МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

ІНЖЕНЕРНИЙ НАВЧАЛЬНО-НАУКОВИЙ ІНСТИТУТ
ім. Ю.М. Потебні
ЗАПОРІЗЬКОГО НАЦІОНАЛЬНОГО УНІВЕРСИТЕТУ

КАФЕДРА ЕЛЕКТРОНІКИ, ІНФОРМАЦІЙНИХ СИСТЕМ ТА ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

Кваліфікаційна робота

перший (бакалаврський)
(рівень вищої освіти)

на тему **Розробка інтерактивної відеогри жанру симулятор бармена з механікою змішування коктейлів на Unity**

Виконав: студент 4 курсу, групи 6.1212-пзс
спеціальності 121 Інженерія програмного забезпечення
(код і назва спеціальності)
освітньої програми Програмне забезпечення систем
(код і назва освітньої програми)
І.М.Фролова
(ініціали та прізвище)
Керівник асистент, PhD, О.О. Жуков
(посада, вчене звання, науковий ступінь, підпис, ініціали та прізвище)
Рецензент директор ТОВ «Діскус»
Р.О. Лютий
(посада, вчене звання, науковий ступінь, підпис, ініціали та прізвище)

Запоріжжя
2026

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

ІНЖЕНЕРНИЙ НАВЧАЛЬНО-НАУКОВИЙ ІНСТИТУТ ім. Ю.М. Потебні ЗАПОРІЗЬКОГО НАЦІОНАЛЬНОГО УНІВЕРСИТЕТУ

Кафедра електроніки, інформаційних систем та програмного забезпечення
Рівень вищої освіти перший (бакалаврський)
Спеціальність 121 Інженерія програмного забезпечення
(код та назва)
Освітня програма Програмне забезпечення систем
(код та назва)

ЗАТВЕРДЖУЮ

Завідувач кафедри Тетяна КРИТСЬКА

“ 01 ” березня 2026 року

З А В Д А Н Н Я
НА КВАЛІФІКАЦІЙНУ РОБОТУ СТУДЕНТОВІ

Фролової Ірини Миколаївни
(прізвище, ім’я, по батькові)

1. Тема роботи Розробка інтерактивної відеогри жанру симулятор бармена з механікою змішування коктейлів на Unity

керівник роботи Жуков Олександр Олексійович, PhD, асистент кафедри електроніки, інформаційних систем та програмного забезпечення
(прізвище, ім’я, по батькові, науковий ступінь, вчене звання)
затверджена наказом ЗНУ від 23.12.2025 р. № 1830-с
Строк подання студентом кваліфікаційної роботи 02.06.2026

2. Вихідні дані бакалаврської роботи
   - комплект нормативних документів ;
   - технічне завдання до роботи.

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

   - огляд та збір літератури стосовно теми кваліфікаційної роботи;;
   - огляд та аналіз існуючих веб-сервісів і програмних рішень-аналогів;
   - проєктування архітектури клієнт-серверної інформаційної системи;
   - реалізація клієнтської та серверної частини веб-додатку;
   - розробка висновків та пропозицій щодо подальшого розвитку системи;

5. Перелік графічного матеріалу (з точним зазначенням обов’язкових креслень) 16 слайдів презентації

6. Консультанти розділів бакалаврської роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата
		Завдання прийняв

7. Дата видачі завдання 01.03.2026

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів бакалаврської роботи	Строк виконання етапів бакалаврської роботи	Примітка
1	Аналіз предметної області	02.03-01.04.2026	виконано
2	Формулювання основної задачі дипломної роботи та узгодження її з науковим керівником	03.04-08.04.2026	виконано
3	Аналіз існуючих аналогів	11.04-14.04.2026	виконано
4	Вибір технологій для реалізації.	15.04-18.04.2026	виконано
5	Аналіз ігрових механік жанру симулятор.	19.04-20.04.2026	виконано
6	Проектування архітектури системи.	21.04-22.04.2026	виконано
7	Реалізація каталогу оголошень, пошуку та фільтрів	22.04-24.04.2026	виконано
8	Розробка системи взаємодії з інтерактивними об'єктами.	25.04-27.04.2026	виконано
9	Реалізація механіки змішування коктейлів.	29.04-01.05.2026	виконано
10	Розробка користувацького інтерфейсу.	02.05-06.05.2026	виконано
11	Розробка системи обслуговування клієнтів та діалогів.	07.05-09.05.2026	виконано
12	Реалізація аудіосистеми та механіки просвічування клієнтів.	10.05-14.05.2026	виконано
13	Рефакторинг коду.	15.05-18.05.2026	виконано
14	Системне тестування ігрових механік	19.05-23.05.2026	виконано
15	Оформлення звіту.	24.05-26.05.2026	виконано
16	Оформлення презентації і отримання рецензії.	27.05-29.05.2026	виконано

Студент _____ _____ І.М. Фролова

Керівник роботи _____ _____О.О. Жуков

Нормоконтроль пройдено

Нормоконтролер _____ _____І.А. Скрипник

АНОТАЦІЯ

Сторінок — 72,

Рисунків — 17,

Таблиць — 2,

Джерел — 22.

Фролова І.М. Розробка інтерактивної відеогри жанру симулятор бармена з механікою змішування коктейлів на Unity: кваліфікаційна робота бакалавра спеціальності 121 «Інженерія програмного забезпечення» / наук. керівник О. О. Жуков. Запоріжжя : ЗНУ, 2026,72 с.

Метою кваліфікаційної роботи є розробка інтерактивної нарративної відеогри в жанрі симулятор бармена з глибокою механікою змішування коктейлів на ігровому рушії Unity, що поєднує у собі динамічну взаємодію з предметами, реалістичну симуляцію рідин та систему обслуговування клієнтів.

У процесі розробки було проаналізовано існуючі аналоги ігор цього жанру, визначено ключові ігрові механіки та спроєктовано модульну архітектуру програмного продукту. Результатом роботи є готовий ігровий додаток «Last Night Last Call», створений на платформі Unity з використанням мови програмування C# та графічного конвеєру URP. Гра містить три ігрові локації, шість типів клієнтів, систему діалогів з підтримкою двох мов локалізації та механіку змішування коктейлів на основі п'яти базових інгредієнтів. Механіка змішування реалізована через процедурну деформацію тривимірного меша рідини та алгоритм поступового змішування кольорів. Логіка системи побудована на паттернах Singleton та Template Method.

Ключові слова: *відеогра, симулятор бармена, Unity, C#, змішування коктейлів, ігрові механіки, інтерактивність, нарративна гра, психологічний хорор, URP, інді-розробка*.

ABSTRACT

Pages — 72,

Drawings — 17,

Tables — 2,

Sources — 22.

Frolova I.M. Development of an Interactive Bartender Simulator Video Game with Cocktail Mixing Mechanics on Unity: bachelor's thesis in specialty 121 "Software Engineering" / scientific supervisor O. O. Zhukov. Zaporizhzhia: ZNU, 2026, 72 p.

The goal of the qualification work is the development of an interactive narrative video game in the bartender simulator genre with deep cocktail mixing mechanics on the Unity game engine, combining dynamic object interaction, realistic liquid simulation, and a customer service system.

