НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
“КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
імені ІГОРЯ СІКОРСЬКОГО”

Факультет інформатики та обчислювальної техніки

Кафедра обчислювальної техніки

До захисту допущено:
Завідувач кафедри
_____ Сергій СТІРЕНКО
(підпис)
“___”_____2021 р.

Дипломний проєкт

на здобуття ступеня бакалавра

за освітньо-професійною програмою “Комп’ютерні системи та мережі”

спеціальності 123 “Комп’ютерна інженерія”

на тему: «Комп’ютерна гра за мотивами настільної гри»

Виконав: студент 4 курсу, групи ІО-71
(шифр групи)
Лозко Леонід Леонідович _____
(прізвище, ім’я, по батькові) (підпис)

Керівник ас. Калюжний О. О. _____
(посада, науковий ступінь, вчене звання, прізвище та ініціали) (підпис)

Консультант (нормоконтроль) д.т.н. проф. Сімоненко В. П. _____
(назва розділу) (посада, вчене звання, науковий ступінь, прізвище та ініціали) (підпис)

Рецензент _____
(посада, вчене звання, науковий ступінь, прізвище та ініціали) (підпис)

Засвідчую, що в цьому дипломному проєкті немає запозичень з праць інших авторів без відповідних посилань.
Студент _____
(підпис)

Київ – 2021 р.

НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
“КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
імені ІГОРЯ СІКОРСЬКОГО”

Факультет інформатики та обчислювальної техніки

Кафедра обчислювальної техніки

Рівень вищої освіти – перший (бакалавр)

Освітньо-професійна програма

“Комп’ютерні системи та мережі”

спеціальність 123 “Комп’ютерна інженерія”

ЗАТВЕРДЖУЮ
Завідувач кафедри
_____ Сергій СТІРЕНКО
(підпис)
“___”_____2021 р.

ЗАВДАННЯ
на бакалаврський дипломний проєкт студента

Лозка Леоніда Леонідовича

1. Тема проєкту *Комп’ютерна гра за мотивами настільної гри*
керівник проєкту Калюжний Олександр Олегович, асистент ,
(прізвище, ім’я, по батькові, науковий ступінь, вчене звання)
затверджені наказом по університету від 11.05 2021 року № 1139-с
2. Термін здачі студентом закінченого проєкту_____
3. Вихідні дані по проєкту *технічна документація, теоретичні та статистичні дані.*
4. Зміст розрахунково-пояснювальної записки (перелік питань, які розробляються) *Опис та аналіз предметної області, аналіз інструментів для розробки програмного забезпечення, розробка програмного забезпечення.*
5. Перелік графічного матеріалу (з точним позначенням обов’язкових креслень) *структурна схема системи, узагальнена схема роботи системи, схема взаємодії програми.*

6. Консультанти проєкту, з вказівкою розділів проєкту, які до них вносяться

Розділ	Консультант	Підпис, дата	
		Завдання видав	Завдання прийняв
Нормоконтроль	Сімоненко В. П.		

7. Дата видачі завдання_____

Календарний план

№ з/п	Найменування етапів дипломного проєкту	Терміни виконання етапів проєкту	Примітки
1.	*Затвердження теми проєкту*	15.12.2020-14.01.2021	
2.	*Вивчення та аналіз завдання*	15.01.2021-14.03.2021	
3.	*Розробка загальної структури системи та її архітектури*	15.03.2021-24.03.2021	
4.	*Розробка структури компонентів системи*	25.03.2021-04.04.2021	
5.	*Програмна реалізація системи*	05.04.2021-19.04.2021	
6.	*Оформлення пояснювальної записки*	20.04.2021-19.05.2021	
7.	*Захист програмного продукту*	20.04.2021	
8.	*Захист*	16.06.2021	

Студент-дипломник _____
(підпис)

Керівник проєкту _____
(підпис)

Анотація

В цьому дипломному проєкті було розроблено комп’ютерну гру на основі існуючої настільної гри на базі безкоштовного ігрового рушія, призначену для проведення додзвілля.

Гравці змагаються один проти одного, граючи кожен за одну з двох команд-суперників.

Для реалізації гри використовується ігровий рушій Unity 2020, мова програмування C# та візуальне середовище Visual Studio 2019.

Annotation

In this diploma project a computer game based on the existing board game was developed with free game engine, designed for leisure.

Players compete against each other, playing for each of the two opposing teams.

To implement the game, the game engine Unity 2020, the C# programming language and the Visual Studio 2019 visual environment.

Опис альбому
до дипломного проєкту
на тему: «Комп’ютерна гра за мотивами настільної гри»

Київ – 2021 року

№ рядка	Формат	*Значення*	*Найменування*	Кіл. листів	№ екземпляра	*Додаток*
			Документація загальна			
			Знову розроблена			
1	***А4***	***ІАЛЦ.467800.001 ОА***	Комп’ютерна гра за мотивами	1		
			настільної гри			
			Опис альбому			
2	***А4***	***ІАЛЦ.467800.002 ТЗ***	Комп’ютерна гра за мотивами	3		
			настільної гри			
			Технічне завдання			
3	***А4***	***ІАЛЦ.467800.003 ПЗ***	Комп’ютерна гра за мотивами	66		
			настільної гри			
			Пояснювальна записка			
4	***А4***	***ІАЛЦ.467800.004 Д1***	Комп’ютерна гра за мотивами	1		
			настільної гри			
			Функціональна схема			
5	***А4***	***ІАЛЦ.467800.005 Д2***	Комп’ютерна гра за мотивами	1		
			настільної гри			
			Принципова схема			
6	***А4***	***ІАЛЦ.467800.006 Д3***	Комп’ютерна гра за мотивами	1		
			настільної гри			
			Структурна схема			

Зм.	*Арк.*	*№ докум.*	*Підп.*	*Дата*	*ІАЛЦ.467800.001 ОА*		
Розроб.		Лозко Л. Л.			*Комп’ютерна гра за мотивами настільної гри* *Опис альбому*	*Літ.*	*Арк.* 1 / *Аркушів* 1
Перев.		Сімоненко В. П.					
Реценз.						*“КПІ імені Ігоря Сікорського” ФІОТ ІО-71*	
Н. Контр.		*Сімоненко В.П.*					
Затв.		*Стіренко С. Г.*					

Технічне завдання
до дипломного проєкту

на тему: «Комп’ютерна гра за мотивами настільної гри»

Київ – 2021 року

ЗМІСТ

1. НАЙМЕНУВАННЯ ТА ОБЛАСТЬ ЗАСТОСУВАННЯ

Це технічне завдання поширюється на розробку комп’ютерної гри за мотивами настільної гри. Область застосування: організація дозвілля гравця.

2. ПІДСТАВИ ДЛЯ РОЗРОБКИ

Підставою для розробки є завдання на виконання бакалаврського проєкту професійної програми “Комп'ютерні системи та мережі” спеціальності 123 “Комп'ютерна інженерія”, затверджене кафедрою Обчислювальної техніки Національного технічного Університету України “Київський Політехнічний інститут імені Ігоря Сікорського”

3. МЕТА ТА ПРИЗНАЧЕННЯ РОЗРОБКИ

Метою цього проєкту є розробка комп’ютерної гри за мотивами настільної гри.

4. ДЖЕРЕЛА РОЗРОБКИ

Джерелами розробки є науково-технічна література, публікації в спеціалізованих періодичних виданнях, технічна документація, публікації в Інтернеті на цю тему.

5. ТЕХНІЧНІ ВИМОГИ

5.1. Вимоги до розробленого продукту

- Зручність та простота графічного інтерфейсу.
- Випадкові стартові умови при кожній ігровій сесії.
- Платформонезалежність.

5.2. Вимоги до програмного забезпечення

- Операційна система MS Windows XP, MS Windows Vista, MS Windows 7, MS Windows 8/8.1, MS Windows 10.

- Microsoft Visual C++ 2005-2008-2010-2012-2013-2015 Redistributable Package.

5.3. Вимоги до апаратної частини

- Коп’ютер на базі Intel Pentium 4 і вище.
- Обсяг оперативної пам’яті не менше 1024 Мбайт.

6. ЕТАПИ РОЗРОБКИ

	Дата
6.1 Вивчення необхідної літератури	21.02.2021
6.2 Складання і узгодження технічного завдання	08.03.2021
6.3 Написання вступної частини та огляд рішень	22.03.2021
6.4 Розробка архітектури додатку	04.04.2021
6.5 Написання програмної частини	11.04.2021
6.6 Тестування та виправлення помилок	04.05.2021
6.7 Оформлення документації дипломного проєкту	15.05.2021

Пояснювальна записка
до дипломного проєкту

на тему: «Комп’ютерна гра за мотивами настільної гри»

Київ – 2021 року

ЗМІСТ

ВСТУП

З часів появи персональних комп'ютерів вони поступово охоплювали все більше коло користувачів. Майже одразу ж після їх появи їх почали використовувати не лише для робочих цілей, а й для проведення дозвілля. Музика, відео, фільми, картинки – все це можна зберігати та відтворювати на комп'ютері в будь-який час. Не останнє місце серед можливостей проведення вільного часу за допомогою комп'ютера є комп'ютерні ігри. Їх особливістю є їх інтерактивність. Навіть якщо гра лінійна, гравець все одно може в будь-який час зупинитись, повернутись назад, щоб роздивитись якісь деталі або і обрати інший шлях.

З кожним роком популярність комп'ютерних ігор і їх кількість росте. Причинами цього є поширеність комп'ютерів у світі завдяки новим технологіям, які дозволяють робити їх все потужнішими та більш дешевими. Також свою роль зіграло широке розповсюдження мережі Інтернет, завдяки якій завантажити гру можна просто з дому, не потрібно шукати ігрові автомати або магазини з компакт-дисками з копією гри. Існує безліч різноманітних ігор від казуальних типу «три в ряд» до складних симуляторів автомобільних гонок та шутерів з реалістичною поведінкою зброї. Кожен може знайти щось для себе. Тому актуальність подібних комп'ютерних розваг з роками не зменшується.

У цьому дипломному проєкті розроблено комп'ютерну гру на основі настільної гри «Морська баталія або Гра в кораблики», метою якої є знищення ворожого флоту. В цій грі гравцям необхідно буде застосувати тактичне та стратегічне мислення для застосування факторів випадковості на свою користь та здобуття перемоги. Причиною розробки стало те, що прочитавши збірник правил цієї гри, виникло бажання пограти у неї. Однак для цього спочатку потрібно було провести певні підготовчі роботи: знайти ватман для ігрової карти, розмітити та розмалювати її, виготовити кораблі. У більшості гравців може не бути бажання, часу та матеріалів для виготовлення необхідного обладнання, отож було прийнято рішення створити комп'ютерний аналог цієї гри для того, щоб ознайомитись з нею могла більша кількість людей.

РОЗДІЛ 1
АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

1.1 Загальний огляд і класифікація настільних ігор

Настільна гра - гра, яку можна грати на столі або іншій відносно рівній поверхні. Гра відбувається за участі відносно невеликої кількості предметів, які можна вмістити в руках гравців або на столі. Однак це поняття досить умовне, адже не всі ігри можливо розмістити на звичайному столі і не для кожної гри він потрібен [2]. Настільні ігри можна класифікувати відповідно до їх механік та необхідного знаряддя:

- Карткові ігри (англ. card games) – основним спорядженням для гри є гральні карти. У загальному характеризуються випадковим початковим станом. Колоди (набори) карт можуть бути як традиційними (колода з 36 або 52 карт з 4 мастями), так і створеними спеціально для конкретної гри.
- Ігри з використанням грального костей (англ. dice games) – гральні кості є основним або допоміжним спорядженням для гри, їх кількість може варіюватись. Використовуються для отримання випадковості, яка в свою чергу є основним компонентом гри або одним з її факторів.
- Ігри з олівцем і папером (англ. paper and pencil games) – ігри з використанням лише паперу та знаряддя для письма, інколи різних кольорів, зазвичай без стирання. Популярні через відсутність потреби в складному обладнанні для гри.
- Настільні ігри з ігровим полем (англ. board games) – використовують попередньо розмічену ігрову поверхню (поле), по якій за певними правилами рухаються фігури. Можуть включати в себе елементи випадковості (нарди) або бути без них (шахи, шашки).
- Ігри з мініатюрами (англ. miniature games) – основним елементом гри є мініатюри (фігури), які також рухаються за певними правилами, однак без

участі спеціально розміченого ігрового поля (Сухопутна баталія, Warhammer 40000).

- Ігри на основі плиток та візерунків на них (англ. tile-based games) – основними елементами гри є плитки, які розташовують або забирають згідно певних правил.
- Рольові ігри (англ. tabletop role-playing games) – моделюють ситуацію у певному світі (реальному або вигаданому). Гравці відіграють певну роль, при цьому вони можуть імпровізувати в рамках встановлених правил.
- Рухливі ігри (англ. moving games) – ігри, які потребують спеціальну поверхню і вимагають від гравців активних рухів. Результат залежить саме від швидкості реакції гравця та його спритності (кікер, настільний хокей).
- Стратегічні ігри (англ. strategy games) – ігри, в яких вміння приймати правильні рішення та продумувати свої ходи та ходи суперника наперед є вирішальним для здобуття перемоги.

Така класифікація ігор є досить абстрактною, оскільки більшість настільних ігор мають ознаки більше, ніж однієї з категорій, тому класифікувати їх досить складно і такі ігри відносять одночасно до кількох категорій (наприклад, шахи є одночасно і настільною грою з ігровим полем і стратегічною грою, адже виграє той, хто глибше продумує ходи суперника і вибудовує кращу стратегію).

Існує класифікація настільних ігор за кількістю гравців, що беруть участь у грі:

- Для одного гравця – грає один гравець, умовою виграшу є певний фінальний стан ігрових предметів (в пасьянсах зазвичай це збір колод карт, в пазлах – відтворення картини зі шматочків).
- Для фіксованої кількості гравців – зазвичай грають два гравці один проти одного (шахи, нарди, шашки), хоча правила деяких ігор вимагають, наприклад, чотирьох гравців (маджонг).

- Для будь-якої кількості гравців – в теорії, може грати будь-яка кількість гравців, на практиці ж для таких ігор існує оптимум кількості гравців, за межами якого грати стає менш цікаво, гравцям важче реалізувати свої можливості. Або ж існує нижня межа кількості гравців (зазвичай в такі ігри один гравець грати не може) та верхня, яка обмежується, наприклад, кількістю фішок гравців або їх персонажів (Епічні сутички бойових магів).

Ця класифікація є більш однозначною, однак ті ж самі пазли можна складати і удвох.

За ступенем кооперації гравців між собою та за метою гри настільні ігри поділяються на:

- Ігри типу «кожен сам за себе» – кожен учасник гри є суперником для всіх інших, відповідно, є лише один переможець.
- Командні ігри – кожен гравець належить до певної команди, переможцями або переможеними вважаються члени всієї команди цілком, однак кожен гравець ходить самостійно. Умовою для перемоги команди може бути або досягнення гравцем якоїсь цілі, або сума балів всіх учасників команди.
- Коаліційні ігри – кожен гравець спочатку є суперником для інших, але гравці можуть об’єднуватись для того, щоб викинути з гри усіх інших. Далі члени коаліції знову стають суперниками.
- Кооперативні ігри – всі учасники гри прямують до спільного результату і відповідно виграють або програють усі разом у разі досягнення чи недосягнення цілі гри.

Класифікують ігри і за динамікою подій, що відбуваються під час процесу гри:

- Покрокові ігри – в процесі гри гравці роблять ходи у відповідній послідовності згідно правил. Обдумувати хід гравець може як завгодно довго, результат залежить не від швидкості реакції, а від правильності ходу учасника гри.

- Динамічні ігри – кожен гравець ходить одночасно, тому на результат впливає не тільки правильність ходу, але й швидкість реакції гравця на зміни в стані гри (усі рухливі ігри можна віднести до цього класу).

Поділ на покрокові ігри та динамічні також не є абсолютним, як приклад, в шведських шахах використовуються елементи покрокової гри (на кожному з двох шахових полів гра відбувається покроково, а обмін фігурами між гравцями – динамічно).

Настільні ігри класифікують також за характером гри і основним фактором, який впливає на результат:

- Інтелектуальні (логічні ігри) – шанси на перемогу залежать від здібностей учасника гри до аналізу ігрової ситуації, що склалася на даний момент і вміння зробити правильний хід.
- Азартні ігри – на результат гри впливає фактор випадковості. Роль випадковості може бути єдиною вирішальною для визначення переможця (гра в кістки, рулетка), або бути одним зі складників гри і впливати лише на можливості гравця під час чергового ходу.
- Ігри на фізичні здібності – швидкість рухів та реакція є вирішальними факторами для перемоги в таких іграх і рухливі ігри є найяскравішими представниками цього класу ігор.

Не всі настільні ігри можна строго віднести до однієї з підгруп. У покері, наприклад, випадкова роздача карт дуже впливає на розвиток гри, але обмеженість інформації дозволяє гравцям блефувати (робити вигляд, що насправді комбінація у них на руках значно гірша чи краща, ніж є насправді) і таким чином вигравати у опонента зі значно кращою комбінацією або провокувати опонентів зі слабшими комбінаціями на підвищення ставок з метою збільшення виграшу.

Відповідно до початкової позиції (стартового розташування фігур на полі, карт на столі тощо) існує поділ на:

- Настільні ігри з незмінною початковою позицією – кожна партія починається за однакових умов для кожного з учасників. Якщо фактор

випадковості відсутній, то однакові ходи призводитимуть до однакового фіналу партії.

