

**МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ВОЛИНСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ
ІМЕНІ ЛЕСІ УКРАЇНКИ
Кафедра комп'ютерних наук та кібербезпеки**

На правах рукопису

**ВОЗНЯК МИКОЛА АНДРІЙОВИЧ
ПРОЄКТУВАННЯ ТА РОЗРОБКА 3D-ГРИ-ГОЛОВОЛОМКИ НА
РУШІЇ GODOT 4.5**

Спеціальність: 122 Комп'ютерні науки
Освітньо-професійна програма: Комп'ютерні науки та інформаційні
технології Робота на здобуття освітнього ступеня «магістр»

Науковий керівник:
ПАСТЕРНАК ЯРОСЛАВ МИХАЙЛОВИЧ,
доктор фізико-математичних наук, професор,
професор кафедри комп'ютерних наук та
кібербезпеки

РЕКОМЕНДОВАНО ДО ЗАХИСТУ

Протокол № _____

засідання кафедри комп'ютерних наук
та кібербезпеки

від _____ 20__ р.

Завідувач кафедри

(_____) _____

(підпис)

ПІБ

ЛУЦЬК – 2025

ЗМІСТ

ВСТУП.....	3
РОЗДІЛ 1 ТЕОРЕТИЧНІ ОСНОВИ СТВОРЕННЯ ІГОР-ГОЛОВОЛОМОК.....	5
1.1 Ігрові головоломки: історія, класифікація та еволюція.....	5
1.1.1 Феномен головоломки як універсального механізму пізнання.....	5
1.1.2 Еволюція головоломок у цифровому середовищі.....	9
1.1.3 Класифікація відеоігор-головоломок.....	16
1.3 Алгоритмічні основи та принципи проектування головоломок.....	19
1.4 Ігровий рушій Godot.....	20
1.4.1 Поняття ігрового рушія та його функції.....	20
1.4.2 Ігровий рушій Godot 4.5 та його основна концепція.....	23
1.4.3 Переваги та недоліки Godot.....	25
1.5 Огляд аналогічних програмних розробок.....	26
РОЗДІЛ 2 ПРОЄКТУВАННЯ ТА РОЗРОБКА ГРИ «RETENTUM».....	29
2.1 Постановка задачі.....	29
2.2 Методологія дослідження	31
2.3 Теоретичні аспекти дослідження	33
2.4 Обґрунтування вибору інструментальних засобів.....	44
2.5 Особливості програмної реалізації.....	46
2.6 Організація тестування та налагодження.....	52
2.7 Аналіз отриманих результатів дослідження, рекомендації щодо використання та впровадження.....	53
ВИСНОВКИ.....	55
СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ.....	56
ДОДАТКИ.....	60

ВСТУП

Сучасна відеоігрова індустрія характеризується не лише стрімким технологічним розвитком, але й активним пошуком нових форм в рамках класичних жанрів. Особливу увагу розробників привертають логічні ігри та головоломки, де ключовим елементом залучення гравця є поєднання глибокої стратегії, зрозумілих механік та унікального ігрового досвіду. Створення інноваційних головоломок, що вимагають від гравця не лише миттєвих рішень, але й довгострокового тактичного планування, залишається актуальним напрямком розробки. Сучасні інструменти, такі як рушій Godot, відкривають широкі можливості для втілення таких концепцій, дозволяючи сконцентруватися на креативному геймдизайні та створенні складних інтерактивних систем.

Метою роботи є проектування та розробка тривимірної гри-головоломки з тактичними елементами на рушії Godot 4.5, основним завданням у якій є блокування руху противника на ігровому полі.

Для досягнення поставленої мети потрібно виконати наступні завдання:

- проаналізувати історичний розвиток головоломок, їх концептуальну природу та еволюцію від механічних форм до цифрових інтерпретацій;
- дослідити алгоритмічні принципи проектування головоломок, зокрема методи забезпечення коректності, розв'язності та структурної цілісності;
- визначити ефективні стратегії балансування складності та побудови прогресії в сучасних ігрових головоломках;
- ознайомитися з ігровим рушієм Godot, його концептуальною моделлю, функціональними можливостями, перевагами та обмеженнями;
- здійснити огляд та аналіз і провести аналіз комп'ютерних ігор, розроблених на базі рушія Godot;
- визначити призначення та сформулювати вимоги до програмного продукту;
- обрати доцільну модель організації процесу розробки;

- підібрати інструментарій, необхідний для створення програмного продукту;
- реалізувати програмний продукт із застосуванням обраних технологій;
- провести тестування та налагодження створеної гри;
- підготувати рекомендації щодо використання та впровадження розробленого продукту.

Об'єкт дослідження – теоретичні та алгоритмічні основи проектування ігор-головоломок.

Предмет дослідження – проектування та розробка тривимірної тактичної гри-головоломки на ігровому рушії Godot.

Результати роботи були представлені на II Міжнародній науково-практичній конференції «Проблеми комп'ютерних наук, програмного моделювання та безпеки цифрових систем» 9-11 червня 2025 року та V Всеукраїнській науково-технічній конференції молодих вчених, аспірантів і студентів «Комп'ютерні ігри і мультимедіа, як інноваційний підхід до комунікації - 2025» 25-26 вересня 2025 року.

РОЗДІЛ 1

ТЕОРЕТИЧНІ ОСНОВИ СТВОРЕННЯ ІГОР-ГОЛОВОЛОМОК

1.1 Ігрові головоломки: історія, класифікація та еволюція

1.1.1 Феномен головоломки як універсального механізму пізнання

Впродовж усієї історії людства головоломки постають як один із найстійкіших механізмів осмислення світу. Вони виникають у різних культурах незалежно одна від одної й слугують способом тренування мислення, моделювання ситуацій та перевірки когнітивних меж людини.

Головоломка – це завдання, для розв’язання якого, як правило, потрібна кмітливість, а не спеціальні знання високого рівня [1]. На відміну від випробувань, що вимагають фізичної сили чи везіння, головоломка створює структурований пізнавальний виклик, подолання якого потребує від людини інтелектуального зусилля в межах визначених правил. Це не просто форма розваги, а фундаментальний інструмент пізнання, який має глибокі психологічні корені.

Щоб краще зрозуміти, чим є головоломка, корисно розкласти її на складові. Зазвичай виокремлюють три ключові елементи: по-перше, чітко визначену мету, тобто конкретний кінцевий стан, до якого ведуть дії; по-друге, набір правил чи обмежень, що визначають допустимі операції та формують простір дії; і по-третє, необхідність інтелектуальної діяльності – застосування логіки, евристик або творчого пошуку, оскільки розв’язання не зводиться лише до простих фізичних маніпуляцій [2]. Ці компоненти роблять головоломку більш ніж просто задачею – вона є вхідною точкою до критичного мислення, винахідливості й проблемно-орієнтованого підходу.

Головоломки є яскравим прикладом моделі пізнавального процесу. У процесі її розв’язання людина взаємодіє з неповністю відомою ситуацією, формує гіпотези щодо можливих стратегій, перевіряє їх методом проб і помилок і коригує свій підхід відповідно до отриманого зворотного зв’язку. Цей когнітивний цикл структурно подібний до наукового методу, де послідовність

«спостереження – гіпотеза – експеримент – висновок» формує основу раціонального дослідження.

У праці *The Puzzle Instinct* [3] Марсель Данезі пропонує концепцію «інстинкту головоломки» – вродженої схильності людини шукати структуру, ставити запитання і прагнути до пояснень навіть у неоднозначних або хаотичних ситуаціях. На його думку, цей інстинкт пов'язаний із фундаментальним пошуком сенсу: головоломки надають компактні, дрібномасштабні моделі великих екзистенційних питань, дозволяючи пережити мініатюрну форму відкриття, що психологічно має ту саму природу, що й опанування складніших життєвих проблем.

Розв'язування головоломок виходить за рамки гри чи тренування інтелекту: це глибинний пізнавальний процес, у якому людина не просто знаходить рішення, а й відтворює базову структуру людського мислення.

Універсальна когнітивна природа головоломок визначила їхнє незалежне виникнення в багатьох культурних традиціях. Найраніші виявлені форми пов'язані з матеріальною культурою ранніх цивілізацій. Археологічні артефакти Месопотамії та долини Інду містять просторові задачі та прототипи механічних конструкцій, що використовувалися як у ритуальному контексті, так і як засоби навчання. Це були предмети, які вимагали маніпуляції формою та об'ємом, закладаючи фундамент для розвитку інженерного мислення.

У античній Греції формуються перші абстрактні головоломки, тісно пов'язані з математичною традицією. Задачі Евкліда, парадокси Зенона, проблеми побудови геометричних фігур та знаменитий «Гордіїв вузол» демонструють перехід від практичних задач до концептуальних пізнавальних моделей. У римській культурі головоломки поширюються у формі логічних задач та стратегічних настільних ігор, як-от латрункулі (рис 1.1), що виконували не тільки розважальну, а й освітню функції [4].



Рисунок 1.1 - Латрункулі, одна з перших покрокових головоломок з механікою блокування ходів противника

Східноазійська традиція репрезентує альтернативний вектор розвитку – інженерно-конструктивний. Тут головоломки еволюціонували як засіб розвитку просторової уяви та комбінаторної логіки. Домінуючими формами стали задачі на геометричну декомпозицію та створення складних фігур та форм. Ці практики вимагали не лінійного алгоритмічного розв’язку, а цілісного бачення об’єкта, базуючись на філософському принципі гармонійної єдності частин і цілого. Такий підхід перетворив головоломку з простої розваги на об’єкт естетичної та інтелектуальної медитації, що тренує терпіння й точність. Яскравим прикладом може виступити китайський танграм – модульна геометрична головоломка з фіксованим набором полігональних елементів (рис 1.2) [5].

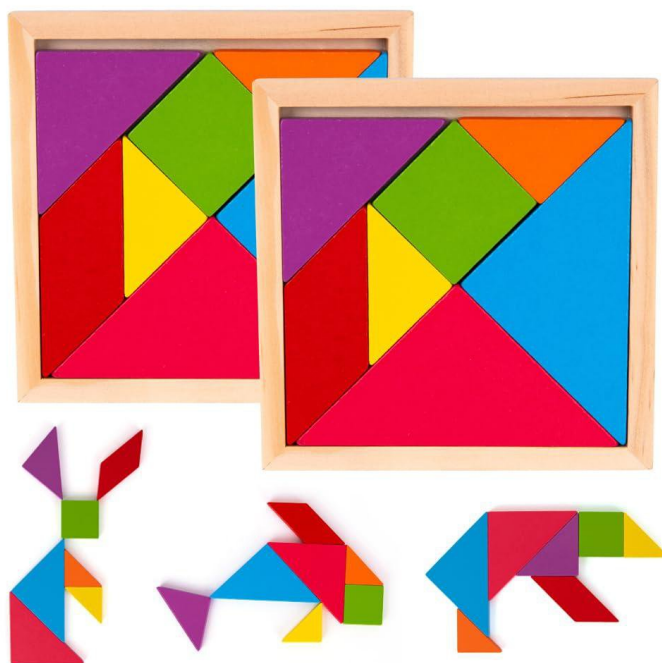


Рисунок 1.2 - Головоломка танграм та приклади зібраних фігур

У Середньовіччі роль головоломок часто зміщувалася від інженерних конструкцій до риторичних, логічних та математичних задач, поширюваних через рукописи. Особливе місце займали задачі, пов'язані з календарем, розподілом ресурсів і алгебраїчними ребусами. Одним із найяскравіших прикладів є збірка задач «*Propositiones ad Acuendos Juvenes*» («Задачі для загострення юнацького розуму») Алкуїна, що містила такі відомі задачі як задача про переправу через річку вовка, кози та капусти та інші логічні головоломки. Ці задачі не лише розважали, а й слугували засобом навчання логіки та арифметики у монастирських та придворних школах [6].

У період Нового часу (XVIII–XIX століття) відбувається інституціоналізація головоломки як культурного продукту. З розвитком друкарства, промислового виробництва та шкільної освіти з'являються перші масові колекції головоломок, логічні вправи й стратегічні настільні ігри. В цей період головоломка перетворюється з інтелектуальної розваги еліт на елемент масової культури.

XX століття радикально розширює типологію головоломок. У цей період з'являються механічні системи, як-от кубик Рубіка та його похідні, логічні задачі на графах, математичні парадокси, кросворди, криптографічні проблеми воєнного періоду та перші комп'ютерні головоломки. Кожен напрям формує власні когнітивні вимоги: просторове мислення, алгоритмічне мислення, дедуктивний аналіз, символічну комбінацію. Така диференціація визначила сучасне розуміння головоломок як міждисциплінарного інтелектуального інструменту.

1.1.2 Еволюція головоломок у цифровому середовищі

З появою комп'ютерів головоломки переходять у цифровий простір і формують основу для сучасних puzzle-ігрових жанрів. Ця цифрова трансформація другої половини XX століття докорінно змінила онтологію головоломок. Комп'ютерна гра, що визначається як інтерактивна програма, керована програмним забезпеченням з метою розваги та виклику [7], суттєво відрізняється від традиційних форм. Якщо механічні та друковані головоломки пропонували статичну задачу з фіксованим набором правил, то цифрове середовище запровадило динамічну систему, що реагує на дії користувача в реальному часі.

На початковому етапі розвитку цифрової індустрії головоломки розвивалися шляхом прямої цифровізації класичних логічних ігор та фундаментальних просторових задач. Обмежені обчислювальні можливості ранніх мікропроцесорів та відсутність розвиненої графіки змушували розробників зосереджуватись виключно на логічних викликах, часто використовуючи головоломки для тестування базових алгоритмів штучного інтелекту.

У контексті аркадних автоматів, що вимагали негайного залучення гравця, з'явилися перші ігри, орієнтовані на фундаментальні просторові задачі. Однією

з перших відеоігор-головоломок, розробленою для аркадних автоматів, стала Amazing Maze, випущена у 1976 році (рис 1.3). Її єдиним завданням було вийти з лабіринту до того, як це зробить керований комп'ютером противник. Цей приклад заклав основу для майбутніх «пошукових» та «навігаційних» піджанрів, де просторове орієнтування поєднувалося з критичним фактором часу або протистояння, трансформуючи статичний лабіринт на динамічну задачу переслідування [8].



Рисунок 1.3 - Amazing Maze в процесі гри

У цей час головоломки часто розроблялися як навчальні ігри, так звані навчально-розважальні програми, використовуючи цифрові платформи для тренування пам'яті та логіки, що відображало освітню функцію головоломок, успадковану від античних традицій.