During the development process, existing analogues of games in this genre were analyzed, key game mechanics were defined, and a modular architecture of the software product was designed. The result of the work is a completed game application called "Last Night Last Call", created on the Unity platform using the C# programming language and the URP (Universal Render Pipeline) graphics pipeline. The game contains three game locations, six types of customers, a dialogue system with support for two localization languages (Russian and English), and a cocktail mixing mechanic based on five basic ingredients. The mixing mechanic is implemented through procedural deformation of a three-dimensional liquid mesh and an algorithm for gradual color blending. The system logic is built on the Singleton pattern (the central SYSTEM class) and the Template Method pattern (a component model with a life cycle).

Keywords: *video game, bartender simulator, Unity, C#, cocktail mixing, game mechanics, interactivity, narrative game, psychological horror, URP, indie development.*

ЗМІСТ

ВСТУП 8
1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ ТА ІСНУЮЧИХ РІШЕНЬ 13
1.1 Огляд літературних джерел 13
1.2 Аналіз існуючих аналогів 15
1.2.1 VA-11 Hall-A: Cyberpunk Bartender Action 15
1.2.2 Coffee Talk 16
1.2.3 The Red Strings Club 17
1.2.4 Bartender: The Right Mix 18
1.2.5 Порівняльна таблиця аналогів 19
1.3 Постановка завдання 21
1.5 Висновки до розділу 1 23
2 ВИМОГИ ДО ОТОЧЕННЯ 24
2.1 Аналіз сучасних технологій для розробки відеоігор 24
2.1.1 Ігровий рушій Unity 24
2.1.2 Графічний конвеєр URP 25
2. 1.3 Додаткові інструменти розробки 26
2. 1.4 Система локалізації 27
2.2 Аналіз вимог до користувачів 27
2.2.1 Гравець 27
2.2.2 Розробник (тестувальник) 28
2.3 Аналіз вимог до апаратного та програмного забезпечення 28
2.3.1 Мінімальні вимоги до апаратного забезпечення гравця 29
2.3.2 Рекомендовані вимоги до апаратного забезпечення гравця 29
2.3.3 Вимоги до програмного забезпечення гравця 30
2.5 Висновки до розділу 2 30
3 РОЗРОБКА ВІДЕОГРИ «Last Night Last Call» 31
3.1 Опис предметної області 31
3.2 Архітектура системи 32
3.2.1 Загальна структура системи 32

3.2.2 Життєвий цикл компонентів системи 34
3.2.3 Ієрархія інтерактивних об'єктів 35
3.2.4 Система рівнів та скінченних автоматів 37
3.3 Функціональні вимоги системи 40
3.4 Функціонал системи 41
3.4.1 Система взаємодії з інтерактивними об'єктами 42
3.4.2 Механіка змішування коктейлів 43
3.4.3 Система обслуговування клієнтів 45
3.4.4 Діалогова система та локалізація 49
3.4.5 Механіка шейкера (збовтування) 51
3.4.6 Система ігрових рівнів та переміщення 52
3.4.7 Механіка просвічування клієнтів крізь пляшку 53
3.4.8 Аудіосистема 55
3.4.9 Система налаштувань та збереження стану 58
3.5 Вимоги до інтерфейсу 60
3.6 Реалізація і тестування 61
3.6.1 Реалізація 61
3.6.2 Апробація 61
3.7 Висновки до розділу 3 66
ВИСНОВКИ 68
СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ 71

ВСТУП

Актуальність теми

Ігрова індустрія на сьогоднішній день є однією з найбільш динамічних та економічно успішних галузей розважального контенту у світі. За даними аналітичних компаній, глобальний ринок відеоігор стабільно зростає кожного року, а частка інді-ігор у цьому зростанні постійно збільшується. Саме інді розробка, завдяки своїй гнучкості та свободі творчого вираження, відкриває можливості для реалізації нестандартних художніх задумів і оригінальних ігрових механік, які часто залишаються поза увагою великих студій-видавців.

Особливу популярність серед інді-проєктів набули ігри жанру симулятор. За даними аналітичної платформи Gamalytic, на платформі Steam станом на 2026 рік представлено понад 25 000 ігор жанру Simulation, сукупна виручка яких перевищує 17,8 мільярда доларів США [1]. Ці показники свідчать про стійкий попит гравців на ігри, які пропонують занурення у певну діяльність, – від управління фермою до роботи за барною стійкою.

Підкатегорія симуляторів бармена є особливо цікавою з огляду на поєднання декількох складових: інтерактивної механіки змішування напоїв, розвинутої діалогової системи, соціальної взаємодії з віртуальними персонажами та атмосферного сетингу. Такі ігри як VA-11 Hall-A: Cyberpunk Bartender Action (2016), The Red Strings Club (2018) та Coffee Talk (2020) стали культовими прикладами того, як за допомогою нескладної ігрової механіки можна створити глибокий нарративний досвід, що захоплює сотні тисяч гравців по всьому світу.

Актуальність розробки відеогри в жанрі симулятор бармена на платформі Unity зумовлена декількома факторами. По-перше, наявним попитом гравців на атмосферні нарративні ігри з невеликим порогом входу. По-друге, потребою у вітчизняних продуктах ігрової індустрії, здатних конкурувати на міжнародних майданчиках цифрової дистрибуції. По-третє, платформа Unity на сьогодні є однією з найбільш доступних та потужних

систем для розробки 3D-ігор, що дозволяє окремим розробникам та невеликим командам створювати якісний продукт без значних фінансових вкладень.

Окремої уваги заслуговує технічний аспект розробки. Реалістична симуляція рідин, процедурна деформація мешів, побудова модульної компонентно-орієнтованої архітектури та реалізація системи скінчених автоматів для управління станами ігрових рівнів – це ті технічні виклики, які постають перед розробником і потребують глибокого розуміння як ігрового рушія, так і принципів об'єктно-орієнтованого програмування.

Оскільки система передбачає реєстрацію користувачів, збереження контактних даних, автентифікацію, завантаження фотографій і обмін повідомленнями, під час її розробки необхідно враховувати базові вимоги безпеки веб-додатків. У роботі такі вимоги враховано через використання хешування паролів, JWT-автентифікації, refresh-token механізму, валідації вхідних даних, перевірки файлів, розмежування ролей та модерації контенту.

Мета дослідження

Розробка інтерактивної відеогри жанру симулятор бармена з реалістичною механікою змішування коктейлів на платформі Unity, що забезпечує занурення гравця у ігровий процес завдяки динамічній взаємодії з предметами, системі обслуговування клієнтів та наративній складовій.

Завдання дослідження

Аналіз існуючих рішень щодо створення ігор-симуляторів. Вивчення архітектури Unity та можливостей мови програмування C# з метою розуміння принципів роботи та використання їх у розробці відеогри. Розробка ефективної архітектури відеогри, яка дозволить забезпечити оптимальну продуктивність та гнучкість. Реалізація ключових ігрових механік: взаємодії з предметами, змішування коктейлів, обслуговування клієнтів та діалогової системи. Проведення тестування розробленого продукту.

Об’єкт і предмет дослідження

Об'єктом дослідження є процес розробки інтерактивної відеогри жанру симулятор на платформі Unity з використанням мови програмування C#.

Предмет дослідження

Предметом дослідження є методи, алгоритми та технології, що застосовуються при реалізації механіки змішування коктейлів, системи взаємодії з інтерактивними об'єктами та компонентно-орієнтованої архітектури відеогри.

Методи дослідження

Проведення детального огляду літературних джерел, наукових публікацій, статей, підручників та інших джерел, що стосуються технологій розробки відеоігор, особливостей платформи Unity; вивчення та аналіз популярних ігор жанру симулятор бармена з метою виявлення їх особливостей, сильних і слабких сторін; методи об'єктно-орієнтованого аналізу та проєктування; апробація результатів.