- Ігри з випадковою стартовою позицією – початкові умови формуються випадковим чином або обираються з певного набору початкових варіантів, також рандомно, що може призвести до нерівності можливостей гравців вже від самого початку гри.
- Ігри з початковою позицією, що повністю залежить від гравця – у учасників гри може бути певний стартовий набір фігур тощо, який вони розміщують на ігровому полі.

Зазвичай правила строго вказують на тип початкової позиції гри, тому такий поділ на підгрупи є досить чітким.

З точки зору математики є ігри:

- З повною інформацією – кожен гравець знає всі можливі ходи свої та своїх суперників, а також має всю інформацію про ігрове поле в будь-який момент часу і про порядок подальших ходів гравців за будь-яких умов, які можуть виникнути в процесі гри.
- З неповною інформацією – до цієї групи належить гра, якщо протягом деякого проміжку часу один або більше гравців володіють інформацією про хід гри, яка є недоступною для усіх або частини суперників, або ж якщо фактор випадковості впливає на результат ходу або на його вибір чи порядок ходів.

Ця класифікація дуже чітко визначає клас гри. Деякі ігри з неповною інформацією будуються саме на розкритті неповноти інформації і їх ціллю є отримання повної інформації про стан гри (морський бій).

Для реалізації у дипломній роботі мною було обрано настільну гру «Морська баталія або гра в кораблики» з деякими модифікаціями, що роблять партії швидшими, а процес гри – більш інтенсивним. Для гри використовується спеціально розмічене ігрове поле, кожен з двох гравців має у своєму розпорядженні однаковий флот з кораблів кількох типів, стартове положення яких є випадковим, гравці ходять по черзі. Отже, класифікувати цю гру можна

як покрокову логічно-стратегічну гру з ігровим полем та випадковою стартовою позицією і повною інформацією для двох гравців типу «кожен сам за себе». Метою гри є кількісна перевага над флотом суперника або захоплення його Королівського порту.

1.2 Опис правил гри

Опис правил гри включає в себе опис ігрового поля, фігур, якими будуть ходити гравці, основні механіки, такі як переміщення, стрільба, захоплення Королівського порту а також умови, за яких та чи інша команда отримує перемогу в партії [1].

1.2.1 Ігрове поле або карта, по якій ходять кораблики

Усе ігрове поле розбите на клітинки. Клітинки бувають кількох типів і їх ігрова роль та кількість залежить від типу:

- Море – морські клітинки займають більшу частину ігрової карти. По ним кораблі можуть переміщатись, на цих полях виконуються всі основні ігрові механіки.
- Суша – сухопутні клітинки є другими за поширеністю після морських. Вони згруповані в острови. Всього островів 14. Біля кожного з островів знаходиться порт тієї чи іншої команди. По сухопутним полям кораблі ходити не можуть, однак вони не заважають стрільбі.
- Порт – складається з морських клітинок більш темного кольору. Всього портів 14 – по 7 у кожної команди. 6 із них є звичайними – в них на початку ігрової партії розташовуються кораблі. Сьомий порт є «королівським». Він складається лише з однієї «портової» клітинки і в нього можуть заходити тільки кораблі суперника для захоплення. По «портовим» полям кораблі ходять як по звичайним морським, але механіка шторму в портах діє інакше, ніж у відкритому морі.

- Рифи – підводні камені. Перешкоджають рух корабля за умов штилю чи свіжого вітру. Якщо корабель під час шторму зносить на риф – він розбивається і тоне.

1.2.2 Кораблі

У кожної сторони є по 28 кораблів. Кораблі можуть переміщатись, стріляти, захоплювати Королівський порт. Такі параметри як дистанція переміщення, стрільби, кількість пострілів та інші залежать від типу корабля та будуть надані у таблиці 1.1 і у таблиці 1.2. Всього є 12 типів кораблів: трипалубний лінкор (по 2 у кожного учасника гри), двопалубний лінкор (по 4), вітрильний фрегат (по 3), тендер (по 2), бриг (по 5), галеон (по 3), паровий фрегат (по 1), парова броненосна батарея (по 1), галера (по 4), паровий корвет (по 1), монітор (по 1), пароплав (по 1).

1.2.3 Вітер і переміщення кораблів

На початку кожного ходу випадковим чином обирається напрям та сила вітру. Всього є 8 напрямків згідно сторін світу: північ, південь, захід, схід, північний захід, північний схід, південний захід, південний схід. Вітру може взагалі не бути (штиль), він може бути середньої сили (свіжий вітер) та бути дуже сильним (шторм). В штиль переміщатись можуть лише кораблі з паровим двигуном (монітор та інші) або з веслами (галера, бриг, тендер), вітрильні ж кораблі, такі як вітрильний фрегат, переміщатись не можуть. Шторм же зносить усі кораблі у відкритому морі у напрямку вітру. Стріляти в шторм кораблі не можуть навіть в портах. Свіжий вітер дозволяє перемістити корабель на максимальну довжину ходу в напрямку вітру, в усіх інших напрямах довжина ходу зменшується на 1 з кожним кроком віддалення від напряму вітру (рис. 1.1). В штиль довжина ходу в усіх напрямках однакова. Перемістити кожен корабель можна лише один раз за хід.

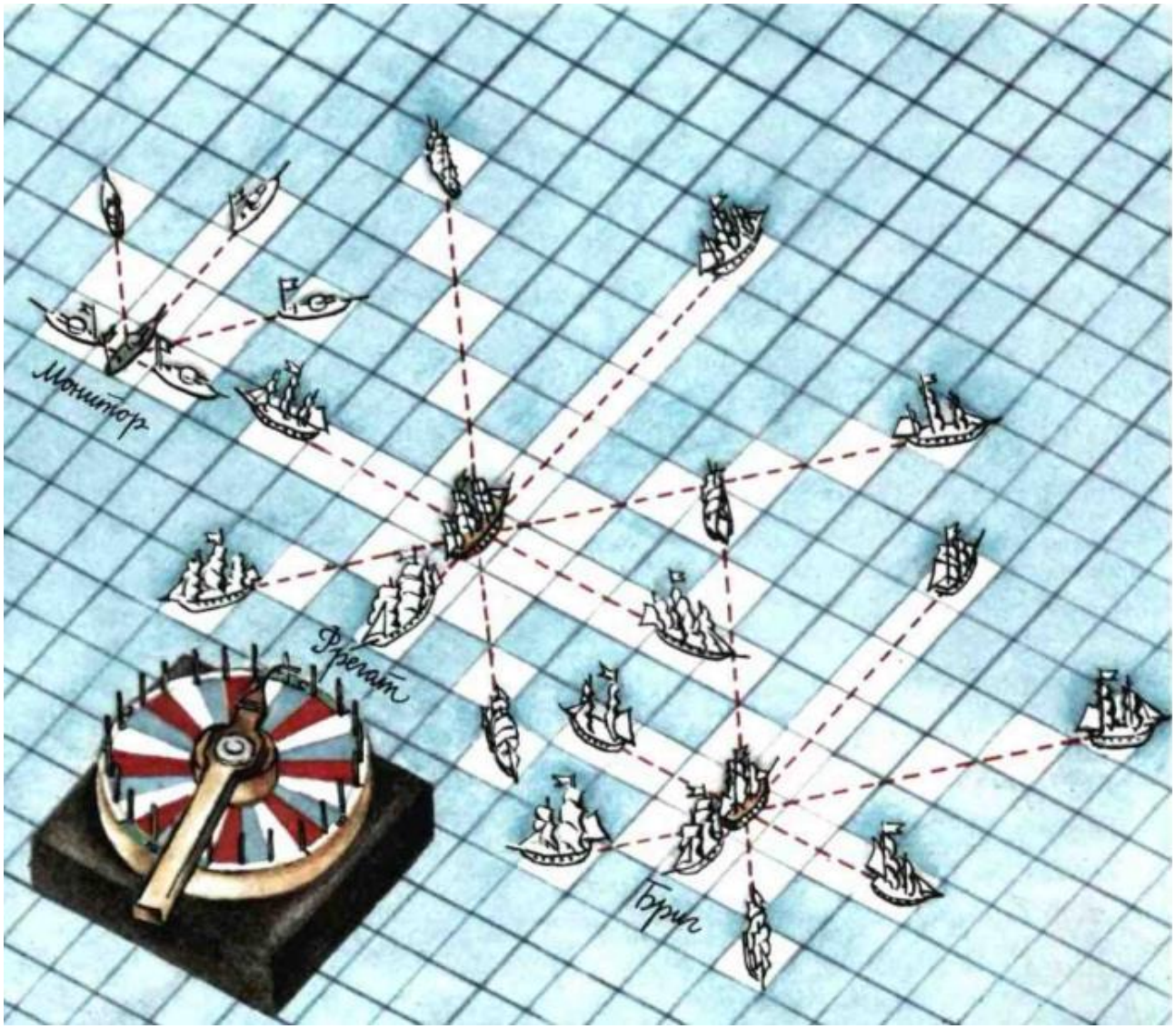


Рисунок 1.1 – Дальність ходу кораблів у кожному напрямі при північному свіжому вітрі

Таблиця 1.1 – Дальність ходу кораблів залежно від сили вітру

Тип корабля	Свіжий вітер (довжина ходу при попутному вітрі)	Штиль	Шторм (зносить корабель)
Трипалубний лінкор	5 клітинок	Не ходить	На 3 клітинки
Двопалубний лінкор	5 клітинок	Не ходить	На 3 клітинки
Вітрильний фрегат	6 клітинок	Не ходить	На 2 клітинки
Бриг	5 клітинок	1 клітинка	На 2 клітинки
Тендер	6 клітинок	1 клітинка	На 2 клітинки
Галеон	4 клітинки	Не ходить	На 3 клітинки

Продовження таблиці 1.1

Галера	5 клітинок	3 клітинки	На 1 клітинку
Парова броненосна батарея	4 клітинки	2 клітинки	На 2 клітинки
Паровий фрегат	6 клітинок	4 клітинки	На 1 клітинку
Паровий корвет	6 клітинок	4 клітинки	На 1 клітинку
Пароплав	5 клітинок	2 клітинки	На 2 клітинки
Монітор	3 клітинки	2 клітинки	На 3 клітинки

1.2.4 Стрільба кораблів

Кожен корабель у свій хід може стріляти. Стрільбу можна вести лише у восьми напрямах згідно сторін світу. Стріляти можна як до переміщення корабля, так і після нього. Стріляти кораблі можуть лише тоді, коли наблизяться один до одного на відстань пострілу і коли суперник знаходиться у зоні обстрілу (рис. 1.2). Влучання у корабель суперника зараховується з 25% шансом. Влучивши в корабель суперника достатню кількість разів, можна його затопити. В такому разі він зникає з ігрового поля.

Рисунок 1.2 – Зона обстрілу трипалубного лінкора

Таблиця 1.2 – Артилерійські можливості кораблів

Тип корабля	Кількість пострілів за один хід	Дальність стрільби гармат	Скільки пробоїн потрібно для його затоплення
Трипалубний лінкор	3	6 клітинок	7
Двопалубний лінкор	2	5 клітинок	5
Вітрильний фрегат	1	5 клітинок	4
Бриг	1	3 клітинки	3
Тендер	1	1 клітинка	2
Галеон	2	3 клітинки	3
Галера	1	3 клітинки	3
Парова броненосна батарея	3	6 клітинок	5
Паровий фрегат	1	3 клітинки	4
Паровий корвет	1	3 клітинки	3
Пароплав	1	3 клітинки	3
Монітор	1	3 клітинки	3

1.2.5 Захоплення Королівського порту

Гравець може вчинити спробу захоплення Королівського порту задля отримання перемоги. Для цього йому потрібно зайти в Королівський порт суперника. У гравця, чий Королівський порт знаходиться під загрозою захоплення, є 3 ходи, за які він має знищити корабель-загарбник або змусити його відступити, інакше йому зарахується поразка. Свої кораблі у свій же Королівський порт заходити не можуть, тому що це унеможливить його захоплення.

1.2.6 Початок гри та умови, за яких гра вважається виграною

У грі беруть учать двоє гравців. Кожен командує жовтим або зеленим флотом. Жовті ходять першими. На початку гри в портах випадковим чином розташовуються 28 кораблів по 4-6 кораблів у кожному порті залежно від місткості.

Гра вважається виграною, якщо:

- Одна зі сторін отримала кількісну перевагу в кораблях (у 4 рази) або якщо у суперника зовсім не лишилось кораблів.
- Корабель зайшов у Королівський порт і простояв там три ходи.
- Один із гравців здався.

1.3 Аналіз існуючих рішень

Існує багато різноманітних реалізацій настільних ігор. Деякі з них мають невеликий розмір та просту графіку (Нарди 2.0, Доміно), частина з них зможе запуститись не на будь-якому комп'ютері та потребує кілька гігабайт вільного простору (Hearthstone, The Elder Scrolls: Legends, Artifact). Популярні операційні системи також містять вбудовані настільні ігри, такі як Mahjong (Ubuntu 11.10 та новіші версії цієї операційної системи, Windows Vista, Windows 7), різноманітні пасьянси включались до складу операційних систем Windows починаючи з версії 3.0. Пасьянс «Косинка» було розроблено в 1988 році і він виявився особливо корисним у навчанні нових користувачів використанню мишки, оскільки такий спосіб взаємодії з операційною системою був не дуже поширеним [3].

Настільні комп'ютерні ігри мають як 2D, так і 3D графіку, яка може бути дуже детальною або навпаки, дуже спрощеною, щоб ніщо не відволікало від процесу гри. В онлайн-сервісі цифрового поширення комп'ютерних ігор і програм Steam, розробленому компанією Valve, наприклад, існує більше десятка реалізацій шахів, як платних, так і безкоштовних, тому кожен розробник намагається залучити гравців саме до своєї версії гри за допомогою певних графічних або геймплейних (таких, що виникають в процесі гри) особливостей.

Оскільки в цьому дипломі буде розроблена гра за мотивами настільної гри з ігровим полем, розглянемо кілька ігор в цьому жанрі.

1.3.1 Шашки

Настільна гра шашки від розробника English Checkers. Незважаючи на назву розробника, підтримує 10 варіантів правил шашок, включаючи англійські шашки (фігури рухаються по стандартній дошці для шахів 8x8 клітинок, можуть рухатись і бити лише вперед, дамки ходять лише на одну клітинку), російські шашки (фігури можуть бити назад, дамки ходять на будь-яку кількість клітинок), міжнародні шашки (поле розміром 10x10 клітинок, правила ходу як у російських шашок, але якщо є кілька варіантів взяття ворожих фігур, гравець мусить обирати варіант з найбільшою кількістю взятих фігур) та інші. Виконана у мінімалістичній 2D графіці (рис. 1.3).

Гра має більше 100 мільйонів скачувань і високі оцінки користувачів через низькі вимоги до апаратного забезпечення, підтримку кількох видів правил, простий дизайн.

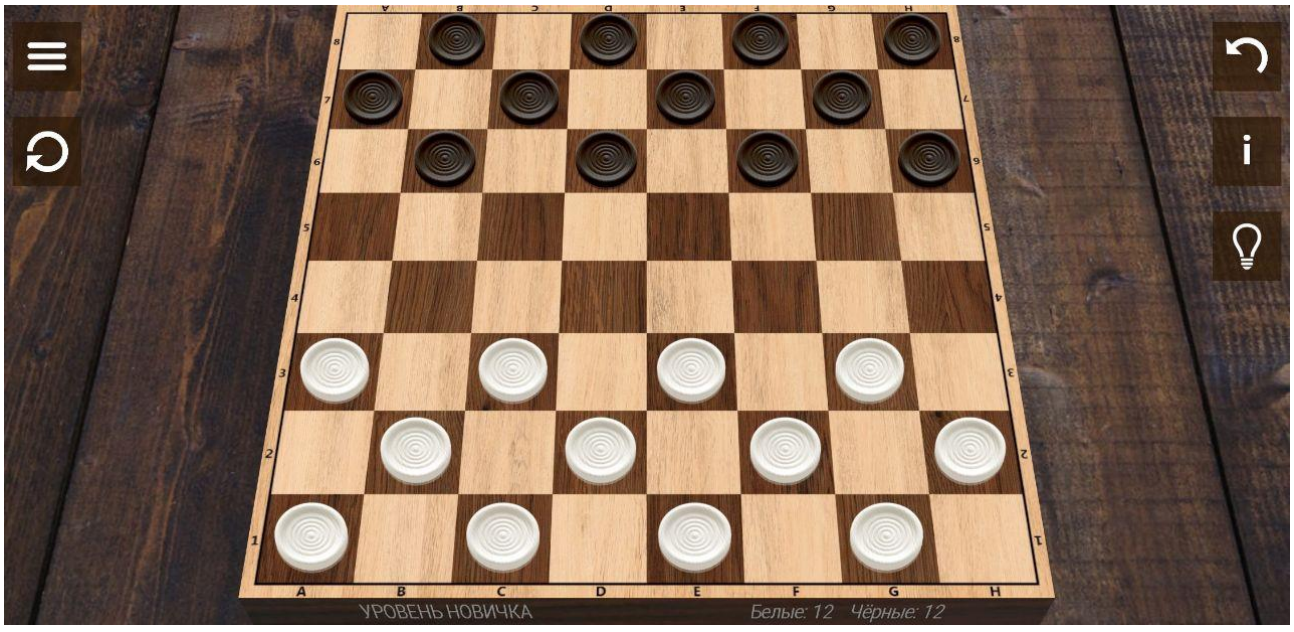


Рисунок 1.3 – Скріншот (знімок екрану) ігрового додатку Шашки

1.3.2 Monopoly Plus

Настільна гра Монополія [5] від розробника Ubisoft Pure. Реалізує класичні правила гри в Монополію. Розрахована на 2-4 гравців. Ігрова дошка розбита на поля, більшість з яких є об'єктами власності. Є кілька специфічних полів, таких як «В'язниця», «Штраф», «Сюрприз» та інші. На початку гри кожному гравцю видається певна сума грошей, які він може витрачати на покупку нічиїх об'єктів власності, орендну плату при потраплянні на чужий об'єкт власності, штрафи та інше. Отримуються гроші при проходженні повного кола полів через поле «Старт», при потраплянні суперника на об'єкт власності гравця у якості орендної плати і при потраплянні на спеціальні поля. Кількість полів, які гравець має пройти за свій хід визначається киданням двох кубиків. Ціллю гри є не стати банкротом, поки не збанкрутують всі суперники.