Технологічний прогрес, зокрема підвищення обчислювальної потужності мікропроцесорів та графічних співпроцесорів 8-бітної ери, призвів до появи складніших і динамічніших ігрових механік. Це дозволило головоломкам вийти за межі простих лабіринтів і почати активну експлуатацію просторового моделювання та складних правил взаємодії.



Рисунок 1.5 - стрибає по діагоналі вниз по піраміді, щоб уникнути ворога

Ця епоха також відзначилася появою комбінованих головоломок, що вимагали поєднання різних навичок. Класичний приклад – Boulder Dash, випущена в 1984 році, яка об'єднала кілька завдань: прокладання шляху, збирання предметів та уникнення пасток чи динамічних загроз [11]. Успіх тут вимагав одночасного застосування швидкої реакції та стратегічного планування ресурсів простору та часу. Для контрасту, у цей же час розвивалися і чисті логічні системи, зокрема Sokoban 1982 року, що використовували простий рушій для створення надзвичайно складних задач із просторового переміщення об'єктів, ставши класикою жанру, орієнтованою на глибокий, не поспішний аналіз [12].

Кульмінацією та яскравим прикладом жанру головоломки стала гра Тетріс, що поєднала простоту правил із часовим тиском, який зростав в кожному новому рівні. Тетріс продемонстрував, як процедурна генерація випадкових фігур може створити нескінченний когнітивний виклик, перетворивши логічну задачу на рефлекторно-координаційний процес [13].

На початку 1990-х років еволюція жанру зазнала чергового значного прогресу завдяки впровадженню носіїв CD-ROM. Збільшений обсяг пам'яті

дозволив інтегрувати у головоломки деталізовану графіку, відеовставки та високоякісний звук, що змінило фокус з абстрактної логіки на імерсивне дослідження середовища.

Знаковою подією цього періоду став вихід гри Myst у 1993 році. Вона відійшла від аркадної динаміки та часових обмежень, властивих Tetris чи Pac-Man, на користь медитативного споглядання та контекстуального аналізу. Головоломки у Myst були інтегровані в наратив та оточення: гравцеві доводилося вивчати механізми, розшифровувати символи та маніпулювати важелями, які існували за внутрішньою логікою вигаданого світу (рис. 1.6). Успіх Myst легітимізував жанр пригодницької головоломки, де рішення задачі стає ключем до розвитку сюжету. Це продемонструвало, що цифрова головоломка може бути не лише математичною задачею, а й засобом розповіді історії [14].



Рисунок 1.6 - Взаємодія з механізмом Puzzle board у світі Myst

Наступний революційний етап настав у другій половині 2000-х років із розвитком фізичних рушіїв таких як Havok та Source. Це дозволило створити піджанр фізичні головоломки, де гравець взаємодіє з об'єктами, що мають масу, інерцію та підкорюються законам ньютонівської механіки.

Апофеозом цього напрямку стала гра Portal, розроблена Valve в 2007 році. Вона об'єднала механіку шутера від першої особи з просторовою головоломкою. Унікальність Portal полягала у використанні порталного пристрою, який дозволяв миттєві переміщення між двома точками простору, зберігаючи імпульс руху (рис. 1.7). Це вимагало від гравця нелінійного мислення та розуміння топології простору, що порушує звичну евклідову геометрію. Гра довела, що головоломка може бути динамічною та видовищною, не втрачаючи інтелектуальної глибини [15].



Рисунок 1.7 - Приклад можливостей порталів у грі Portal

Новий етап розвитку головоломок нерозривно пов'язаний із поширенням сенсорних інтерфейсів мобільних пристроїв. Відсутність фізичних кнопок та можливість прямої тактильної взаємодії з сенсорним екраном породили унікальні типи механік.

Знаковим прикладом адаптації жанру під специфіку сенсорного керування стала гра Cut the Rope, випущена студією ZeptoLab у 2010 році (рис. 1.8). Проєкт відмовився від емуляції віртуальних кнопок на користь прямої тактильної взаємодії: основною механікою став жест свайпу (проведення пальцем по екрану), що імітував фізичну дію розрізання мотузки.

Геймплей базувався на симуляції фізичних законів – гравітації, інерції маятника та натягу нитки, вимагаючи від гравця не лише логічного розрахунку траєкторії, а й точного таймінгу. Cut the Rope продемонструвала, як мобільні платформи змогли трансформувати складні фізичні задачі у доступний казуальний формат, де бар'єр між мисленнєвим процесом гравця та ігровою дією було мінімізовано завдяки інтуїтивному інтерфейсу [16].



Рисунок 1.8 - Навчальні рівні гри Cut the Rope

Сьогодні жанр головоломок характеризується гібридизацією та глибокою інтеграцією в інші жанри. Елементи пазлів стають невід'ємною складовою ігор жанру Action-RPG, наприклад, святилища у The Legend of Zelda: Breath of the Wild та кооперативних ігор, наприклад, It Takes Two, де рішення задачі вимагає скоординованих дій двох гравців. Окрім того, розвиток технологій віртуальної реальності (VR) відкриває нову сторінку в історії жанру, дозволяючи гравцеві фізично взаємодіяти з тривимірними об'єктами у віртуальному просторі, як у іграх Superhot або The Room VR. Це повертає цифрову головоломку до її тактильних, механічних витоків, але на якісно новому технологічному рівні.

Таким чином, технологічний розвиток поступово змінив сам характер цифрових головоломок. Вони перестали бути статичними задачами з

фіксованими умовами та перетворилися на інтерактивні системи, де важливими є не лише логічні побудови, а й просторове мислення, робота з фізичними моделями, реагування на змінні стани середовища та іноді й контекст сюжету. Розвившись, жанр став набагато різноманітнішим та вийшов далеко за межі традиційних форм, поєднуючи в собі елементи різних типів геймплею.

1.1.3 Класифікація відеоігор-головоломок

Сучасний спектр ігор-головоломок є надзвичайно широким і характеризується високим рівнем гібридизації. Проте, для аналізу доцільно класифікувати їх не за візуальним стилем, а за домінуючою ігровою механікою (core mechanic) та типом когнітивного навантаження, яке покладається на гравця. На основі аналізу ігрового процесу та взаємодії «гравець-система», можна виділити наступні ключові категорії відеоігор-головоломок.

Однією з найпоширеніших категорій є екшн-головоломки (Action Puzzles), які поєднують логічне мислення з необхідністю швидкої реакції. У таких іграх, класичними представниками яких є Tetris або Lumines, ключовим фактором виступає часовий тиск, що змушує гравця приймати рішення в реальному часі. Успіх тут залежить не лише від стратегічного планування, а й від сенсомоторної координації та здатності гравця увійти в стан потоку, оперуючи динамічними об'єктами.

На противагу динамічним системам, великий сегмент ринку займають головоломки типу «три в ряд» та злиття (Tile-Matching & Merge). Цей піджанр, представлений іграми на кшталт Candy Crush Saga чи 2048, базується на маніпуляції елементами на статичному або напівстатичному полі з метою створення певних комбінацій. Когнітивне навантаження тут зосереджене на локальному плануванні та пошуку патернів, хоча науковці зазначають, що високий рівень випадкових значень у таких іграх часто знижує роль чистої логіки на користь азартної складової.

З розвитком технологій симуляції окремим кластером виділилися фізично-орієнтовані головоломки (Physics-Based Puzzles). У таких проектах, як World of Goo або Angry Birds, вирішення задачі базується на інтуїтивному розумінні та використанні законів ньютонівської механіки: гравітації, інерції, тертя. Особливістю цього типу є емерджентність геймплею, що дозволяє гравцеві вирішувати одну й ту саму задачу різними способами, взаємодіючи з інтерактивною моделлю фізичного світу, а не з жорстко заданим алгоритмом.

Класичні традиції механічних ігор продовжують головоломки переміщення та навігації, де завданням є переміщення об'єкта через систему перешкод як-от в Sokoban, яке стало фактичною назвою піджанру, чи Rush Hour. На відміну від фізичних головоломок, тут панує дискретність та повна детермінованість: успіх вимагає глибокого, покрокового прорахунку наслідків. Еволюція цього напрямку призвела до появи мета-головоломок, таких як Baba Is You, де об'єктом маніпуляції стають не лише предмети, а й самі правила гри, що вимагає від гравця гнучкого системного мислення.

Окрему нішу займають головоломки дослідження (Exploration & Adventure Puzzles), де завдання органічно інтегровані в навколишнє середовище та наратив. У проектах на кшталт The Witness, Myst або The Talos Principle рішення вимагає не стільки швидкості реакції, скільки спостережливості, збору розрізної інформації та цілісного розуміння логіки ігрового світу. Відсутність таймерів та акцент на атмосфері створюють простір для індуктивного навчання, де гравець самостійно відкриває правила через спостереження та експериментування, без прямих інструкцій від системи.

Специфічним нішевим сегментом є алгоритмічні та програмістські головоломки (Coding & Zach-likes), названі на честь розробника Зака Барта. Ігри цього типу, такі як SpaceChem або Human Resource Machine, вимагають від гравця конструювання складних систем або написання псевдокоду для автоматизації процесів. На відміну від класичних головоломок з одним правильним рішенням, тут домінують відкриті задачі з можливістю оптимізації за різними параметрами: швидкістю виконання, ефективністю використання

ресурсів або компактністю алгоритму. Когнітивне навантаження зосереджене на системному мисленні, декомпозиції складних проблем на елементарні операції та налагодженні процесів.

Варто згадати про жанр пошуку предметів (Hidden Object Puzzle Adventure), який фокусується на візуальному сприйнятті та концентрації уваги. Завдання більшості головоломок такого типу полягає у знаходженні певного визначеного ряду об'єктів, замаскованих у складних, детально опрацьованих статичних сценах. Хоча цей піджанр характеризується мінімальною інтерактивністю порівняно з іншими типами головоломок, він ефективно тренує увагу, концентрацію та здатність до фігуро-фонового розрізнення, що робить його популярним серед певної аудиторії гравців.

Особливе місце в екосистемі головоломок посідають логічні головоломки (Logic Puzzles), що представляють собою чисту форму інтелектуального виклику без залежності від рефлексів чи випадковості. До цієї категорії належать як цифрові адаптації традиційних паперових головоломок – Sudoku, Nonogram (Picross), Kakuro, – так і оригінальні відеоігрові концепції на кшталт Minesweeper чи The Witness. Когнітивна специфіка цього піджанру полягає в послідовному виключенні неможливих варіантів і побудові ланцюгів логічних висновків, що робить такі ігри популярним інструментом у дослідженнях проблемно-орієнтованого навчання. Логічні головоломки активізують декларативні когнітивні процеси, а відсутність часового тиску перетворює процес гри на медитативну практику алгоритмічного мислення.

Попри певну частку умовності, запропонована типологія дозволяє систематизувати абсолютну більшість існуючих на ринку відеоігор-головоломок. Враховуючи тенденцію до еклектики в сучасному геймдизайні, окремі проекти можуть не вписуватися в рамки однієї категорії, балансуючи на межі кількох піджанрів. Проте, фокусування саме на домінуючій механіці та типі когнітивної діяльності дає змогу ефективно класифікувати навіть найбільш нестандартні ігрові системи, розглядаючи їх або як варіації вищеназваних типів, або як їх гібридні форми [17].

1.3 Алгоритмічні основи та принципи проектування головоломок

Проектування головоломок у відеоіграх є подвійним завданням: з одного боку, це створення цікавого досвіду для гравця, а з іншого — побудова надійної логічної системи. В цьому контексті алгоритмічна база виступає не як самоціль, а як інструмент забезпечення якості: будь-яка головоломка розглядається як простір станів, де завдання гравця полягає у пошуку маршруту від початкової конфігурації до цільової через послідовність дозволених операцій. Фундаментальною вимогою до такої системи є її розв'язність, що тісно пов'язує геймдизайн із теорією алгоритмів та поняттям обчислюваних функцій. Головоломка вважається коректно поставленою задачею лише тоді, коли існує алгоритм, здатний перевести систему з початкового стану в фінальний за скінченну кількість кроків [18]. Якщо такого алгоритму не існує (шлях у графі станів розірваний), задача переходить у розряд некоректних або необчислюваних. При цьому розв'язок не обов'язково має бути єдиним. Наявність множини розв'язків наближає головоломку до більш творчих задач, де алгоритм перевірки оцінює не точний збіг із єдиним еталоном, а відповідність результату заданим умовам.

Ключовим фактором, що визначає якість та цікавість головоломки, є її обчислювальна складність, яка поділяє всі ігрові задачі на дві основні категорії. Першу категорію складають задачі класу P, які розв'язуються за поліноміальний час; у контексті гри вони часто сприймаються як тривіальні, оскільки гравець здатен швидко прорахувати лінійний шлях до мети без глибокого аналізу, що суттєво знижує інтелектуальний виклик [19]. Другу категорію формують задачі класу NP, зокрема NP-повні задачі (як-от Судоку чи Сапер), які характеризуються високою складністю пошуку рішення; вони забезпечують гравцеві необхідний рівень напруги, адже вимагають застосування евристик та інтуїції замість простого перебору, проте дозволяють миттєво переконатися у

правильності знайденого результату [20]. Саме ця фундаментальна асиметрія — складність знаходження розв’язку при легкості його перевірки — є основою захопливого ігрового процесу.

Для практичного забезпечення розв’язності, особливо в контексті процедурної генерації рівнів, розробники застосовують комплексні алгоритмічні стратегії, що базуються на конструктивних та перевірчих методах. Одним із найнадійніших підходів є метод зворотного проєктування (Reverse Design), при якому побудова рівня розпочинається з фінального, вже вирішеного стану, до якого послідовно застосовуються зворотні ігрові операції; такий алгоритм гарантує математичну наявність шляху до перемоги ще на етапі створення структури. Альтернативою виступає стохастичний метод "генерації та перевірки", де система спочатку створює випадкову конфігурацію ігрового поля, а потім запускає віртуального агента для пошуку шляху, відбраковуючи ті варіанти, які виявляються некоректними або непрохідними [21].

Окрім базової генерації, критично важливим аспектом є застосування теорії графів для контролю топології ігрового простору та калібрування його складності. Використання алгоритмів побудови мінімального кістякового дерева (наприклад, алгоритмів Пріма або Крускала) дозволяє створювати ідеально зв’язні лабіринти без ізольованих зон, гарантуючи доступність кожної ключової точки на карті. Паралельно з цим, алгоритми аналізу простору станів оцінюють метрики складності, такі як коефіцієнт розгалуження та мінімальна кількість ходів до цілі, що дозволяє динамічно адаптувати згенерований рівень під навички гравця, уникаючи як тривіальних рішень, так і надто складних конфігурацій.

1.4 Ігровий рушій Godot

1.4.1 Поняття ігрового рушія та його функції

Ігровий рушій (англ. Game engine) – це ключовий програмний компонент будь-якої комп’ютерної гри, що забезпечує реалізацію всіх її технічних аспектів.