Практичне значення одержаних результатів

Практичне значення одержаних результатів дослідження полягає у створенні готового до використання інтерактивного ігрового продукту, який може бути опубліковано на майданчиках цифрової дистрибуції. Розроблена відеогра демонструє практичне застосування архітектурних паттернів та компонентно-орієнтованого підходу Unity для створення інді-ігор з нарративним фокусом. Отримані у процесі роботи напрацювання – зокрема реалізація механіки змішування рідин через процедурну деформацію меша – можуть бути використані іншими розробниками як референсне рішення для створення подібних ігор у жанрі симулятора. Результати дослідження можуть сприяти обміну знаннями та досвідом серед інді-ігрової спільноти.

Глосарій

Unity – міжплатформове середовище розробки відеоігор та інтерактивних додатків, що підтримує створення 2D- та 3D-контенту для широкого спектру платформ.

C# – об'єктно-орієнтована мова програмування загального призначення, яка є основною мовою розробки скриптів у середовищі Unity.

GameObject – базовий об'єкт у Unity, що представляє будь-яку сутність на ігровій сцені. До нього можна додавати компоненти, які визначають його поведінку.

Component – скрипт або логічна одиниця, яку можна прикріпити до GameObject для надання йому певного функціоналу (наприклад, фізики, анімації, рендерингу).

MonoBehaviour – базовий клас Unity, від якого успадковуються всі скрипти-компоненти. Надає методи життєвого циклу, такі як Start, Update, Awake.

ScriptableObject – серіалізований клас Unity для зберігання даних поза ігровою сценою, що використовується для створення конфігураційних ресурсів та пресетів.

Prefab (префаб) – шаблон ігрового об'єкта у Unity, який дозволяє повторно використовувати об'єкт зі збереженими налаштуваннями компонентів.

URP (Universal Render Pipeline) – універсальний графічний конвеєр Unity, що забезпечує оптимізоване відображення графіки на широкому спектрі пристроїв.

Mesh (меш) – геометричне представлення 3D-моделі у вигляді набору вершин, ребер і полігонів.

Shader (шейдер) – спеціалізована програма, що виконується на графічному процесорі та визначає, як саме буде відображено пікселі чи вершини графічного об'єкта.

Raycast – метод трасування променя у тривимірному просторі, який використовується для визначення об'єктів, з якими перетинається заданий напрямок.

Singleton – паттерн проєктування, який гарантує, що клас має лише один екземпляр, та надає глобальну точку доступу до нього. State Machine (скінченний автомат) – обчислювальна модель, яка описує поведінку системи через набір станів та переходів між ними.

NPC (Non-Player Character) – персонаж у відеогрі, яким керує не гравець, а внутрішня логіка програми.

FBX – формат файлу для обміну тривимірними моделями, анімаціями та сценами між різними програмами тривимірної графіки.

Game Design Document (геймдизайн-документ, GDD) – документ, що описує концепцію, механіки, персонажів, сюжет та інші аспекти відеогри.

1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ ТА ІСНУЮЧИХ РІШЕНЬ

1.1 Огляд літературних джерел

Розробка інтерактивної відеогри жанру симулятор бармена на платформі Unity є комплексною задачею, яка об'єднує знання з кількох суміжних галузей: геймдизайну, комп'ютерної графіки, об'єктно орієнтованого програмування та проєктування інтерфейсів користувача. Для системного підходу до реалізації кваліфікаційної роботи було проведено аналіз літературних джерел, які охоплюють зазначені напрями.

Фундаментальною працею в галузі геймдизайну є книга Джессі Шелла [2], у якій автор розкриває принципи проєктування ігрових механік, елементи ігрового досвіду та підходи до побудови емоційного зв'язку з гравцем. Підхід «лінз» дозволяє поглянути на гру з різних боків: ігровий процес, наратив, естетика, технології. Ці концепції були використані при проєктуванні ігрових сцен та діалогової системи.

У сфері розробки на платформі Unity основним джерелом виступає офіційна документація Unity Technologies [3], яка містить детальний опис API середовища, приклади використання компонентів та посібники з оптимізації. Документація Unity постійно оновлюється відповідно до нових версій платформи та є авторитетним джерелом інформації для розробників будь-якого рівня підготовки. Також значну допомогу надає книга Джеремі Гібсона Бонда [4], де поєднано теорію геймдизайну з практичними прикладами реалізації на Unity.

Питання архітектурних паттернів у контексті ігрового програмування детально розглянуто у праці Роберта Ністрома [5]. Автор адаптує класичні паттерни [6] до специфіки розробки ігор та вводить ігрово-специфічні паттерни, такі як Game Loop, Update Method, Component, Event Queue, Service Locator. Більшість з них безпосередньо застосовано при розробці відеогри: так, клас SYSTEM у реалізованому проєкті відповідає паттерну Service

Locator, а базовий клас SystemComponent – паттерну Template Method.

Для поглибленого розуміння компонентно-орієнтованого підходу корисним є джерело [7] Entity Component System (ECS), який дедалі більше використовується у сучасних іграх завдяки своїй продуктивності та гнучкості. Хоча у кваліфікаційній роботі використовується класична об'єктно орієнтована архітектура з MonoBehaviour, розуміння ECS дало змогу правильно структурувати компоненти та уникнути тісного зв'язування коду.

Аспекти процедурної генерації графіки, зокрема деформація мешів для симуляції рідин, розкриті у роботі Еріка Ленгьеля [8] та у численних технічних статтях на ресурсах Gamasutra (нині Game Developer) [9] та GDC Vault. Математичний апарат, що лежить в основі реалізації механіки рідини, ґрунтується на лінійній алгебрі – зокрема, операції скалярного множення для обчислення відстані від точки до площини.

Окремої уваги заслуговують аналітичні джерела, які надають статистичну картину ігрової індустрії. Платформа Gamalytic [1], що агрегує дані з Steam, дозволяє оцінити фінансові показники ігор, кількість проданих копій, середню тривалість сесії та інші метрики. Ці дані використано в роботі для обґрунтування актуальності обраної теми та порівняння з існуючими аналогами.

Таким чином, аналіз літератури показав, що проблематика розробки інтерактивних відеоігор у жанрі симулятор досить широко представлена у наукових публікаціях та професійній літературі. Водночас специфічна ніша симуляторів бармена з глибокою механікою змішування залишається малодослідженою в українському академічному середовищі, що підтверджує актуальність даної роботи.

1.2 Аналіз існуючих аналогів

Для формування цілісного уявлення про жанр симулятор бармена було проведено аналіз чотирьох найбільш відомих представників цього жанру. Розглянуто такі параметри як: концепція гри, ключові ігрові механіки, стиль графіки, сюжетна складова, показники продажів та рецензій, а також сильні та слабкі сторони кожного з продуктів.

1.2.1 VA-11 Hall-A: Cyberpunk Bartender Action

VA-11 Hall-A: Cyberpunk Bartender Action – інді-гра від венесуельської студії Sukeban Games, видана у 2016 році [10]. Гра поєднує в собі елементи візуальної новели та симулятора бармена у сеттингу кіберпанку. Геймплей гри зображено на рис. 1.



Рисунок *1 – Гра VA-11 Hall-A: Cyberpunk Bartender Action*

Ключовою механікою гри є приготування коктейлів за рецептами або на основі побажань клієнта («щось солодке», «щось міцне», «щось прохолодне»).