Гра має змішані відгуки користувачів. Основними недоліками у рецензіях вказані періодичні вильоти під час гри, високі системні вимоги через велику кількість анімацій та об'єктів.

Рисунок 1.4 – Скріншот ігрового додатку Monopoly Plus

1.3.3 Dota Auto Chess

Dota Auto Chess [4] – стратегічна ігрова модифікація для комп'ютерної гри Dota 2. Розроблена ігровою студією Drodo Studio на базі ігрового рушія Source. Гра створена за мотивами шахів та Dota 2. У грі беруть участь до восьми гравців, і вони розподілені між вісьма різними шахівницями. Гравці обирають із загального пулу з понад п'ятдесяти фігур шахів перед кожним туром, який вони розміщують на шаховій дошці. Кожна шахова фігура належить до певної раси та класу з індивідуальними здібностями. Після початку кожного раунду фігури протиборчих сторін автоматично б'ються. У деяких раундах гравці борються не з крипами (монстрами), а з іншими гравцями. Кожен гравець керує персонажем, кур'єром, який відстежує рівень здоров'я, рівень золота та рівень досвіду гравця. Досвід кур'єра визначає кількість шахових фігур на дошці. В кінці кожного раунду рівень кур'єрського досвіду буде збільшуватися на одиницю, поки не досягне десяти. Під час битви гравці накопичують золото, за допомогою якого вони можуть купувати нові фігури перед кожним раундом, щоб посилити свої комбінації для перемоги. З кожним програним раундом здоров'я кур'єра падає, і його смерть означає поразку відповідного гравця. Гра закінчується, коли залишається один гравець.

Гра отримала високі оцінки користувачів за свою новизну, цікавий геймплей та творчий підхід до правил гри. Популярність гри призвела до визнання нового жанру «авто-батлер» і надихнула на створення цілого ряду ігор в цьому жанрі. Однак через претензії з боку Valve гру довелось кардинально змінити і перейменувати.

Рисунок 1.5 – Скріншот ігрового додатку Dota Auto Chess

Висновки до розділу 1

У першому розділі було розглянуто поняття настільних ігор, їх класифікації. Настільні ігри класифікують за багатьма ознаками, кожна з яких є в тій чи іншій мірі важливою. Однак навіть за однією класифікацією більшість ігор можна віднести одразу до кількох типів, оскільки багато з них мають кілька ознак через їх механіки гри та правила.

Було розглянуто правила гри, що буде реалізована у рамках цієї дипломної роботи. Вони були взяті з оригінальної книги правил настільної гри “Морська баталія або гра в кораблики” з незначними змінами, які роблять ігровий процес менш затягнутим та більш активним. Основні правила лишились незмінними. Два гравці-суперники керують своїми флотами з 28 кораблів 12 різних типів. Вони можуть атакувати своїми кораблями ворожі щоб затопити якомога більшу їх кількість і отримати перемогу за кількісною перевагою або ж рухатись до протилежного краю карти щоб захопити королівський порт суперника і таким чином перемогти. У грі присутні елементи випадковості у вигляді стартового розташування кораблів, шансу на вдалий постріл по супернику та вітру, який може як допомогти виграти, так і стати на заваді і навіть стати причиною затоплення корабля, якщо він буде штормової сили і кине його на рифи. Тому навіть з найкращим розумінням гри існує імовірність зазнати черги невдач, які призведуть до поразки, однак уміння будувати тактику гри, витягувати із ситуації, що склалась, максимальну користь все одно є вирішальними у цій грі

Також було розглянуто кілька прикладів реалізацій комп’ютерних ігор за мотивами настільних ігор. Проаналізувавши існуючі комп’ютерні імплементації настільних ігор, було зроблено висновок, що дійсно близьких до оригіналу ігор не дуже багато, в основному, це варіації кількох ігор з різним графічним оформленням або ігри, в яких від оригінальних настільних ігор залишилась тільки ідея або ігрова історія. Однак є приклади вдалого симбіозу настільних ігор та інших жанрів.

Розділ 2
АНАЛІЗ ТЕХНОЛОГІЙ ДЛЯ РЕЛІЗАЦІЇ ПРОЄКТУ

2.1 Аналіз відомостей про ігрові рушії

Ігровий движок – це основне ядро гри, основне програмне забезпечення, на якому побудовані всі інші частини гри. Програмний код, який можна використовувати для створення варіантів гри, доповнень до неї або навіть абсолютно нового ігрового світу.

Визначення вперше з’явилося в середині 90-х, коли почали з’являтися ігри, схожі на головного шутера того часу - Doom. Одночасно у вільному доступі почали з’являтися ігрові двигуни, на основі яких як сторонні розробники, так і звичайні користувачі могли спробувати писати власні ігри.

З тих пір ігрові двигуни стали більш технічно складними, тривалими та вимагають коду. Але в той же час, як і на початку свого існування, вони містять жорстко фіксовані дані:

- Ігрова логіка – все, що надбудовується над базовим абстрактним рівнем додатку, будь-яка специфіка на кшталт ігрових станів, об’єктів і операцій з ними;
- Фізика об’єктів – правила, за якими об’єкти взаємодіють один з одним на з їх оточенням, симуляція реальних фізичних законів нашого світу у світі віртуальному;
- Правила візуалізації об’єктів – правила, за якими двовимірні або тривимірні об’єкти відображаються на екрані монітора під час процесу гри;
- Ігровий процес загалом – компонент, що відповідає за інтерактивну взаємодію гри та гравця.

Всі інші компоненти гри написані "зверху" на двигуні, і їх багато. Тому, навіть використовуючи один і той же двигун, віртуальні світи виявляються абсолютно різними.

Однак все ж існують певні обмеження. Наприклад, один і той же двигун не можна використовувати для стратегічних і екшн-ігор, рольових ігор та тактики. Зазвичай один двигун призначений для ігор одного або декількох суміжних жанрів [6].

2.2 Аналіз сучасних ігрових рушіїв

Існує багато ігрових рушіїв. Великі компанії при розробці AAA (triple-a, читається «тріпл-ей» - неформальний термін, що використовується для позначення класу високобюджетних проектів) ігор створюють власні ігрові рушії, які найкраще оптимізовані саме під вимоги їх проектів. В той же час існують ігрові рушії, які розроблялися для широкого кола розробників, що позбавляє їх від потреби створювати власний. Це сильно спрощує розробку для студій, що складаються всього з кількох чоловік. В цій дипломній роботі буде зроблено огляд саме таких ігрових рушіїв, і знайдено оптимальний для поставлених вимог.

2.2.1 Unity

Unity – це кросплатформне IDE (Integrated Development Environment – інтегроване середовище розробки) для створення комп’ютерних ігор, а також кросплатформний ігровий рушій. Unity дає можливість створювати додатки, що можуть працювати у понад 20 різних операційних системах, включаючи персональні комп’ютери, мобільні пристрої, ігрові приставки, і навіть веб-додатки та інші. Unity вийшов у 2005 році і з того часу постійно підтримується і розвивається.

Одними із основних переваг Unity є наявність графічного середовища для розробки, підтримка кросплатформних розробок та модульна система компонентів. До недоліків можна віднести ускладнене підключення зовнішніх бібліотек та труднощів при роботі зі схемами, які містять у своєму складі багато компонентів.

Тисячі ігор, додатків та симуляцій написані в Unity, охоплюючи багато платформ і жанрів. У той же час Unity використовується і незалежними студіями, і великими розробниками.

Редактор Unity (рис. 2.1) має простий інтерфейс Drag & Drop, який складається з різних вікон і має можливість налаштування під потреби розробника, тому ви можете налагодити гру прямо в редакторі. Рушій підтримує мову сценаріїв C#, до версії 2017.1 підтримувалась також модифікація мови JavaScript, а також була підтримка діалекту мови Python Boo, видаленого у версії 5. Фізичні обчислення виконуються фізичним двигуном PhysX від NVIDIA.

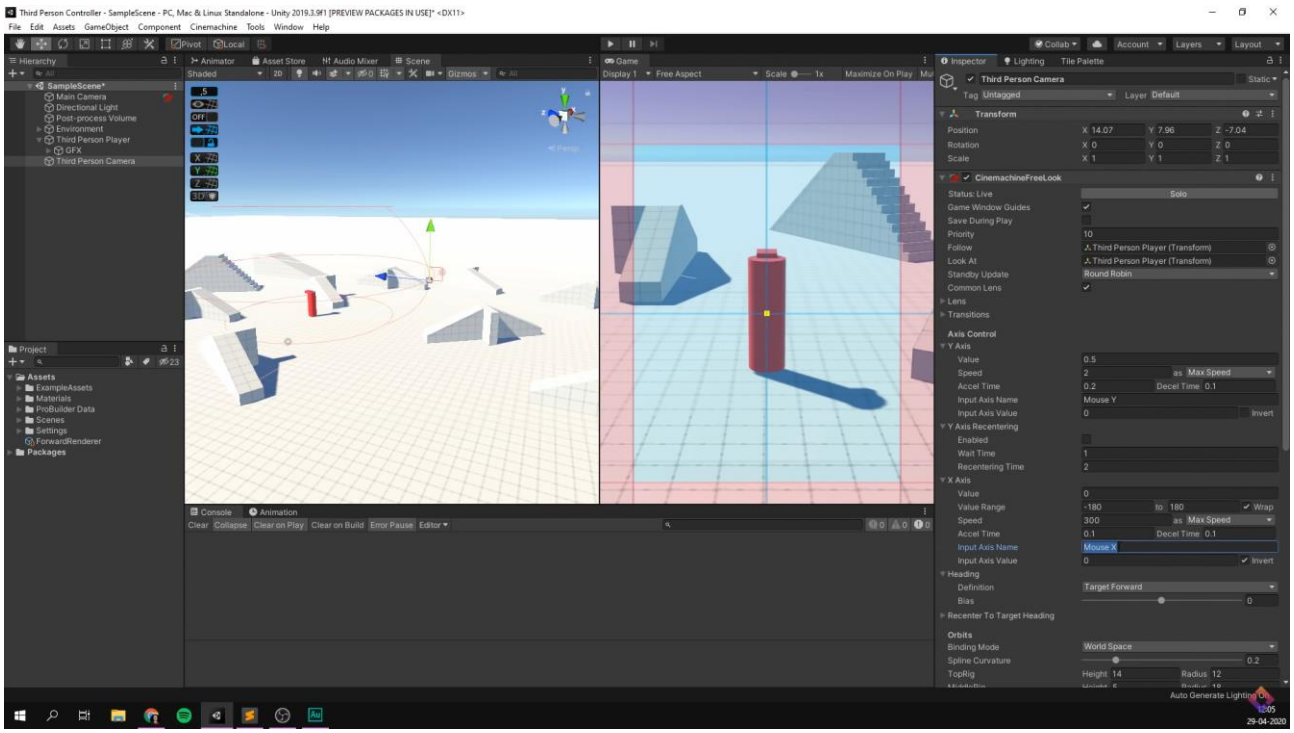


Рисунок 2.1 – Скріншот інтерфейсу редактора Unity

Проект в Unity складається зі сцен (рівнів) – окремих файлів, що містять їх ігровий світ із власним набором об’єктів, налаштувань та сценаріїв. Сцени можуть містити як, власне, об'єкти (моделі), так і порожні ігрові об'єкти, що не мають моделі ("манекени"). Об'єкти, в свою чергу, містять у собі набори компонентів, з якими відбувається взаємодія сценаріїв. Крім того, об’єкти мають назву (в Unity припустиме існування двох або більшої кількості об’єктів з однаковими іменами), можлива присутність тегу (мітки) і шар, на якому його буде відображено. Крім того, будь-який об’єкт на сцені повинен мати

компонент Transform - який зберігає координати розташування, обертання та розміру об’єкта по всіх трьох осях. Об'єкти з видимою геометрією також мають за замовчуванням компонент Mesh Renderer, що робить об'єктну модель видимою.

Колізії (зіткнення) можуть застосовуватися до об'єктів (в Unity вони називаються colliders – колайдери), яких існує кілька типів.

Unity також підтримує фізику тканин та твердих тіл, а також фізику Ragdoll («лялькова» фізика – використовується для обрахування неконтрольованого падіння живих істот). Редактор має систему наслідування об’єктів: дочірні об'єкти повторюватимуть всі зміни положення, обертання та масштабу батьківського об'єкта. Скрипти (сценарії) в редакторі приєднуються до об'єктів як окремі компоненти.

Під час імпорту текстури в Unity ви можете генерувати альфа-канал, mip-рівні, карту нормалей, карту світла, карту відображень, але ви не можете прикріпити згенеровану текстуру до моделі безпосередньо – буде створений матеріал, до якого буде призначено шейдер, а потім до моделі буде прикріплено цей матеріал. Редактор Unity підтримує написання та редагування шейдерів. Також редактор Unity має компонент для створення анімації, однак її можна також створити у 3D-редакторі заздалегідь та імпортувати разом із моделлю, а після цього розділити на файли.

Unity 3D має підтримку системи Level Of Detail (скорочено LOD), яка полягає в заміні високодеталізованих моделей на менш деталізовані при збільшенні відстані від камери гравця, і навпаки, а також системи Occlusion culling, яка полягає в тому, що геометрія та зіткнення об’єктів, які не потрапили у поле зору ігрової камери, не візуалізуються, що робить меншим навантаження на центральний процесор і таким чином оптимізує проект. Коли компілюється проект, для системи Windows створюється виконуваний (з розширенням .exe) файл гри, а дані гри (включаючи всі рівні гри та бібліотеки динамічних посилань) створюються в окремій папці.

Рушій підтримує багато популярних форматів. Моделі, матеріали, текстури, сценарії, звуки можуть бути упаковані у формат .unityassets і передані до рук інших розробників, або викладені для вільного доступу. Той самий формат використовується у внутрішньому Unity Asset Store, де розробники можуть ділитися різними елементами, які потрібні їм під час створення ігор безкоштовно та за гроші. Для використання магазину активів Unity потрібно мати обліковий запис розробника Unity. У Unity є всі компоненти, необхідні для створення багатокористувацької гри. Ви також можете використовувати метод контролю версій, який підходить вашому користувачеві. Наприклад, Tortoise SVN або Source Gear.

Unity містить Unity Asset Server, набір інструментів спільної розробки, заснований на Unity, який є надбудовою, що додає контроль версій та ряд інших серверних рішень [8].

Серед переваг цього ігрового рушія потрібно відмітити:

- Інтерфейс Drag&Drop, що є зручним у використанні, має можливість налаштування під свої потреби та дозволяє швидко розташувати необхідні об’єкти та одразу ж протестувати роботу додатку.
- Використання високорівневої мови програмування C#, яка має низький поріг входження та широкий вибір уже реалізованих інструментів, якими розробник може скористатись при розробці гри.
- Кросплатформність – ігри, можна збирати під великий список платформ, що дозволяє випустити гру з мінімальними змінами у вихідному коді на кількох платформах одночасно. Середовище розробки також є кросплатформним, тому для створення ігор також можна використовувати різні платформи.
- Модульна система компонентів, за допомогою якої конструюються ігрові об’єкти, до яких додаються різні функціональні елементи, що полегшує створення прототипів та їх просте модифікування.

- Велика кількість безкоштовних та платних ресурсів (текстури, моделі, звуки тощо) та плагінів (сторонні бібліотеки для розширення функціоналу рушія) на сервісі Unity Asset Store.
- Велика кількість навчальних матеріалів на різноманітних ресурсах а також докладна документація, що дозволяє швидко знайти необхідні матеріали для вирішення поставленої задачі [14][17][18][19][20].
- Велика спільнота розробників, що можуть допомогти у розв'язанні якоїсь проблеми, порадити кращі шляхи вирішення задачі [7].

Недоліками цього ігрового рушія є:

- Обмеженість візуального редактора при роботі з багатокомпонентними схемами.
- Необхідність самостійно налаштовувати роботу із зовнішніми бібліотеками.

Приклади ігор, розроблених на ігровому рушії Unity: Among Us, Escape from Tarkov, Superhot [11].

Рисунок 2.2 – Скріншот ігрового додатку Among Us

Рисунок 2.3 – Скріншот ігрового додатку Escape from Tarkov

Рисунок 2.4 – Скріншот ігрового додатку Superhot

2.2.2 Unreal Engine

Unreal Engine (зараз актуальна версія Unreal Engine 4, однак уже анонсована Unreal Engine 5) - це популярний і широко використовуваний ігровий движок, розроблений Epic Games.

Він використовується в багатьох сучасних іграх AAA, таких як бойовий королівський шутер Fortnite від розробників цього рушія або інших хітових іграх, таких як Psychonix's "Rocket League".

Рушій дозволяє розробляти різні платформи від ПК до консолей, таких як PS4, Xbox One та Nintendo Switch. Одна із причини, через яку він настільки широко використовується, це гнучкість роботи між цими різними платформами.