Він полегшує процес створення гри через стандартизацію та впорядкування її внутрішньої архітектури. Суттєвою перевагою використання рушіїв є здатність розробляти кросплатформові проєкти, які можуть функціонувати на різноманітних пристроях одночасно, зокрема на комп'ютерах, ігрових приставках та смартфонах [22].

Сучасні ігрові рушії представляють собою комплексні програмні платформи, які об'єднують інструменти розробки та виконавче середовище. Вони забезпечують не лише технічну основу для гри, але й надають зручні інтерфейси для дизайнерів, художників та програмістів, дозволяючи команді працювати паралельно над різними аспектами проєкту. Модульна архітектура рушіїв дозволяє використовувати лише необхідні компоненти, оптимізуючи продуктивність кінцевого продукту.

Базову функціональність гри забезпечує саме її рушій, який складається з численних підсистем та модулів [23]:

- рушій візуалізації, або рендерер, забезпечує відтворення графічних елементів та візуальних ефектів;
- фізичний модуль відповідає за симуляцію фізичних процесів у віртуальному просторі, включаючи гравітацію, колізії та взаємодію об'єктів;
- аудіосистема обробляє звукові ефекти та музичне оформлення;
- система скриптування надає можливість розробникам кодувати поведінку ігрових елементів і механіки геймплею;
- система анімації забезпечує плавність руху та трансформації ігрових персонажів і об'єктів;
- модуль ігрового ШІ контролює поведінку неігрових персонажів та інших компонентів гри, які потребують логіки прийняття рішень;
- мережевий модуль забезпечує функціонал мультиплеєра та синхронізацію інформації між клієнтськими частинами;
- менеджер пам'яті ефективно розподіляє та вивільняє ресурси оперативної пам'яті протягом роботи програми;

- багатопоточність забезпечує паралелізацію обчислювальних процесів для збільшення продуктивності;
- граф сцени є ієрархічним представленням ігрового світу, що спрощує процес проектування рівнів і візуалізацію складних віртуальних локацій.

Застосування ігрового рушія дозволяє економити час та ресурси в процесі розробки, адже базові технічні модулі вже створені та оптимізовані. Це надає можливість розробникам концентруватися на формуванні ігрового контенту, такого як візуальне оформлення, анімаційні послідовності, аудіосупровід та ігрову логіку. До того ж, ігрові рушії зазвичай є повторно використовуваними, що дозволяє розробляти різноманітні ігрові проекти на базі одного рушія, модифікуючи виключно ігровий контент.

Сучасний ринок містить широкий спектр ігрових рушіїв, але на даний момент високий попит зосереджений навколо трьох платформ – Unity, Unreal Engine та Godot. Вони формують основне технологічне ядро індустрії завдяки своїй екосистемі, функціональності та активним спільнотам розробників.

Unity вважається одним із найбільш затребуваних і функціональних рушіїв, що забезпечує створення ігор для широкого спектру платформ, таких як персональні комп'ютери, ігрові консолі, смартфони та веб-платформи. Рушій характеризується інтуїтивним інтегрованим середовищем розробки, ефективними засобами для генерації ігрового контенту та можливістю кросплатформної розробки з використанням мови програмування C# [24].

Unreal Engine, розроблений компанією Epic Games, представляє собою провідне рішення для створення високоякісних тривимірних ігор та інтерактивних додатків. Рушій відзначається передовими технологіями візуалізації, досконалим фізичним модулем і здатністю відтворювати фотореалістичну графіку. Unreal Engine також пропонує комплексний редактор локацій та розширений набір інструментів для формування ігрового наповнення [25].

Godot – це безкоштовний ігровий рушій з відкритим вихідним кодом, який активно завойовує популярність завдяки своїй доступності та універсальності. Він дозволяє розробляти двовимірні та тривимірні ігри для різноманітних платформ, зокрема персональних комп'ютерів, мобільних гаджетів та веб-браузерів. Godot відрізняється зрозумілим інтерфейсом, практичними засобами для створення ігрового контенту та підтримкою декількох мов програмування, включаючи GDScript, C# та C++ [26].

Слід відзначити, що окрім перелічених рушіїв, на ринку представлена велика кількість альтернативних рішень, як платних, так і з відкритим кодом, кожне з яких характеризується унікальними особливостями та перевагами залежно від специфіки проєкту та преференцій команди розробників. Вибір конкретного рушія зазвичай визначається обсягом проєкту, наявними ресурсами, необхідним функціоналом та цільовими платформами для реалізації. Незалежно від обраного інструменту, ігровий рушій виконує критично важливу функцію в процесі створення відеоігор, надаючи технологічний фундамент та інструментарій для побудови захопливих віртуальних світів і унікального ігрового досвіду.

1.4.2 Ігровий рушій Godot 4.5 та його основна концепція

Godot 4.5 – це передовий рушій із відкритим програмним кодом, призначений для розробки ігор та інтерактивних застосунків. Його ключовою ідеєю є створення комфортного та адаптивного інструментарію для розробників, який дозволяє продуктивно втілювати задуми, незалежно від розміру проєкту чи рівня кваліфікації команди.

Базовою філософією Godot є надання творцям максимальних можливостей та автономії у реалізації їхніх задумів. На противагу окремим платним рушіям, Godot не встановлює жорстких методологій чи лімітів, а навпаки, забезпечує повну творчу самостійність розробнику.

Ключовим компонентом Godot виступає система вузлів (nodes), що представляють собою ієрархічні одиниці, з яких конструюється архітектура гри чи програми. Кожен вузол здатен включати різноманітні елементи, зокрема спрайти, тривимірні моделі, фізичні об'єкти, скриптові компоненти, анімаційні послідовності та інше. Подібна структура забезпечує зручне компонування складних сцен із можливістю багаторазового застосування окремих компонентів, підтримуючи модульність та адаптивність у процесі створення.

Godot прагне гарантувати доступний та зрозумілий підхід до створення ігрових проєктів, не втрачаючи при цьому функціональних можливостей та ефективності. Рушій пропонує власну мову програмування GDScript, яка характеризується простотою освоєння, особливо для початківців, проте одночасно залишається достатньо гнучкою та функціональною для масштабних проєктів. Водночас Godot забезпечує сумісність із C# та іншими мовами програмування завдяки відкритій та експансивній архітектурі, що дозволяє інтегрувати зовнішні бібліотеки та засоби розробки.

У Godot 4.5 реалізовано розширений інструментарій для створення як двовимірних, так і тривимірних проєктів в одному інтегрованому середовищі. Для 2D розробки рушій надає практичну систему тайлових карт та графічних спрайтів, тоді як для 3D доступний повноцінний арсенал можливостей, що охоплює імпорт тривимірних моделей, анімаційних послідовностей, систем освітлення та фізичних симуляцій. Така універсальність дає змогу розробникам без труднощів переключатися між двовимірними та тривимірними проєктами, зберігаючи єдине робоче середовище [27].

Загалом Godot 4.5 втілює принципи відкритості, адаптивності та зручності використання, що перетворює його на привабливе рішення для широкого спектру творців, від невеликих незалежних команд до великих студій розробки, а також освітніх ініціатив. Орієнтація на свободу та повний контроль над процесом створення, об'єднана з потужним функціоналом та інструментами, робить Godot унікальним феноменом серед ігрових рушіїв.

1.4.3 Переваги та недоліки Godot

Рушій Godot характеризується низкою сильних сторін, зокрема:

- Godot має повністю відкритий вихідний код (open source), що надає можливість кожному сприяти розробникам у модифікації та розширенні функціоналу рушія, враховуючи запити спільноти;
- проєкти, реалізовані в Godot, можуть бути без проблем портовані на різноманітні платформи, такі як Windows, macOS, Linux, Android, iOS, веб-браузери та ігрові консолі, без істотних модифікацій кодової бази. Це уможливорює охоплення широкої аудиторії за допомогою єдиного проєкту;
- Godot створювався з пріоритетом на продуктивність та оптимізацію, що гарантує ефективне використання системних ресурсів і високу швидкодію навіть на мобільних пристроях;
- Godot є абсолютно безоплатним і не передбачає виплату роялті чи ліцензійних платежів, що перетворює його на привабливе рішення для індивідуальних розробників та невеликих студій;
- Godot забезпечує мультимовну підтримку. Окрім вбудованої скриптової мови GDScript, Godot підтримує C#, а також дозволяє інтегрувати інші мови програмування завдяки відкритій архітектурі;
- Godot пропонує всеосяжний набір інструментів для розробки як 2D, так і 3D ігор та застосунків у єдиному інтегрованому середовищі, включаючи систему тайлмепів, імпорт 3D моделей, анімацій, систем освітлення та фізичних симуляцій.

Водночас Godot має певні слабкі сторони:

- порівняно з комерційними рушіями, зокрема Unity чи Unreal Engine, Godot характеризується меншою користувацькою базою та меншою кількістю доступних ресурсів, що може ускладнити пошук технічної підтримки чи навчальних матеріалів;

- оскільки Godot є open-source проєктом, окремі специфічні функції можуть бути не повністю імплементовані або потребувати додаткових зусиль для інтеграції;
- як open-source рушій, Godot може стикатися з проблемами сумісності з певними сторонніми бібліотеками чи інструментами, якщо вони не були спеціально розроблені для Godot;
- попри потужну систему нодів та сцен, Godot не забезпечує такого ж рівня гнучкості у візуальному скриптуванні, як окремі комерційні рушії;
- на відміну від більш популярних рушіїв, Godot має обмеженіший вибір готових асетів, плагінів та екстеншенів, доступних для придбання або безкоштовного використання.

Незважаючи на зазначені обмеження, Godot є потужним та гнучким інструментом, особливо для незалежних розробників, малих студій та освітніх проєктів. Його відкритість, безкоштовність, кросплатформеність та активна спільнота розробників перетворюють його на привабливий варіант для тих, хто цінує свободу та контроль над власними проєктами.

1.5 Огляд аналогічних програмних розробок

У рамках дослідження було проаналізовано декілька ігор логічного піджанру головоломок, зокрема «Bloxorz» та «Chat Noir», які репрезентують різні підходи до реалізації логічних механік.

«Bloxorz» (<https://www.coolmathgames.com/0-bloxorz>) – просторова головоломка від Damien Clarke, що є еталонним представником піджанру ізометричних головоломок з фізичними механіками. Гра створена на Flash-платформі у 2007 році та демонструє досконале поєднання тривимірної візуалізації з логічними завданнями.

Основна мета полягає у переміщенні прямокутного блока розміром $1 \times 1 \times 2$ клітини у цільову лунку, де блок має встати вертикально. Механіка гри

ґрунтується на керуванні орієнтацією блока за допомогою стрілок клавіатури: кожен рух перекочує блок, змінюючи його положення з горизонтального на вертикальне і навпаки. Гра містить 33 рівні зростаючої складності з інтеграцією додаткових механік: м'яких перемикачів (активуються будь-якою частиною блока), жорстких перемикачів (активуються лише вертикально розташованим блоком), телепортів (розділяють блок на два незалежні кубики), крихких платформ та рухомих мостів.

Bloxorz вимагає алгоритмічного мислення, просторової уяви та планування послідовності кроків, що робить гру популярним інструментом у освітніх програмах для розвитку обчислювального мислення. З точки зору теорії складності обчислень, головоломки типу Bloxorz класифіковані як PSPACE-повні задачі (рис. 1.9) [28].



Рисунок 1.9 - Знімок екрану гри «Bloxorz»

«Chat Noir» (<https://www.gamedesign.jp/sp/cat/>) – класична логічна головоломка від Taro Ito, що є яскравим прикладом піджанру стратегічних головоломок з елементами блокування простору. Гра реалізована на основі Flash-технології (пізніше портована на JavaScript) та представляє собою покрокову головоломку на гексагональній сітці 9×9 .

Мета гри полягає у блокуванні шляхів чорного кота, щоб запобігти його втечі за межі ігрового поля. Механіка гри базується на по черговості ходів: гравець блокує одну клітину, після чого кіт виконує хід у напрямку найкоротшого шляху до краю поля за алгоритмом Дейкстри. Гра вимагає від гравця стратегічного планування, просторового мислення та передбачення алгоритму руху суперника.

Особливістю Chat Noir є мінімалістичний дизайн, відсутність часових обмежень та випадкове розташування початково заблокованих клітин у кожній новій грі, що забезпечує високу реіграбельність. Гра має чіткі паралелі з давньою настільною грою Го, використовуючи схожі принципи контролю території та блокування простору (рис. 1.10) [29].

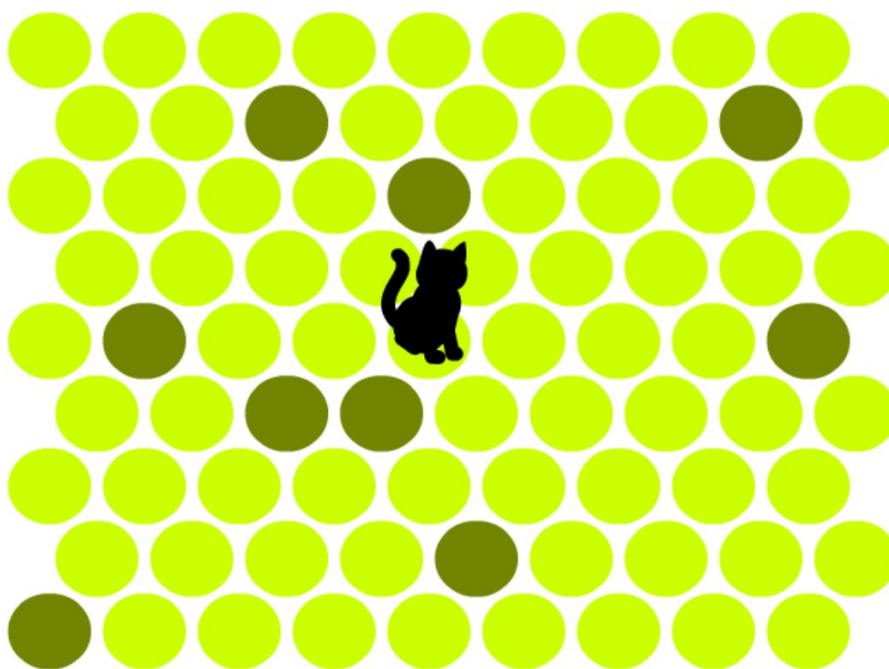


Рисунок 1.10 - Знімок екрану гри «Chat Noir»

РОЗДІЛ 2

ПРОЄКТУВАННЯ ТА РОЗРОБКА ГРИ «RETENTUM»

2.1 Постановка задачі

Програмний засіб «Retentum» є відеогрою жанру логічна головоломка з елементами покрокової стратегії. Основним призначенням гри є надання гравцю інтелектуального виклику через розв'язання тактичних задач на обмеженому ігровому просторі, де необхідно стратегічно планувати послідовність дій для блокування шляху противника. Гра демонструє можливості ігрового рушія Godot 4.5 у створенні 3D-проєктів з фіксованою ортогональною камерою та реалізацією складної покрокової ігрової логіки.