Гравець обирає потрібні інгредієнти з фіксованого переліку та регулює їх пропорції. Від якості приготованого напою залежить настрій клієнта, розвиток сюжету та розмір чайових, що впливають на особисте життя Джилл.

Сильні сторони: глибокий наратив з добре пропрацьованими персонажами; оригінальна стилістика пікселарт; атмосферний музичний супровід; виразна кіберпанк-естетика; проста, але ефектна механіка змішування. Слабкі сторони: суто 2D-стилістика, яка не дозволяє передати тактильність процесу приготування напоїв; обмежена варіативність ігрового процесу; відсутність просторової взаємодії з оточенням.

1.2.2 Coffee Talk

Coffee Talk – інді-гра в жанрі візуальної новели та симулятора бариста, розроблена індонезійською студією Toge Productions та видана у 2020 році [11]. Гра відображає атмосферу нічного кафе в альтернативній версії Сіетла. Гравець виконує роль бариста, який готує напої та вислуховує історії своїх клієнтів. Ігровий процес показано на рис. 2.



Рисунок *2 – Гра Coffee Talk*

Основна механіка гри – приготування гарячих напоїв з обмеженого набору базових інгредієнтів (кава, молоко, шоколад, зелений чай, імбир тощо). Гравець комбінує три компоненти у визначеній послідовності, щоб отримати потрібний напій. Особливістю є можливість малювання латте-арту на поверхні напою, що впливає на емоційний стан клієнта. Від поданого напою залежить хід розмови та варіанти розвитку окремих сюжетних ліній.

Сильні сторони: візуальний стиль у стилі lo-fi пікселарт; заспокійлива атмосфера та музика; соціально значущі теми, що піднімаються в діалогах; нескладне керування, орієнтоване на широку аудиторію. Слабкі сторони: обмежений геймплей, який переважно зводиться до вибору напою та прочитання тексту; відсутність значного впливу рішень гравця на розвиток сюжету; коротка тривалість проходження.

1.2.3 The Red Strings Club

The Red Strings Club – нарративна пригодницька гра, розроблена іспанською студією Deconstructeam та видана Devolver Digital у 2018 році [12]. Дія гри відбувається у кіберпанк-суспільстві майбутнього. Одним з трьох головних героїв є бармен Донован, який готує особливі коктейлі для відвідувачів бару «The Red Strings Club», де вирішуються корпоративні та політичні інтриги. Геймплей гри зображено на рис. 3.



Рисунок *3 – Гра Coffee Talk*

Ключові механіки гри різноманітні: окрім приготування коктейлів, гравцю доступні вирізання імплантатів на гончарному колі та телефонні дзвінки з імперсонацією. Механіка бармена побудована на емоційному маніпулюванні: Донован повинен обрати такий коктейль, який викличе у клієнта потрібну емоцію і тим самим розв'яже йому язик. Інгредієнти представлені як абстрактні рідини різних кольорів, які гравець змішує в келихах різної форми.

Сильні сторони: оригінальний художній стиль з яскравими неоновими кольорами; глибока філософська складова наративу; поєднання декількох механік у рамках однієї гри; сильний саундтрек. Слабкі сторони: коротка тривалість; лінійність сюжету при формальному виборі варіантів; обмежена варіативність механіки змішування.

1.2.4 Bartender: The Right Mix

Bartender: The Right Mix – браузерна та мобільна аркадна гра, розроблена студією Miniclip, є одним з найбільш ранніх представників жанру, що здобули широку аудиторію. Хоча гра не має розвиненого наративу, її механіка змішування коктейлів є однією з найбільш пропрацьованих у своєму сегменті з погляду фізики. Ігровий процес показано на рис. 4.



Рисунок *4 – Bartender: The Right Mix*

Сутність гри полягає в тому, щоб з набору пляшок з алкоголем поступово наповнювати келих, дотримуючись заданого співвідношення інгредієнтів. Після наповнення гравець потрясає келих та отримує оцінку залежно від точності дотримання рецепту. Відомим є те, що надмірне або недостатнє збовтування призводить до невдалого результату, а надлишкове наливання – до вибуху клієнта.

Сильні сторони: найбільш деталізована механіка наливання рідини з пляшки, яка надихнула багато подальших симуляторів; наявність елементу шейкера; доступність широкій аудиторії через браузерний запуск. Слабкі сторони: застаріла графіка; відсутність сюжету та атмосфери; примітивна модель клієнтів; обмежена реіграбельність.

1.2.5 Порівняльна таблиця аналогів

Як показано на рисунку 5, загальна статистика ринку ігор-симуляторів на платформі Steam демонструє, що цей жанр є одним з найбільших та найприбутковіших у сегменті інді-ігор.

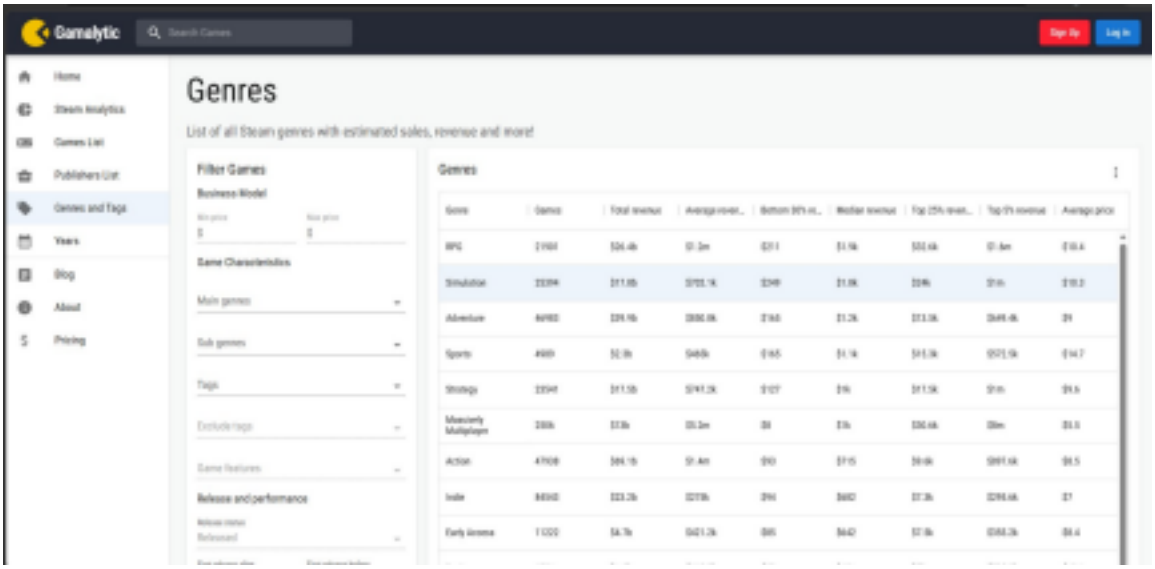


Рисунок *5* – *Статистика жанрів відеоігор на платформі Steam за даними Gamalytic*

За наведеними даними, жанр Simulation налічує понад 25 тисяч ігор з сукупним валовим доходом 17,8 мільярда доларів США [1], що свідчить про стійкий попит гравців та комерційну привабливість цього напряму для розробників.

Для узагальнення аналізу було складено порівняльну таблицю основних аналогів за ключовими параметрами (табл. 1).