Більш досвідчені розробники можуть використовувати мову C++ для створення власних сценаріїв, які працюють в ігровому движку. Розробники-початківці можуть використовувати його дуже потужну систему візуальних сценаріїв Blueprint, які в основному є збірними блоками коду, які ви можете додати до своїх об'єктів для взаємодії.

Він також має потужні інструменти для створення матеріалів та анімації для художників, які дозволяють швидко робити складні сцени. Налаштування деяких із цих функцій спочатку може здатися страшним, але існує багато прикладів, в яких можна просто змінювати параметри, поки результат не задовольнить вас.

На додаток до всіх цих функцій є величезна кількість документації, яка допоможе вам вивчити систему.

Великою перевагою Unreal Engine є те, що він абсолютно безкоштовний для використання. Незалежно від того, чи ви розробник-любитель, чи студія AAA, за користування Unreal плата не стягується. Натомість розробники рушія стягують оплату в розмірі 5% від усіх ігрових доходів, які становлять понад 3000 доларів на квартал, щоб заробляти гроші і надалі розвивати цей продукт.

Unreal Engine існує з 1998 року. Вперше він був використаний для гри «Unreal», Unreal Engine 2 вийшов у 2002 році, Unreal Engine 3 вийшов у 2006 році, а поточна версія UE 4 вийшла в 2014 році.

Завдяки своєму тривалому перебуванню в ігровій індустрії, він має величезну кількість послідовників та безліч сторонніх підручників та онлайн-форумів.Завдяки всім цим ресурсам ви зможете навчитися найкращому способу

використання цього механізму та знайти рішення для будь-яких проблем, які можуть виникнути на шляху.

Для того, щоб залишатись конкурентоздатним на ринку, до нього постійно додаються нові функції, щоб він завжди був одним із найпередовіших засобів для ігрових розробок. Його поєднання художніх та орієнтованих на розробку інструментів забезпечує повну творчу гнучкість.

Він використовує широко прийнятий робочий процес PBR (Physically Based Rendering – фізично коректна візуалізація) для своїх матеріалів та візуалізації. Це в поєднанні з динамічним освітленням або «запіканням» тіні та світла дозволяє отримати дуже близьке до реальності зображення, яке працює у режимі реального часу.

Функція Blueprints, дозволяє складати прості сценарії, які взаємодіють між собою. Все це використовує візуальний інтерфейс, тому навіть якщо ви ніколи не кодували у своєму житті, кілька коротких посібників допоможуть досить легко створити робочу гру.

Коли «Unreal» (гра) була випущена компанією Epic в 1998 році, виникли проблеми з її багатокористувацькими системами, які на той час ще були досить передовими. Розробники витратили близько року, щоб це виправити, і це врешті-решт призвело до появи цілком нової гри «Unreal Tournament» у 1999 році. Цей багатокористувацький фреймворк є одним з основних моментів продажу сьогодні. У вересні 2017 року розробники внесли модифікації, які дозволяють до 100 різних гравців підключатись і грати разом в одній грі, тобто можливості справді дуже широкі.

Unreal Engine має хороший вбудований фізичний рушій, що дозволяє застосовувати фізику м’яких тіл, ефекти частинок та інші прості речі, такі як гравітація. Фреймворк простий у використанні для виготовлення будь-яких предметів, таких як гойдалки, які реагують на прикладену до них силу.

Гнучкою є також система текстур та матеріалів. Після додавання текстурних карт, які можна розробити в інших програмах, можна

використовувати сотні різних зразків матеріалів, щоб нескінченно налаштовувати кінцевий результат, щоб отримати те, що необхідно.

Також є вбудовані інструменти, які значно полегшують створення віртуальної або доповненої реальності. Зростання популярності VR в останні роки неможливо ігнорувати, тому завжди корисно мати доступ до механізму, який може легко допомогти створювати контент для цих платформ, включаючи Oculus Rift та HTC Vive.

Unreal Engine також має інструменти, які дозволяють легко створювати власну місцевість. Можна малювати за допомогою миші власні матеріали та змішувати їх, щоб, наприклад, трава плавно перетворювалась в стежку. Далі можна змішувати дерева, траву і фарби для створення суцільних лісів або квіткових галявин. Можливо створити цілі ландшафти за лічені хвилини, і вони вже оптимізовані для запуску у ігровому додатку, якщо використовуються рослинні сітки, що постачаються з Unreal Engine.

Браузер вмісту (рис. 2.5) є невід'ємною частиною інтерфейсу користувача і дозволяє швидко дослідити всю структуру файлів вашої сцени, щоб знайти різні ресурси та перетягнути їх прямо у область перегляду.

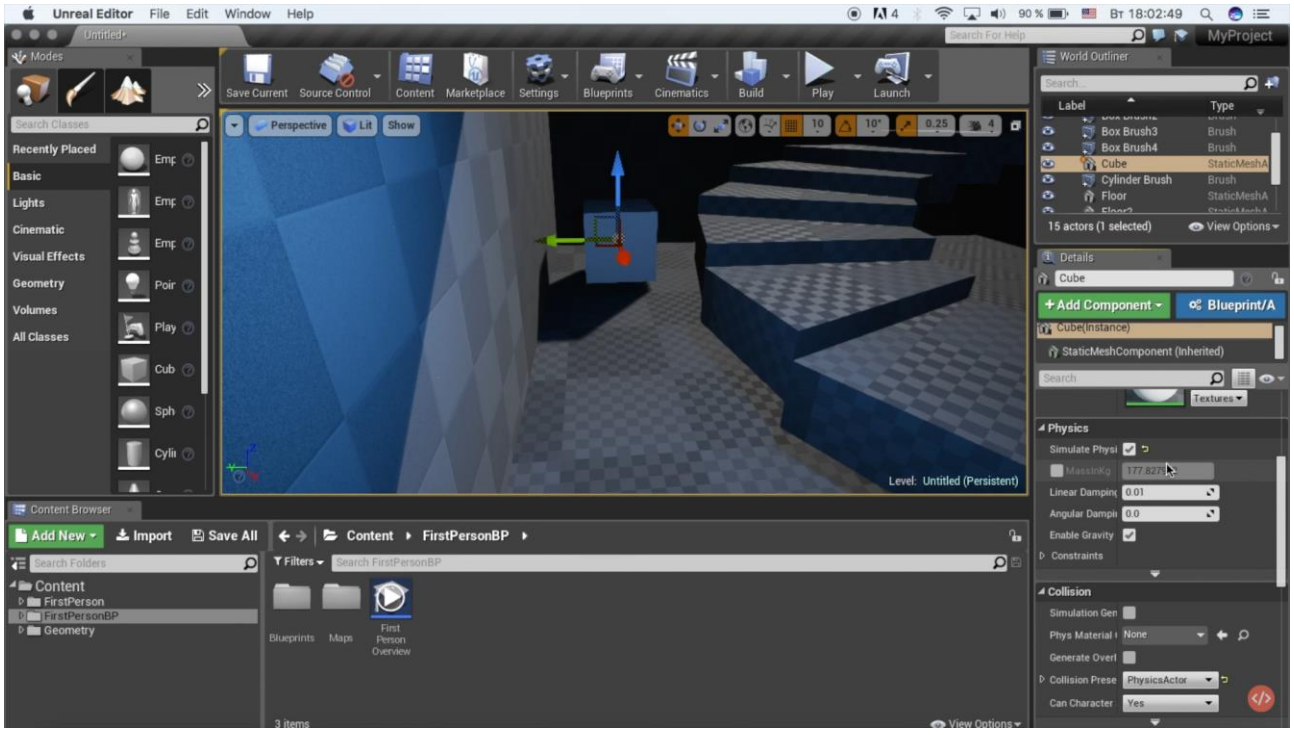


Рисунок 2.5 – Скріншот інтерфейсу редактора Unreal Engine 4

Останньою особливістю, яку слід зазначити, є ринок. Він містить велику кількість ресурсів Epic та інших розробників, які використовують Unreal Engine. З нього можна завантажити художні заготовки або повноцінні розробки проектів, які потім можна доробити та вдосконалити. Також можна використовувати їх для вивчення інструментів, які використовував розробник, щоб краще зрозуміти їх. Частина цього вмісту на ринку безкоштовна, але більшість із них має високу ціну. Можна продавати власні розробки на ринку, при цьому, Epic стягує 12% суми усіх продажів на свою користь [10].

Перевагами цього ігрового рушія є:

- Великий час існування на ринку, постійна підтримка і оновлення, завжди актуальні технології.
- Система Blueprint, яка дає змогу створити гру навіть без написання коду на мові C++.
- Сервіс Unreal Marketplace, на якому можна отримати доступ до великої кількості ассетів та плагінів, більшість з них є платними, однак список безкоштовних доволі широкий, також Epic Games щомісячно відкриває безкоштовний доступ до деяких платних ресурсів та плагінів.
- Велика кількість докладної документації та навчальних матеріалів, які допомагають зрозуміти функціонал рушія [15].
- Використання мови C++ дозволяє краще оптимізувати проект за рахунок можливостей цієї мови.
- Відкритий вихідний код, що дає можливість стороннім розробникам долучитись до покращення цього рушія, додавання нового корисного функціоналу.

Серед мінусів можна виділити:

- Високий поріг входження – функціонал Unreal Engine занадто обширний для розробника-початківця. Також, незважаючи на можливість створення гри без написання коду завдяки системі Blueprint, для більш-менш серйозних розробок необхідно вивчати мову C++, яка не є дуже легкою для вивчення.

- Обмежена кількість інструментів для розробки 2D проектів.

Приклади ігор, розроблених на ігровому рушії Unreal Engine: Dishonored, Borderlands 2, Bioshock 2 [12].

Рисунок 2.6 – Скріншот ігрового додатку Dishonored

Рисунок 2.7 – Скріншот ігрового додатку Borderlands 2

Рисунок 2.8 – Скріншот ігрового додатку Bioshock 2

2.2.3 CryEngine

CryEngine - це ігровий движок, створений німецькою приватною компанією Crytek в 2002 році, що спочатку використовувався в шутері від першої особи Far Cry. CryEngine - комерційний рушій, який пропонується для ліцензування іншим компаніям. З 30 березня 2006 року всі права на рушій належать Ubisoft [9].

Набір для розробки програмного забезпечення CryEngine (SDK), який спочатку називався Sandbox Editor, є поточною версією редактора рівнів, що використовується для створення рівнів на базі рушія CryEngine від Crytek. У програмі також передбачені інструменти для полегшення написання сценаріїв, анімацій та створення об’єктів. Він включений в різні ігри компанії Crytek (включаючи, але не обмежуючись іграми Crysis і Far Cry) і широко використовується для створення модифікацій. Стиль редагування заснований на концепції «пісочниці», з акцентом на великі місцевості та вільний стиль програмування рівнів. Редактор також може використовуватись для дизайну приміщень.

Рисунок 2.9 – Скріншот інтерфейсу редактора CryEngine 5

На відміну від поширених редакторів, таких як UnrealEd, які використовують "субтрактивний" стиль редагування, який полягає в забиранні ділянок із заповненого світового простору, пісочниця має "аддитивний" стиль. Об'єкти додаються до загального порожнього простору. Концентрація пісочниці на потенційно величезній (теоретично сотні квадратних кілометрів) місцевості означає, що вона використовує алгоритмічну форму малювання текстур та об'єктів на ландшафті. В ній використовуються різні параметри для визначення розподілу текстур або типів рослинності. Це зроблено для заощадження часу та можливості редагування великих ландшафтів, зберігаючи при цьому можливість вільного і безсистемного переміщення в пісочниці "реального світу". Це відрізняється від деяких стилів редагування, які часто використовують "фальшиві фони", щоб створити ілюзію великих місцевостей.

Порівняно з 3D-редактором Blender, який можна використовувати для ігрового дизайну, редактор Sandbox має можливість одним натисканням клавіші редактору перейти прямо до поточного дизайну («What You See Is What You Play» – що ви бачите, в те і граєте). Гру не потрібно завантажувати, оскільки ігровий движок вже працює в редакторі. «Ігрове» зображення відображається в 3D-частині редактора. Редактор також підтримує всі функції CryEngine, такі як

транспортні засоби та фізика, сценарії, вдосконалене освітлення (включаючи динамічне освітлення, рухомі тіні), технологію Polybump, шейдери, 3D аудіо, інверсну кінематику персонажів та змішування анімацій, динамічну музику, систему м'яких частинок у реальному часі, вбудований FX редактор, відкладене освітлення, карти нормалей а також вдосконалену модульну систему штучного інтелекту.

Переваги ігрового рушія CryEngine:

- Фотореалістична графіка. При розробці рушія велика увага приділялась саме графічним можливостям. Порівняно із іншими рушіями, графіка, яку забезпечує рушій, набагато реалістичніша.
- Підтримка передових технологій при розробці ігрового рушія.
- Система візуальних сценаріїв FlowGraph, яка дозволяє створювати ігри без написання коду.
- Інструмент Statoscope, що дає можливість візуально відстежити витрати комп’ютерних ресурсів при запуску проекту.

Серед недоліків движка варто виділити:

- Скромний вибір готових плагінів та ресурсів, порівняно із магазинами двох найпопулярніших рушіїв Unity та Unreal Engine.
- Складність компіляції готової версії продукту.
- Невелика спільнота розробників.
- Невелика кількість навчальних матеріалів [16].
- Відсутність інструментів для розробки 2D проектів.

Також варто відмітити, що інструмент FlowGraph програє по функціональності системі Blueprints рушія Unreal Engine і з його використанням майже неможливо створити складний рівень.

Приклади ігор, розроблених на ігровому рушії CryEngine: Sniper Ghost Warrior 3, Prey, Ryse: Son of Rome [13].

Рисунок 2.10 – Скріншот ігрового додатку Sniper Ghost Warrior 3

Рисунок 2.11 – Скріншот ігрового додатку Prey

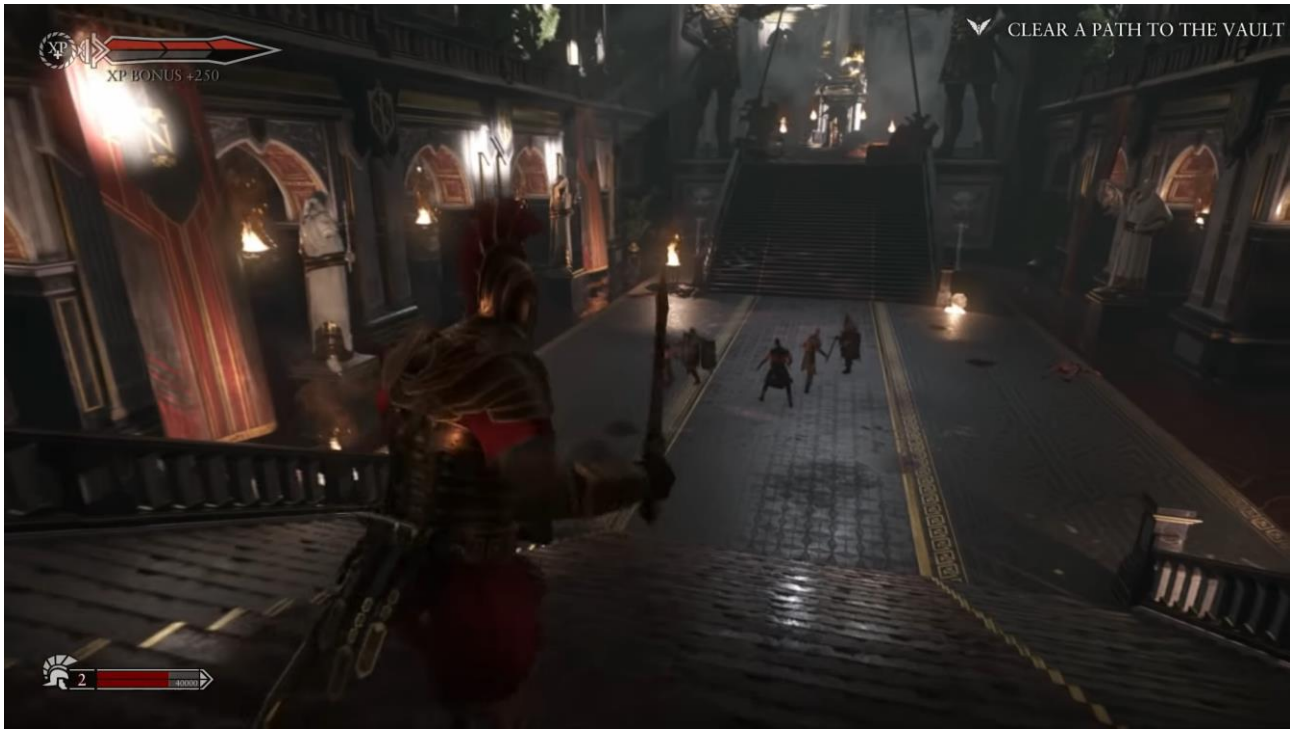


Рисунок 2.12 – Скріншот ігрового додатку Ryse: Son of Rome

Висновки до розділу 2

У цьому розділі було проведено аналіз популярних ігрових рушіїв, які поширюються безкоштовно (плата за їх використання стягується лише з розроблених з їх використанням проектів, що досягли певного фінансового успіху).

Ігрові рушії у цілому спрощують розробку гри, оскільки мають широкий набір інструментів для її розробки та багато реалізованого функціоналу. Тому їх використання необхідне для економії великої кількості часу і грошей при розробці гри. Створити власний ігровий движок з нуля можуть собі дозволити лише великі компанії, що мають значну кількість співробітників і можуть виділити на це значну кількість часу та бюджет. Готові рушії дають можливість навіть невеликим студіям або поодиноким розробникам, у яких обмежені людські та фінансові ресурси створити конкурентоспроможний продукт, що може зацікавити свою аудиторію та стати успішним та дохідним. Такі проєкти називаються інді-іграми та із появою загальнодоступних рушіїв поширюються все більше і більше.

