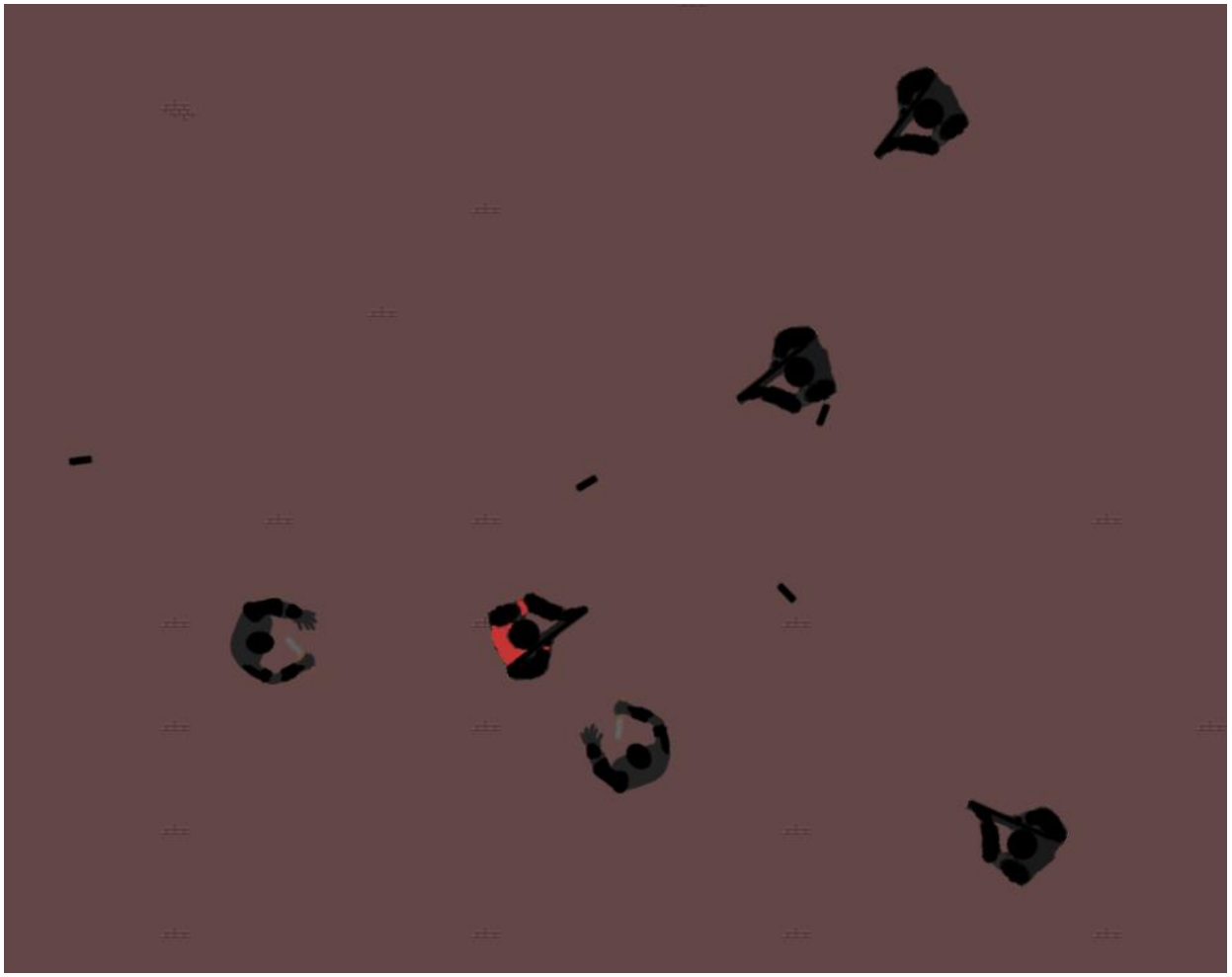


Рисунок 1.33. Бій гравця з кількома супротивниками

У коді логіка поведінки стрільців у бою побудована в такий спосіб. Що Після аналізу гравця, стрілець відразу перевіряє чи був він у русі або стояв на місці, якщо він був у русі, то продовжить його і почне стріляти, якщо стояв, то відразу відкриє вогонь. Функція стрілянини викликається у другій перевірці. Таким чином, можна передбачити першу дію юніту, однак через деякий час він змінить свій стан і може почати рухатися, навіть якщо до цього стояв:

```
if hostile.global_position.distance_to(patrolLocation) < 5:
        patrolLocationReached = true
        hostileVelocity = Vector2.ZERO
        movementTimer.start()
    if abs(hostile.rotation - angleToPlayer) < 0.5:
        weapon.shoot()
```

Наступні створюю противника ближнього бою. І його я налаштовую схожим чином, однак він матиме кординальні відмінності.

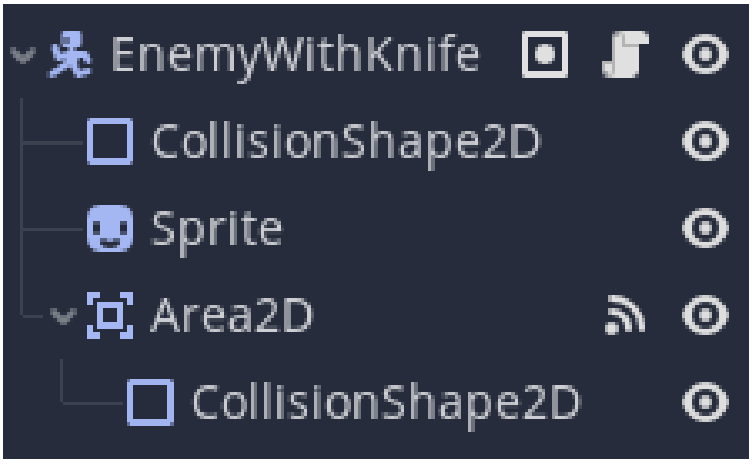


Рисунок 1.34. Структура об'єктів ворога зі зброєю ближнього бою

Для цього ворога я не прописую зайві рядки коду зі зброєю, а просто додаю можливість вбити торкання гравця. І додаю можливість завжди знати розташування гравця. Після чого скалюю його швидкість залежно від відстані до гравця. Тим самим підвищуючи швидкість за великих відстаней. Таким чином, юніт майже миттєво опинятиметься у полі зору гравця, але чим ближче він підійде тим менше буде його швидкість, це надасть гравцю адреналіну і не залишатиме негативних відчуттів від гри. А також приведе гравця в рух і не дасть йому стояти на місці. Завдяки чому гравець буде змушений ходити по карті а там розумітимуть стрілецькі юніти, які мають свій радіус огляду.

Оглядаючи структуру колайдерів та зон - даний ворог не має чим похизуватися.

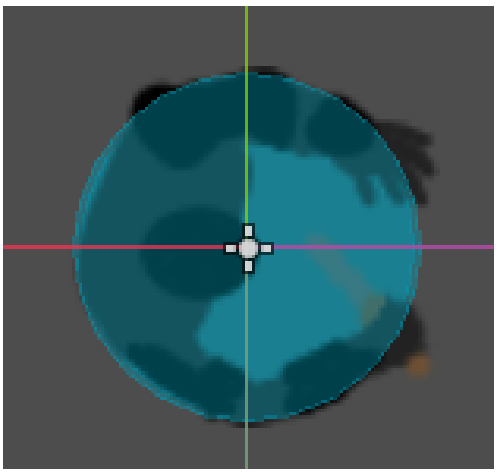


Рисунок 1.35. Загальний вигляд юніта з усіма зонами сприйняття

Рисунок 1.36. Функціональна схема поведінки супротивника зі зброєю ближнього бою

1.6 Реалізація поведінки штучного інтелекту та алгоритмів взаємодії з ігровим оточенням для персонажів 2D-гри

Спроектувавши всі алгоритми, створивши карту, персонажа та об'єкти, необхідні для написання коду штучного інтелекту, я зміг зробити два юніти, які мають набір команд, необхідний для створення бойових сцен у грі. Всі функції та логіка поведінки були написані в окремих модулях, що дозволить скористатися методологією поліморфізму при подальшому розвитку проекту для створення нових типів юнітів.