Таблиця *1 – Порівняння аналогів ігор жанру симулятор бармена*

Параметр	**VA-11 Hall-A**	**Coffee Talk**	**The Red Strings Club**	**Bartender: TRM**	**Midnight Jam**
Рік випуску	2016	2020	2018	2010	2026
Графічний стиль	2D пікселарт	2D пікселарт	2D арт	2D flash	3D URP
Сеттинг	Кіберпанк	Фентезі	Кіберпанк	Умовний	Нічне місто
Механіка шейкера	–	–	–	Є	Є
Нарратив	Глибокий	Середній	Глибокий	Немає	Глибокий
Копій продано	~1,1 млн	~386 тис.	~383 тис.	– free	–
Оцінка Steam, %	97	95	94	–	–

Як видно з таблиці 1, існуючі аналоги переважно використовують двовимірну графіку, що обмежує можливості передачі тактильної взаємодії з предметами. Розроблюваний у рамках кваліфікаційної роботи продукт «Last

Night Last Call» займає нішу, яка комбінує переваги кращих аналогів: тривимірне середовище з продуманою механікою змішування (у дусі Bartender: The Right Mix), глибокий наратив (як у VA-11 Hall-A та The Red Strings Club) і атмосферність (як у Coffee Talk).

1.3 Постановка завдання

На основі аналізу предметної області та проведеного порівняння існуючих аналогів, сформульовано постановку завдання кваліфікаційної роботи. Розроблюваний продукт має задовольнити такі вимоги:

1. Тривимірне середовище від першої особи, яке забезпечує інтуїтивну взаємодію гравця з інтерактивними об'єктами (пляшки, шейкер, келих, льодогенератор, кран з водою).
2. Механіка змішування коктейлів, що включає: наливання рідини з пляшки у келих або шейкер; візуалізацію рівня рідини з фізично коректною поведінкою при нахилах; плавне змішування кольорів відповідно до доданих інгредієнтів; додавання льоду; механіку збовтування (шейк) через відстеження руху миші.
3. Система обслуговування клієнтів, що передбачає: появу клієнтів на сцені за заданим сценарієм або випадковим чином; формування ними замовлення (набір інгредієнтів, наявність льоду, вимога збовтування); перевірку приготованого напою на відповідність замовленню; реакцію клієнта на результат обслуговування.
4. Діалогова система з підтримкою локалізації на двох мовах, що відповідає найбільшим мовним аудиторіям платформи Steam.
5. Нарративна складова у вигляді послідовності ігрових рівнів з різними локаціями та сюжетними подіями.
6. Модульна архітектура, побудована на принципах об'єктно-орієнтованого проєктування з застосуванням архітектурних паттернів.

7. Реалізація на платформі Unity версії 6 або вище з використанням мови програмування C# та графічного конвеєру URP.
8. Готова збірка для операційної системи Windows з можливістю запуску без встановлення додаткового програмного забезпечення.

Відповідно до цих вимог було розпочато проєктування архітектури та розробку відеогри, яка отримала робочу назву «Last Night Last Call».

1.5 Висновки до розділу 1

1. Проведено огляд літературних джерел, присвячених геймдизайну, розробці відеоігор на платформі Unity, архітектурним патернам у ігровому програмуванні та процедурній графіці. Встановлено, що обрана тематика потребує комплексного підходу та володіння знаннями з кількох суміжних галузей.
2. Здійснено детальний аналіз чотирьох провідних аналогів у жанрі симулятор бармена: VA-11 Hall-A, Coffee Talk, The Red Strings Club та Bartender: The Right Mix. Для кожної гри визначено її сильні та слабкі сторони, ключові механіки та показники комерційного успіху. Встановлено, що сукупний попит на ігри цього сегменту є стабільно високим.
3. Складено порівняльну таблицю аналогів, на основі якої обґрунтовано нішу розроблюваного продукту. Показано, що «Last Night Last Call» пропонує комбінацію переваг кращих аналогів: тривимірне середовище з розвинутою фізичною механікою, глибокий наратив і атмосферний сетинг.
4. На основі проведеного аналізу сформульовано постановку завдання кваліфікаційної роботи, визначено функціональні вимоги до розроблюваного продукту.

2 ВИМОГИ ДО ОТОЧЕННЯ

2.1 Аналіз сучасних технологій для розробки відеоігор

Розробка сучасної тривимірної відеогри є комплексним процесом, що вимагає використання низки спеціалізованих технологій. Вибір технологічного стеку безпосередньо впливає на продуктивність розроблюваного продукту, зручність процесу розробки та час виходу продукту на ринок. У цьому розділі наведено аналіз технологій, обраних для реалізації відеогри «Last Night Last Call».

2.1.1 Ігровий рушій Unity

Unity – це міжплатформовий ігровий рушій, розроблений компанією Unity Technologies. Рушій підтримує створення 2D- та 3D-ігор, інтерактивних додатків, симуляцій та додатків віртуальної й доповненої реальності. Станом на 2026 рік Unity є одним з найпопулярніших ігрових рушіїв у світі, на якому створено як значну частку інді-ігор, так і багато AAA-проєктів.

Переваги Unity, що обумовили його вибір для реалізації кваліфікаційної роботи:

- Безкоштовна ліцензія Unity Personal для індивідуальних розробників та невеликих команд, чий річний дохід не перевищує встановлений поріг. 2. Компонентно-орієнтована архітектура, яка дозволяє гнучко комбінувати функціональність через додавання та налаштування компонентів до ігрових об'єктів.
- Мова програмування C# – сучасна, добре задокументована мова з розвинутою стандартною бібліотекою та великою спільнотою розробників [13].
- Вбудована підтримка фізичного рушія PhysX від NVIDIA, що забезпечує реалістичну симуляцію зіткнень, гравітації та інших фізичних процесів.

- Підтримка експорту на понад 25 платформ: Windows, macOS, Linux, iOS, Android, WebGL, консолі PlayStation, Xbox, Nintendo Switch тощо.
- Інтегрований редактор з візуальними інструментами роботи зі сценою, анімаціями, шейдерами, системою часток.
- Потужна екосистема додатків та ресурсів через Unity Asset Store. 8. Велика спільнота розробників, численні навчальні матеріали, офіційна документація та форуми.

Для реалізації проєкту була обрана версія Unity 6 з графічним конвеєром URP, що забезпечує баланс між якістю графіки та продуктивністю.

2.1.2 Графічний конвеєр URP

URP – це універсальний графічний конвеєр, розроблений компанією Unity Technologies як сучасна заміна застарілому Built-in Render Pipeline [14]. URP забезпечує оптимізоване відображення графіки на широкому спектрі пристроїв – від мобільних телефонів до високопродуктивних ігрових комп'ютерів.

Ключові особливості URP, які вплинули на його вибір для проєкту:

- Scriptable Render Pipeline: можливість налаштування процесу рендерингу через код, що дозволяє адаптувати графічний конвеєр під специфічні потреби проєкту.
- Shader Graph: візуальний редактор шейдерів, який дозволяє створювати складні графічні ефекти без написання коду шейдерів вручну [15]. Shader Graph було використано для створення ефектів неонової підсвітки, відображень на мокрому асфальті та спеціальних шейдерів рідини.
- Оптимізована система освітлення з підтримкою реального часу, попередньо розрахованого освітлення та змішаного освітлення.
- Підтримка post-processing ефектів: bloom, depth of field, motion blur, color grading, що дозволяють створити характерну для нічного атмосферного сетингу візуальну стилістику.

- . Висока продуктивність, яка дозволяє забезпечити стабільну частоту кадрів навіть на середніх за потужністю комп'ютерах.