Ігровий рушій CryEngine дає змогу реалізувати чудову графіку, тому якщо метою гри є вразити гравця фотореалістичними краєвидами, цей рушій є гарним вибором, однак потребує чималих зусиль для оптимізації великих проектів. Не підходить для використання у цьому дипломному проєкті через брак інструментів для розробки двовимірних ігор

Ігровий рушій Unreal Engine використовується при розробці багатьох сучасних високобюджетних проектів. Його система Blueprints також може допомогти початківцям при розробці власної гри. Він має чудовий потенціал при оптимізації великих проектів.

Ігровий рушій Unity є чудовим вибором для старту при розробці ігор. Він має велику кількість готових ресурсів, навчального матеріалу та документації. Також серед усіх проаналізованих ігрових рушіїв у нього найкращий інструментарій для розробки 2D гри.

РОЗДІЛ 3
РЕАЛІЗАЦІЯ ГРИ

3.1 Вибір технологій

Проаналізувавши особливості та вимоги до гри та порівнявши можливості ігрових рушіїв, описаних в 2 розділі цієї дипломної роботи, було зроблено висновок, що Unity є найкращим вибором для виконання поставленої задачі, оскільки він має найбільше інструментів для реалізації 2D проєктів. Також цей ігровий рушій має просту та зрозумілу структуру, що дозволяє скоротити час розробки. Всі ігрові сцени складаються з об'єктів з певним набором компонентів, які можна просто додавати або видаляти, присутня система префабів (заготовок), яка дозволяє швидко розмістити велику кількість об'єктів з однаковим або схожим набором компонентів та характеристик.

Ігровий рушій Unity підтримує скрипти, написані на C#, тому використовуватись буде саме ця мова. Однак, текстового редактора у візуальному середовищі розробки немає, тому необхідно обрати середовище для розробки коду. В Unity присутній власний компілятор коду на C#, тому використовувати можна навіть блокнот, однак за замовчуванням використовується Microsoft Visual Studio. Він і є найкращим вибором для розробки коду, оскільки після встановлення модуля «Game development with Unity», середовище розробки дозволяє відслідковувати помилки у коді, які призведуть до неможливості компіляції, автоматично доповнювати назви класів, функцій, змінних при вводі початкових літер, швидко знаходити документацію та опис функцій, що використовуються при розробці. Всі ці можливості дозволяють моментально відслідковувати помилки та не витрачати зайвий час на пошук власних друкарських помилок на кшталт пропущеної крапки з комою.

Цільовою платформою було вирішено обрати персональні комп'ютери, оскільки розробка відбувається на ПК і є можливість одразу ж проводити тести готового продукту. Однак, завантаживши необхідні модулі в Unity через

менеджер проєктів та загрузок UnityHub, з’являється можливість збілдити проєкт для будь-якої платформи (за замовчуванням є можливість збілдити гру лише для ПК).

3.2 Компоненти гри

Для реалізації гри необхідні дві сцени: одна для головного меню і одна для безпосередньо ігрового процесу. За замовчуванням при створенні сцени генеруються два об’єкти: камера (англ. Main Camera) і джерело світла (англ. Directional Light). Джерело світла в цьому проєкті не потрібне, оскільки в головному меню всі об’єкти – це написи і кнопки, які розміщуються на полотні (англ. Canvas), а сама гра виконана в 2D. Камера ж потрібна для виведення зображення на екран.

3.2.1 Префаби

Префаби (англ. Prefabs) – шаблони об’єктів з певними наборами компонентів. Вважається гарною практикою взагалі будь-які об’єкти робити у вигляді префабів, тому що так можна змінити певну характеристику в усіх розташованих об’єктів на основі певного шаблону, при цьому їх індивідуальні ознаки не зміняться. В розроблюваному проєкті всі об’єкти було створено у вигляді заготовок, за виключенням об’єктів, що розташовуються на полотні, таких як панелі, написи, кнопки.

3.2.1.1 MainCamera

MainCamera – шаблон для камери. Використовується в головному меню та в ігровій сцені.

Ігрова камера має чотири компоненти: Transform, Camera, Physics 2D Raycaster і скрипт CameraMovements. У камери в головному меню є тільки Transform і Camera.

Компонент Transform (перетворення) присутній у будь-якого об’єкта і містить в собі інформацію про розташування у просторі, поворот об’єкта і його масштабування.

Компонент Camera (камера) містить інформацію про фон, дистанцію прорисовки, згладжування та інші налаштування. Також є можливість вибору проекції камери (перспектива або ортографічна). Перспектива використовується для 3D ігор, де необхідно передати на плоский екран тривимірне зображення, у якого відстань від різних об’єктів до камери різна. Для 2D гри було обрано ортографічну камеру, остільки “глибини” у зображення нема. Фоновим кольором було обрано аквамариновий для асоціації з морем (рис. 3.1).

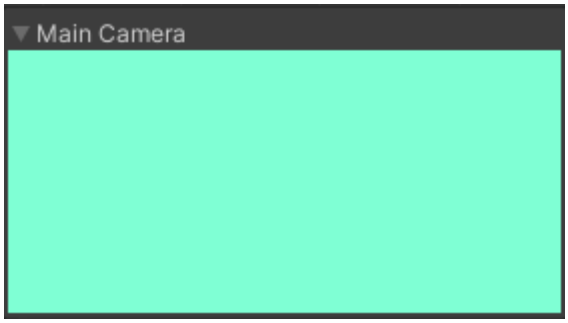


Рисунок 3.1 – Фоновий колір ігрової камери

Physics 2D Raycaster – використовується для знаходження двовимірних об’єктів, які мають колайдери (фізичні форми) для подальшої їх обробки подіями. У розроблюваній грі цей компонент потрібний для того, щоб запускати різні події після захоплення фігури мишею, її пересування тощо.

3.2.1.2 GameManager

GameManager – шаблон об’єкта, що використовується при запуску ігрової партії. Окрім компоненту Transform містить однойменний сценарій GameManager.

3.2.1.3 Board

Board – префаб, який містить компоненти Transform і скрипт Board, у якому реалізовано основну логіку гри.

3.2.1.4 Cell

Cell – шаблон клітинки ігрового поля. Складається з компонентів Transform, Sprite Renderer і скрипт Cell, у якому встановлюються параметри об’єкта.

Компонент Sprite Renderer використовується для візуального відображення об’єктів на екрані. Містить поля для вибору спрайту (графічного зображення), його кольору (якщо присутні прозорі частини), перевороту відносно осей X та Y, режиму відображення (простий, з кількох шарів або тайловий – багато дрібних частинок, що утворюють велике зображення). Також є можливість вибрати матеріал і позицію в черзі відображення (якщо необхідно зобразити один об’єкт поверх іншого).

3.2.1.5 Figure

Figure – шаблон фігури (кораблика), які є основою геймплею. Як і префаб Cell містить компоненти Transform, Sprite Renderer і додатково Box Collider 2D та скрипт Figure.

Компонент Box Collider 2D потрібен для можливості взаємодії гравця з об’єктами та об’єктів між собою. В ньому можна змінити матеріал, який буде впливати на фізику об’єкта, виставити прапор тригера, який буде активовано, якщо колайдер об’єкта зіткнеться з іншим колайдером, розміри колайдера, його зміщення відносно об’єкта. В розроблюваному проєкті використовується для того, щоб фігури можна було захоплювати та переміщати мишею по ігровому полю.

3.2.2 Спрайти

Спрайти – це двовимірні графічні ігрові об’єкти. Оскільки ігрове поле та його елементи генеруються після початку гри, всі спрайти необхідно помістити у папку Resources. Ця особливість Unity дозволяє завантажити їх з коду не вказуючи шлях напряму і не задаючи строгих обмежень на розміщення папки з грою.

Всі спрайти було створено за допомогою графічного редактора Photoshop, розроблений компанією Adobe Systems. Програма є платною, однак існує безкоштовна семиденна пробна версія.

Спрайти мають розмір 200*200 пікселів. В редакторі Unity для кожного спрайта у налаштуваннях необхідно змінити тип з Default на Sprite (2D and UI), щоб зображення можна було використовувати як повноцінний спрайт і прив’язувати до об’єкту у компоненті Sprite Renderer.

3.2.2.1 Спрайти клітинок ігрового поля

Всі клітинки ігрового поля залежно від їх типу забарвлені у свій колір. Клітинки, що позначають воду – блакитні, порти – темно-сині, рифи – бордові, землю – коричневі, клітинки, на які можна перемістити захоплений мишею корабель – сині, а клітинки, на яких знаходяться кораблі, які можна атакувати – червоні. Також біля портів знаходяться клітинки з прапором, який показує приналежність порту тій чи іншій стороні а також королівські порти (рис. 3.2).

Рисунок 3.2 – Спрайти клітинок ігрового поля

3.2.2.2 Спрайти кораблів

У грі існує 12 типів кораблів: бриг, галеон, галера, трипалубний лінкор, двопалубний лінкор, монітор, парусний фрегат, парова броненосна батарея, пароплав, паровий корвет, паровий фрегат, тендер. У кожної сторони кораблі однакові, відрізняються лише кольором прапора. Також спрайти зелених кораблів є дзеркальними до жовтих відносно осі Y і спрямовані у напрямку королівського порту суперника (рис. 3.3).

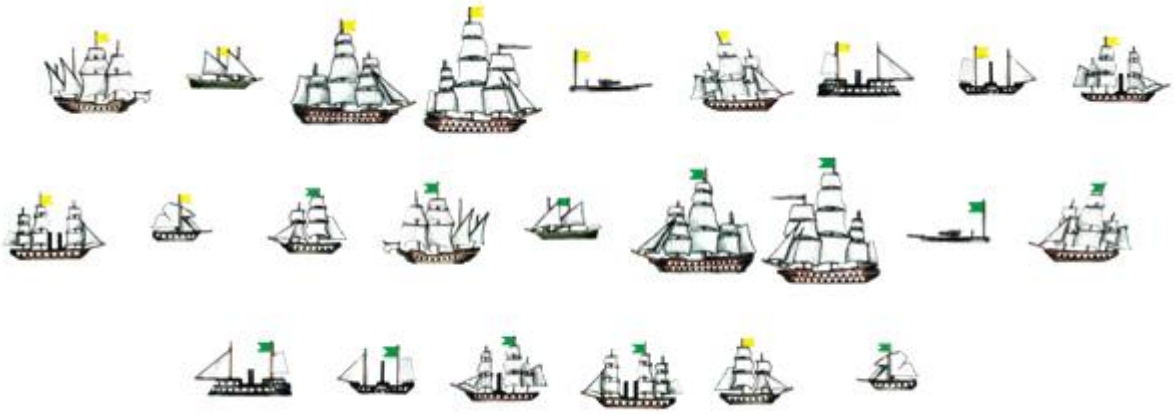


Рисунок 3.3 – Спрайти кораблів

3.2.3 Сцена головного меню

Сцена головного меню – це сцена, яка завантажується при запуску додатку. У неї є такі об’єкти, як:

- Main Camera – розміщений префаб Main Camera. У нього видалено компоненти Physics 2D Raycaster та скрипт CameraMovements. Початкова позиція незмінна, однак вона не має значення, оскільки всі елементи головного меню розташовані на полотні, яке рендериться не в режимі камери. Однак, камера є обов’язковим елементом сцени.
- Canvas – полотно, яке є батьківським об’єктом для будь-яких кнопок та написів у сцені. Має такі компоненти, як Rect Transform (заміна компонента Transform), Canvas, в якому можна налаштувати режим рендеру, порядок сортування елементів та цільовий дисплей, Canvas Scaler – компонент, в якому можна налаштувати режим масштабування (у розроблюваному проєкті було обрано масштабування за розміром вікна додатку), режим підлаштування під екран (після кількох тестів було обрано підлаштування під висоту екрану), а також Graphic Raycaster – аналог Physich 2D Raycaster для дочірніх об’єктів полотна.
- EventSystem – об’єкт для обробки подій. Створюється автоматично при додаванні в сцену полотна, або будь-якого об’єкта, що розташовується на полотні (в такому випадку полотно також буде автоматично додано). Має стандартний компонент Transform, Event System – власне компонент для

обробки подій і Standalone Input Module – компонент для налаштування введення інформації.

- Panel – дочірній об’єкт полотна. Замість компонента Transform має компонент Rect Transform, який визначає розташування панелі на полотні, Canvas Renderer та Image, який містить фонове зображення. Власне зображення встановлено не було, а був обраний фоновим кольором насичений аквамариновий. Має три дочірніх об’єкта: дві кнопки (Start та Exit) і текст.
- Text – текст з назвою гри. Має компоненти Rect Transform, Canvas Renderer і Text – компонент, в якому налаштовується власне текст, шрифт, розмір та тип шрифту, вирівнювання та інші характеристики. Якщо потрібно обрати шрифт, відмінний від стандартного, необхідно скачати його та помістити в папку з асетами.
- Start – кнопка для початку гри. Має компоненти Rect Transform, Image, Shadow, що створює ефект тіні біля об’єкта, сценарій StartGame, що запускає гру і Button – компонент, в якому можна налаштувати ефекти при натисненні кнопки для візуального відображення натискання а також скрипт, який необхідно виконати при натисканні кнопки. Також має дочірній об’єкт Text, аналогічний тексту з назвою гри, але його прив’язано не до полотна, а до кнопки.
- Exit – кнопка для виходу із гри. Має аналогічну структуру і склад компонентів до кнопки Start, відрізняється лише розташуванням, прив’язаним скриптом (у цієї кнопки він має назву ExitGame) та текстом, розташованим на ній.

3.2.4 Геймплейна сцена

Геймплейна сцена – сцена, в якій реалізовано безпосередньо ігровий процес. Містить наступні об’єкти:

- Main Camera – префаб Main Camera, розміщений без змін. Початкове положення у компоненті Transform відповідає центру ігрового поля, розмір ортографічної камери такий, що видно усе поле.
- EventSystem – об’єкт з аналогічними компонентами і параметрами до того, який міститься у сцені головного меню.
- Canvas – об’єкт, який має майже такий самий набір параметрів і компонентів, як і полотно головного меню. Але, на відміну від об’єкта Canvas сцени MainMenu, підлаштовується не під висоту, а під ширину екрану, щоб панелі та кнопки на ньому завжди знаходились по краям екрану. Також має дві дочірні кнопки (Restart та Surrender) і дві панелі (ShipInfo та MessagePanel)
- Board – розташований у сцені префаб Board. Змін немає.
- GameManager – незмінений шаблон GameManager розміщений у сцені.
- Restart – дочірня кнопка полотна, з набором компонентів, аналогічним до кнопок головного меню, але зі своїм сценарієм з назвою Button. Відповідає за передачу ходу супернику, якщо гравець вже зробив свій хід. Компонент Image має фоновий колір та стандартне зображення кнопки в Unity UISprite. Також має дочірній об’єкт Text, однак у нього замість компонента Text присутній компонент Image зі спрайтом передачі ходу.
- Surrender – такий же дочірній об’єкт полотна, як і Restart. При натисканні на неї активний гравець здається та отримує поразку. Скрипт, прив’язаний до неї має назву SurrenderButton. Має такий же фоновий колір і зображення, як і об’єкт Restart, а у дочірнього об’єкта Text фоновим зображенням є білий прапорець.
- ShipInfo – дочірня панель полотна. Спочатку неактивна. Активується при натисненні лівою кнопкою миші на корабель. Містить інформацію про фігуру, таку як: назва корабля, кількість здоров’я, кількість пострілів, дистанцію атаки. Має 5 дочірніх об’єктів: один із зображенням (Image) і 4 з текстовими полями (Text). Image – при активації у ньому відображається спрайт корабля, по якому відбувся клік лівою кнопкою

миші. Об’єкт Text містить тип корабля та його назву, Text (1) – інформацію про кількість пострілів, яка необхідна для затоплення корабля, Text (2) – кількість пострілів, що залишилась на цьому ходу та максимальну кількість пострілів, Text (3) – максимальну відстань, на яку корабель може здійснити постріл.

- MessagePanel – дочірня панель полотна. Займає весь екран та містить напівпрозоре сіре тло для затемнення екрану при виводі повідомлень. Має дочірній об’єкт Panel.
- Panel – панель із аквамариновим фоном, на якій відображається інформація щодо результатів пострілів, затоплених кораблів, перемог та вітру на початку кожного ходу. Є батьківським об’єктом для тексту Message та кнопки MessageButton. Message містить власне текст повідомлення, а MessageButton – сценарій MessagePanel, який робить панель MessagePanel та всі дочірні об’єкти неактивними при натисканні на кнопку.

3.2.5 Сценарії

В скриптах міститься код, що відповідає за всі події та ігровий процес додатку. Кожен сценарій містить одноіменний клас. Всі сценарії є компонентами тих чи інших об’єктів, за роботу з якими вони відповідають. Також слід зазначити, що всі створювані розробником гри на Unity класи наслідуються від класу MonoBehaviour.

3.2.5.1 Board

Board – клас, в якому реалізується основна логіка гри. Через редактор Unity в нього передаються об’єкти MessagePanel та ShipInfo, які будуть активуватись за певних умов, а також префаби Cell та Figure (рис. 3.4).