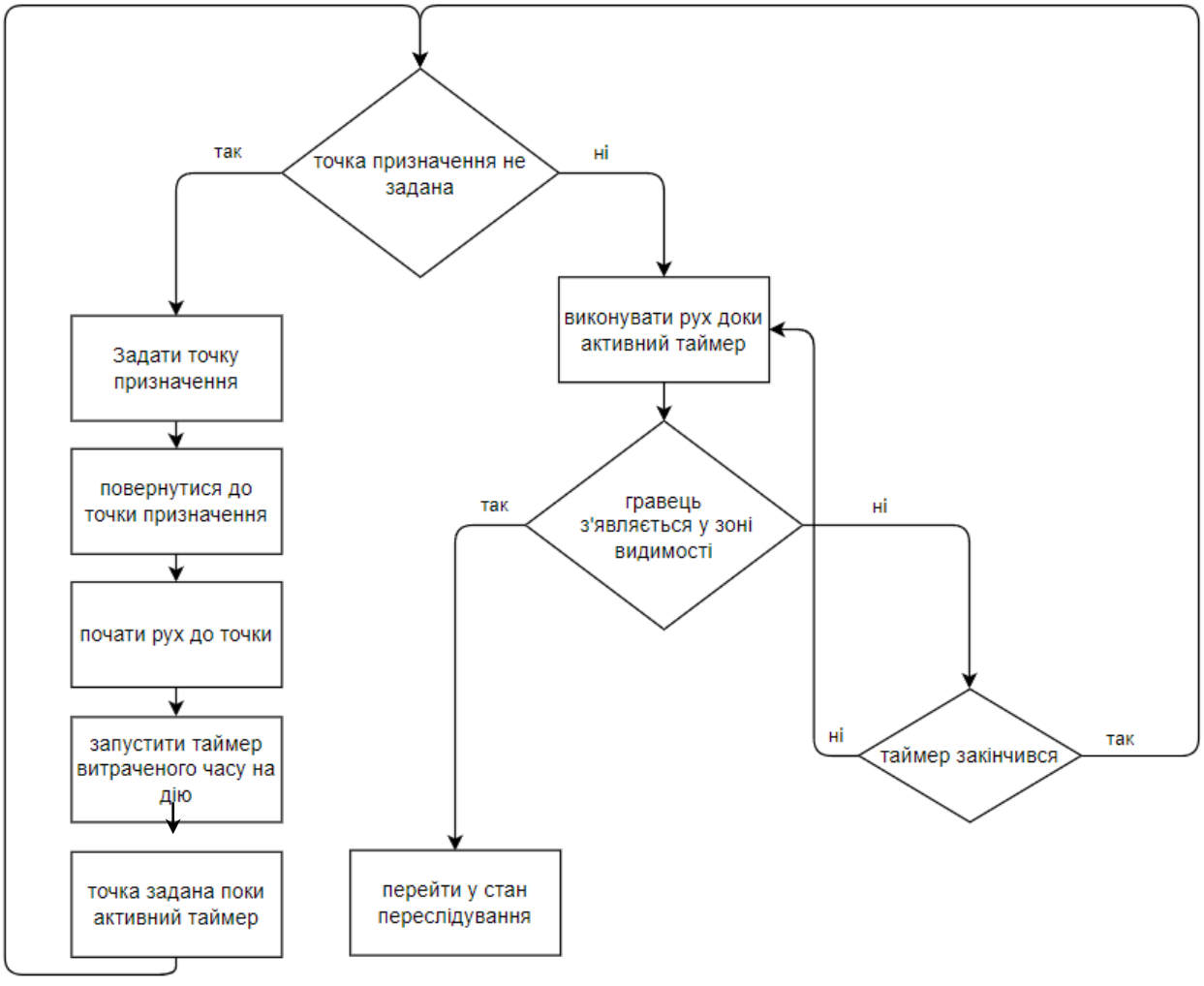


Рисунок 1.37. Функціональна схема патрулювання

Для реалізації патрулювання я використовував генерацію випадкової координати у просторі, у певній області, до якої зрештою

повинен буде дійти юніт. Поки персонаж рухається до точки - активний таймер, який не дозволяє генірувати нові точки, щоб юніт рухався природно, а не крутився на місці мене постійно пункт призначення:

```
var patrolRange = 200
    var random_x = rand_range(-patrolRange, patrolRange)
    var random_y = rand_range(-patrolRange, patrolRange)
    patrolLocation = Vector2(random_x, random_y) + origin
    patrolLocationReached = false
    hostileVelocity = hostile.global_position.direction_to (patrolLocation) * 100
```

Юніт має зону сприйняття. Яка дозволяє йому аналізувати об'єкти, що потрапляють до неї, використовуючи описаний у скрипті алгоритм станів. Завдяки ній він розрізняє супротивників і союзників користуючись сортуванням за групою "Свій-чужий". Гравець належить до групи "Player", а всі його противники до "Enemy" саме з них відбувається перевірка.

```
func _on_PlayerDetection_body_exited(body):
        if body.is_in_group("player"):
            set_state(States.PATROL)
            player = null
```

Також дана область є радіусом стрільби юніта і власне областю видимості. Завдяки чому ця зона безпосередньо впливає на його пересування по карті, тому що спираючись на отримані дані із зони юніт будує найбільш прийнятний маршрут.