Програмний засіб повинен відповідати наступним функціональним вимогам.

1. Коректна робота тайлів як одиниці ігрового поля, а саме:
 - підтримка типів тайлів з різною ігровою поведінкою;
 - наявність звичайного тайлу та можливість його знищення кліком миші;
 - наявність мертвого тайлу, що не підлягає взаємодії;
 - наявність тайлів з різними модифікаторами та можливість зняття з них модифікаторів кліком миші;
 - наявність кінцевих тайлів, які не можна знищити;
 - різні текстури тайлів в залежності від локації.
2. Генерація та відображення ігрового поля, а саме:
 - наявність різних форм ігрового поля;
 - наявність різних візерунків ігрового поля;
 - генерація поля у відповідності з візерунком, визначеним локацією та формою, вибраною випадковим чином;
 - розміщення кінцевих тайлів по периметру ігрового поля;
 - розміщення звичайних тайлів у внутрішній частині поля;

- розміщення тайлів з модифікаторами;
 - розміщення мертвих тайлів для балансування.
3. Система візуальних локацій з послідовним розблокуванням, що включає мінімум 5 різних локацій з унікальними візуальними стилями, типами тайлів та патернами їх розміщення, з можливістю розширення у майбутньому.
4. Систему крейджаних позначок з мінімум 7 різними типами, кожен з яких має унікальний вплив на поведінку противника, включаючи нейтральні позначки, позначки що змінюють рух противника, телепортують його, або модифікують ігрове поле.
5. Систему штучного інтелекту противника, що включає:
- поява противника на одному з центральних тайлів ігрового поля на початку партії;
 - алгоритм пошуку найкоротшого шляху до найближчого кінцевого тайлу на основі A^* з елементом випадковості для імітації природної непередбачуваності;
 - можливість руху противника у восьми напрямках;
 - режим випадкового блукання при неможливості побудувати шлях до цілі;
 - система емоційних станів з візуальною індикацією;
 - унікальна реакція на кожен тип крейджаної позначки.
6. Коректний перебіг гри:
- покрокова система: одна дія гравця чергується з одним ходом противника;
 - перевірка після кожного ходу противника досягнення кінцевого тайлу для визначення програшу;
 - перевірка відсутності доступних сусідніх тайлів для подальшого руху противника для визначення виграшу;
 - відображення відповідного екрану результату партії.

7. Система прогресії та розблокування контенту:

- відстеження загальної кількості зіграних партій;
- поступове розблокування нових локацій на основі кількості зіграних раундів;
- відображення нової локації після першого переможного раунду з моменту його розблокування;
- розблокування нових типів крейдяних позначок у процесі прогресу;
- збереження прогресу між ігровими сесіями.

8. Камера та візуалізація:

- фіксована ортогональна камера з ізометричним оглядом ігрового поля;
- оптимальне позиціонування камери для забезпечення повного огляду поля будь-якої форми;
- відображення всіх елементів гри за допомогою 3D-моделей.

2.2 Методологія дослідження

Для реалізації відеогри було обрано ітераційний підхід до розробки, що є однією з варіацій гнучких методологій Agile і демонструє високу ефективність у проєктах зі змінюваними специфікаціями та значним рівнем варіативності. Ітераційний підхід ґрунтується на циклічному принципі створення програмного продукту, де весь процес поділяється на послідовні ітерації, також відомі як спринти.

Кожен спринт є завершеним циклом, що включає всі етапи створення програмного забезпечення: збір і аналіз вимог, дизайн архітектури, проєктування структури, створення візуальної складової, роботу з ігровим рушієм, написання програмного коду та завершальне тестування (рис. 2.1). На початковому спринті реалізується базова функціональність, а в наступних циклах специфікації

переглядаються, наявні можливості оптимізуються, виправляються дефекти та інтегруються нові функції.



Рисунок 2.1 - Цикл ітераційної моделі розробки

Ітераційний підхід забезпечує високу адаптивність у процесі розробки, дозволяючи оперативно реагувати на зміни специфікацій на будь-якому етапі та модифікувати функціональність системи відповідно до нових вимог. Цей метод надає можливість швидко отримати базову працездатну версію програми, яку можна тестувати з сторонніми користувачами для збору зворотного зв'язку ще на ранніх стадіях. Завдяки систематичному аналізу проміжних результатів забезпечується постійний контроль якості та ефективна пріоритизація завдань на кожному спринті. Розбиття процесу на короткі цикли значно знижує ризики невизначеності та надає гнучкість у коригуванні вектору розробки залежно від отриманих результатів і змін у вимогах. Такий підхід також сприяє

оптимальному розподілу ресурсів, дозволяючи команді концентрувати зусилля на найбільш критичних компонентах у межах поточного спринту [30].

Застосування ітераційного підходу для створення відеогри є обґрунтованим рішенням, оскільки дозволяє поетапно реалізовувати та валідувати ключові ігрові механіки, адаптивно реагувати на зауваження користувачів, коригувати пріоритети та вносити необхідні модифікації на кожному спринті. Це забезпечує формування якісного ігрового досвіду, що відповідатиме очікуванням гравців.

2.3 Теоретичні аспекти дослідження

Процес створення програмного продукту був структурований як ряд послідовних етапів, подолання кожного з яких доповнювало проект новими, орієнтовно dokonаними рішеннями.

Розробка основної ігрової механіки базувалася на критичному переосмисленні класичної головоломки Chat Noir з метою створення більш інтуїтивної та візуально зрозумілої системи взаємодії.

Ключовою концептуальною зміною стала інверсія логіки ігрового простору: замість візуального позначення недоступних клітинок на суцільному полі було прийнято рішення про фізичне видалення клітинок з ігрового простору. Така трансформація дозволила створити більш наочну просторову модель, де гравець безпосередньо формує геометрію поля, а не оперує умовними позначеннями.

Окрім того, було переглянуто цільову механіку: якщо в оригінальній концепції противник прагне покинути ігрове поле через будь-яку периферійну клітинку, то в розробленій грі введено чіткі візуально виділені кінцеві клітинки, досягнення яких противником означає програш. Така система не лише спрощує розуміння цілей для гравця, але й створює додаткові можливості для стратегічного планування, оскільки противник рухається до конкретних точок, а не до абстрактного краю поля.

Ще однією важливою особливістю став перехід від шестикутної сітки до квадратної з діагональним рухом, що збільшило кількість можливих напрямків переміщення з шести до восьми. Таке ускладнення могло б зламати ігровий баланс, однак концепція кінцевих клітинок дає можливість регулювати кількість можливих точок програшу, компенсуючи додаткову свободу руху противника.

У сукупності ці рішення трансформували початкову кор-механіку до стану, що суттєво віддаляється від свого класичного прообразу, формуючи фактично нову системну модель із власною логікою взаємодії (рис. 2.2). Завдяки цим змінам вдалося зберегти фундаментальну тактичну глибину оригіналу, водночас істотно підвищивши читабельність ігрового стану та інтуїтивність взаємодії.

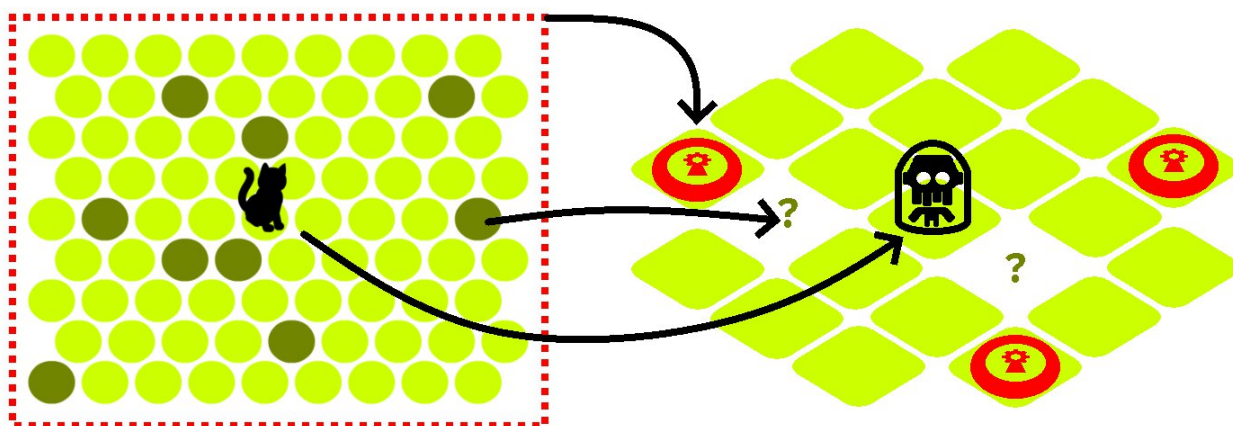


Рисунок 2.2 - Концептуальне порівняння відмінностей між Chat Noir та Retentum

Наступним етапом розробки стало формування сеттінгу гри, який мав не лише створити унікальну атмосферу, але й органічно доповнити абстрактну природу головоломки наративним контекстом.

Для гри було обрано поєднання хорор-естетики та затишної атмосфери — контраст, що рідко використовується в дизайні головоломок. Замість класичного надмірного тиску хорор тут працює як фоновий емоційний акцент: легка тривожність підсилює концентрацію й утримує напругу, тоді як затишний

візуальний шар забезпечує комфортний, неконфліктний простір для стратегічного мислення.

Ігровий світ представлений як плавуча лімінальна локація з плит певного матеріалу, що зависла в абстрактній порожнечі (рис 2.3). Лімінальні простори – це інтернет-естетика, що зображує порожні, перехідні місця, які мають моторошний, покинутий і часто сюрреалістичний вигляд. Такі локації викликають відчуття дискомфорту через свою знайомість та водночас неприродну пустотність – вони нагадують місця, де зазвичай мають бути люди, але натомість панує абсолютна відсутність життя [31]. У контексті гри лімінальність посилює відчуття ізоляції та абстрактності ігрового простору.

Для візуального оформлення гри розглядалися такі підходи як 2D, 3D та псевдо-3D. Остаточним рішенням стала повноцінна 3D-графіка, що зумовлено кількома практичними та естетичними факторами. По-перше, тривимірні моделі дозволяють уникнути необхідності в створенні множинних варіацій однієї анімації для різних кутів повороту персонажа. По-друге, 3D-середовище забезпечує значно більшу гнучкість у створенні та модифікації анімацій, дозволяючи швидко адаптувати рухи персонажа до різних ігрових ситуацій без необхідності перемальовування кадрів.



Рисунок 2.3 - Фінальна реалізація поля, кругла варіація

Особливої уваги заслуговує вибір зовнішнього вигляду противника – демона-мавпи в клітці (рис. 2.4). Цей образ є незвичайним рішенням, проте логічно вписується в концепцію гри. Мавпа без обмежень могла б швидко пересуватися ігровим полем завдяки своїй природній спритності, однак клітка обмежує її рухливість, пояснюючи покрокову механіку. Окрім функціонального обґрунтування, клітка надає противнику чітку мотивацію: досягти кінцевих клітинок, де розташовані механізми відкриття, щоб здобути свободу. Для посилення незвичайності образу та перетворення мавпи на впізнаваного маскота гри їй була додана кам'яна маска, що формує образ демонічної сутності та підкреслює містичну природу противника. Такий наративний контекст перетворює абстрактну задачу блокування руху на історію про утримання та втечу, додаючи емоційного виміру кожній партії.



Рисунок 2.4 - Фінальна реалізація противника та кінцевої клітинки

Згідно з обраним сетінгом було створено низку унікальних локацій, кожна з яких посилює атмосферу лімінальності та сюрреалістичності ігрового світу. Ці простори свідомо позбавлені ознак активного життя. Така естетична спрямованість підсилює відчуття ізоляції та абстрактності, перетворюючи кожную локацію на окремий сюрреалістичний досвід.

Plaza – стартова локація, побудована з однотипних сірих бетонних тайлів без виразного рельєфу (рис. 2.5). Це базова, нейтральна локація, що вводить

гравця у світ гри. За задумом вона нагадує спорожнілу міську площу – звичний публічний простір, який тепер перебуває в неприродній тиші.



Рисунок 2.5 - Реалізація локації Plaza

Paititi Meadow – це поле зелених трав'яних тайлів із поодинокими вставками темно-синіх дорогоцінних каменів (рис. 2.6). Назва локації відсилає до легенди про загублене інкське місто Пайтіті, нібито сповнене скарбів, і формує відчуття давнього, покинутого святилища.

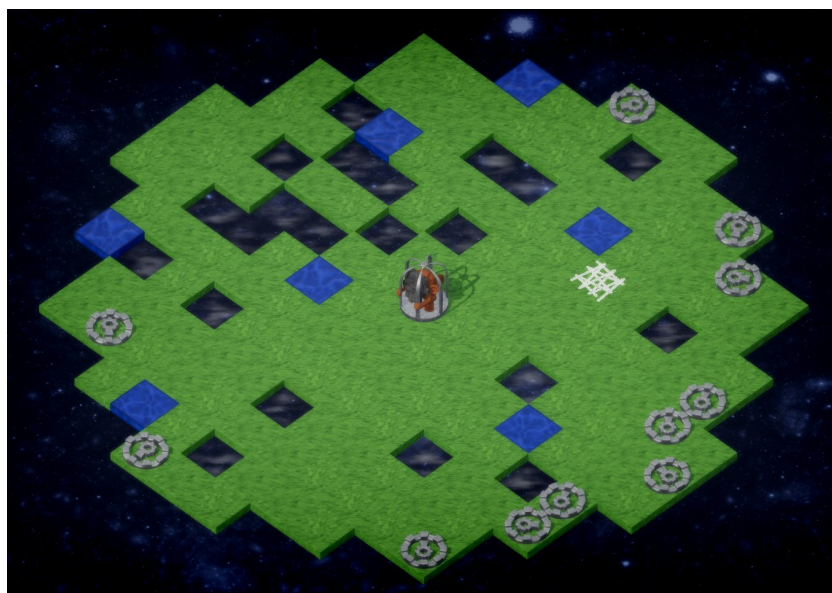


Рисунок 2.6 - Реалізація локації Paititi Meadow

Ballroom складається з чорних та білих керамічних плиток, розміщених у шахматному порядку (рис. 2.7). Бальна зала – класичний приклад лімінального простору.