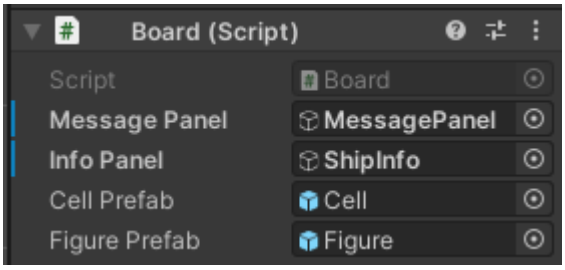


Рисунок 3.4 – поле вибору об’єктів, які передаються в сценарій

Клас Board має наступні змінні:

- Cell [,] allCells – матриця, що містить всі комірки ігрового поля.
- Figure [,] allFigures – матриця з інформацією про всі фігури, які є на ігровому полі.
- bool yellowTurn – змінна, яка відповідає за те, чия черга ходити.
- int howLongYKingPortIsBusy – кількість ходів поспіль, яку кораблі зелених простояли в королівському порту жовтих.
- int howLongGKingPortIsBusy – кількість ходів поспіль, яку кораблі жовтих простояли в королівському порту зелених.
- Figure activeShip – активний корабель.
- int windDirection – напрям вітру від 0 до 7.
- int windPower – сила вітру від 0 до 2.
- List<int[]> possibleMoves – список клітинок, в які активний корабель може здійснити хід.
- List<int[]> possibleShots – список клітинок, що активний корабель може обстріляти.
- Dictionary<string, string> ukrainianNames – словник співвідношень анлійських та українських типів кораблів. Англійські назви використовуються у коді, українські – при виводі повідомлень гравцям.
- Dictionary<string, int[]> ships – словник з характеристиками кожного типу корабля, такими, як дальності ходу під час штилю, свіжого вітру, шторму, кількість пострілів за хід, максимальна дистанція пострілу, кількість пострілів, яку корабель може втримати.
- int landCells – список координат клітинок суші.

- int portCells – список координат клітинок портів.
- int reefCells – список координат клітинок рифів.
- int yFortCells – список координат жовтих фортів.
- int gFortCells – список координат зелених фортів.
- List<String> yShips – список всіх кораблів жовтих.
- List<String> gShips – список всіх кораблів зелених.
- List<String> namesForShips – список всіх назв для кораблів.

Методи, реалізовані в цьому класі:

- bool hasElement(int[,] array, int x, int y) – метод, що перевіряє наявність координат клітинки (x, y) у списку (array) при розміщенні клітинок у просторі.
- void Create() – метод, який на початку гри створює ігрове поле, заповнюючи його клітинками (відповідно до списків їх координат) а також розміщує кораблі в портах свого кольору. При цьому кожному кораблю випадковим чином присвоюється унікальне ім’я зі списку назв кораблів і розміщуються вони по портах також випадково.
- void CalcAnyDirection(int newMoveX, int newMoveY, int newDistance, Vector2Int newPosition) – метод для виявлення клітинок, у які активний корабель може здійснити хід. Змінні newMoveX, newMoveY приймають значення від -1 до 1 і визначають напрям, у якому потрібно розраховувати хід.
- int[] CalcStorm(int newMoveX, int newMoveY, int newDistance, Vector2Int newPosition) – метод для виявлення позиції, в яку корабель потрапить внаслідок шторму. На відміну від методу CalcAnyDirection, враховує клітинки рифів, як такі, в які корабель може потрапити і за наявності їх на шляху повертає саме їх координати.
- int[] CalcWindMoves(int newMaxMoveDistance) – метод для визначення максимальної довжини ходу активнного корабля при свіжому вітрі у кожному напрямі.

- void CalculateMoves(int newStormDistance, int newMaxMoveDistance, int newMaxCalmDistance, Vector2Int newPosition, Figure newFigure) – метод, що обчислює та підсвічує всі можливі клітинки, в які активний корабель може походити за штилю та свіжого вітру. Якщо ж дме вітер штормової сили, примусово переміщає усі кораблі активного гравця відповідно до їх довжини ходу та напряму вітру і знищує всі кораблі, що потрапили на рифи.
- void EndMove() – завершує хід активного корабля та міняє колір підсвічених клітинок на стандартний.
- void NextTurn() – метод, який передає хід супернику. Підраховує кількість кораблів кожної команди для перевірки умови перемоги за кількісною перевагою над суперником. Також за наявності корабля суперника у королівському порту збільшує лічильник на одиницю і перевіряє умову перемоги шляхом захоплення королівського порту. “Перезаряджає” гармати кораблів активного гравця, випадковим чином визначає напрям та силу вітру під час нового ходу і виводить відповідні повідомлення.
- IEnumerator WaitForClosingMessage() – кожен фрейм перевіряє, чи активне вікно з повідомленням.
- IEnumerator Storm(int id) – примусово “розряджає” гармати кораблів активного гравця і викликає метод для перевірки поля, в яке вітер змістить корабель. Рекурсивно викликається для кожного корабля гравця, який має ходити.
- void CalcAnyDirectionShots(int newShotX, int newShotY, int newDistance, Vector2Int newPosition) – метод для виявлення клітинок, у яких знаходиться корабель суперника, який активний корабель може обстріляти. Змінні newMoveY, newMoveX приймають значення від -1 до 1, використовуються для позначення напряму у якому розраховується постріл.

- void CalculateShots(int newMaxShotDistance, Vector2Int newPosition, Figure newFigure) – метод, що знаходить і підсвічує всі клітинки, які знаходяться в радіусі атаки активного корабля і містять кораблі суперника.
- void EndShoting() – метод, який робить активний корабель неактивним та очищає список клітинок, які можуть бути атаковані і заміняє їх спрайти на стандартні.
- void Shoot(Figure newFigure) – випадковим чином перевіряє, чи влучив корабель по супернику і виводить відповідне повідомлення.

3.2.5.2 Button

Button – клас, який відповідає за події при натисканні кнопки Restart. Отримує через редактор Unity об’єкт MessagePanel, який буде активувати, а також містить змінну isYellowTurn. У нього присутній метод void Restart(), який перевіряє, чий хід було завершено, передає його іншому гравцю і активує відповідне повідомлення.

3.2.5.3 CameraMovements

CameraMovements – сценарій, який використовується для того, щоб можна було пересувати камеру по ігровому полю та наближати певні його ділянки. Містить у собі об’єкт класу Camera. У скрипті є лише один метод – void Update(), який є стандартним методом класу MonoBehaviour. Метод Update викликається кожен фрейм (ігровий кадр) та перевіряє кілька умов:

- Якщо натиснена клавіша “q” – поступово зменшує параметр камери ortographicSize, який відповідає за розмір камери. Таким чином, зображення збільшується ніби ігрове поле наближається до неї, оскільки вона починає “бачити” меншу площу і заповнювати нею весь екран.
- Якщо натиснена клавіша “e” – поступово збільшує параметр камери ortographicSize. Таким чином, ігрове поле можна побачити повністю, але у зменшеному вигляді. Наближення/віддалення зображення відбувається до певних меж.

- Якщо натиснена клавіша “w” – поступово збільшує параметр position.y компонента Transform. Параметр відповідає за Y-координату розташування об’єкта (в цьому випадку – ігрової камери) в просторі. Компонент Transform автоматично береться від об’єкта, до якого прив’язується скрипт. Так відбувається переміщення камери до верхньої частини ігрового поля.
- Якщо натиснена клавіша “s” – поступово зменшує параметр position.y компонента Transform і камера пересувається вниз по ігровому полю.
- Якщо натиснена клавіша “a” – поступово зменшує параметр position.x компонента Transform. Параметр відповідає за X-координату просторового розташування об’єкта. Камера переміщається ліворуч.
- Якщо натиснена клавіша “d” – поступово збільшує параметр position.x компонента Transform. Камера переміщається праворуч. Всі переміщення обмежені, тому камера постійно відображає хоча б частину ігрового поля.

3.2.5.4 Cell

Cell – клас, що містить інформацію про комірку. Має наступні змінні:

- SpriteRenderer spriteRenderer – компонент Sprite Renderer об’єкта Cell, до якого прив’язаний скрипт.
- Transform transform – компонент Transform цього ж об’єкта.
- Board board – ігрове поле, на якому розміщуються клітинки.
- Vector2Int position – позиція клітинки на ігровому полі.
- string cellType – містить інформацію про тип комірки.

Також у класу є метод void Setup(Board newBoard, Vector2Int newPosition, string newCellType), що встановлює початкові налаштування клінинки, такі як тип, спрайт, позиція.

3.2.5.5 ExitGame

ExitGame – клас, який містить події, виконувані при натисканні кнопки Exit. Має метод void Exit(), який закриває додаток.

3.2.5.6 Figure

Figure – клас, який містить інформацію про об’єкт Figure, до якого прив’язаний сценарій з цим класом, а також методи для взаємодії з кораблями. Через редактор Unity в нього передається об’єкт Figure. Окрім класу MonoBehaviour наслідується також від класів IDragHander, що відповідає за подію перетягування мишею, IEndDragHandelr, що відповідає за подію закінчення перетягування, IBeginDragHandler, який відповідає за подію початку перетягування, а також IPointerClickHandler, який відповідає за подію натискання лівою клавішею мишки на об’єкті. Має змінні:

- GameObject messagePanel – панель, на якій виводяться повідомлення.
- GameObject infoPanel – панель, на якій виводиться інформація про виділений корабель.
- SpriteRenderer spriteRenderer – компонент Sprite Renderer об’єкта Figure, до якого прив’язаний скрипт.
- Transform transform – компонент Transform вищевказаного об’єкта.
- Board board – ігрове поле.
- Vector2Int position – позиція корабля на ігровому полі.
- int maxMoveDistance – максимальна дальність ходу при попутному свіжому вітрі.
- int maxCalmDistance – максимальна дальність ходу при відсутньому вітрі (в штиль).
- int stormDistance – відстань, на яку корабель зносить під час шторму.
- bool canMove – інформація про те, чи може пересуватись корабель.
- bool isYellow – інформація про колір корабля.
- int maxShots – максимальна кількість пострілів за хід.
- int remainShots – кількість пострілів, що залишилась на цьому ходу.
- int health – кількість пострілів, необхідна для затоплення.
- int maxShotDistance – максимальна дальність пострілу.
- string nameShip – назва корабля.

- bool occupiesYKingPort – показує, чи знаходиться корабель на клітинці жовтого королівського порту.
- bool occupiesGKingPort – показує, чи знаходиться корабель на клітинці зеленого королівського порту.

Методи цього класу:

- void Setup(Board newBoard, Vector2Int newPosition, int newMaxMoveDistance, int newMaxCalmDistance, int newStormDistance, int newMaxShots, int newMaxShotDistance, int newHealth, string newName, GameObject newMessagePanel, GameObject newInfoPanel) – початкові налаштування корабля.
- void OnBeginDrag(PointerEventData eventData) – стандартний метод класу IBeginDragHandler. Під час початку перетягування прораховує можливі ходи корабля.
- void OnEndDrag(PointerEventData eventData) – стандартний метод класу IEndDragHandler. Після завершення перетягування перевіряє, чи нова позиція корабля є такою, в яку він міг походити, якщо ні – повертає корабель у вихідне положення, якщо так – не дозволяє більше переміщати корабель, міняючи змінну canMove, а також перевіряє, чи нова позиція є клітинкою королівського порту і міняє відповідну змінну, якщо так.
- void OnDrag(PointerEventData eventData) – стандартний метод класу IDragHandler. Під час перетягування міняє позицію корабля згідно позиції курсора.
- void OnPointerClick(PointerEventData eventData) – стандартний метод класу IPointerClickHandler. Під час натискання на корабель виводить відповідну панель з інформацією про нього. Якщо цей корабель знаходиться на клітинці під атакою, прораховує постріл по ньому, якщо у корабля в цьому ході ще залишились постріли – обраховує можливі клітинки для атаки.
- void StormCall() – забирає можливість корабля стріляти і пересуватись під час поточного ходу і викликає метод для обрахування ходу в шторм.

- void Destroy(string reason) – виводить панель з повідомленням про затоплення корабля та причину цього. Видаляє корабель з ігрового поля.
- void Reload() – “перезаряджає” озброєння корабля, встановлюючи значення змінної remainShots рівним maxShots.
- void Discharge() – “розряджає” озброєння корабля, встановлюючи значення змінної remainShots рівним 0.
- void Shot() – зменшує кількість пострілів, що залишилась на одиницю.
- void GotShot() – зменшує кількість здоров’я на 1, якщо кількість здоров’я стала рівною 0, викликає метод знищення корабля.

3.2.5.7 GameManager

GameManager – клас, що відповідає за початок гри. З редактора Unity отримує об’єкт Board. Містить вбудований метод void Start() класу MonoBehaviour, який запускається при завантаженні сцени та запускає генерацію ігрового поля.

3.2.5.8 InfoPanel

InfoPanel – клас, який відповідає за панель виводу інформації про корабель. У редакторі Unity йому передається панель ShipInfo та її дочірні об’єкти: Image, Text(2), Text, Text(1), Text(3), на яких і буде відображатись інформація. Має два методи:

- void ShowMessage(string name, string spriteName, int health, int maxHealth, int shots, int maxShots, int maxShotDistance) – виводить інформацію про корабель, на якому відбувся клік мишею.
- void OnPointerClick(PointerEventData eventData) – стандартний метод класу IPointerClicker, від якого наслідується клас InfoPanel, деактивує панель інформації при натисканні на неї.

3.2.5.9 MessageButton

MessageButton – клас, що відповідає за подію при натисненні кнопки на панелі повідомлень. Отримує об’єкт MessagePanel з редактора Unity. Містить метод void ClosePanel(), який деактивує панель повідомлень та залежно від типу повідомлення, може викликати метод передачі ходу або завершення гри.

3.2.5.10 MessagePanel

MessagePanel – клас, що відповідає за показ повідомлення на панелі повідомлень. В редакторі Unity йому передається об’єкт текстового поля Message. Також має змінну string nextAction, яка впливає на метод ClosePanel класу MessageButton. У класу є метод void ShowMessage(string newMessage, int messageID), який виводить текст в текстове поле і встановлює значення змінної nextAction.

3.2.5.11 StartGame

StartGame – клас, що відповідає за подію при натисканні кнопки Start у головному меню. Містить метод void StartNewGame(), який завантажує сцену з безпосередньо грою.

3.2.5.12 SurrenderButton

SurrenderButton – клас, який відповідає за подію кнопки Surrender. В редакторі Unity передається об’єкт MessagePanel для виводу інформації про здачу супернику. Має змінну bool isYellowTurn для визначення активного гравця і метод void Surrender(), який виводить повідомлення про гравця, який здався і завершує гру.

Висновки до розділу 3

У цьому розділі для реалізації проєкту було обрано такі засоби: ігровий рушій Unity, середовище розробки Visual Studio, в якості платформи для реалізації – персональні комп'ютери. Причинами цього стали особливості цих засобів, які роблять їх зручнними у використанні та дозволяють зменшити час, витрачений на розробку гри.

Було розроблено головне меню з написами та кнопками з використанням різних компонентів. Розроблено скрипти, які містять події, виконувані при натисканні цих кнопок. Розроблено сцену з геймплеєм гри. Основою ігрового процесу є ігрове поле, яке складається з комірок різних типів, а також фігури кораблів, якими гравці ходять, воюють, захоплюють королівський форт суперника. Існує існує чотири основних види комірок, які відрізняються кольором, призначенням, можливістю ставити на них фігури. Присутні 12 видів кораблів, які відрізняються спрайтами, дальністю ходу, дистанцією атаки, кількістю здоров'я та можливих пострілів за один хід.

За допомогою графічного редактора було створено спрайти для комірок різних типів. Також було створено додаткові спрайти для позначення комірок можливих ходів та кораблів під загрозою атаки. Спрайти кораблів було взято з книги, в якій описуються правила оригінальної настільної гри та оброблено у тому ж графічному редакторі для відповідності розмірів фігур розмірам клітинок ігрового поля.

Розроблено сценарії, в яких описується взаємодія елементів ігрового додатку. Створено методи для запуску гри, виходу з гри, генерації ігрового поля, розміщення кораблів, обчислення можливих ходів, вираховування кораблів під атакою, знищення кораблів, виведення повідомлень та інформації, перевірки умов для перемоги, виходу до головного меню після завершення гри, переміщення та масштабування ігрової камери, реалізації механіки штормів, проведення атаки на корабель суперника, передачі ходу та інших ігрових можливостей.

РОЗДІЛ 4
ТЕСТУВАННЯ ГРИ

4.1 Головне меню

При старті гри запускається головне меню гри. У головному меню є кнопки для початку гри та для виходу з гри (рис. 4.1).