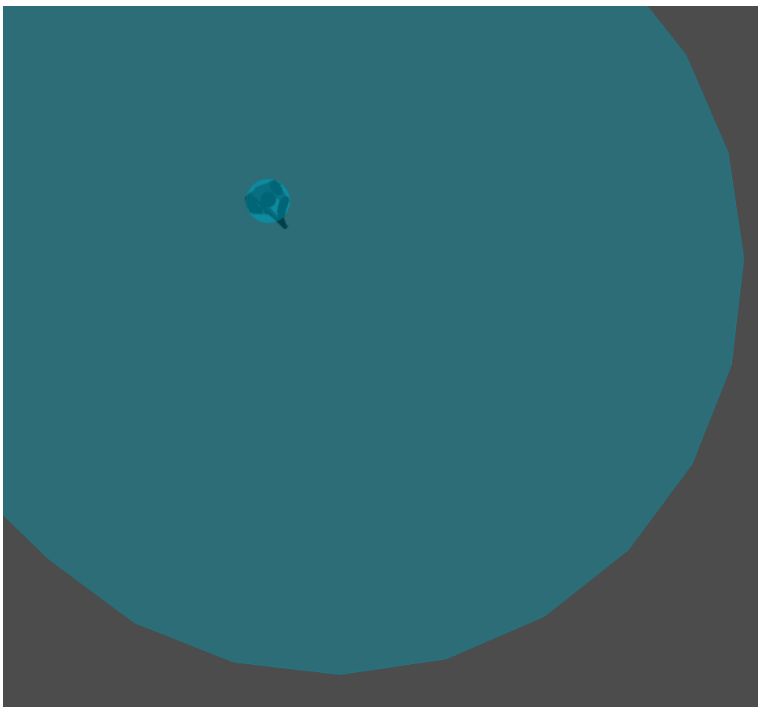


Рисунок 1.38. Область сканування та прокладення маршруту

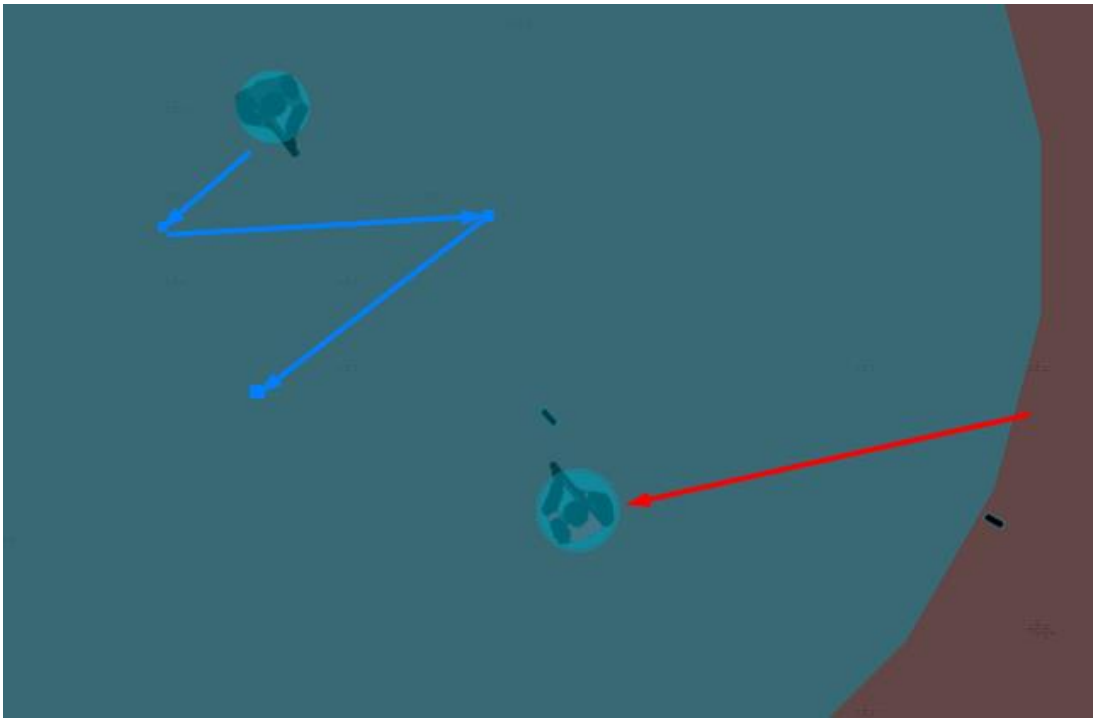


Рисунок 1.39. Вхід гравця до зони та прокладання маршруту

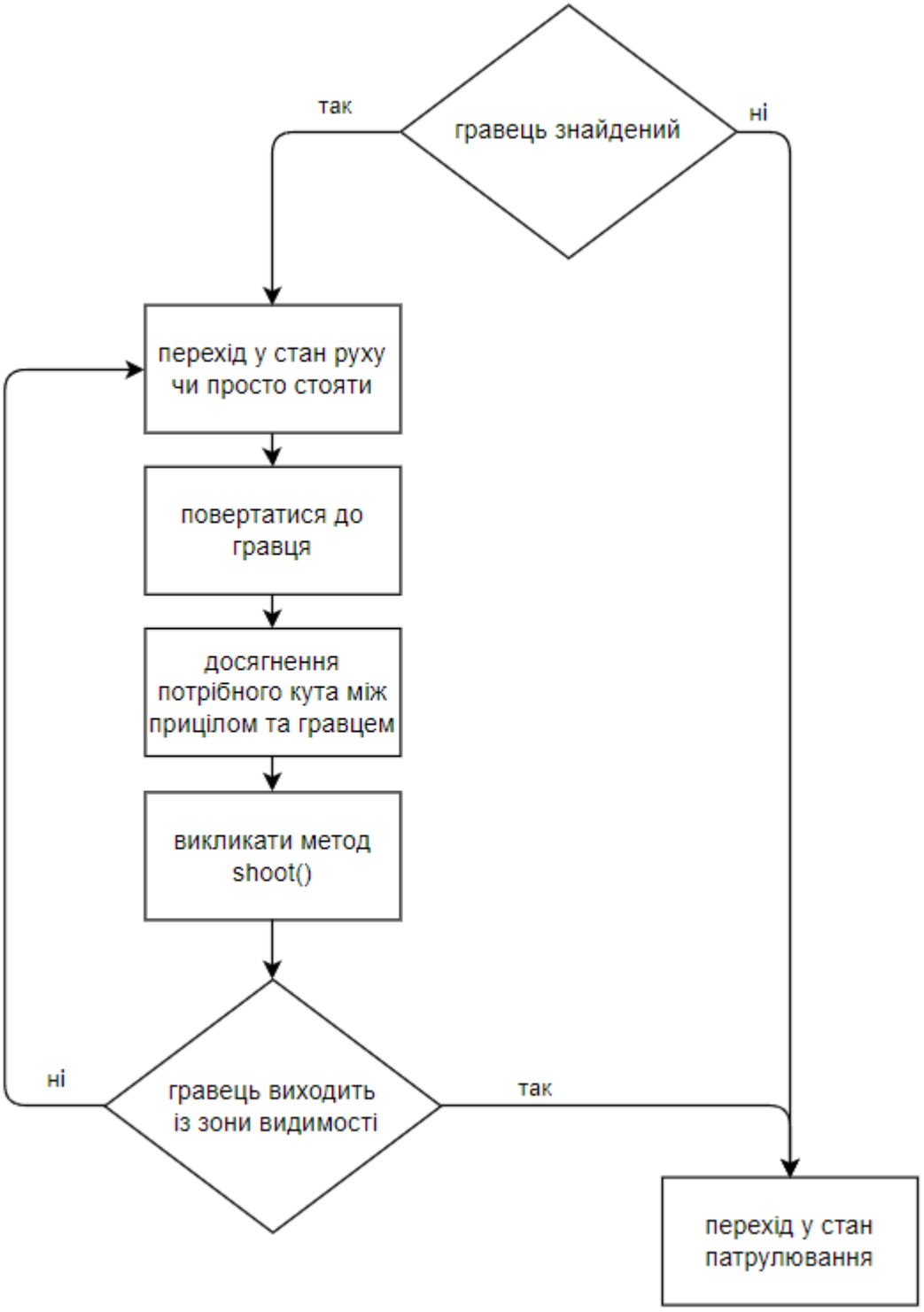


Рисунок 1.40. Функціональна схема переслідування

Поява противників на карті прописано так, щоб вони з'являлися у випадкових місцях на карті, але не за її межами, при цьому вони не можуть один в одному і в гравці, якщо точка появи збігається з місцем якогось юніту, вона генерується по новій.

```
func _get_random_spawn_position() -> Vector2:
    var enemyPosition = Vector2(rand_range(-160, 1200), rand_range(200, 1200))
    while enemyPosition.x < 400 and enemyPosition.y > -80 or enemyPosition.x < 400 and enemyPosition.y > 100:
        enemyPosition = _get_random_spawn_position() # just get a new one instead
        return enemyPosition
```

Контроль куль на карті та їх поява загалом відстежується у глобальному скрипті . Без нього було б можливе проходження куль через текстури, некоректне спрацювання скрипту на вбивство гравця та потенційну шлоду союзним юнітам.

```
signal bullet_fired(bullet, position, direction)
func bulletSpawned(enemybullet: EnemyBullet, position: Vector2, direction: Vector2):
    add_child(enemybullet)
    enemybullet.global_position = position
    enemybullet.set_direction(direction)
```

Після реалізації всіх модулів, об'єктів та сцен. Я зв'язав їх в одне ціле на карті за допомогою глобальних скриптів та зміг провести тестування гри. Далі будуть перераховані результати отримані під час тестування та міркування про можливу модернізацію програми.

Рисунок 1.41. Командна робота двох типів юнітів

Завдяки командній роботі двох штучних інтелектів гравець запросто може опинитися в пастці, загнаний спритними бігунами до майстер-стрільців, що призведе до програшу гравця.

Також було передбачено ускладнення, за допомогою спеціальної формули я прискорив появу ворогів, а також з певного моменту їх починає з'являтися більше як зображено на рисунку 1.42. Також була передбачена безпечна зона для гравця в радіусі якої не можуть з'являтися супротивники. Інакше це могло б призводити до незаслужених програшів.

З часом гра стає досить складною, що дозволяє гравцям залишатися в грі заради того, щоб ставити нові рекорди за часом виживання. І якби я розвивав проект далі, то точно б додав елементи GUI для відображення часу проведеному в раунді, таймер перезарядки зброї, ривка і найголовніше лічильник очок, адже з ним куди зручніше відстежувати свій прогрес у грі.

Рисунок 1.42. Пізній етап гри

2 ЕКОНОМІЧНА ЧАСТИНА

2.1 Резюме

В даному дипломному проекті реалізована поведінки штучного інтелекту та алгоритмів взаємодії з ігровим оточенням для персонажів 2D-гри у жанрі top down shooter. Створення механізму взаємодії між гравцем на сцені та не ігровими персонажами.

Окрім виконання поставленого завдання, завдяки використанню принципів модульної розробки, була закладена основа для майбутнього удосконалення ігрових персонажів, їх взаємодії між собою та гравцем. Гнучкі можливості мови програмування GDScript дають змогу в подальшому додавати до гри нові елементи, ігрові механіки та модернізувати алгоритми штучного інтелекту

Ефективність кожного програмного продукту визначається його якістю та ефективністю процесу розробки. Якість ПП визначається наступними складовими: з точки зору користувача; з позиції використання ресурсів; виконання вимог до програмного забезпечення.

Оцінка якості програмного продукту з точки зору користувача визначається необхідним на стадії функціонування розміром оперативної пам'яті ЕОТ, витратами машинного часу, пропускною спроможністю каналів передачі даних. Оцінка якості програмного продукту включає визначення трудомісткості і вартості його створення.

2.2 Визначення трудомісткості розробки програмного забезпечення

Тривалість розробки програмного продукту залежить від його обсягу, трудомісткості розробки, кваліфікації виконавців, а також планових термінів, визначених умовами ринку. Методом структурної аналогії по відповідних каталогах аналогів програмного забезпечення визначається обсяг програмних засобів, у тисячах умовних машинних команд програми аналога.

Таблиця 2.1 Каталог аналогів

Найменування ПП	Обсяг функції ПП – V_o, усл. машинних командах.
1. ПП автоматизації засобів по каталогу	680 – 7000
2. ПП автоматизованих розрахунків	1300 – 8600
3. ПП загальної математики і ПП імітаційного моделювання	7800 – 8800

У таблиці 2.1 представлені аналоги програмного забезпечення, функції яких, у більшому або меншому ступені, виконує розроблений програмний продукт. Для нашого варіанта виділено сірим кольором.