2. 1.3 Додаткові інструменти розробки

Для реалізації проєкту, окрім Unity та мови C#, використано наступні додаткові інструменти та технології:

- Blender – безкоштовне програмне забезпечення з відкритим кодом для створення тривимірних моделей, анімації, рендерингу та симуляції [16]. У проєкті Blender використовувався для моделювання локацій (нічний клуб, VHS-магазин, кінотеатр), пляшок, келихів, меблів та інших об'єктів оточення. Моделі експортувались у формат FBX, який нативно підтримується Unity.
- Формат FBX (Filmbox) – це промисловий стандарт обміну тривимірними моделями, анімаціями та сценами між різними програмами тривимірної графіки. Формат підтримує збереження геометрії, матеріалів, UV координат, скелетних анімацій та інших даних.
- Visual Studio – інтегроване середовище розробки від Microsoft, яке тісно інтегрується з Unity та надає потужні інструменти для розробки скриптів на C#: підсвітка синтаксису, інтелектуальне автодоповнення IntelliSense, інтегрований налагоджувач, можливість прикріплення до процесу Unity для відлагодження у реальному часі.
- Git та системи контролю версій – для ведення історії змін проєкту, резервного копіювання та організації роботи використовувався Git. Для великих бінарних файлів (моделі, текстури, аудіофайли) застосовано Git LFS (Large File Storage).
- Audacity – безкоштовний аудіоредактор з відкритим кодом, використовувався для підготовки звукових ефектів: обрізання, нормалізації гучності, застосування ефектів (реверберація, відлуння, затухання).

2. 1.4 Система локалізації

Для забезпечення можливості гри різними мовами розроблено власну систему локалізації на основі текстових файлів. Кожна мова зберігається у окремому файлі, де діалоги подано у структурованому форматі «сторона_діалогу|текст|опціональний_скрипт». Такий підхід дозволяє легко додавати нові мови без зміни коду програми – достатньо створити новий файл з перекладами.

2.2 Аналіз вимог до користувачів

Важливою частиною проєктування програмного продукту є визначення цільової аудиторії та формулювання вимог до різних категорій користувачів. Для відеогри «Last Night Last Call» визначено дві основні ролі користувачів: гравець та розробник (тестувальник).

2.2.1 Гравець

Основною та найчисленнішою групою користувачів є гравці – особи, які запускають готовий ігровий додаток з метою отримати розважальний досвід.

Функції гравця:

- запуск гри з робочого столу або меню «Пуск» операційної системи;
- переміщення персонажа по ігровому середовищу за допомогою клавіатури та миші;
- взаємодія з інтерактивними об'єктами (пляшки, келихи, шейкер тощо);
- приготування коктейлів шляхом комбінування різних інгредієнтів; – спілкування з клієнтами через діалогову систему;
- проходження ігрових рівнів з розвитком сюжетної лінії;
- налаштування параметрів гри (гучність звуку, керування тощо).

Вимоги до гравця:

- базові навички користування персональним комп'ютером з операційною системою Windows;

- знання основ взаємодії з ігровими додатками (використання клавіатури та миші);
- вік 16+ років (з урахуванням тематики гри, яка передбачає приготування алкогольних коктейлів);

2.2.2 Розробник (тестувальник)

Другою важливою групою користувачів є розробники та тестувальники, які працюють з проєктом через середовище Unity Editor.

Функції розробника:

- перегляд та редагування сцен гри;
- створення та модифікація префабів і ігрових об'єктів;
- написання та відлагодження скриптів на мові C#;
- налаштування параметрів компонентів у Inspector;
- тестування гри у редакторі у режимі Play Mode;
- створення білду гри для цільової платформи (Windows);
- виявлення та виправлення дефектів програмного продукту.

Вимоги до розробника:

- знання мови програмування C# на середньому або вищому рівні;
- досвід роботи з середовищем Unity;
- розуміння принципів об'єктно-орієнтованого програмування;
- базові знання тривимірної математики (вектори, матриці, кватерніони);
- здатність до самостійного вивчення документації та використання ресурсів спільноти розробників.

2.3 Аналіз вимог до апаратного та програмного забезпечення

Так як розроблюваний продукт є тривимірною відеогрою з освітленням реального часу, використанням пост-ефектів та процедурною деформацією мешів, він висуває певні вимоги до обчислювальних ресурсів пристрою, на

якому запускається. Мінімальні характеристики визначено з урахуванням статистики апаратного забезпечення користувачів Steam [17].

2.3.1 Мінімальні вимоги до апаратного забезпечення гравця

Для комфортного запуску гри пристрій повинен відповідати таким мінімальним характеристикам:

1. Центральний процесор: Intel Core i3 4-го покоління або AMD Ryzen 3 1-го покоління з тактовою частотою не менше 2,5 ГГц.
2. Оперативна пам'ять: не менше 4 ГБ.
3. Графічний процесор: NVIDIA GeForce GTX 750 Ti, AMD Radeon R7 260X або інтегрована графіка з підтримкою DirectX 11.
4. Обсяг вільного місця на жорсткому диску: 2 ГБ.
5. Звукова карта: будь-яка сумісна з DirectX.
6. Роздільна здатність екрану: не менше 1280×720 точок.

2.3.2 Рекомендовані вимоги до апаратного забезпечення гравця

Для забезпечення оптимального ігрового досвіду з високою частотою кадрів та максимальними налаштуваннями графіки рекомендується пристрій з такими характеристиками:

1. Центральний процесор: Intel Core i3 4-го покоління або AMD Ryzen 3 1-го покоління з тактовою частотою не менше 2,5 ГГц.
2. Оперативна пам'ять: 8 ГБ або більше.
3. Графічний процесор: NVIDIA GeForce GTX 1060 6 ГБ або AMD Radeon RX 580 та новіші.
4. Обсяг вільного місця на SSD-накопичувачі: 2 ГБ.
5. Звукова карта: з підтримкою об'ємного звуку для більшого занурення в атмосферу.
6. Роздільна здатність екрану: 1920×1080 точок або вище.

2.3.3 Вимоги до програмного забезпечення гравця

Для запуску гри на пристрої користувача необхідно таке програмне забезпечення:

1. Операційна система Windows 10 версії 64-bit або новіша.
2. Встановлена версія .NET Framework 4.7.2 або новіша (зазвичай встановлена за замовчуванням).
3. Оновлені драйвери графічної карти з підтримкою DirectX 11 або вищої версії.

2.5 Висновки до розділу 2

1. Проведено аналіз сучасних технологій для розробки відеоігор. Обґрунтовано вибір ігрового рушія Unity, мови програмування C# та графічного конвеєру URP як базових технологій проєкту. Додатково описано допоміжні інструменти: Blender для моделювання, Visual Studio для розробки, Git для контролю версій, Audacity для обробки звуку.
2. Розроблено власну систему локалізації на основі текстових файлів, що дозволяє легко додавати нові мови без модифікації коду.
3. Визначено дві основні ролі користувачів системи: гравець та розробник (тестувальник). Для кожної ролі сформульовано функції, які вона виконує, та вимоги до її знань і навичок.
4. Визначено мінімальні та рекомендовані вимоги до апаратного й програмного забезпечення пристрою гравця, а також вимоги до середовища розробника. Зазначені характеристики забезпечують стабільну роботу продукту на широкому спектрі сучасних комп'ютерів.

3 РОЗРОБКА ВІДЕОГРИ «Last Night Last Call»

3.1 Опис предметної області

Предметна область цієї кваліфікаційної роботи охоплює процес розробки інтерактивної тривимірної відеогри в жанрі симулятор бармена з елементами нарративної пригодницької гри та психологічного хорору.