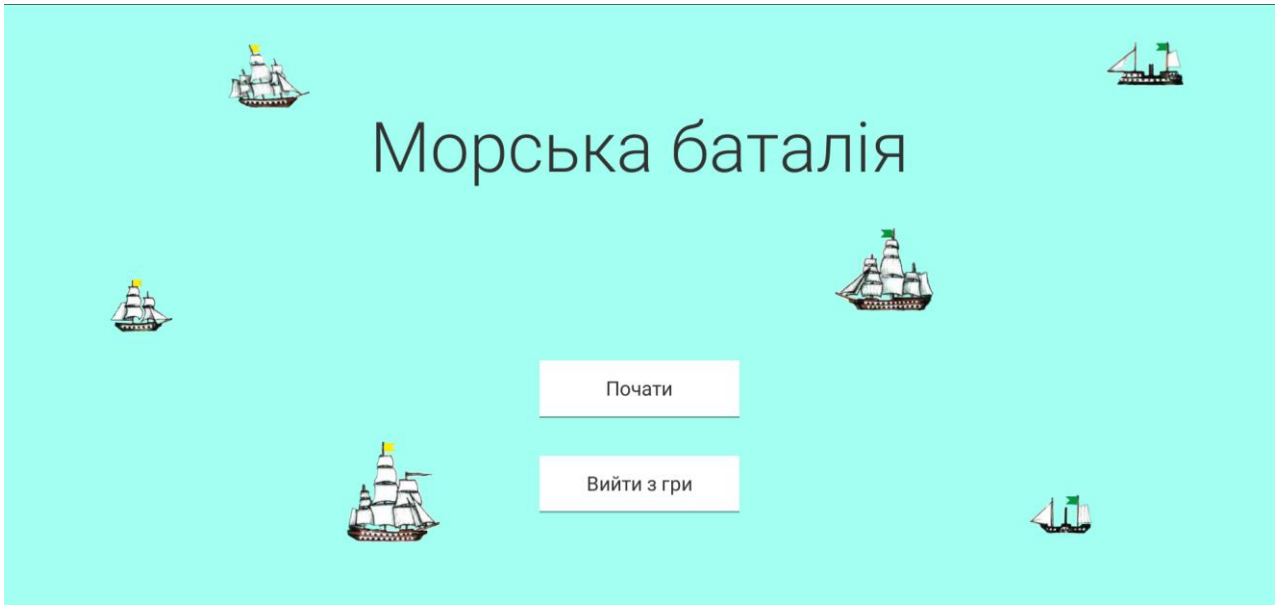


Рисунок 4.1 – Головне меню гри

4.2 Ігровий процес

При натисненні кнопки “Почати” завантажується гейплейна сцена. Переміщення камери по ігровому полю відбувається за допомогою клавіш W, A, S, D, а масштабування клавішами Q, E (рис. 4.3). Всі інші ігрові взаємодії відбуваються за допомогою лівої кнопки миші.

Після початку одразу з’являється повідомлення про те, що ходять жовті, потім – повідомлення про вітер (рис. 4.2).

Рисунок 4.2 – Активна панель повідомлення

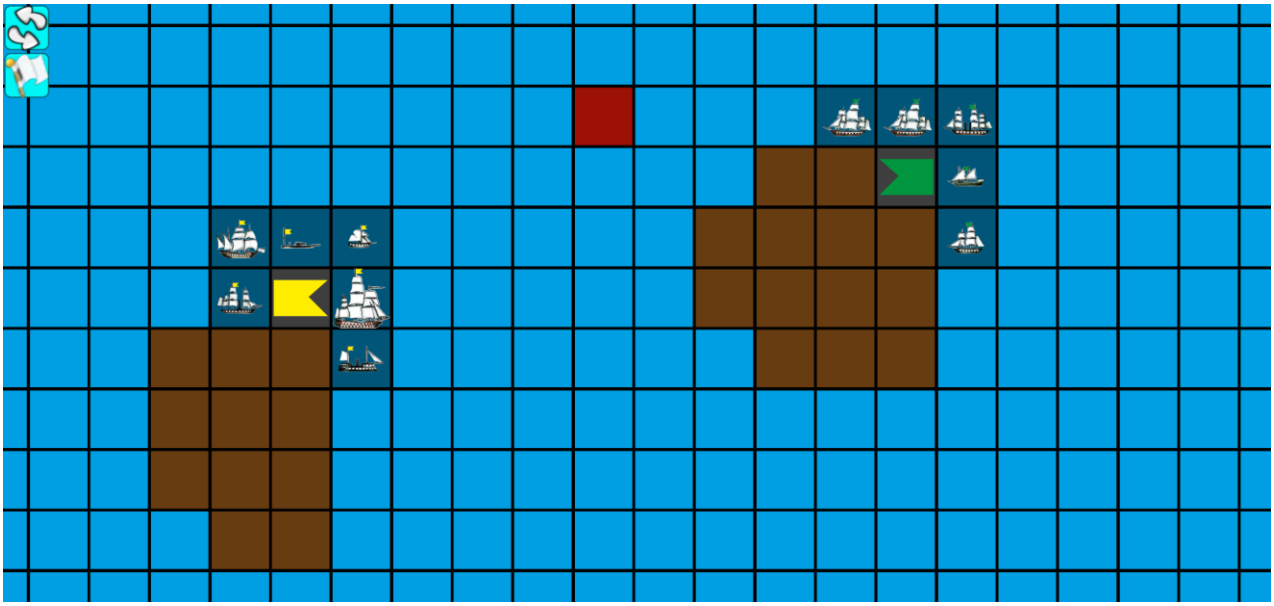


Рисунок 4.3 – Максимально наближена ділянка ігрового поля

При кліку на кораблі показується інформація про його тип, назву, здоров’я, постріли та дистанцію атаки (рис. 4.4). Назви кораблів унікальні, тому їх неможливо переплутати, хоча для кожної гри вони розподіляються випадково.

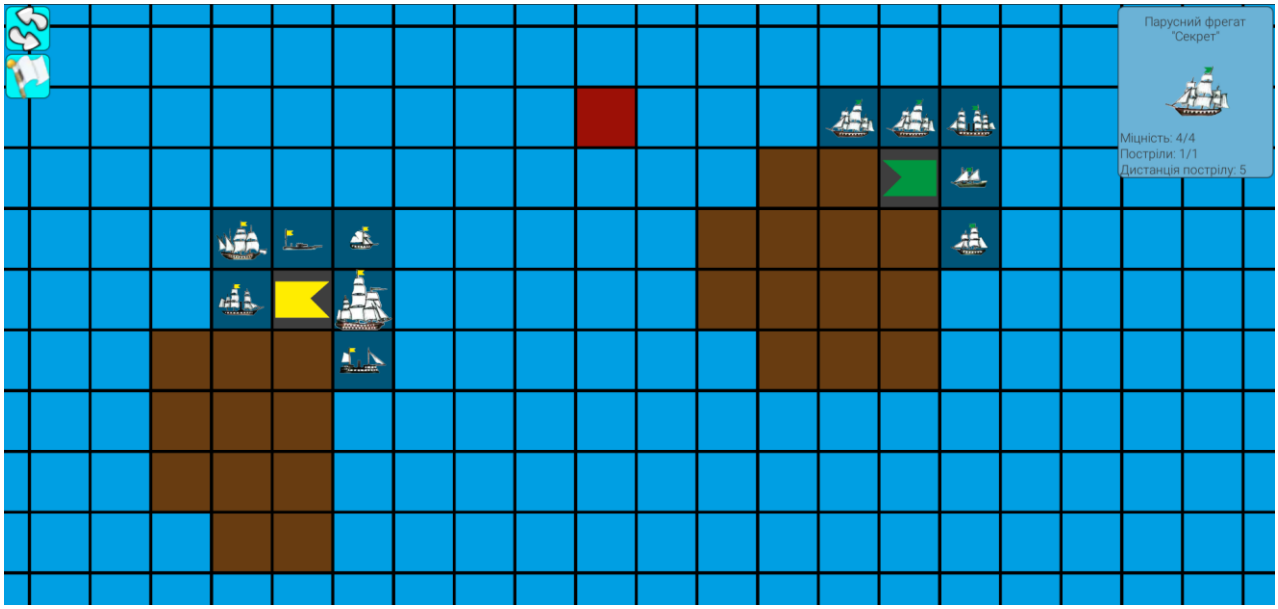


Рисунок 4.4 – Активна панель інформації про корабель

Перетягування корабля активує виділення клітинок, в які можна помістити корабель (рис. 4.5).

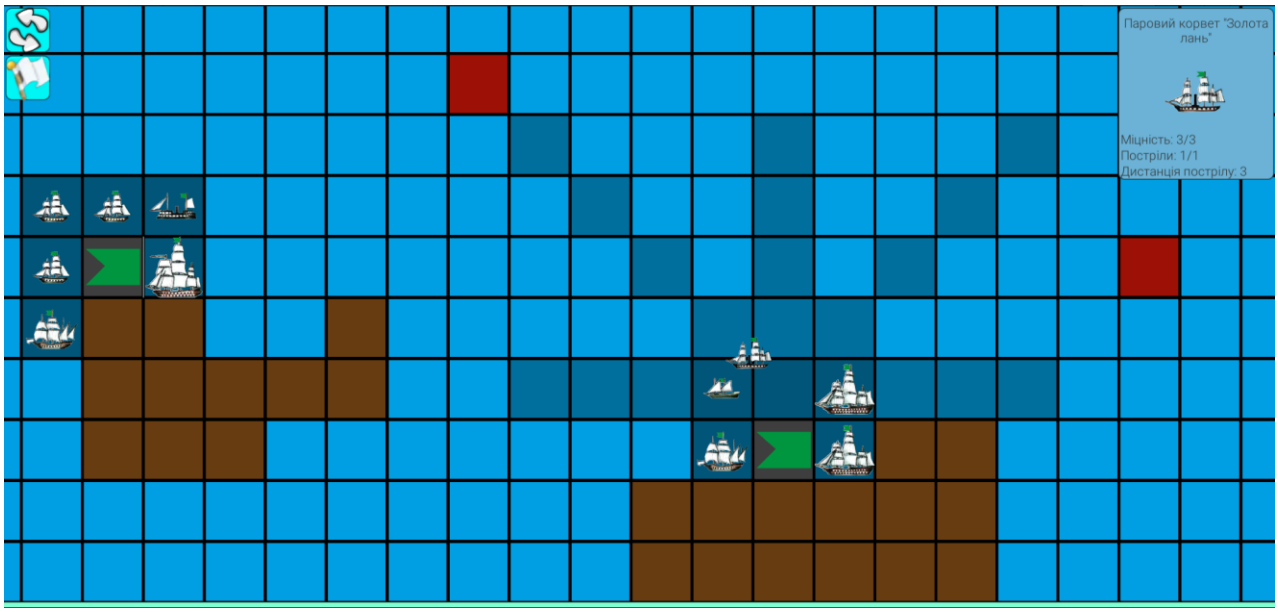


Рисунок 4.5 – Підсвічення клітинок для ходу

Клік на кораблі, у якого не закінчились постріли, підсвічує ворожі кораблі, які знаходяться в радіусі атаки (рис. 4.6).

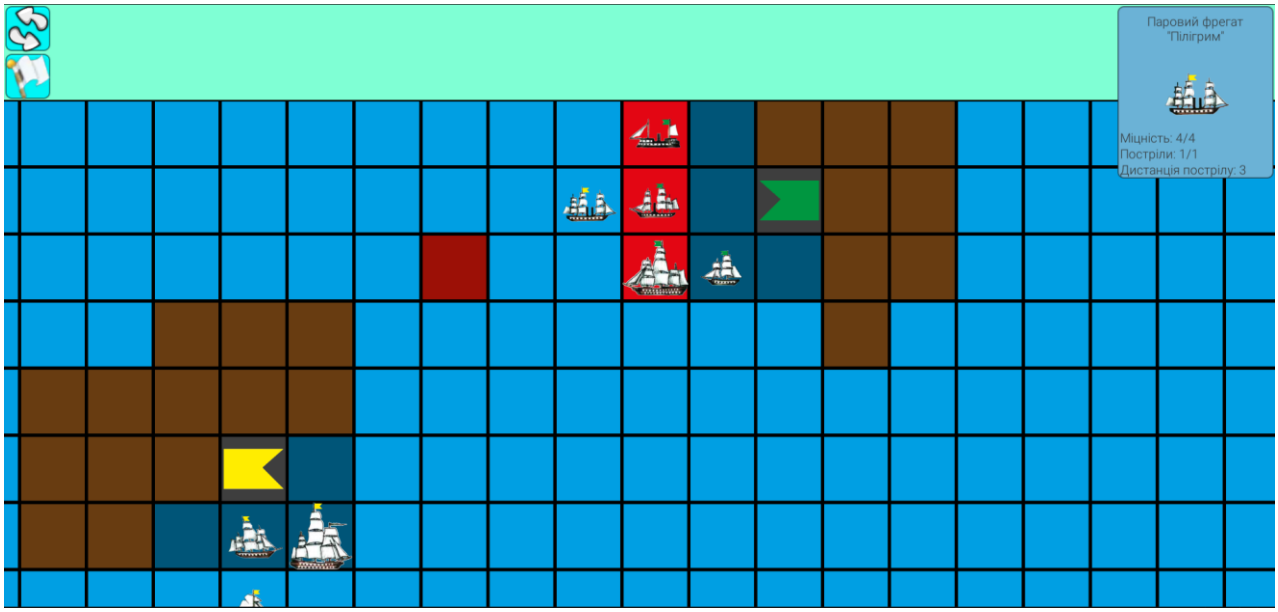


Рисунок 4.6 – Підсвічення клітинок для атаки

У випадку влучного пострілу виводиться повідомлення про успіх, а у вікні інформації про корабель, в який стріляли, відображається зменшення очок здоров'я (рис. 4.7).

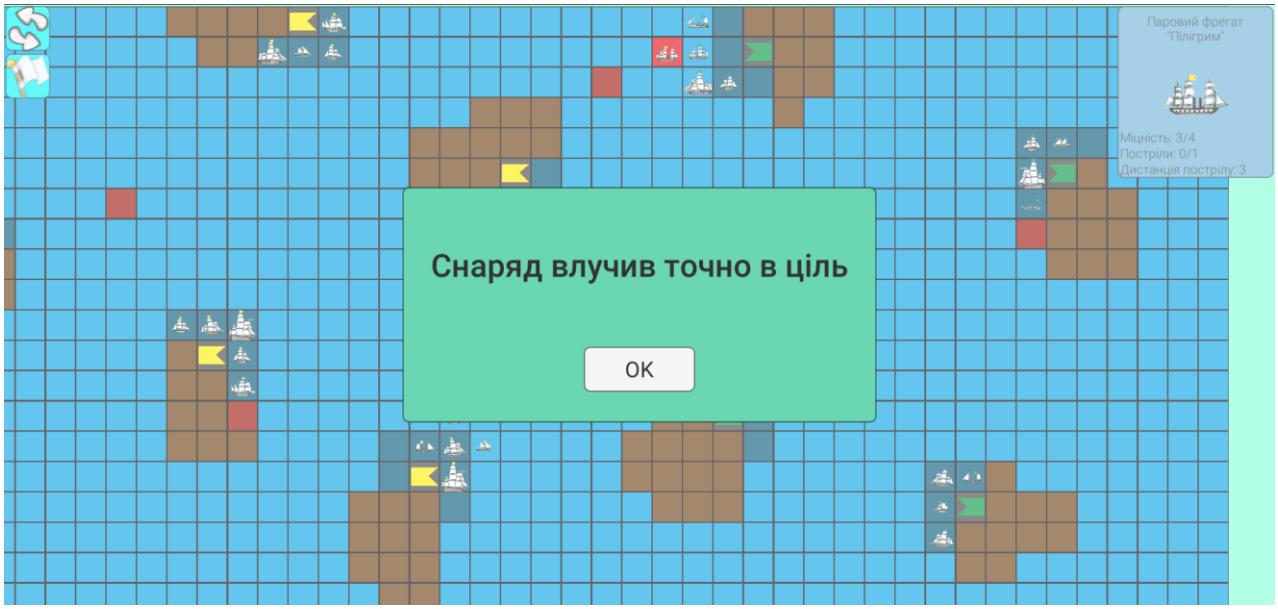


Рисунок 4.7 – Інформація про влучний постріл

За сприятливих вітрів найпростішим способом перемогти є захоплення ворожого королівського порту (рис. 4.8), оскільки шанс влучання пострілу рівний 25% і швидких кораблів у грі не так багато, тому вони не скоро наздоженуть суперника і не факт, що влучать.

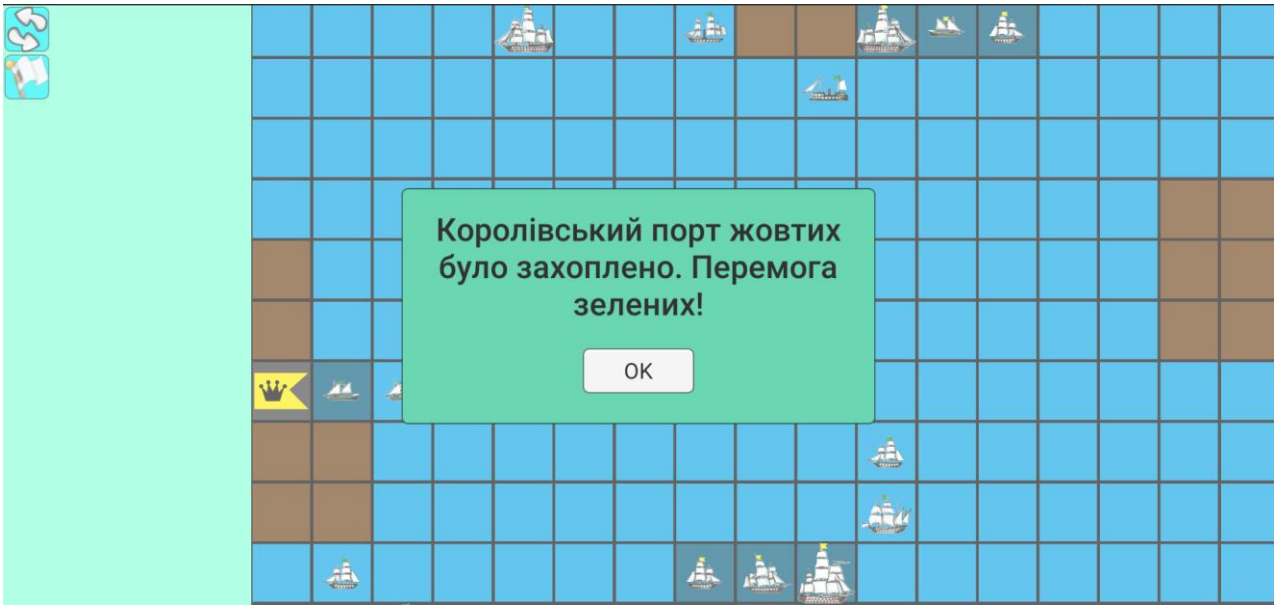


Рисунок 4.8 – Команда зелених захопила Королівський порт команди жовтих

Висновки до розділу 4

У цьому розділі було протестовано ігрові аспекти та геймплей. Усі об’єкти працюють належним чином. На панелі повідомлень відображаються повідомлення, які відповідають подіям, що відбуваються у грі: причина знищення корабля з вказанням його назви, напрям і сила вітру, чия черга ходити, причина перемоги тієї чи іншої сторони. Панель інформації виводить коректну інформацію про корабель, на якому було натиснуто курсор. Кнопка передачі ходу запускає відповідну подію, та виводить повідомлення про те, який гравець має здійснювати хід. При натисненні кнопки здачі активується панель з повідомленням про здачу супернику та встановлюється подія завершення гри для кнопки інформаційної панелі. Клік на панелі інформації закриває її. Клік на кнопці, що розміщена на панелі повідомлень закриває її та може активувати одну з двох подій: завершення гри і перехід до головного меню або перехід ходу до іншого гравця.