Вибравши аналог ПП, що містить Vo в умовних машинних командах, трудомісткості визначати на основі таблиці 2.2.

Таблиця 2.2 Трудомісткість

Обсяг ПП, тис.умов.машинних команд	Норма часу, люд/год
1.00	229
2.00	244
3.00	262

На підставі отриманого значення, по довіднику, визначається укрупнена норма часу на розробку аналога програмного забезпечення (коректується поправочним коефіцієнтом враховуючої умови розробки ПП, тобто в умовах комп'ютера, Кк=0,7÷0,8): Тар = 229 х 0,8 = 183,2 (люд/годин).

Трудомісткість програмного продукту визначається по кожному етапу розробки окремо на підставі трудомісткості аналога з урахуванням складності розробки, ступеня новизни і ступеня використання в розробці стандартних модулів на підставі формул:

$$T_{ТЗ} = T^a p \times L_1 \times K_H \quad (2.1)$$

$$T_{ТП} = T^a p \times L_2 \times K_H \quad (2.2)$$

$$T_{РП} = T^a p \times L_3 \times K_H \times K_T \quad (2.3)$$

Для розрахунку необхідні наступні коефіцієнти:

Li – питома вага і-го етапу розробки (див. табл. 2.3);

Кн – поправочний коефіцієнт, що враховує ступінь новизни (таблиця 2.4);
Кт – поправочний коефіцієнт, що враховує ступінь використання в розробці типових програм (таблиця 2.5).

Таблиця 2.3 Значення питомих коефіцієнтів трудомісткості стадії в загальній трудомісткості розробки ПП.

Код стадії	Ступінь новизни		
	А	Б	В
ТЗ (L_1)	0,15	0,12	0,12
ТП (L_2)	0,16	0,15	0,11
РП (L_3)	0,55	0,58	0,61

Таблиця 2.4 Значення поправочного коефіцієнта, що враховує ступінь новизни

Код ступеня новизни	Ступінь новизни	Значення $К_н$
А	Принципово нові ПО	1,75 – 1,2
Б	ПЗ – розвиток параметричного ряду	1,0 – 0,8
В	ПЗ маючий аналог	0,7

Таблиця 2.5 Значення коефіцієнта ступеня використання в розробці типових програм

Ступінь охоплення реалізованих функцій розроблювального ПЗ типовими програмами, %	Значення $К_т$
60 і вище	0,6
40-60	0,7
20-40	0,8
До 20	0,9

Тепер розраховуємо трудомісткість по кожному етапу окремо:

Трудомісткість технічного завдання

$$Т_{тз}=Та*L_1*К_н=183,2 \ (*0,12*0,7= 15,38 \text{ (люд/годин)}$$

Трудомісткість розробки технічного проекту

$$Т_{тп}=Та*L_2*К_н=183,2 \ (*0,11*0,7 = 14,11 \text{ (люд/годин)}$$

Трудомісткість розробки робочого проекту

$$Т_{рп}=Та*L_3*К_н*К_т=183,2 \ (*0,61*0,7*0,8= 62,58 \ \text{(люд/годин)}$$

Для подальших розрахунків визначили кількість папера, витраченого на кожен етап: технічне завдання $N_{тз}$=3 (стр), розробка ТП $N_{тп}$=19(стр), розробка робочого проекту $N_{рп}$=14 (стр), пояснювальна записка відповідно $N_{пз}$=40 (стр)

Таблиця 2.6 Розрахунок трудомісткості ПП

Найменування етапів	Розрахунок, годин.		
1.ТЗ	$T_{Ртз}$=15,38	$T_{кк}=0,7*N_{тз}$= 0,7*3=2,1	$T_{нк}=0,15*N_{тз}=0,15*3=0,45$
2.Розробка ТП	$T_{Ртп}$=14,11	$T_{кк}=0,7*N_{тп}=0,7*19=13,3$	$T_{нк}=0,15*N_{тп}=0,15*19=2,85$
3.Розробка РП	$T_{Ррп}$= 62,58	$T_{кк}=0,7*N_{рп}=0,7*14=9,8$	$T_{нк}=0,15*N_{рп}=0,15*14=2,1$
4.Розробка ПЗ	$T_{пз}=1,5**N_{пз}$= 1,5*36 =54	$T_{кк}=0,7*N_{тз}=0,7*36=25,2$	$T_{нк}=0,15*N_{пз}=0,15*36$ =5,4
Усього, в т.ч.:	207,6		
- на розробку	$\Sigma T_р$=146.1		
- контроль керівника		$\Sigma T_{кк}$=50,7	
- нормоконтроль			$\Sigma T_{нк}$=10,8

2.3 Розрахунок ціни програмного продукту

У цьому розділі для визначення ціни розраховуємо основну заробітну плату виконавців, матеріальні витрати, вартість машино – години і витрати на розробку ПО. Розрахунок основної заробітної плати виконавців приведений у таблиці 2.7. Відповідно до статті 8 «Закону про Державний бюджет України на 2021» встановлено мінімальну заробітну плату у місячному розмірі з 1 січня 2022 року - 6700 гривень; мінімальну погодинну тарифну ставку – 40,46 грн.

Таблиця 2.7. - Розрахунок основної заробітної плати виконавців.

Найменування робіт	Трудомісткість робіт, години	Погодинна тарифна ставка, грн.	Розрахунок, грн.
1.Розробка ПП	146,1	40,46	5911,21
2.Контроль керівника	50,7	80,00	4056,00
3.Нормоконтроль	10,8	70,00	756,00
Усього	-	-	ΣЗо= 10723,21

Зробимо розрахунок матеріальних витрат на розробку ПП. Розрахунок зведемо в таблицю 2.8.

Таблиця 2.8 Розрахунок матеріальних витрат на розробку ПО

Найменування матеріальних витрат	Тип, модель	Кількість	Ціна одиниці, грн.	Вартість, грн.
Папір	Лист А4	71	3.0	210,0
Разом	-	-	-	$В_{м\,і}$ =210,0
Транспортно – заготівельні Витрати (10%)				$Втр\_з = 0{,}1 \times Вм1$= 0,1*210=21,00
Усього				$В_м$=$В_{мі}$+$В_{тр\ з=}$ 231.00

На підставі отриманих даних по окремих статтях витрат складена калькуляція планової собівартості в цілому ПП за формою, приведеною в таблиці 2.9.

Таблиця 2.9 Розрахунок статей витрат планової собівартості

Стаття витрат	Значення, грн.	Формула розрахунку
1. Матеріали	231,00	$В_м$ (див. табл. 2.8.)
2. Основна заробітна плата	10723,21	$З_о$ (див. табл. 2.7.)
3.Додаткова заробітна плата	1072,32	$Зд = 0{,}1 \times Зо$ = 10723,21*0,1
4.Відрахування до єдиного фонду соціального внеску	2595,00	$Вє.с.в. = 0{,}22 \times (Зо + Зд)$ =0,22* (10723,21+1072,32)
5. Накладні витрати	4289,28	$Внак. = 0{,}4 \times Зо$ =0,4 * 10723,21
6. Повна собівартість	18910,81	$С_{пов}$= $В_м$ +$З_о$ + $З_д$ +Вє.с.в. + Внак.= 231,00+10723,21+1072,32+2595,00 + 4289,28

Розмір прибутку, що включен в ціну, визначаємо по наступній формулі:

$П=(С_п*Р)/100=(18910,81*10)/100=1891,08$грн

Де р – плановий рівень рентабельності (10-15%).

Оптова ціна (кошторисна вартість) визначається по формулі:

$Ц_о = С_п + П$ = 18910,81+1891,08 = 20801,89 грн;

Ціна реалізації розробленого програмного продукту, становитиме:

$Ц_р = Ц_о + ПДВ$ = 20801,89 +20801,89 *0.2 = =24962, 27 грн;

3 ОХОРОНА ПРАЦІ

У даному розділі ДП вирішується питання охорони праці програміста на стадії реалізації поведінки штучного інтелекту та алгоритмів взаємодії з ігровим оточенням для персонажів 2D-гри у жанрі top down shooter.

3.1 Аналіз небезпечних і шкідливих факторів, що впливають на програміста при розробці даного програмного комплексу

Робота з ПК супроводжується підвищеним ступенем напруженості трудового процесу. При систематичному впливі виробничих факторів, які не відповідають нормативним показникам, зростає рівень професійно зумовленої захворюваності працюючих та можуть виникнути професійні захворювання органів зору, руху, нервової системи. На робочому місці користувача ПК виникають небезпечні та шкідливі фактори: підвищений рівень шуму, несприятливі мікрокліматичні умови, недостатній рівень освітленості, шкідливі речовини.

3.2 Гігієнічні вимоги до виробничого середовища

Вивчення умов праці на робочому місці користувача ПК є необхідною умовою запобігання негативних наслідків впливу небезпечних та шкідливих факторів.