Гра поєднує кілька жанрових елементів. Основою ігрового процесу є симулятор: гравець виконує роль бармена на ім'я Джо, який обслуговує клієнтів у пересувному барі, що переміщується між різними локаціями нічного міста. Гравець самостійно готує коктейлі, комбінуючи рідкі інгредієнти, додаючи лід та застосовуючи механіку збовтування. Важливе значення має також нарративна складова: гру супроводжує сюжет, у якому головний герой переживає втрату дружини Сари та намагається впоратися з цим через роботу. Елементи психологічного хорору в грі реалізовано через механіку просвічування клієнтів крізь спеціальну пляшку, яка дозволяє гравцю побачити справжню природу деяких відвідувачів – істот, прихованих під людською зовнішністю. Теоретичні засади побудови відеоігрового хорору розкрито у роботі Б. Перрона [18]. Ця сюжетна лінія містить пряме посилання на класичний фільм «Invasion of the Body Snatchers» (постер якого розміщено в інтер'єрі фургона) та розкриває тему двоїстості реальності, характерну для стану емоційної травми.

Ігровий процес відбувається у трьох основних локаціях, кожна з яких має власну атмосферу та сюжетні події:

1. Нічний клуб – локація з неоновими вивісками, пальмами, музикою та строкатими клієнтами. Це перший рівень, на якому гравець знайомиться з основними механіками гри.
2. VHS-магазин – локація у вигляді атмосферного районного магазину з відеокасетами, які у грі є сюжетно важливими предметами.

3. Кінотеатр – фінальна локація, яка є кульмінацією сюжету гри. Переміщення між локаціями відбувається через механіку водіння фургону, під час якого відбувається зміна декорацій та завантаження нового рівня.

Фургон виконує функцію «хабу» – простору між ігровими сесіями обслуговування клієнтів.

Ігровий світ населяють різноманітні клієнти, кожен з яких має власну особистість та конкретне замовлення. Гравець може зустріти як випадкових відвідувачів, так і сюжетно важливих персонажів, що прогресують через декілька рівнів. Поведінка клієнтів реалізована на основі простих скінченних автоматів, теоретичні засади побудови яких розкрито у праці Міллінгтона та Фанджа [19].

3.2 Архітектура системи

Для реалізації відеогри «Last Night Last Call» було спроєктовано модульну архітектуру на основі компонентної моделі Unity з застосуванням низки класичних архітектурних паттернів. Архітектура забезпечує чітке розділення відповідальностей між компонентами системи, що спрощує підтримку коду та його подальший розвиток.

3.2.1 Загальна структура системи

Основу архітектури утворює центральний клас SYSTEM, реалізований за паттерном Service Locator. Цей клас надає єдину точку доступу до всіх підсистем гри: системи керування гравцем, системи клієнтів, системи діалогів, аудіосистеми, системи інтерактивних об'єктів, системи кольорів, системи переміщення між локаціями та інших. Такий підхід дозволяє уникнути тісного зв'язування між окремими компонентами гри. Діаграму класів поверхневих модулів системи наведено на рис. 6.

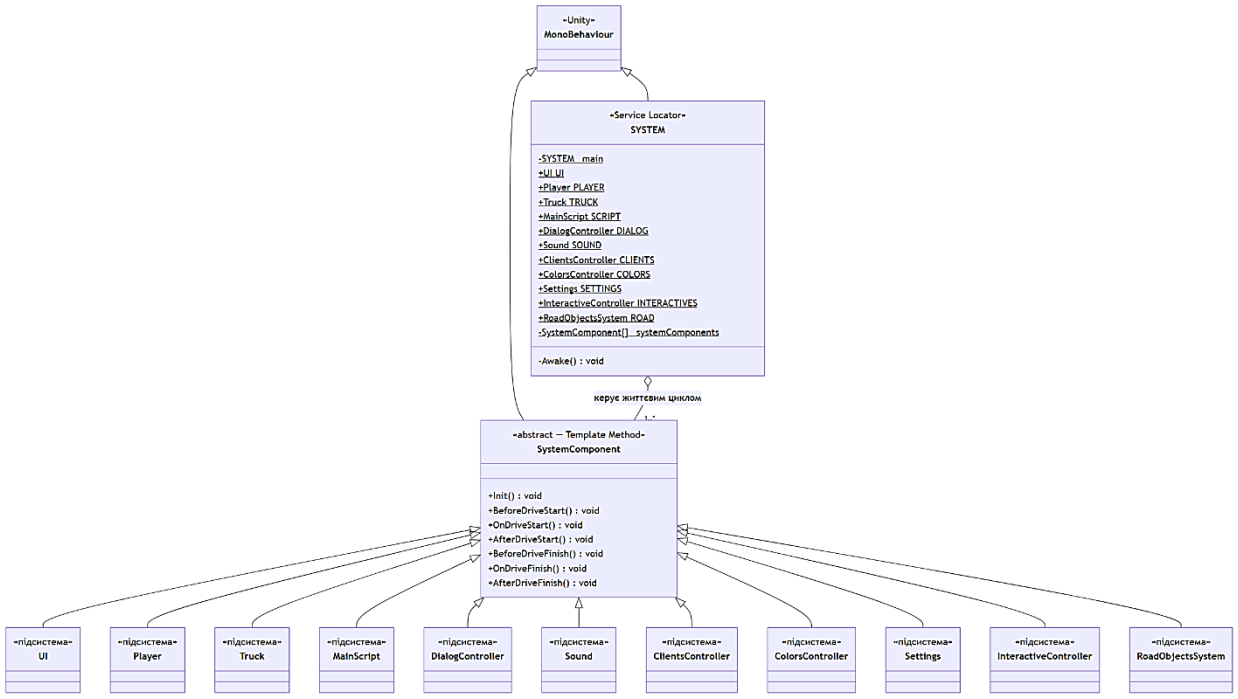


Рисунок *6 – Діаграма класів поверхневих модулів системи*

Лістинг 1 містить фрагмент коду класу SYSTEM, у якому декларуються статичні посилання на всі підсистеми гри.

Лістинг 1 – Фрагмент класу SYSTEM

```
public class SYSTEM : MonoBehaviour {
 private static SYSTEM _main;
 public static UI UI => _main._ui;
 public static Player PLAYER => _main._player;
 public static Truck TRUCK => _main._truck;
 public static MainScript SCRIPT => _main._script;
 public static DialogController DIALOG => _main._dialog;
public static Sound SOUND => _main._sound;
 public static ClientsController CLIENTS => _main._clients;
public static ColorsController COLORS => _main._colors;
public static Settings SETTINGS => _main._settings;
 public static InteractiveController INTERACTIVES =>
_main._interactives;
 public static RoadObjectsSystem ROAD => _main._road;
```

```
 private static SystemComponent[] _systemComponents;
 private void Awake() {
 _main = this;
 _systemComponents = new SystemComponent[] {
 _colors, _sound, _settings, _player, _truck,
 _road, _ui, _dialog, _clients, _interactives, _script,  };
 for (int i = 0; i < _systemComponents.Length; i++) {
_systemComponents[i].Init();
 }
 }
}
```

Кожна з підсистем є окремим компонентом (MonoBehaviour) [20], прикріпленим до спеціального ігрового об'єкта на сцені. Через статичні властивості класу SYSTEM будь-який скрипт у проєкті може отримати доступ до будь-якої підсистеми у формі SYSTEM.UI, SYSTEM.PLAYER, SYSTEM.SOUND тощо. Такий підхід значно спрощує написання коду та робить його більш зрозумілим.