Рисунок 2.7 - Реалізація локації Ballroom

Padded Cell повністю складається з тайлів помаранчевого буферу (рис. 2.8). Локація натхненна камерою з м'якими стінами для утримання небезпечних пацієнтів, а помаранчевий колір перегукується з традиційним відтінком тюремної роби у США, створюючи асоціацію із ув'язненням і ізоляцією.

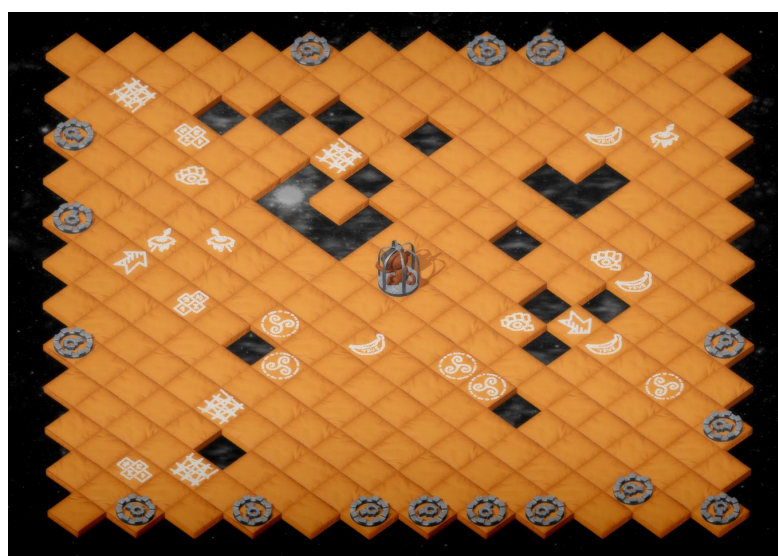


Рисунок 2.8 - Реалізація локації Padded Cell

Fluffy Prison складається з тайлів червоного та рожевого хутра із різними візерунками у формі сердець (рис. 2.9). Локація поєднує м'які, затишні текстури з похмурим контекстом пустоти, створюючи контраст між приємним на дотик середовищем і тривожною атмосферою простору.

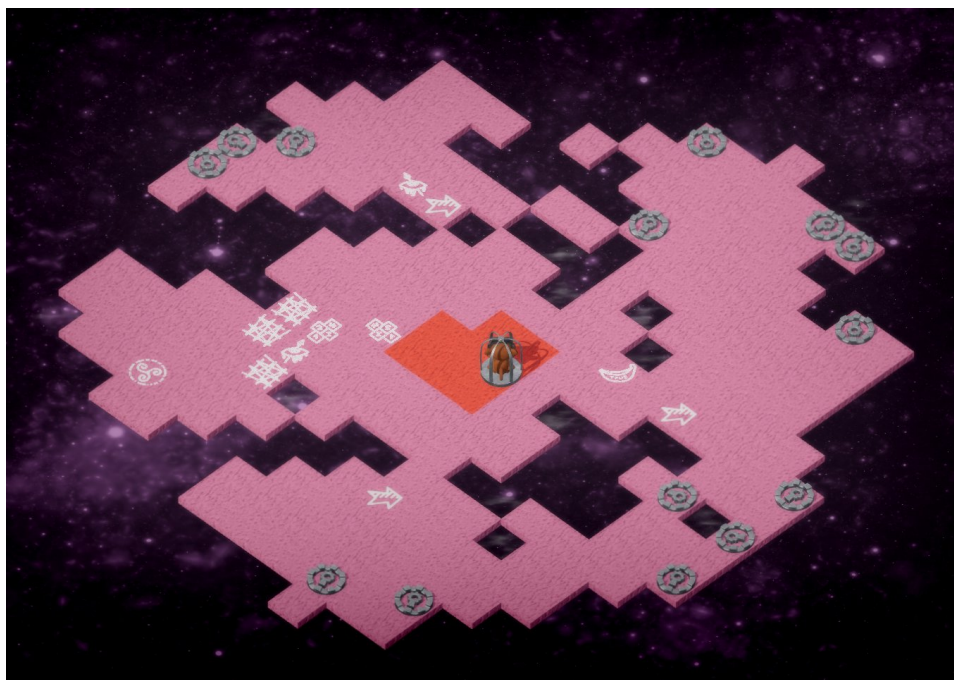


Рисунок 2.9 - Реалізація локації Fluffy Prison

Office складається з тайлів офісного килиму і може набувати великої кількості варіацій завдяки семи доступним кольорам та трьом візерункам місцевості: суцільне покриття, картатий і геометричні квадрати (рис. 2.10). Порожній офіс – типовий лімінальний простір сучасності: знайоме місце щоденної роботи, яке без людей набуває дещо моторошного, порожнього вигляду.

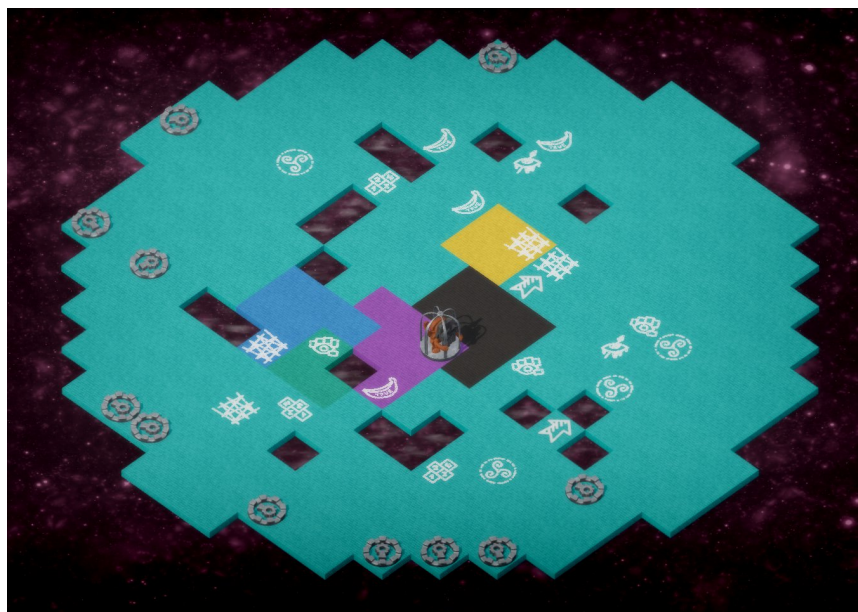


Рисунок 2.10 - Реалізація локації Office

Resilience Meadow складається з тайлів зеленої трави та іржавих плит із кількома варіаціями візерунків (рис. 2.11). Назва локації відсилає до анімаційного короткометражного фільму Юні Чої «Resilience», присвяченого темі відродження природи [32]. Поєднання природних і індустриальних елементів передає світ, де природа поступово поглинає залишки цивілізації.

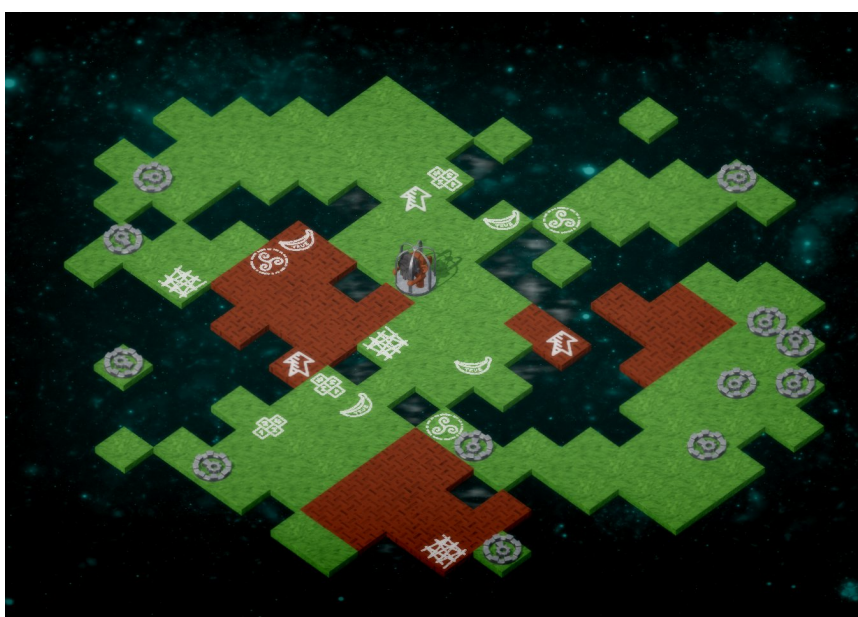


Рисунок 2.11 - Реалізація локації Resilience Meadow

Factory складається з тайлів сірого бетону та іржавих плит із кількома варіаціями візерунків (рис. 2.12). Промислова локація передає відчуття покинутості: колись гучні виробничі простори тепер порожні та занепалі.

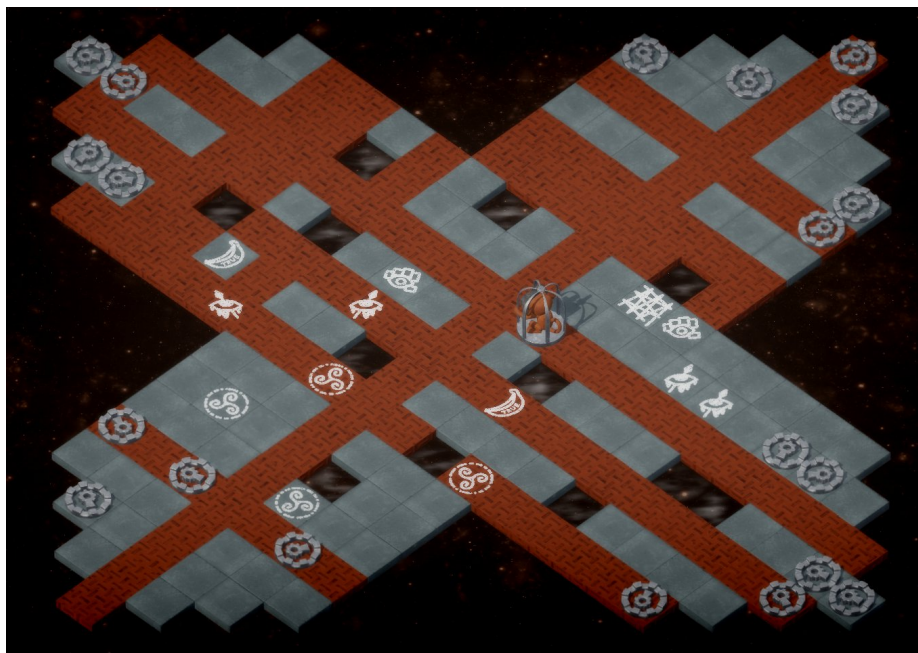


Рисунок 2.12 - Реалізація локації Factory

Spaceyard є фінальною локацією, що з'являється після 125 зіграних партій, та представляє собою сюрреалістичну мішанину з усіх можливих типів тайлів та візерунків місцевості. Також локація має унікальний шумовий візерунок, що символізує кульмінацію подорожі гравця через усі попередні простори, об'єднуючи їх у єдиний хаотичний колаж (рис. 2.13).



Рисунок 2.13 - Локація Spaceyard з шумовим візерунком локації

Важливим нововведенням, що значно розширило стратегічну глибину гри, стало додавання крейдяних модифікаторів клітинок. Матеріал та концепція крейди були обрані ще на етапі проектування першої локації Plaza, де сіре бетонне покриття природно асоціювалося з міським середовищем, де на асфальті часто з'являються крейдяні малюнки. Крейда виявилася ідеальним рішенням для механіки дворівневої взаємодії з клітинками: її легко стерти в реальному житті, що інтуїтивно пояснює необхідність додаткового кліку для видалення модифікатора перед знищенням самої клітинки.

Система крейдяних модифікаторів включає сім різних типів, кожен з яких унікально впливає на поведінку противника та стан ігрового поля [33]:

- Scribble – нейтральний вступний модифікатор без особливого ефекту, що слугує для ознайомлення гравця з механікою крейдяних позначень;
- Decoy – позитивний модифікатор, що змушує противника зупинитися та пропустити наступний хід при попаданні на позначену клітинку;
- Hopscotch – негативний модифікатор, що змушує противника здійснити подвійний хід у напрямку руху, якщо кінцева клітинка доступна, або залишитися на місці у протилежному випадку;

- Scotoma – позитивний модифікатор, який при попаданні противника знищує частину клітинок ігрового поля, потенційно блокуючи шляхи до виходу;
- Revelation – негативний модифікатор, який при попаданні противника відновлює частину клітинок ігрового поля, створюючи нові шляхи для руху;
- Sigil – нейтральний модифікатор з непередбачуваним ефектом, що телепортує противника у випадкову точку острова, радикально змінюючи ситуацію на полі;
- Guidance – нейтральний модифікатор, що повертає противника у вказаному крейдою напрямку та змушує рухатися у цьому напрямку на наступному ході, якщо клітинка доступна, роблячи наступний крок передбачуваним.

Візуальне оформлення та концепцію кожного крейдяного малюнку було ретельно продумано, більшість з них має відсилання до тих чи інших культурних явищ, що додає додатковий смисловий шар та робить ігрові елементи більш впізнаваними та запам'ятовуваними (рис. 2.14).



Рисунок 2.14 - Зображення всіх крейдяних модифікаторів з підписами

Така різноманітність модифікаторів вимагає від гравця не лише тактичного планування блокування шляхів, але й прогнозування взаємодії противника з крейдяними позначеннями, що значно збагачує стратегічну складову гри.

2.4 Обґрунтування вибору інструментальних засобів

У процесі створення гри було використано ряд інструментальних засобів, підібраних з огляду на їх функціональні можливості, зручність експлуатації та доступність. Основними інструментами, що використовувалися в даному проєкті, є наступні.

Як ігровий рушій було обрано Godot 4.5. Godot є безкоштовним відкритим інструментом із потужною спільнотою розробників та активною підтримкою. Поточна версія 4.5 демонструє істотні вдосконалення архітектури рушія та новий функціонал, що спрощує процес створення ігор. Godot також характеризується зручністю експлуатації та помірними системними вимогами, що робить його привабливим вибором для розробників будь-якого рівня.

Програмний код гри було повністю написано мовою програмування GDScript. Це нативна мова програмування Godot, спеціально створена для цього рушія. Її синтаксис схожий на Python, що полегшує процес вивчення та застосування. GDScript глибоко інтегрований із Godot, забезпечуючи безперебійну взаємодію між кодом та редактором рушія, що дозволяє ефективно втілювати ігрову логіку [34].