3.2.1 Вимоги до приміщення

Слід відзначити, що площа одного робочого місця оператора ПК не повинна бути меншою за 6м^2, а об'єм не менший за 20м^3 , Також приміщення повинно мати як штучне, так і природне освітлення. Воно має бути забезпечено системою опалення, кондиціювання повітря або припливно-витяжною вентиляцією.

3.2.2 Освітлення

Освітлення приміщення, у якому працює користувач персонального комп’ютера (програміст), використовується змішане освітлення, тобто сполучення природного й штучного освітлення. Природне освітлення - здійснюється через

вікна в зовнішніх стінах будинку. Штучне освітлення - використовується при недостатньому природному освітленні й здійснюється за допомогою двох систем: загального й місцевого освітлення. Для загального освітлення приміщення, де перебуває робоче місце програміста, використовуються газорозрядні лампи типу ЛД. Нормами для даних робіт установлена необхідна освітленість робочого місця E_H=300 лк (для робіт високої точності, коли найменший розмір об'єкта розрізнення дорівнює 0,3 – 0,5 мм).

3.2.3 Шум

У робочих приміщеннях основними джерелами акустичних шумів є шуми ПЕОМ. ЕОМ є також джерелами шумів електромагнітного походження (коливання елементів електромеханічних пристроїв під впливом змінних магнітних полів). Допустимий еквівалентний рівень шуму для робочого місця оператора складає 65 дБА. Під час виконання робіт з ПК у виробничих приміщеннях значення характеристик вібрації на робочих місцях мають не перевищувати допустимі відповідно до ДСанПіН 3.3.2.007-98, ДСН 3.3.6-039-99

3.2.4 Вимоги до організації робочого місця працівника

Створення зручного робочого місця повинно бути одним з головних пріоритетів роботодавця. Створення зручного робочого місця повинно бути одним з головних пріоритетів роботодавця.

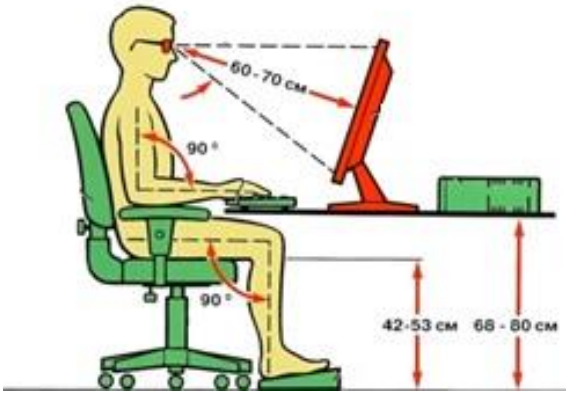


Рисунок 3.1 Правильне розміщення робочого місця

Для людини, що працює, потрібно створити санітарні умови, які б дали змогу їй плідно працювати, не перевтомлюючись та зберігати своє здоров'я. Для цього треба, щоб енергетичні витрати при праці компенсувалися відпочинком та умовами оточуючого середовища. Ці умови створюються забезпеченням для працюючого:

1) зручного робочого місця;
2) чистого повітря;
3) нормованої освітленості;
4) захисту від шуму та вібрацій;
5) захисту від дії шкідливих речовин та випромінювань;
6) робочим одягом та різними засобами індивідуального захисту;
7) побутовими приміщеннями та спеціальними службами, що призначені
8) створювати безпечні та нормальні умови праці.

Обладнання і організація робочого місця з ВДТ мають забезпечувати відповідність конструкцій всіх елементів робочого місця та їх взаємного розташування ергономічним вимогам з урахуванням характеру і особливостей трудової діяльності (ДСаПіН 3.3.2.-007-08). Конструкція робочого місця й взаємне розташування всіх його елементів (сидіння, органи керування, засобу відображення інформації) відповідають антропометричним, фізіологічним і психологічним вимогам, а також характеру роботи. Конструкція робочих меблів повинна забезпечувати можливість індивідуального регулювання відповідно росту працюючих для підтримки зручної пози.

Робочий стіл повинен бути пофарбований матовою фарбою. Дисплей розташований так, що його верхній край перебуває на рівні очей на відстані близько 70 см, що укладається в у припустимі рамки від 60 до 90 см

Робоче місце розташоване перпендикулярно віконним прорізам, це зроблено з тією метою, щоб виключити пряму й відбиту мерехтливість екрана від вікон і приладів штучного освітлення, якими є лампи накалювання. Частота мерехтіння екрана $f_{мер}$=100 Гц, що відповідає умові $f_{мер}$>70 Гц.

3.2.5 Мікроклімат

Найбільш значним фактором продуктивності й безпеки праці є виробничий мікроклімат, що характеризується температурою й вологістю повітря, швидкістю його руху, а також інтенсивністю радіації, і повинен відповідати ГОСТ 12.1.005-88 і СНиП 2.04.05-86, тому значення параметру мікроклімату повинно становити: температуру повітря від 18-22 градусів Цельсію, вологість повітря від 40%-60%, та швидкість повітря від 0,1-0,2 м/с.

Для підтримки в приміщеннях нормального, що відповідає гігієнічним вимогам складу повітря, видалення з нього шкідливих газів, пару і пилу використають вентиляцію

Механічна вентиляція (кондиціонери вентилятори і т.ін.) залежно від напрямку руху повітряних потоків, може бути витяжною, нагнітаючої і нагнітаючо-витяжною. При природній вентиляції (за допомогою вікон) повітря надходить у приміщення й віддаляється з нього внаслідок різниці температур, а також під дією вітру.

3.2.6 Електробезпека

Приміщення, де використовуються імпульсні джерела живлення відповідно до ОНТП24-86 і ПУЕ-87 відноситься до класу приміщень без підвищеної небезпеки поразки персоналу електричним струмом, оскільки відносна вологість повітря не перевищує 75%, температура не більш 35ºС, відсутні хімічно агресивні середовища. Живлення електроприладів усередині приміщення здійснюється від двухфазної мережі з заземленою нейтралю напругою 220 В і частотою 50 Гц із використанням автоматів токового захисту. У приміщенні повинна бути застосована схема заземлення.

Заземлення повинно бути зроблено за допомогою гнучкого сплетеного мідного проводу діаметром порядку 1,5 мм2. Для зменшення значень напруг дотику і відповідних їм величин струмів, при нормальному й аварійному режимах роботи устаткування необхідно виконати повторне захисне заземлення нульового прово-

ду. Відповідно до ГОСТ-12.2.007.0-75 все устаткування (крім ЕОМ - II клас) відноситься до I класу, воно має робочу ізоляцію відповідно до вимог ГОСТ 12.1.009-76. Підключення устаткування виконане відповідно до вимог ПБЕ та ПУЕ. Додаткових заходів по електробезпечності не потрібно.

3.3 Пожежна безпека

Коли від пожежі захищаються приміщення з персональними комп'ютерами, то слід ураховувати специфіку вогнегасних речовин у вогнегасниках, які призводять під час гасіння до псування обладнання. Ці приміщення рекомендується оснащувати вуглекислотними вогнегасниками з урахуванням граничнодопустимої концентрації вогнегасної речовини.

Будинки, споруди, приміщення, технологічні установки повинні бути забезпечені первинними засобами пожежогасіння: вогнегасниками, ящиками з піском, покривалами з негорючого теплоізоляційного полотна, грубововняної тканини чи повсті, іншим пожежним інструментом, які використовуються для локалізації і ліквідації пожеж у початковій стадії їхнього розвитку.

Для зазначення місцезнаходження первинних засобів пожежогасіння слід установлювати відповідні знаки згідно з чинними державними стандартами. Знаки слід розміщувати на видних місцях на висоті 2-2,5 м від рівня підлоги як у середині, так і поза приміщеннями (у разі потреби).

Переносні вогнегасники повинні розміщуватися шляхом:

1) навішування на вертикальні конструкції на висоті не більше 1,5 м від рівня підлоги до нижнього торця вогнегасника і на відстані від дверей, достатній для її повного відчинення;

2) установлення в пожежні шафи пожежних кранів, або у спеціальні тумби; 3) навішування вогнегасників на кронштейни, розміщення їх у тумбах або пожежних шафах повинне забезпечувати можливість прочитання маркувальних написів на корпусі.

ВИСНОВКИ

Метою дипломного проектування було реалізація поведінки штучного інтелекту та алгоритмів взаємодії з ігровим оточенням для персонажів 2D-гри у жанрі top down shooter. Створення механізму взаємодії між гравцем на сцені та не ігровими персонажами.

Для виконання поставленої задачі необхідно було створити 2D-гру в жанрі top down shooter. Для цієї цілі було використано ігровий двигун GODOT, що має широкий і зрозумілий функціонал, а також безкоштовну ліцензію. Написання коду проводилось через середу розробки GODOT.