При старті нової гри кораблям присвоюються випадкові імена, клітинка розташування також обирається випадково серед портів того ж кольору. Ігрове поле генерується правильно, всі клітинки прилягають одна до одної та не накладаються. При спробі перемістити корабель, який вже походив, нічого не відбувається, перенесення корабля до клітинки, в яку не можна походити або за межі ігрового поля повертають його у вихідну позицію. Стрільба виконується коректно, влучання трапляється приблизно в чверті випадків. При отриманні пробоїни у корабля зменшується кількість здоров’я. При падінні цього параметра до 0, він зникає з поля.

Механіка шторму коректно працює для всіх кораблів – їх переміщає в потрібному напрямку, у цей хід вони не можуть стріляти та переміщатись, при потраплянні на рифи вони знищуються.

Усі умови для виграшу перевіряються та виконуються належним чином. При входженні корабля у королівський порт починає змінюватись лічильник, при виході або знищенні – лічильник обнуляється.

Висновки

Цей дипломний проект присвячено розробці комп’ютерної гри на основі настільної гри з назвою “Морська баталія або гра в кораблики” з використанням існуючого ігрового рушія.

У першому розділі було розглянуто основні поняття, що стосуються настільних ігор, основні характеристики різних класів настільних ігор. Також було проаналізовано правила настільної гри, яка стала основою для проєкту та кілька прикладів існуючих реалізацій настільних ігор, більш чи менш вдалих та їх основні плюси і мінуси.

Оскільки було прийнято рішення розробляти гру на базі існуючого ігрового рушія, в другому розділі було проаналізовано три найбільш популярні ігрові рушії, виділено плюси та мінуси кожного. Більша увага приділялась особливостям рушіїв, які найбільш відповідають особливостям гри, розроблюваної в цьому дипломі.

Третій розділ присвячено безпосередньо розробці гри. Описано основні об’єкти та класи проєкту в Unity, їх зв’язок та призначення. У грі використовується проста графіка, кількість елементів не дуже велика, тому її можна запустити на комп’ютерах зі слабким апаратним забезпеченням. Через особливості створення проєктів на Unity, гру можна збілдити не лише для ОС Windows, а і для інших актуальних операційних систем. Єдиним обмеженням є наявність підтримки миші або її емулятора, або ж тачскріна у пристрою, на якому запускатиметься гра.

У четвертому розділі було продемонстровано процес гри, основні елементи та ігрові моменти і механіки. Проведено тестування різних ситуацій, які можуть виникнути під час гри. Показано головне меню, безпосередньо гру, інтерфейс.

Результатом виконання дипломного проєкту є ігровий додаток, який можна запустити на операційній системі Windows та провести вільний час, змагаючись із другом в умінні використовувати ситуацію на свою користь та продумувати тактику боїв.

СПИСОК ВИКОРИСТАНОЇ ЛІТЕРАТУРИ

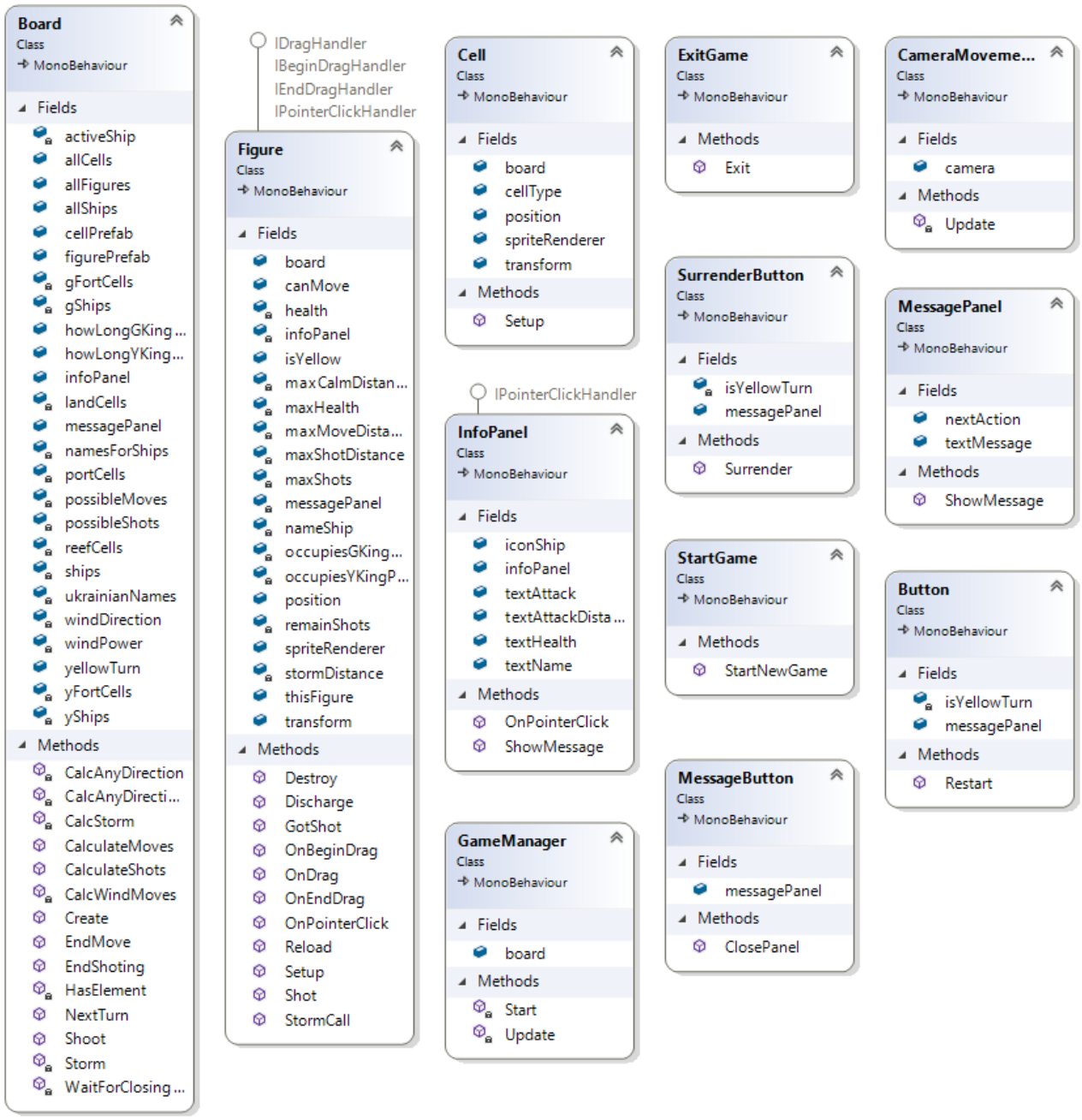
1. Орлов О. П., Попов Р. Б. / Кораблики и солдатики / Ленинград, издательство «Детская литература» – 1989 – №493 – 48 с.
2. Что такое настольные игры и для чего они нужны? | Интернет-магазин Nastolkino [Електронний ресурс] – 2018 – Режим доступу: https://nastolkino.com.ua/chto-takoe-nastolnye-igry-i-dlya-chego-oni-nuzhny/ (дата звернення 27.05.2021)
3. Why we can't stop playing computer solitaire. [Електронний ресурс] – 2008 – Режим доступу: https://slate.com/human-interest/2008/05/why-we-can-t-stop-playing-computer-solitaire.html (дата звернення 27.05.2021)
4. Dota Auto Chess. Только свежий и оригинальный контент. [Електронний ресурс] – Режим доступу: https://d2chess.com (дата звернення 27.05.2021)
5. MONOPOLY® PLUS в Steam [Електронний ресурс] – Режим доступу: https://store.steampowered.com/app/562810/MONOPOLY_PLUS/ (дата звернення 27.05.2021)
6. Что такое игровой движок | Funduk.ua [Електронний ресурс] – 2018 – Режим доступу: https://funduk.ua/technoblog/gaming-raznoe/chto-takoe-igrovoy-dvizhok/ (дата звернення 27.05.2021)
7. Топ-10 игровых движков: выбери свой | App2top [Електронний ресурс] – 2014 – Режим доступу: https://app2top.ru/game_development/top-10-igrovy-h-dvizhkov-vy-beri-svoj-45170.html (дата звернення 27.05.2021)
8. Что такое Unity 3D [Електронний ресурс] – Режим доступу: http://web.spt42.ru/index.php/chto-takoe-unity-3d (дата звернення 27.05.2021)
9. Что такое CryEngine [Електронний ресурс] – Режим доступу: http://web.spt42.ru/index.php/chto-takoe-cryengine (дата звернення 27.05.2021)
10. What is Unreal Engine? [Електронний ресурс] – Режим доступу: https://conceptartempire.com/what-is-unreal-engine/ (дата звернення 27.05.2021)

11. Made with Unity | Unity [Електронний ресурс] – Режим доступу: https://unity.com/ru/madewith (дата звернення 27.05.2021)
12. ТОП 10 лучших игр на Unreal Engine | GAME.DATA [Електронний ресурс] – 2017 – Режим доступу: https://gamedata.club/article-top/1324-top-10-luchshih-igr-na-unreal-engine.html (дата звернення 27.05.2021)
13. ТОП 10 лучших игр на движке CryEngine | GAME.DATA [Електронний ресурс] – 2017 – Режим доступу: https://gamedata.club/article-top/2181-top-10-luchshih-igr-na-dvizhke-cryengine.html (дата звернення 27.05.2021)
14. Очные и онлайн-курсы и обучение разработке 2D-, 3D-, AR- и VR-приложений | Дистанционное обучение [Електронний ресурс] – Режим доступу: https://unity.com/ru/learn (дата звернення 27.05.2021)
15. Unreal Engine 4 • » Уроки [Електронний ресурс] – Режим доступу: https://uengine.ru/category/ue4-tutorials (дата звернення 27.05.2021)
16. CRYENGINE | Tutorials [Електронний ресурс] – Режим доступу: https://www.cryengine.com/tutorials (дата звернення 27.05.2021)
17. Hocking J. / Unity in Action: Multiplatform game development in C#. Second edition / Manning Publications – 2018 – 400c.
18. Halpern J. / Developing 2D Games with Unity: Independent Game Programming with C# / Apress, United States – 2018 – 408 c.
19. Patrick F. / Unity From Zero to Proficiency (Foundations): A step-by-step guide to creating your first game – 2019 – 240 c.
20. Nugent T. Manning J. Buttfield-Addison P. / Unity Game Development Cookbook: Essentials for Every Game / O'Reilly Media – 2019 – 408 c.

Додаток А
ФУНКЦІОНАЛЬНА СХЕМА

до дипломного проєкту
на тему: «Комп’ютерна гра за мотивами настільної гри»

Київ – 2021 року

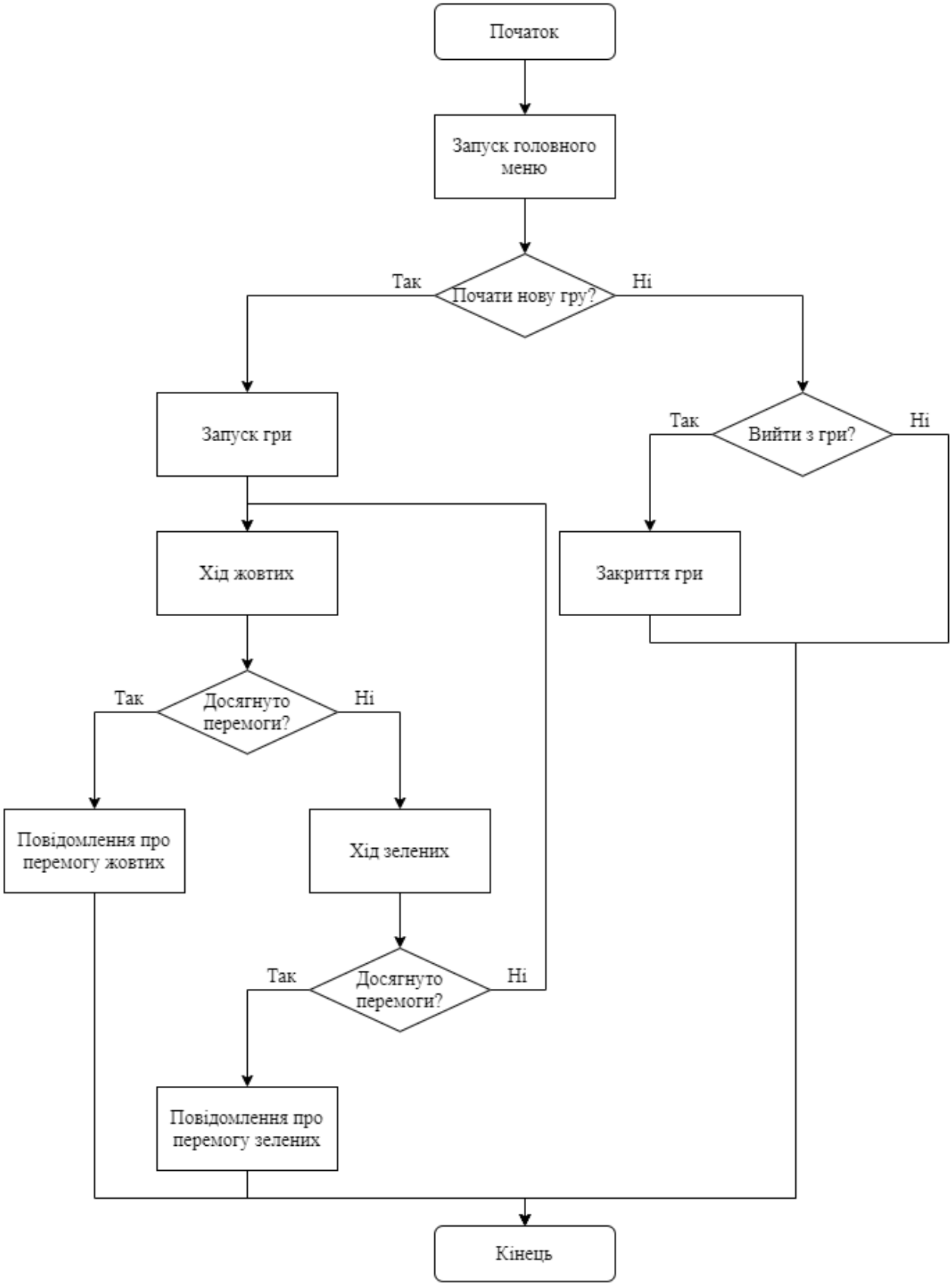


					ІАЛЦ.467800.004 Д1	
Зм.	Арк.	№ докум.	Підпис	Дата		
Розробив		Лозко Л. Л.			Комп'ютерна гра за мотивами настільної гри	Літ. / Арк. 1 / Аркушів 1
Перевірив		Калюжний О. О.				
Реценз.						«КПІ імені Ігоря.Сікорського» ФІОТ, гр. ІО-71
Н. Контр.		Сімоненко В.П.				
Затв.		Стіренко С.Г.			Функціональна схема	

Додаток Б
ПРИНЦИПОВА СХЕМА

до дипломного проєкту
на тему: «Комп’ютерна гра за мотивами настільної гри»

Київ – 2021 року



					ІАЛЦ.467800.005 Д2	
Зм.	*Арк.*	*№ докум.*	*Підпис*	*Дата*		
Розробив		*Лозко Л. Л.*			*Комп'ютерна гра за мотивами настільної гри*	*Літ.* / *Арк.* / *Аркушів*: 1 / 1
Перевірив		*Калюжний О. О.*				
Реценз.						*«КПІ імені Ігоря.Сікорського» ФІОТ, гр. ІО-71*
Н. Контр.		*Сімоненко В.П.*				
Затв.		*Стіренко С.Г.*			*Принципова схема*	

Додаток В
СТРУКТУРНА СХЕМА

до дипломного проєкту
на тему: «Комп’ютерна гра за мотивами настільної гри»

Київ – 2021 року

					ІАЛЦ.467800.006 Д3		
Зм.	*Арк.*	*№ докум.*	*Підпис*	*Дата*			
Розробив		*Лозко Л. Л.*			*Комп'ютерна гра за мотивами настільної гри*	*Літ.*	*Арк.* 1 / *Аркушів* 1
Перевірив		*Калюжний О. О.*					
Реценз.						*«КПІ імені Ігоря.Сікорського» ФІОТ, гр. ІО-71*	
Н. Контр.		*Сімоненко В.П.*					
Затв.		*Стіренко С.Г.*			*Структурна схема*		