Для виготовлення графічних елементів та редагування зображень було застосовано безкоштовний растровий графічний редактор Paint.NET. Програма має легкий та інтуїтивно зрозумілий інтерфейс, широкий набір інструментів та підтримку .dll плагінів, що дозволяє розширювати її функціональність до будь-яких потрібних меж. Paint.NET є простим у використанні та достатньо потужним для задоволення потреб у створенні графіки для гри [35].

Для створення всіх 3D-асетів у проєкті було використано Blender – безкоштовний open-source інструмент, що забезпечує повний виробничий

пайплайн для ігрової графіки. Blender надає можливості полігонального моделювання, скульптингу, ретопології, UV-розгортки, процедурного та ручного текстурювання, включно з роботою з PBR-матеріалами. Крім того, інструмент містить засоби ригінгу та створення анімаційних циклів, що дозволяє готувати готові до експорту анімовані моделі. Вбудовані редактори матеріалів і нодова система забезпечують точний контроль над шейдерами, а експортери формату glTF/FBX дозволяють коректно інтегрувати моделі в ігровий рушій. Завдяки комплексності пайплайну Blender став основним інструментом для виготовлення всіх необхідних 3D-моделей, анімацій та текстур у цьому проєкті [36].

Для контролю версій коду протягом розробки гри було застосовано систему контролю версій Git та хмарний сервіс GitHub. Git є розподіленою системою контролю версій, яка дозволяє ефективно відстежувати зміни в коді, створювати гілки розробки, об'єднувати їх та повертатися до попередніх версій у разі потреби. GitHub у свою чергу забезпечує безпечне хмарне сховище для коду та надає зручний веб-інтерфейс для керування репозиторіями.

У процесі створення гри часто виникала потреба в реалізації діаграм, мокапів та схематичних зображень інтерфейсу. Для цього було застосовано онлайн-інструмент Diagrams.net. Цей безкоштовний ресурс дозволяє легко створювати різноманітні схеми, діаграми та макети, забезпечуючи зручний інтерфейс та широкий набір форм і фігур [37].

Для організації дизайн-документації, створення попунктових планів та ведення заміток було використано програму Obsidian. Це потужний інструмент для керування базами знань, який дозволяє структурувати та зв'язувати інформацію за допомогою гіпертексту, забезпечуючи зручний доступ та навігацію в документах [38].

Кожен із цих інструментів був ретельно підібраний з урахуванням його переваг, функціональності та відповідності потребам розробки гри. Їх комбінація дозволила ефективно реалізувати задум проєкту та забезпечити високу якість кінцевого продукту.

2.5 Особливості програмної реалізації

Програмна реалізація гри містить певні архітектурні рішення та технічні особливості, спрямовані на оптимізації продуктивності та забезпечення коректної роботи ігрової логіки.

Центральним компонентом системи є спеціально розроблена бібліотека OrthoGridMap, призначена для управління 3D-сіткою в ортогональній проєкції. Оскільки ігрове поле є ізометричним та повернутим на 45° , використання звичайних декартових координат призводило б до того, що логіка роботи з сіткою відбувалася б дивно відносно візуального представлення поля. Бібліотека реалізує систему перетворення координат зі зсувом (staggered coordinates), яка дозволяє працювати з повернутим полем як зі звичайним (рис. 2.15).

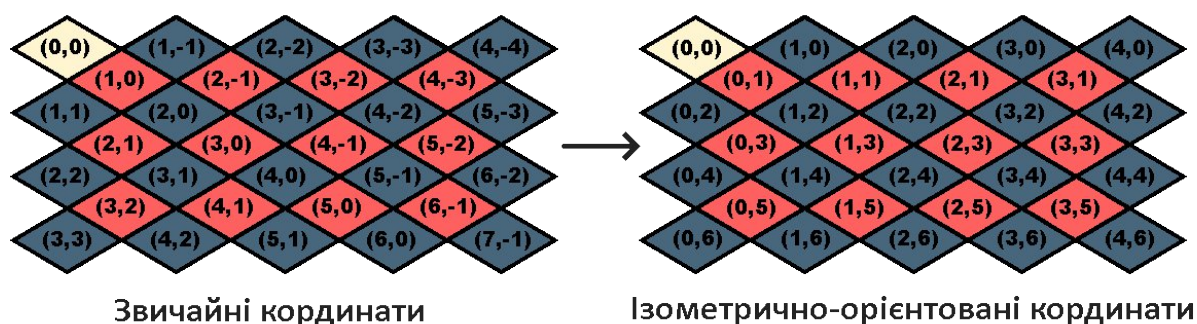


Рисунок 2.15 – Концепції перетворення звичайних координат у ізометрично-орієнтовані координати зі зсувом

Функціонал бібліотеки включає методи конвертації між системами координат (`get_staggered_from_true`, `get_true_from_staggered`) (рис. 2.16), трансформації між сітковими та світовими координатами (`grid_to_world`, `world_to_grid`), маніпуляції з об'єктами сітки (`set_cell_item`, `remove_cell_item`, `has_cell_item`) та пошук сусідніх клітинок з урахуванням всіх восьми напрямків руху, включаючи діагональні (`get_neighbors`, `get_neighbor_position`). Система

автоматично враховує парність/непарність координат для коректного обчислення сусідніх позицій у Staggered-системі.

Бібліотека OrthoGridMap активно використовується допоміжними скриптами, зокрема `shape_mapper` для генерації форм ігрового поля та `terrain_mapper` для створення візерунків поверхні.

```
## Converts true isometric coordinates to staggered isometric
static func get_staggered_from_true(grid_pos: Vector2i) -> Vector2i:
    # [not tested]
    var x = grid_pos.x
    var z = grid_pos.y
    var new_z = x + z
    var z_corrected = z + (1 if new_z % 2 != 0 else 0)
    var new_x = floori((x - z_corrected) / 2.0)
    return Vector2i(new_x, new_z)

## Converts staggered isometric coordinates to true isometric
static func get_true_from_staggered(grid_pos: Vector2i) -> Vector2i:
    var x = grid_pos.x
    var z = grid_pos.y
    var new_x = z + x - floori(z / 2.0)
    var new_z = z - x - floori(z / 2.0)
    if z % 2 != 0:
        new_z -= 1
    return Vector2i(new_x, new_z)
```

Рисунок 2.16 - Функції перетворення зі звичайних координат у координати зі зсувом і навпаки

Оскільки кожна нова ігрова сесія генерує нове поле з плиток, а об'єкт плитки є досить затратним для створення, було використано патерн об'єктного пулінгу (Object Pooling). Object Pooling – це техніка управління пам'яттю, за якої об'єкти створюються один раз і зберігаються в пулі для повторного використання замість постійного створення та знищення нових екземплярів [39].

Вся сітка плиток створюється один раз під час завантаження гри через глобальний singleton POOLGRID в окремому потоці, щоб не блокувати основний потік виконання (рис. 2.17).

```

10 ~ func _ready():
11     ~ is_generating = true
12     ~ initialize_grid()
13
14 ~ func initialize_grid():
15     ~ if grid_map != null:
16         ~ return
17     ~ grid_map = OrthoGridMap.new()
18     ~ if $is_gaps_on:
19         ~ grid_map.cell_size = Vector2(1.05, 1.05)
20     ~ else:
21         ~ grid_map.cell_size = Vector2(0.99, 0.99)
22     ~ grid_map.name = "GameField"
23     ~ if thread == null or not thread.is_started():
24         ~ thread = Thread.new()
25         ~ thread.start(_generate_tiles_in_thread)
26
27 ~ func _generate_tiles_in_thread():
28     ~ var tiles_to_add: Array = []
29     ~ for y in GRID_Y_MAX:
30         ~ for x in GRID_X_MAX:
31             ~ var tile = G_TILE_SCENE.instantiate()
32             ~ tiles_to_add.append({
33                 ~ position = Vector2i(x, y),
34                 ~ instance = tile
35             ~ })
36     ~ call_deferred("_add_batch", tiles_to_add)
37
38 ~ func _add_batch(tiles: Array) -> void:
39     ~ for item in tiles:
40         ~ grid_map.set_cell_item(item.position, item.instance)
41     ~ if thread != null:
42         ~ thread.wait_to_finish()
43     ~ thread = null
44     ~ is_generating = false
45     ~ SIGNAL.loading_poolgrid_completed.emit()

```

Рисунок 2.17 - Функція генерації сітки тайлів в окремому потоці

Між раундами плитки не знищуються, а лише скидаються до початкового стану через метод `reset_tiles()` (рис. 2.18), що виключає затратні операції створення та знищення сотень об'єктів та забезпечує плавність ігрового процесу.

```

~ func reset_tiles():
~ ~ for position in grid_map._grid_objects:
~ ~     ~ var tile = grid_map._grid_objects[position]
~ ~     ~ tile.tile_config = TileConfig.new()

```

Рисунок 2.18 - Функція скидання налаштувань тайлів

Окремої уваги заслуговує штучний інтелект противника (Monke). Його логіка побудована на поєднанні трьох ключових компонентів: пошук шляху (pathfinding) для навігації, машину станів (state machine) для керування

поведінковими режимами та процедурної анімації для забезпечення візуальної достовірності.

Пошук шляху реалізовано через алгоритм A^* , але з навмисною відмовою від ідеального пошуку шляху. Метод `find_path_A_STAR` містить механізм стохастичної модифікації ваг ребер графа: `astar.set_point_weight_scale(id, 1.0 + randf_range(-0.5, 0.5))`. Це впроваджує так званий «фактор дурості» (stupidity factor), завдяки якому агент не завжди обирає математично найкоротший шлях, а може робити "помилкові" петлі або обходи, що робить його поведінку менш передбачуваною та більш органічною для гравця (рис. 2.19).

```
func find_path_A_STAR(start: Vector2i, end: Vector2i) -> Array:
>| #A* algorithm
>| var astar = AStar2D.new()
>| var points: Dictionary = {}
>| var point_id = 0
>|
>| #Add all tiles, ignore NULL and DEAD
>| for cell: Vector2i in grid_map._grid_objects:
>|   >| var tile: Tile = grid_map.get_cell_item(cell)
>|   >| if tile.tile_core.tile_type != 6.TileType.NULL and tile.tile_core.tile_type != 6.TileType.DEAD:
>|   >|   >| var id = point_id
>|   >|   >| point_id += 1
>|   >|   >| astar.add_point(id, Vector2(cell.x, cell.y))
>|   >|   >| points[cell] = id
>|   >|   >|
>|   >|   >| var weight: float = 1.0 + randf_range(-0.5, 0.5) # monke stupidity factor
>|   >|   >| astar.set_point_weight_scale(id, weight)
>|   >|   >| if scary_cells.has(cell) && tile.tile_core.tile_type == 6.TileType.CHALKED:
>|   >|   >|   >| astar.set_point_disabled(id, true)
>|   >|   >|
>| #Add connections
>| for cell: Vector2i in points.keys():
>|   >| var neighbors: Array[Vector2i] = grid_map.get_neighbors(cell)
>|   >| for neighbor: Vector2i in neighbors:
>|   >|   >| if points.has(neighbor):
>|   >|   >|   >| var from_id: int = points[cell]
>|   >|   >|   >| var to_id: int = points[neighbor]
>|   >|   >|   >| if not astar.are_points_connected(from_id, to_id):
>|   >|   >|   >|   >| astar.connect_points(from_id, to_id, true)
>|   >|   >|   >|
>| #Path find
>| if points.has(start) and points.has(end):
>|   >| var start_id: int = points[start]
>|   >| var end_id: int = points[end]
>|   >| var point_path: PackedVector2Array = astar.get_point_path(start_id, end_id)
>|   >|
>|   >| #Map to Vector2i
>|   >| var cell_path: Array[Vector2i] = []
>|   >| for point: Vector2 in point_path:
>|   >|   >| cell_path.append(Vector2i(point))
>|   >| return cell_path
>|
>| return []
```

Рисунок 2.19 - Функція пошуку шляху алгоритмом A^*

Додатково система використовує короточасну пам'ять через масив `scary_cells`: вузли, у яких агент нещодавно зазнав невдачі, тимчасово виключаються з графа навігації. Це моделює інстинктивне уникнення небезпечних ділянок поля і створює ілюзію адаптивної поведінки (рис 2.20).

```
func add_scary_cell(cell: Vector2i, scary_time: int = 6):
>| if not cell in scary_cells:
>| >| scary_cells.append(cell)
>| scary_cell_timers[cell] = scary_time

func scary_cells_audit() -> void:
>| var cells_to_remove: Array[Vector2i] = []
>|
>| for cell in scary_cells:
>| >| if cell in scary_cell_timers:
>| >| >| scary_cell_timers[cell] -= 1
>| >| >| if scary_cell_timers[cell] <= 0:
>| >| >| >| cells_to_remove.append(cell)
>|
>| for cell in cells_to_remove:
>| >| scary_cells.erase(cell)
>| >| scary_cell_timers.erase(cell)
>| >| set_move_mode(MoveMode.SEARCH)
```

Рисунок 2.20 – Функції, що відповідають за пам'ять невдалих ходів

Також алгоритм поведінки антагоніста побудований на адаптивній логіці, яка пріоритезує цілеспрямоване переслідування, але коректно обробляє тупикові ситуації: якщо метод `find_path_A_STAR` не знаходить доступного маршруту до цілі, система миттєво перемикається на режим випадкового блукання через `perform_random_step`, що імітує природну розгубленість.

Взаємодія з ігровим світом централізована у функції `make_move`, яка реалізує послідовність «спочатку крок, потім реакція»: агент спершу фізично займає клітинку, і лише після цього активується програмна обробка її властивостей. Визначальну роль у цій механіці відіграють крейджні модифікатори, які виступають детермінантами подальшої поведінки: одні з них

(як GUIDANCE чи DECOY) жорстко задають вектор наступної дії або примусово переривають хід, тоді як інші (SIGIL, HOPSCOTCH) ініціюють миттєві рекурсивні виклики переміщення, дозволяючи агенту виконувати складні каскадні маневри в межах одного ігрового ходу. Огляд загального алгоритму калькуляцій при здійсненні руху подано у рис. 2.21.