Виконання роботи розпочалось з аналізу питання, продумування концепції гри, пошуку необхідних даних, пошуку спрайтів, проектування самої гри, алгоритмів поведінки, а також майбутнього ігрового персонажа. Також було проведено проектування взаємодії елементів із персонажами гри, їх взаємозв’язків. Виконання реалізації було проведено згідно із спроектованими функційними схемами та алгоритмами.

В результаті було створено 2D-гру в жанрі top down shooter, в якому був реалізований головний ігровий функціонал, не ігрові персонажі та об’єкти, імплементовано алгоритми штучного інтелекту для не ігрових персонажів, механізми їх взаємодії з оточенням на гравцем. Поставлена мета дипломного проектування виконана.

Окрім виконання поставленого завдання, завдяки використанню принципів модульної розробки, була закладена основа для майбутнього удосконалення ігрових персонажів, їх взаємодії між собою та гравцем. Гнучкі можливості мови програмування GDScript дають змогу в подальшому додавати до гри нові елементи, ігрові механіки та модернізувати алгоритми штучного інтелекту.

ПЕРЕЛІК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Beginning Game Development with Godot / за ред. Maithili Dhule. Singapore: Apress, 2022. 447 с.
2. Godot Engine Game Development Projects / за ред. Chris Bradfield. NewYorker: Packt Publishing, 2018. 300 с.
3. Godotengine.org: [Веб-сайт]. URL: https://docs.godotengine.org (дата звернення: 07.06.2023).
4. Godottutorials.com: [Веб-сайт]. URL: https://godottutorials.com/ (дата звернення: 07.09.2023
5. Gdquest.com: [Веб-сайт]. URL: https://www.gdquest.com/tutorial/godot/learning-paths/beginner/ (дата звернення: 07.06.2023).
6. Youtube.com:_jmbiv_Godot_Top-down_Shooter_Tutorial: [Веб-сайт]. URL:https://www.youtube.com/watch?v=gXkkNSfxLRI&list=PLpwc3ughKbZexDyPexHN2MXLliKAovkpl (дата звернення: 07.09.2023).
7. Lucidchart.com: [Веб-сайт]. URL: https://www.lucidchart.com/pages/ (дата звернення: 07.06.2023).
8. Gamedev.net: [Веб-сайт]. URL: https://www.gamedev.net/articles/programming/artificial-intelligence/the-totalbeginners-guide-to-game-ai-r4942/ (дата звернення: 07.06.2023).
9. Youtube.com: PlugWorld: [Веб-сайт]. URL: https://youtu.be/YqkfTjXzA9w (дата звернення: 07.06.2023).

ДОДАТОК А ФРАГМЕНТ МОДУЛІВ ПРОГРАМИ

Скрипт World

```
extends Node2D

onready var bulletManager = $BulletManager

var enemyMelee = preload("res://EnemyWithKnife.tscn")
var enemyShooter = preload("res://EnemyWithGun.tscn")

func _ready():
	randomize()
	GlobalSignals.connect("bullet_fired", bulletManager, "bulletSpawned")

func _get_random_spawn_position() -> Vector2:
	var enemyPosition = Vector2(rand_range(-160, 1200), rand_range(200, 1200))
	while enemyPosition.x < 400 and enemyPosition.y > -80 or enemyPosition.x < 400 and enemyPosition.y > 100:
		enemyPosition = _get_random_spawn_position() # just get a new one instead
	return enemyPosition

func _on_EnemySpawnTimer_timeout():
	GlobalSpawner.instance_node(enemyMelee, _
	get_random_spawn_position(), self)
	GlobalSpawner.instance_node(enemyShooter, _
	get_random_spawn_position(), self)
	if $EnemySpawnTimer.wait_time < 2:
		GlobalSpawner.instance_node(enemyMelee, _
		get_random_spawn_position(), self)


func _on_SpawnBoost_timeout():
	if $EnemySpawnTimer.wait_time > 1:
		$EnemySpawnTimer.wait_time *= 0.95
```

Скрипт Weapon

```
extends Node2D
class_name Weapon

export (PackedScene) var EnemyBullet
```

```
onready var gunDirection = $GunDirection
onready var endOfGun = $EndOfGun
onready var fireCooldown = $FireCooldown

func shoot():
	if fireCooldown.is_stopped():
		var bulletInstance = EnemyBullet.instance()
		bulletInstance.position = endOfGun.global_position
		var direction = (gunDirection.global_position - endOfGun.global_position).normalized()
		GlobalSignals.emit_signal("bullet_fired", bulletInstance, endOfGun.global_position, direction)
		fireCooldown.start()
```

Скрипт Shooter_AI

```
extends Node2D

signal state_changed(newState)

enum States{
STAND,
PATROL,
ENGAGE
}

onready var playerDirection = $PlayerDetection
onready var patrolTimer = $PatrolTimer
onready var movementTimer = $MovementTimer

var currentState: int = -1 setget set_state
var hostile: KinematicBody2D = null
var player: Player = null
var weapon: Weapon = null
var origin: Vector2 = Vector2.ZERO
var patrolLocation: Vector2 = Vector2.ZERO
var patrolLocationReached: bool = false
var hostileVelocity: Vector2 = Vector2.ZERO

func _ready() -> void:
	set_state(States.PATROL)

func _physics_process(delta: float) -> void:
	match currentState:
	States.STAND:
```

```
                pass
        States.PATROL:
                if not patrolLocationReached:
                        hostile.move_and_slide(hostileVelocity)
                        hostile.rotation = lerp(hostile.rotation,
                        hostile.global_position.direction_to(patrolLocation).angle()
                        , 0.1)
                        if hostile.global_position.distance_to(patrolLocation) < 5:
                                patrolLocationReached = true
                                hostileVelocity = Vector2.ZERO
                                patrolTimer.start()
        States.ENGAGE:
                if player != null:
                        hostile.move_and_slide(hostileVelocity)
                        var angleToPlayer = hostile.global_position.direction_to
                        (player.global_position).angle()
                        hostile.rotation = lerp(hostile.rotation, angleToPlayer, 0.1)
                        if hostile.global_position.distance_to(patrolLocation) < 5:
                                patrolLocationReached = true
                                hostileVelocity = Vector2.ZERO
                                movementTimer.start()
                        if abs(hostile.rotation - angleToPlayer) < 0.5:
                                weapon.shoot()
        _:
                print("Error. State not found.")

func set_state(newState: int):
        if newState == currentState:
                return

        if newState == States.PATROL:
                origin = global_position
                patrolTimer.start()
                patrolLocationReached = true

        currentState = newState
        emit_signal("state_changed", currentState)

func initialize(hostile, weapon: Weapon):
        self.hostile = hostile
        self.weapon = weapon

func _on_PlayerDetection_body_shape_entered(body_rid, body,
body_shape_index, local_shape_index):
```

```
        if body.is_in_group("player"):
                set_state(States.ENGAGE)
                player = body
                if body.is_in_group("enemy"):
                        set_state(States.PATROL)

func _on_PlayerDetection_body_exited(body):
        if body.is_in_group("player"):
                set_state(States.PATROL)
                player = null

func Muvement():
        var patrolRange = 200
        var random_x = rand_range(-patrolRange, patrolRange)
        var random_y = rand_range(-patrolRange, patrolRange)
        patrolLocation = Vector2(random_x, random_y) + origin
        patrolLocationReached = false
        hostileVelocity = hostile.global_position.direction_to(patrolLocation) *
        100

func _on_PatrolTimer_timeout():
        Muvement()

func _on_Movement_timeout():
        Muvement()
```

Скрипт Player

```
extends KinematicBody2D
class_name Player

onready var fireCooldown = $FireCooldown
onready var dash = $Dash
onready var dashCooldown = $DashCooldown
onready var endOfGun = $EndOfGun

var dash_used = false
var speed = 300
var bullet_speed = 1000
var bullet = preload("res://Bullet.tscn")

func _physics_process(delta):
        var motion = Vector2()

        if Input.is_action_pressed("up"):
```

```
            motion.y -= 1
        if Input.is_action_pressed("down"):
            motion.y += 1
        if Input.is_action_pressed("left"):
            motion.x -= 1
        if Input.is_action_pressed("right"):
            motion.x += 1
        if (Input.is_action_pressed("dash") && dash_used == false):
            dashCooldown.start()
            dash.start()
            speed = 1000
            dash_used = true

        motion = motion.normalized();
        motion = move_and_slide(motion * speed)
        look_at(get_global_mouse_position())

        if Input.is_action_just_pressed("shoot"):
            fire()

func fire():
    if fireCooldown.is_stopped():
        var bullet_instance = bullet.instance()
        bullet_instance.position = endOfGun.global_position
        bullet_instance.rotation_degrees = rotation_degrees
        bullet_instance.apply_impulse(Vector2(),    Vector2(bullet_speed,
        0).rotated(rotation))
        get_tree().get_root().call_deferred("add_child", bullet_instance)
        fireCooldown.start()

func kill():
    get_tree().reload_current_scene()

func _on_Area2D_body_entered(body):
    if "Enemy" in body.name:
        kill()

func _on_Dash_timeout():
    speed = 300

func _on_DashCooldown_timeout():
    dash_used = false
```