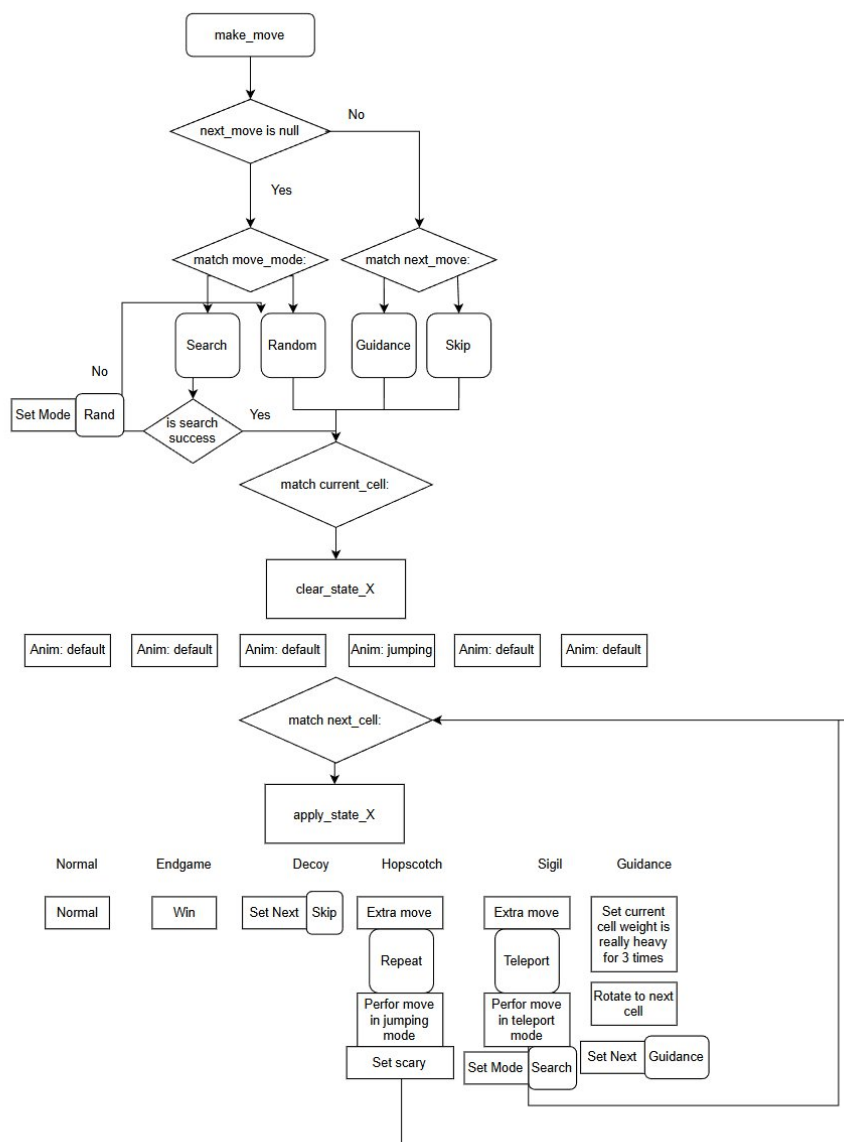


Рисунок 2.21 – Схема, що використовувалася при розробці алгоритму руху противника

2.6 Організація тестування та налагодження

Протягом створення відеогри ретельне тестування здійснювалося на всіх етапах розробки, відповідно до циклу обраної ітераційної моделі. Процес тестування включав як самостійну перевірку розробником, так і залучення зовнішніх осіб у ролі бета-тестерів.

Основний обсяг роботи припадав на поєднання інструментальної та мануальної перевірки функціональності. При впровадженні кожної нової можливості застосовувався вбудований налагоджувач (debugger) рушія Godot, який дозволяв відстежувати стан змінних, послідовність виконання методів та виявляти логічні помилки на етапі розробки. Інструментальне налагодження виконувалося з використанням точок зупинки (breakpoints), покрокового виконання коду та аналізу стека викликів, що забезпечувало оперативне виявлення та усунення програмних дефектів.

Паралельно з інструментальним налагодженням проводилося систематичне мануальне тестування кожної доданої частини функціоналу в реальних ігрових умовах. Це передбачало практичну перевірку роботи механік у різноманітних сценаріях взаємодії, включаючи як очікувані, так і нестандартні ігрові ситуації. Такий підхід дозволяв виявити помилки, які не проявлялися при статичному аналізі коду, але могли виникати при конкретних комбінаціях ігрових подій.

Окрім самостійного тестування розробником, до процесу перевірки залучалася група бета-тестерів різного віку та з відмінним рівнем досвіду у відеоіграх. Залучення зовнішніх тестувальників дозволило отримати об'єктивну оцінку ергономічності та збалансованості ігрового процесу. На основі зібраних відгуків було проведено коригування ігрової механіки, налаштування складності рівнів, оптимізацію системи керування та покращення загального користувацького досвіду.

Загалом, застосований комплексний підхід до тестування та налагодження, що поєднував інструментальне налагодження, мануальну перевірку

функціональності та залучення незалежних тестувальників, забезпечив високу якість кінцевого програмного продукту та дозволив мінімізувати кількість дефектів у фінальній версії гри.

2.7 Аналіз отриманих результатів дослідження, рекомендації щодо використання та впровадження

Розроблена 3D-гра-головоломка має потенційно широкий спектр застосування та може бути корисною у різних сферах людської діяльності. Основною цільовою аудиторією є підлітки та дорослі користувачі, які цінують інтелектуальні виклики.

Перш за все, гра рекомендована для розваг та проведення вільного часу, проте також може використовуватись для навчальних та розвиваючих цілей. Покрокова механіка та необхідність стратегічного планування роблять її привабливою для гравців, які надають перевагу виваженому, тактичному геймплею.

Велика кількість різноманітних ігрових елементів на полі робить геймплей нелінійним та поглиблює стратегічну складову. Кожна партія вимагає індивідуального підходу та адаптації тактики до конкретної ситуації, що стимулює когнітивну активність гравця та забезпечує високу реіграбельність.

Гру можна використовувати в якості навчального інструменту для розвитку стратегічного та просторового мислення. Необхідність аналізувати можливі маршрути переміщення противника, планувати послідовність дій та передбачати їх наслідки сприяє тренуванню важливих розумових навичок, таких як прогнозування, тактичне планування та прийняття рішень в умовах обмежених ресурсів (рис 2.22).

Окрім того, гра може застосовуватись у освітніх закладах як допоміжний засіб для розвитку алгоритмічного мислення та базових принципів теорії графів, оскільки ігровий процес фактично моделює задачі пошуку шляху та блокування вузлів у графі.

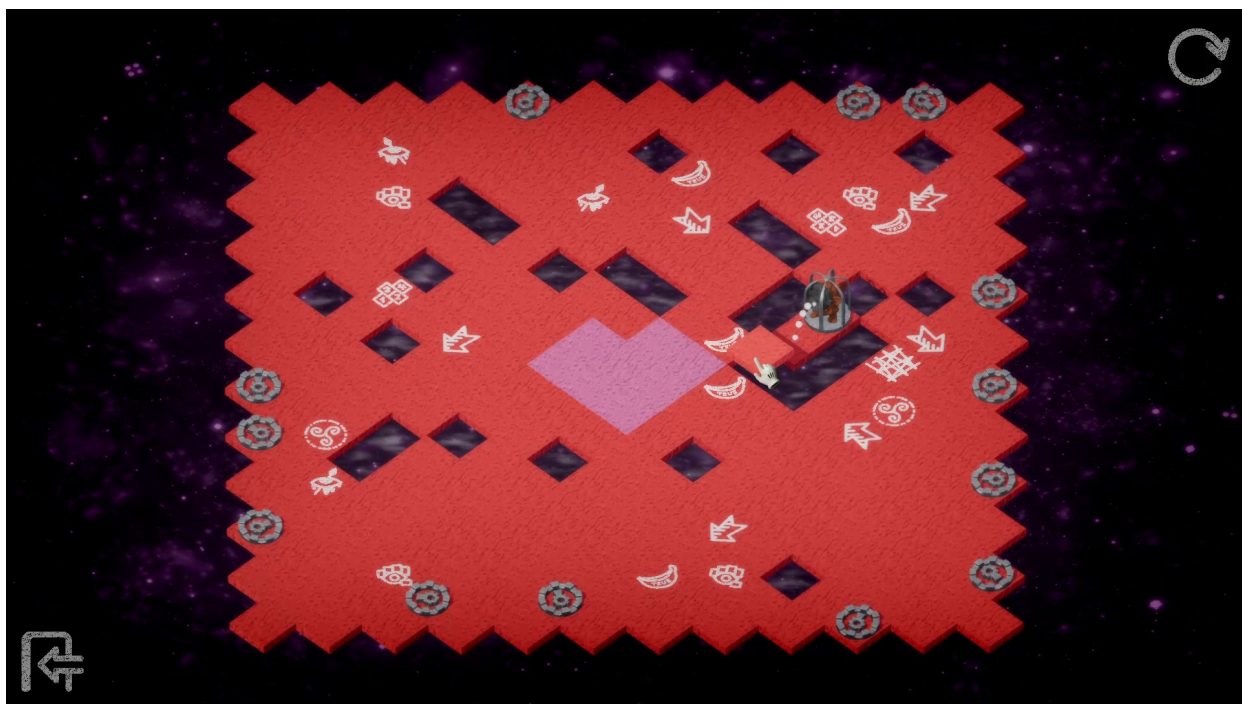


Рисунок 2.22 - Один з рівнів гри «Retentum» в процесі гри

ВИСНОВКИ

У даній роботі було досліджено теоретичні основи проектування ігор-головоломок, включаючи їх історичний розвиток та алгоритмічні принципи забезпечення коректності та розв'язності. Проаналізовано стратегії балансування складності та побудови прогресії в сучасних головоломках. Вивчено можливості ігрового рушія Godot 4.5, його архітектурні особливості та функціональні можливості для створення тривимірних інтерактивних систем.

На основі проведеного аналізу було розроблено концепцію тривимірної тактичної гри-головоломки «Retentum» з покроковою механікою, де основна ідея полягає у блокуванні руху противника на ігровому полі шляхом стратегічного знищення частин поля. Контент гри доповнений системою різноманітних ігрових елементів, з унікальними властивостями, що створюють нелінійний геймплей з поглибленою стратегічною складовою.

Реалізація розробленої концепції була здійснена в ігровому рушії Godot 4.5. Було створено повноцінний програмний продукт, що містить всі елементи, необхідні для справного функціонування.

Було проведено комплексне тестування та налагодження програмного продукту, що дозволило оптимізувати ігрову механіку та забезпечити високу якість кінцевого продукту.

Розроблений програмний засіб демонструє успішне втілення принципів проектування головоломок у тривимірному просторі та може використовуватися як для розважальних, так і для освітніх цілей, зокрема для розвитку стратегічного та алгоритмічного мислення. Створена архітектура та реалізовані ігрові системи мають значний потенціал і можуть слугувати основою для подальшого розвитку та масштабування проєкту.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Учасники проектів Вікімедіа. Головоломка – Вікіпедія. Вікіпедія. URL: <https://uk.wikipedia.org/wiki/Головоломка> (дата звернення: 02.12.2025).
2. Michalewicz M. Puzzle-based Learning: Introduction to critical thinking, mathematics, and problem solving. Credibility Corporation Pty Ltd, 2008. 346 p.
3. Danesi M. The Puzzle Instinct: The Meaning of Puzzles in Human Life. Indiana University Press, 2004. 288 p.
4. Solly M. The Best Board Games of the Ancient World. Smithsonian Magazine. URL: <https://www.smithsonianmag.com/science-nature/best-board-games-ancient-world-180974094/> (date of access: 02.12.2025).
5. Classical Chinese Puzzles. ChinesePuzzles.org. URL: <https://chinesepuzzles.org/> (date of access: 02.12.2025).
6. Contributors to Wikimedia projects. Propositiones ad Acuendos Juvenes - Wikipedia. Wikipedia, the free encyclopedia. URL: https://en.wikipedia.org/wiki/Propositiones_ad_Acuendos_Juvenes (date of access: 02.12.2025).
7. Contributors to Wikimedia projects. Video game - Wikipedia. Wikipedia, the free encyclopedia. URL: https://en.wikipedia.org/wiki/Video_game (date of access: 02.12.2025).
8. Contributors to Wikimedia projects. The Amazing Maze Game - Wikipedia. Wikipedia, the free encyclopedia. URL: https://en.wikipedia.org/wiki/The_Amazing_Maze_Game (date of access: 02.12.2025).
9. Contributors to Wikimedia projects. Loco-Motion (video game) - Wikipedia. Wikipedia, the free encyclopedia. URL: [https://en.wikipedia.org/wiki/Loco-Motion_\(video_game\)](https://en.wikipedia.org/wiki/Loco-Motion_(video_game)) (date of access: 02.12.2025).

10. Autoren der Wikimedia-Projekte. Q*bert – Wikipedia. Wikipedia – Die freie Enzyklopädie. URL: https://de.wikipedia.org/wiki/Q*bert (date of access: 02.12.2025).
11. Contributors to Wikimedia projects. Boulder Dash (video game) - Wikipedia. Wikipedia, the free encyclopedia. URL: [https://en.wikipedia.org/wiki/Boulder_Dash_\(video_game\)](https://en.wikipedia.org/wiki/Boulder_Dash_(video_game)) (date of access: 02.12.2025).
12. Contributors to Wikimedia projects. Sokoban - Wikipedia. Wikipedia, the free encyclopedia. URL: <https://en.wikipedia.org/wiki/Sokoban> (date of access: 02.12.2025).
13. Contributors to Wikimedia projects. Tetris - Wikipedia. Wikipedia, the free encyclopedia. URL: <https://en.wikipedia.org/wiki/Tetris> (date of access: 02.12.2025).
14. Contributors to Wikimedia projects. Myst - Wikipedia. Wikipedia, the free encyclopedia. URL: <https://en.wikipedia.org/wiki/Myst> (date of access: 02.12.2025).
15. Contributors to Wikimedia projects. Portal (video game) - Wikipedia. Wikipedia, the free encyclopedia. URL: [https://en.wikipedia.org/wiki/Portal_\(video_game\)](https://en.wikipedia.org/wiki/Portal_(video_game)) (date of access: 02.12.2025).
16. Contributors to Wikimedia projects. Cut the Rope - Wikipedia. Wikipedia, the free encyclopedia. URL: https://en.wikipedia.org/wiki/Cut_the_Rope (date of access: 02.12.2025).
17. Different Types of Puzzle Games for All Ages. thegamingtales.com. URL: <https://thegamingtales.com/different-types-of-puzzle-games-a-complete-guide-for-all-ages/> (date of access: 02.12.2025).
18. Russell S. J., Norvig P. Artificial Intelligence: A Modern Approach. Pearson Education, Limited, 2020.
19. Computers and intractability: A guide to the theory of NP-completeness / ed. by Johnson, David S., 1945- joint author. San Francisco : W. H. Freeman, 1979. 338 p.