Скрипт GlobalSpawner

```
extends Node

func instance_node(node, location, parent):
	var node_instance = node.instance()
	parent.add_child(node_instance)
	node_instance.global_position = location
	return node_instance
```

Скрипт GlobalSignals

```
extends Node

signal bullet_fired(bullet, position, direction)
```

Скрипт EnemyWithKnife

```
extends KinematicBody2D
class_name EnemyWithKnife

var player: Player = null
var motion = Vector2.ZERO

func _physics_process(delta):
	var Player = get_parent().get_node("Player")
	position += (Player.position - position)/75
	look_at(Player.position)
	move_and_collide(motion)

func _on_Area2D_body_entered(body):
	if "Bullet" in body.name:
		queue_free()
```

Скрипт EnemyWithGun

```
extends KinematicBody2D
class_name EnemyWithGun

onready var shooter_AI = $Shooter_AI
onready var weapon = $Weapon

var motion = Vector2()

func _ready() -> void:
	shooter_AI.initialize(self, weapon)

func _on_Area2D_body_entered(body):
```

```
	if "Bullet" in body.name:
		queue_free()
```

Скрипт EnemyBullet

```
extends Area2D
class_name EnemyBullet

var speed = 10
var direction = Vector2.ZERO

func _physics_process(delta: float) -> void:
	if direction != Vector2.ZERO:
		var velocity = direction * speed
		global_position += velocity

func set_direction(direction: Vector2):
	self.direction = direction
	rotation += direction.angle()

func _on_KillTimer_timeout():
	queue_free()

func _on_EnemyBullet_body_entered(body: Node) -> void:
	if body.has_method("kill"):
		get_tree().reload_current_scene()
	if  not "Enemy" in body.name:
		queue_free()
```

Скрипт BulletManager

```
extends Node2D

func bulletSpawned(enemybullet: EnemyBullet, position: Vector2, direction: Vector2):
	add_child(enemybullet)
	enemybullet.global_position = position
	enemybullet.set_direction(direction)
```

Скрипт Bullet

```
extends RigidBody2D

onready var killTimer = $KillTimer

func _ready() -> void:
	killTimer.start()
```

ДОДАТОК Б СЛАЙДИ МУЛЬТИМЕДІЙНОЇ ПРЕЗЕНТАЦІЇ

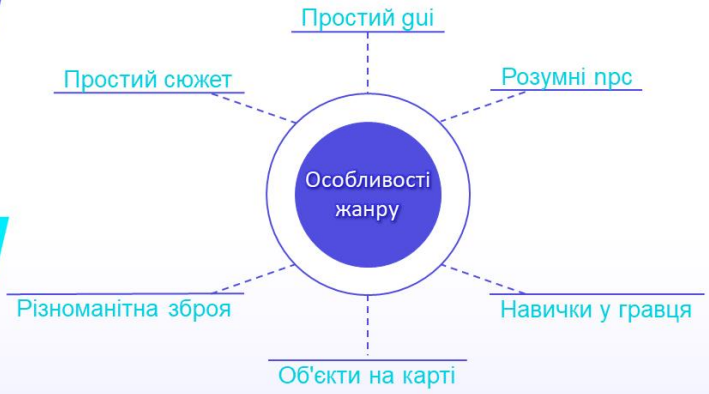
Реалізація поведінки штучного інтелекту та алгоритмів взаємодії з ігровим оточенням для персонажів 2D-гри у жанрі top down shooter

Дипломний проект студента: Мірошкіна Андрія Ігоровича

Керівник дипломного проекту: Джабраілов Д. В.

Особливості ігрового жанру top down shooter

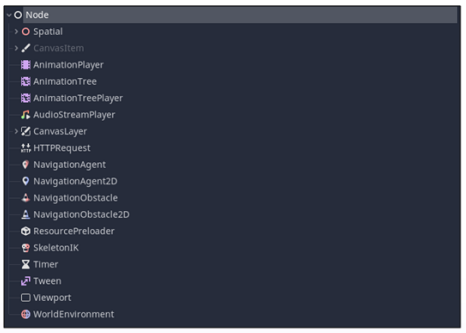
Даний жанр з'явився майже разом з відеоіграми і його можна вважати родоначальником багатьох жанрів наявних на сьогоднішній день.



Приклади ігор

Особливості Godot Engine – приклади ігор що базуються на ньому

Godot Engine – об'єктно орієнтований, тому об'єкти мають зрозумілу користувача ієрархію розміщення. У godot все є сценами, будь то персонаж гравця чи ігрова карта. А самі сцени складаються з вузлів, а ось вузли мають досить велике дерево різних об'єктів.

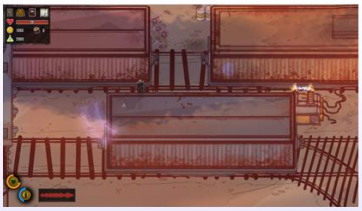


Щодо редактора сцен, я б відзначив посилальні теги для швидкого переходу між етапами розробки сцени:

- *Тег полів;*
- *Тег вузлів;*
- *Тег скриптів;*
- *Тег відображення;*

Особливості Godot Engine – приклади ігор що базуються на ньому

На Godot розробляють переважно інді проекти. І як приклади я обрав дві гри: Lumencraft та The Zone: Stalker Stories. Обидві гри мають елементи Top down shooter



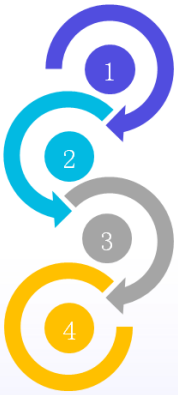
Етапи створення макету 2D-гри в жанрі top down shooter

Середовище розробки

Так як у мене інді проект, то Godot Engine ідеально підходить для розробки одному.

Тестування

На етапі тестування я перевіряв створений функціонал і вносив виправлення за потреби.



Ідея

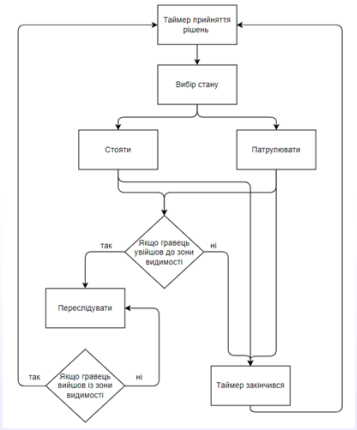
Спочатку потрібно визначиться з сеттингом, придумати де відбуватимуться події та обміркувати основні механіки гри.

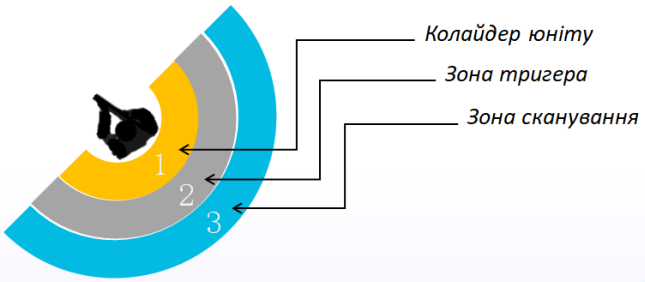
Розробка

На даному етапі я проектував ігрову локацію, створював персонажа гравця та прописував алгоритм дій для штучного інтелекту.

Опис реалізованих елементів поведінки штучного інтелекту

Для реалізації поведінки штучного інтелекту я використовував алгоритм та зону сканування. Яка дозволяє йому аналізувати об'єкти, що потрапляють до неї, використовуючи описаний у скрипті алгоритм станів та розподіляти юнітів за методикою "Свій-чужий".





Опис взаємодії між ігровими персонажами

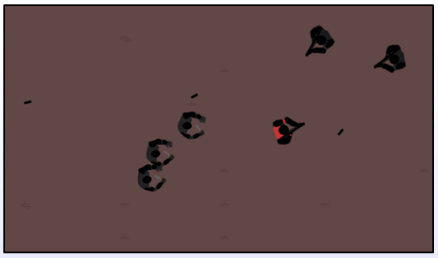
Ворог
Патрулює місцевість чи стоїть на місці.

Зона сканування
Гравець увійшов до зони сканування. Відпрацювання сигналу

Алгоритм станів
Спрацювання алгоритму зміни стану.