20. Demaine E. D. Games, Puzzles, and Computation. A K Peters, Limited, 2009. 250 p.
21. Shaker N., Togelius J., Nelson M. J. Procedural Content Generation in Games. Cham : Springer International Publishing, 2016. URL: <https://doi.org/10.1007/978-3-319-42716-4> (date of access: 02.12.2025).
22. Учасники проєктів Вікімедіа. Ігровий рушій – Вікіпедія. Вікіпедія. URL: https://uk.wikipedia.org/wiki/Ігровий_рушій (дата звернення: 02.12.2025).
23. Contributors to Wikimedia projects. Game engine - Wikipedia. Wikipedia, the free encyclopedia. URL: https://en.wikipedia.org/wiki/Game_engine (date of access: 02.12.2025).
24. Contributors to Wikimedia projects. Unity (game engine) - Wikipedia. Wikipedia, the free encyclopedia. URL: [https://en.wikipedia.org/wiki/Unity_\(game_engine\)](https://en.wikipedia.org/wiki/Unity_(game_engine)) (date of access: 02.12.2025).
25. Contributors to Wikimedia projects. Unreal Engine - Wikipedia. Wikipedia, the free encyclopedia. URL: https://en.wikipedia.org/wiki/Unreal_Engine (date of access: 02.12.2025).
26. Contributors to Wikimedia projects. Godot (game engine) - Wikipedia. Wikipedia, the free encyclopedia. URL: [https://en.wikipedia.org/wiki/Godot_\(game_engine\)](https://en.wikipedia.org/wiki/Godot_(game_engine)) (date of access: 02.12.2025).
27. Why Godot is right for you. Godot Engine. URL: <https://godotengine.org/features/> (date of access: 02.12.2025).
28. Bloxorz. Flash Gaming Wiki | Fandom. URL: <https://flashgaming.fandom.com/wiki/Bloxorz> (date of access: 02.12.2025).
29. Chat Noir - Walkthrough, Tips, Review. Jay is games. URL: <https://jayisgames.com/review/chat-noir.php> (date of access: 02.12.2025).
30. What is Agile? | Atlassian. Atlassian. URL: <https://www.atlassian.com/agile> (date of access: 02.12.2025).
31. Contributors to Wikimedia projects. Liminal space (aesthetic) - Wikipedia. Wikipedia, the free encyclopedia. URL: [https://en.wikipedia.org/wiki/Liminal_space_\(aesthetic\)](https://en.wikipedia.org/wiki/Liminal_space_(aesthetic)) (date of access: 02.12.2025).

32. Resilience (Short 2023) | Short. IMDb. URL: <https://www.imdb.com/title/tt29041340/> (date of access: 02.12.2025).
33. Возняк М. А. Розробка дизайну гри-головоломки Retentum. Матеріали V Всеукраїнської науково-технічної конференції молодих вчених, аспірантів та студентів «Комп'ютерні ігри і мультимедіа, як інноваційний підхід до комунікації - 2025». 2025. URL: https://ontu.edu.ua/download/konfi/2025/Abstracts_Computer_games_and_multimedia_innovative_approach_electronic_communication_2025.pdf (дата звернення: 02.12.2025).
34. GDScript. GDScript. URL: <https://gdscript.com/> (date of access: 02.12.2025).
35. Ласкаво просимо на сайт Paint.NET. Paint.NET. URL: <https://paintnet.org.ua/> (дата звернення: 02.12.2025).
36. Blender 5.0 Manual. Blender Documentation - blender.org. URL: <https://docs.blender.org/manual/en/latest/> (date of access: 02.12.2025).
37. Contributors to Wikimedia projects. diagrams.net - Wikipedia. Wikipedia, the free encyclopedia. URL: <https://en.wikipedia.org/wiki/Diagrams.net> (date of access: 02.12.2025).
38. Obsidian - Sharpen your thinking. Obsidian. URL: <https://obsidian.md/> (date of access: 02.12.2025).
39. Object Pool · Optimization Patterns · Game Programming Patterns. Game Programming Patterns. URL: <https://gameprogrammingpatterns.com/object-pool.html> (date of access: 02.12.2025).

ДОДАТКИ

ДОДАТОК А

ТЕХНІЧНЕ ЗАВДАННЯ

1. Вступ

«Retentum» – тривимірна тактична гра-головоломка з покроковою механікою, розроблена в ігровому руші Godot 4.5. Гра представляє собою переосмислення класичної головоломки з додаванням системи крейджних модифікаторів, множинних візуальних локацій та системи прогресії.

2. Підстави для розробки

Розроблена програма була створена в рамках кваліфікаційної роботи на здобуття освітнього ступеня «магістр».

3. Призначення розробки

Програмний продукт призначений для розваг, розвитку стратегічного, просторового та алгоритмічного мислення у цільовій аудиторії підлітків та дорослих користувачів. Гра може використовуватися як у розважальних, так і в освітніх цілях для тренування навичок тактичного планування та прийняття рішень в умовах обмежених ресурсів.

4. Вимоги до програми чи програмного продукту

4.1 Вимоги до функціональних характеристик

Гра повинна забезпечувати:

- коректну роботу плиток як одиниці ігрового поля з підтримкою різних типів: звичайні, кінцеві, мертві, з крейджними модифікаторами;
- генерацію та відображення ігрового поля різних форм з відповідними візерунками;
- систему візуальних локацій з послідовним розблокуванням, що включає 9 різних локацій з унікальними візуальними стилями;
- систему крейджних модифікаторів з мінімум 7 різними типами, кожен з яких має унікальний вплив на поведінку противника;

- систему штучного інтелекту противника з алгоритмом пошуку шляху, можливістю руху у восьми напрямках та реакцією на крейдянні модифікатори;
- коректний перебіг гри з покроковою системою, перевіркою умов перемоги та програшу;
- систему прогресії та розблокування контенту з відстеженням кількості зіграних партій та збереженням прогресу;
- фіксовану ортогональну камеру з ізометричним оглядом та відображення всіх елементів за допомогою тривимірних моделей;
- інтуїтивно зрозумілий мінімалістичний інтерфейс та коректну обробку введення користувача.

4.2 Вимоги до надійності

На цільових платформах програма повинна працювати стабільно та без критичних помилок. Всі ігрові механіки повинні функціонувати коректно за будь-яких комбінацій ігрових подій.

4.3 Умови експлуатації

Для використання програми необхідно та достатньо мати її виконуваний файл та підтримку DirectX для рендерингу тривимірної графіки.

4.4 Вимоги до складу і параметрів технічних засобів

Програма функціонує на персональних комп'ютерах з операційною системою Windows 10 або новішою. Мінімальні системні вимоги включають процесор з підтримкою SSE2, 2 ГБ оперативної пам'яті та відеокарту з підтримкою OpenGL 3.3 / DirectX 11.

4.5 Вимоги до інформаційної і програмної сумісності

За умов дотримання вимог до складу і параметрів технічних засобів, додаткових вимог до інформаційної і програмної сумісності не виникає.

4.6 Вимоги до маркування і упаковки

Немає вимог до маркування і упаковки.

4.7 Вимоги до транспортування і збереження

Вимоги до транспортування та зберігання відсутні.

5 Вимоги до програмної документації

Програмна документація повинна включати технічне завдання та керівництво користувача.

6 Техніко-економічні показники

Програмний продукт не передбачає жодних витрат з боку користувача.

7 Стадії і етапи розробки

Етапи розробки відображені в наступних пунктах.

1. Дослідження теоретичних основ проектування ігор-головоломок та вивчення можливостей ігрового рушія Godot 4.5.

2. Розробка концепції гри, визначення основних ігрових механік, сеттінгу та візуального стилю.

3. Розробка програмного продукту за ітераційною моделлю з паралельним тестуванням на кожній ітерації.

4. Комплексне тестування гри з залученням бета-тестерів та внесення необхідних коригувань.

8 Порядок контролю і приймання

Програма повинна бути розроблена в часових межах, виділених під реалізацію магістерської кваліфікаційної роботи та прийнята в день її здачі разом з усіма супутніми матеріалами.

ДОДАТОК Б

КЕРІВНИЦТВО КОРИСТУВАЧУ

Загальні відомості

Найменування гри: «Retentum».

Функціональне призначення

Програмний засіб призначений для виконання наступних задач:

- надання користувачу очікуваного та справного повного ігрового процесу;
- реалізацію ігрового процесу з покроковою механікою блокування противника на ігровому полі, взаємодією з різними типами плиток, крейдяними модифікаторами та системою штучного інтелекту противника;
- забезпечення системи прогресії з поступовим розблокуванням нових візуальних локацій та ігрових елементів;
- відображення необхідної ігрової інформації та коректне реагування на дії користувача.

Умови застосування програми

Для запуску гри необхідний комп'ютер або ноутбук з операційною системою Windows 10 або новішою, процесор з підтримкою SSE2, 2 ГБ оперативної пам'яті, відеокарта з підтримкою OpenGL 3.3 / DirectX 11 та 500 МБ доступного простору на жорсткому диску.

Повідомлення оператору та опис роботи програми

При запуску гри, перед користувачем відкривається ігровий рівень, що виглядає наступним чином (рис. 3.1).

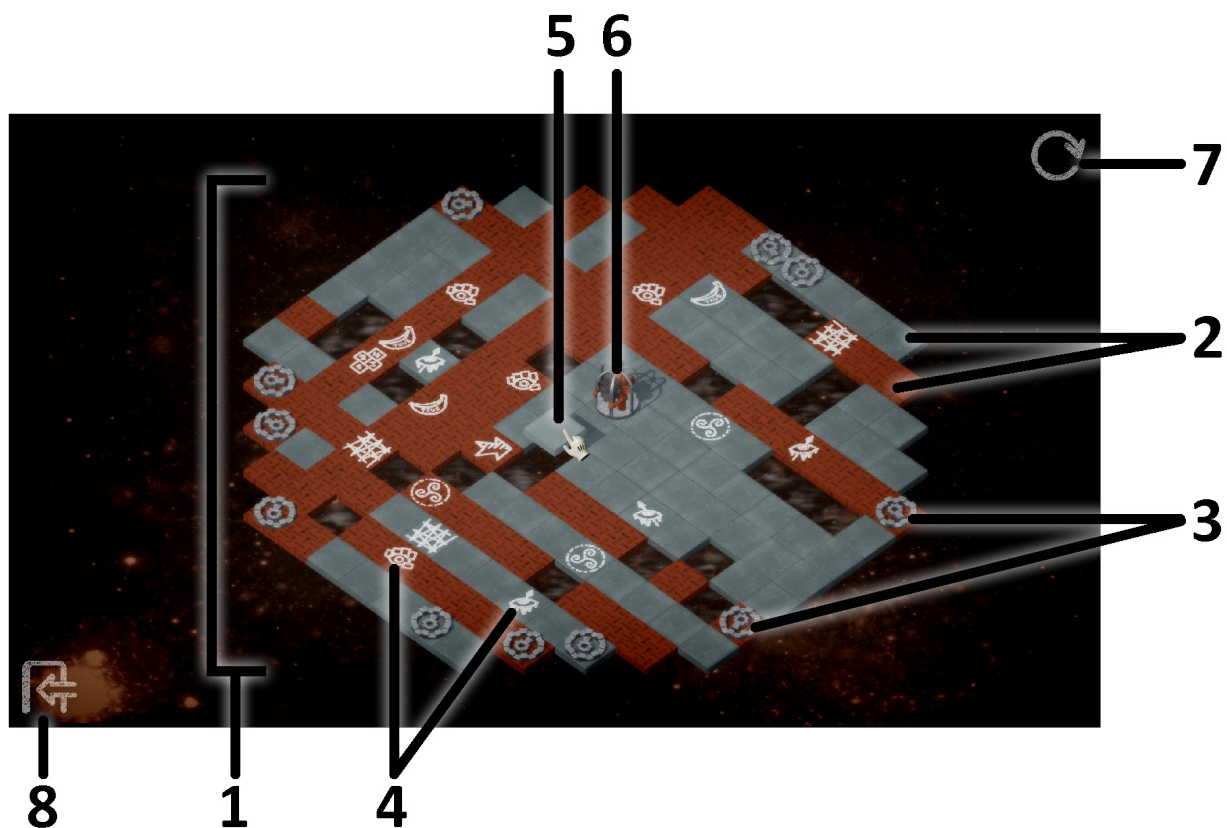


Рисунок 3.1 - Схема одного з рівнів гри

- 1 – поле гри
- 2 – звичайні плити
- 3 – кінцеві плити
- 4 – плити з крейдяними ефектами
- 5 – активна плита з наведеним курсором
- 6 – противник
- 7 – кнопка регенерації рівня
- 8 – кнопка виходу з гри

АНОТАЦІЯ

Возняк М. А. – Проєктування та розробка 3D-гри-головоломки на рушії Godot 4.5 – Рукопис.

Кваліфікаційна робота за спеціальністю 122 Комп'ютерні науки. – Волинський національний університет імені Лесі Українки, Луцьк. – 2025 р.

Робота присвячена проєктуванню та розробці тривимірної тактичної гри-головоломки «Retentum» на рушії Godot 4.5. У роботі досліджено теоретичні основи створення ігор-головоломок: їх історію та принципи проєктування. Основним завданням гри є блокування руху противника на ігровому полі шляхом стратегічного знищення клітинок. Гра доповнена системою крейджаних модифікаторів, що створюють нелінійний геймплей з поглибленою стратегічною складовою. Ігровий світ представлено як лімінальний простір, що поєднує хорор-естетику та козі-атмосферу. Робота охоплює всі етапи створення гри: від концептуалізації та дизайну до програмної реалізації та тестування.

Ключові слова: розробка відеоігор, гра-головоломка, тактична стратегія, Godot 4.5, 3D-графіка, ізометрична проекція, штучний інтелект, алгоритм A*, покрокова механіка, лімінальні простори, козі-хоррор, геймдизайн, системи прогресії.

ANNOTATION

Vozniak M. A. – Design and development of a 3D puzzle game on the Godot 4.5 engine – Manuscript.

Qualification work for the specialty 122 Computer Sciences. – Lesya Ukrainka Volyn National University, Lutsk. – 2025.

The work is dedicated to the design and development of the 3D tactical puzzle game "Retentum" using the Godot 4.5 engine. The work investigates the theoretical foundations of creating puzzle games, including their history and design principles. The main objective of the game is to block the opponent's movement on the game board through the strategic destruction of cells. The game is augmented by a system of chalk modifiers, which create non-linear gameplay with a deep strategic component. The game world is presented as a liminal space that combines horror aesthetics with a cozy atmosphere. The work covers all stages of game creation: from conceptualization and design to software implementation and testing.

Keywords: video game development, puzzle game, tactical strategy, Godot 4.5, 3D graphics, isometric projection, artificial intelligence, A* algorithm, turn-based mechanics, liminal spaces, cozy horror, game design, progression systems.