Противники
Противники, що знаходяться в зоні сканування, теж нападуть на гравця

Скріншоти реалізованої гри



ВСП «ОДЕСЬКИЙ ТЕХНІЧНИЙ ФАХОВИЙ КОЛЕДЖ ОНТУ»

ВІДГУК

керівника на дипломний проект здобувача (здобувачки) освіти

відділення комп'ютерних систем

Мірошкіна Андрія Ігорович

(прізвище, ім`я та по батькові)

Спеціальність: 121 "Інженерія програмного забезпечення"

Освітня програма: «Розробка програмного забезпечення»

Тема дипломного проекту: Реалізація поведінки штучного інтелекту та алгоритмів взаємодії з ігровим оточенням для персонажів 2D-гри у жанрі top down shooter

ХАРАКТЕРИСТИКА ДИПЛОМНОГО ПРОЕКТУ

а) обсяг і якість виконання проекту (графічного матеріалу і розрахунково-пояснювальної записки) Дипломний проект виконано відповідно технічному завданню. Пояснювальна записка містить 71 сторінок. У пояснювальній записці виконано опис предметної області, способи реалізації штучного інтелекту в іграх. Проведено проектування та реалізація поведінки штучного інтелекту для гри в жанрі top down shooter. Графічна частина складається з 8 слайдів мультимедійної презентації, які передбачені технічним завданням. Якість виконання пояснювальної записки та графічної частини добра, розробку виконано в повному обсязі.

б) самостійність роботи над проектом: Протягом всього строку дипломного проектування та переддипломної практики здобувач освіти Мірошкін А.І. поступово та послідовно виконував всі етапи розробки. Всі роботи здобувач освіти виконував самостійно, з оглядом на рекомендації керівника.

в) теоретична підготовка випускника (випускниці): Здобувач освіти Мірошкін А.І. під час роботи над дипломним проектом вивчив достатню кількість літературних джерел та матеріалів за даною тематикою.

Вважаю, що теоретична підготовка дипломника добра і він готовий до захисту дипломного проекту

г) вміння розв'язувати виробничі та конструкторські питання

Під час дипломного проектування здобувач освіти Мірошкін А.І. мав змогу самостійно приймати рішення з способів та варіантів реалізації штучного інтелекту в іграх, та показав вміння організовано працювати над поставленим завданням, проводити розробку коду за допомогою GDScript та працювати у ігровому двигуні Godot Engine.

Оцінка розрахункової частини (5) Відмінно

Оцінка графічної частини (4) Добре

Загальна оцінка (5) Відмінно

Прізвище, ім'я, по батькові керівника дипломного проекту

Джабраілов Дмитро Володимирович

Місце роботи і посада керівника дипломного проекту

ВСП "Одеський технічний фаховий коледж ОНТУ", викладач комісії комп'ютерних технологій та програмної інженерії

Підпис

«09» 06 2023 р.

ДОЗВІЛ
НА РОЗМІЩЕННЯ
ВИПУСКНГО ДИПЛОМНОГО ПРОЕКТА
В ЕЛЕКТРОННОМУ РЕПОЗИТАРІЇ ВСП «ОТФК ОНТУ»

Ми, що нижче підписалися,

Мірошкін Андрій Ігорович,
здобувач освіти гр. 4РП-06, та

Джабраілов Дмитро Володимирович,
керівник дипломного проекту,

не заперечуємо щодо розміщення електронного варіанту пояснювальної записки до випускного дипломного проекту молодшого спеціаліста на тему:

«Реалізація поведінки штучного інтелекту та алгоритмів взаємодії з ігровим оточенням для персонажів 2D-гри у жанрі top down shooter» (автор роботи – Мірошкін А.І., керівник роботи – Джабраілов Д.В.)

виконаного у ВСП «Одеський технічний фаховий коледж Одеського національного технологічного університету» в 2023 році, у повному обсязі в електронному репозитарії ВСП «ОТФК ОНТУ» для вільного доступу через мережу Інтернет.

Несемо відповідальність за ідентичність електронного та друкованого варіантів випускної кваліфікаційної роботи, і даємо згоду на обробку персональних даних.

Виконавець _____ / Мірошкін А.І. /

Керівник _____ / Джабраілов Д.В./

«09» 06 2023 р.

UNICHECK

Ім'я користувача:
Наталія Вікторівна Копусь

ID перевірки:
1015552095

Дата перевірки:
11.06.2023 23:22:37 EEST

Тип перевірки:
Doc vs Internet + Library

Дата звіту:
11.06.2023 23:29:43 EEST

ID користувача:
100011688

Назва документа: 4РП-06 Мірошкін А.І.

Кількість сторінок: 53 Кількість слів: 7208 Кількість символів: 52042 Розмір файлу: 4.12 MB ID файлу: 1015204354

21.6%
Схожість

Найбільша схожість: 7.24% з Інтернет-джерелом (https://docs.godotengine.org/uk/stable/getting_started/introduction/...

21.6% Джерела з Інтернету | 537 ..... Сторінка 55

Не знайдено джерел з Бібліотеки

0% Цитат

Вилучення цитат вимкнене

Вилучення списку бібліографічних посилань вимкнене

0%
Вилучень

Немає вилучених джерел

Модифікації

Виявлено модифікації тексту. Детальна інформація доступна в онлайн-звіті.

Замінені символи | 31

ВСП «ОДЕСЬКИЙ ТЕХНІЧНИЙ ФАХОВИЙ КОЛЕДЖ ОНТУ»

РЕЦЕНЗІЯ

на дипломний проект (роботу) здобувача (здобувачки) освіти

відділення комп'ютерних систем

Мірошкіна Андрія Ігоровича

(прізвище, ім`я та по батькові)

Спеціальність 121 " Інженерія програмного забезпечення "

Освітня програма «Розробка програмного забезпечення»

Керівник дипломного проекту (роботи) Джабраілов Дмитро Володимирович

(прізвище, ім`я та по батькові)

Тема дипломного проекту (роботи) *Реалізація поведінки штучного інтелекту та алгоритмів взаємодії з ігровим оточенням для персонажів 2D-гри у жанрі top down shooter*

Обсяг розрахунково-пояснювальної записки 71 сторінок

Обсяг графічної (презентаційної) частини 8 аркушів (слайдів)

ХАРАКТЕРИСТИКА ДИПЛОМНОГО ПРОЕКТУ (РОБОТИ)

а) заключення про ступінь відповідності виконаного дипломного проекту (роботи) завданню

Представлений на рецензію дипломний проект повністю відповідає меті проектування та технічному завданню. Тематика дипломного проекту присвячена актуальному питанню реалізації поведінки штучного інтелекту в іграх та поведінки не ігрових персонажів.

б) характеристика виконання кожного розділу дипломного проекту (роботи)

Дипломний проект складається зі вступу, трьох розділів, висновків, переліку використаних джерел. У технологічному розділі розглянуто проблематику штучного інтелекту в іграх, його різновиди та аналоги реалізації. Виконано проектування основних програмних елементів для реалізації штучного інтелекту в 2D-грі. Реалізовано проект із ігровим процесом та працюючим штучним інтелектом ворогів.

в) оцінка якості виконання пояснювальної записки та графічної частини дипломного проекту (роботи) Графічна частина виконана на достатньому рівні у вигляді презентації із використанням офісного пакету Microsoft PowerPoint та Visio. Пояснювальна записка виконана акуратно та у відповідності до норм оформлення документів із використанням офісного пакету Microsoft Word. Загальна якість виконання документації – відмінна, академічного плагіату у роботі не виявлено

г) перелік позитивних якостей дипломного проекту (роботи)

1. Проведено розгляд актуального питання штучного інтелекту в іграх;
2. Виконано макет повноцінної гри;
3. Різноманітні вороги та патерни їх поведінки.

д) основні недоліки дипломного проекту (роботи)

1. Не достатній розділ слайдів презентації.

Оцінка розрахункової частини Відмінно

Оцінка графічної частини Відмінно

Загальна оцінка Відмінно

Прізвище, ім'я, по батькові рецензента Стайкуца Сергій Володимирович

Місце роботи і посада рецензента Державний університет інтелектуальних технологій і зв'язку, к.ф.н., доцент кафедри КБ та ТЗІ, пом.декану факультету інформаційних технологій та кібербезпеки

Підпис:

« 16 » червня 2023 р.

ПІДПИС ПОСВІДЧУЮ
НАЧАЛЬНИК ВІДДІЛУ
КАДРІВ ДУІТЗ

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ * ДЕРЖАВНИЙ УНІВЕРСИТЕТ ІНТЕЛЕКТУАЛЬНИХ ТЕХНОЛОГІЙ І ЗВ'ЯЗКУ * ВІДДІЛ КАДРІВ * 43997335 *