3.2.2 Життєвий цикл компонентів системи

Усі підсистеми успадковуються від базового класу SystemComponent, який реалізує паттерн Template Method. Цей клас визначає набір віртуальних методів, які викликаються у певному порядку на різних етапах життєвого циклу гри: під час ініціалізації, перед початком переміщення між локаціями, після переміщення, перед завершенням локації, після завершення тощо.

Лістинг 2 наведено приклад визначення таких методів у базовому класі SystemComponent та їх перекриття у класі MainScript.

Лістинг 2 – Базовий клас SystemComponent та його нащадок MainScript

```
public abstract class SystemComponent : MonoBehaviour {
 public virtual void Init() { }
```

```
 public virtual void BeforeDriveStart() { }
 public virtual void OnDriveStart() { }
 public virtual void AfterDriveStart() { }
 public virtual void BeforeDriveFinish() { }
 public virtual void OnDriveFinish() { }
 public virtual void AfterDriveFinish() { }
}
public class MainScript : SystemComponent {
 public override void OnDriveStart() {
 base.OnDriveStart();
 if (_activeLoadedLevelIndex > -1)
 _levels[_activeLoadedLevelIndex].AfterFinish();
_activeLoadedLevelIndex = _activeLevelIndex;
 _levels[_activeLevelIndex].OnDriveStart();
 }
}
```

Диспетчеризацію подій життєвого циклу між усіма підсистемами виконує клас SYSTEM. На кожну подію він послідовно викликає відповідний метод у кожного зареєстрованого компонента. Це дозволяє кожній підсистемі належним чином підготуватися до зміни стану гри без необхідності знати про існування інших підсистем.

3.2.3 Ієрархія інтерактивних об'єктів

Для уніфікованого представлення всіх інтерактивних об'єктів у грі розроблено ієрархію класів з базовим класом Interactive на вершині. Ця ієрархія дозволяє гравцю однаково взаємодіяти з будь-яким інтерактивним об'єктом – пляшкою, келихом, шейкером, дверима, льодогенератором – через виклик спільних методів CanUse, Use, CanCombine, Combine.

Загальна ієрархія інтерактивних об'єктів гри показана на рис. 7.

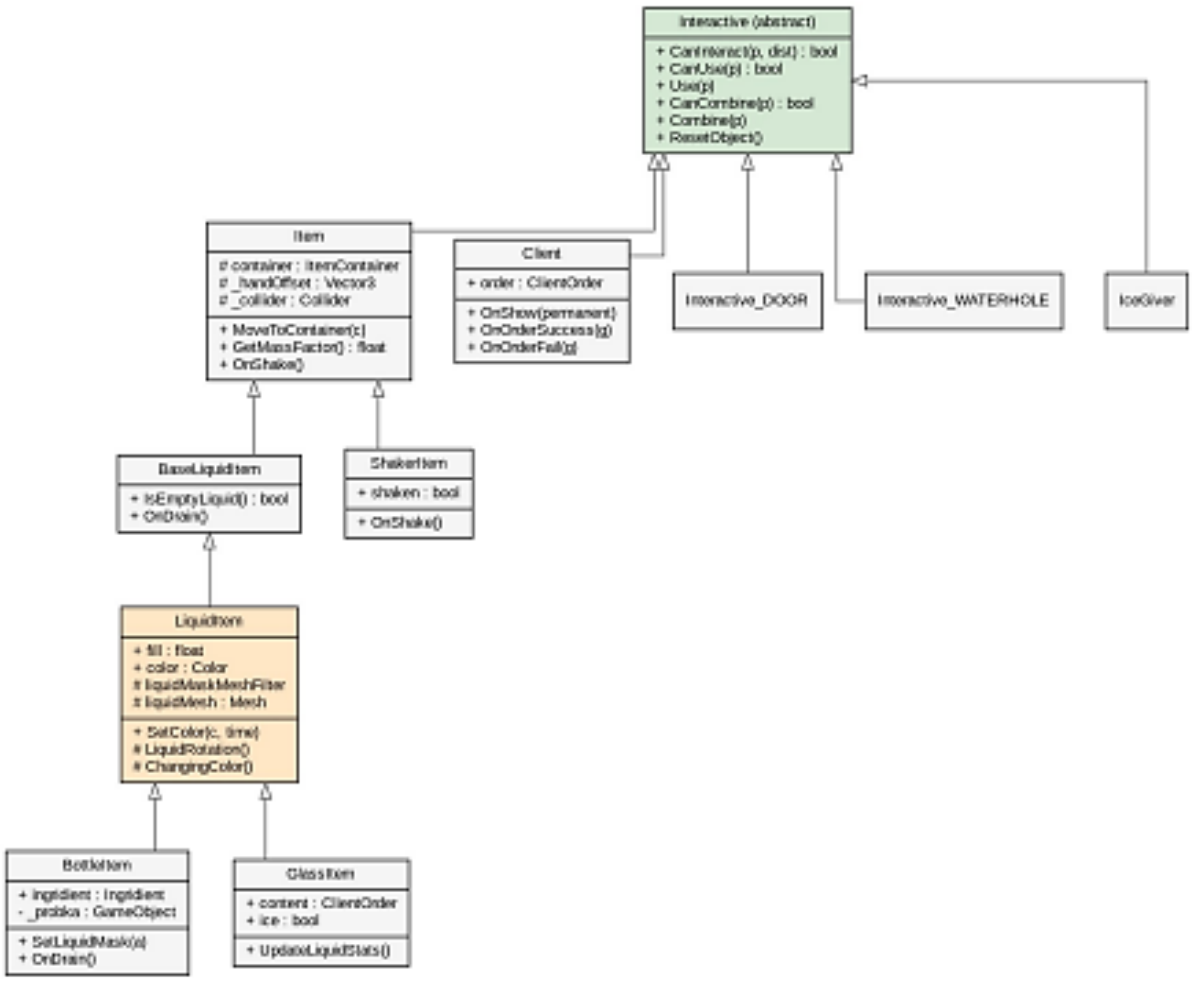


Рисунок *7 – Ієрархія класів інтерактивних об'єктів*

Особливо важливим є ланцюжок спадкування Interactive → Item → BaseLiquidItem → LiquidItem → BottleItem. Базовий клас Interactive описує загальний інтерфейс взаємодії. Клас Item додає поняття контейнера (предмет завжди знаходиться у якомусь контейнері – у руці гравця або на стійці). BaseLiquidItem та LiquidItem реалізують логіку роботи з рідиною – зберігання об'єму, кольору, ступеня заповнення. Клас BottleItem додає специфіку пляшки – закорковану горловину, конкретний тип інгредієнта.

Лістинг 3 містить фрагмент базового класу Item.

Лістинг 3 – Фрагмент класу Item

```
public class Item : Interactive {
```

```
 public ItemContainer container { get; protected set; }
 protected Vector3 _baseLocalPosition;
 protected Quaternion _baseLocalRotation;
 public override bool CanUse(Player player) {
 return player.grabber.itemInHand == null;
 }
 public override void Use(Player player) {
 player.grabber.PlaceItem(this,
player.grabber.handItemContainer);
 _collider.gameObject.layer = Layer.IgnoreRaycast;  }
 public virtual void MoveToContainer(ItemContainer
container, bool  sound = true) {
 this.container?.SetItem(null);
 this.container = container;
 transform.SetParent(container.transform);
 container.SetItem(this);
 // ...
 }
}
```

3.2.4 Система рівнів та скінченних автоматів

Ігрові рівні у проєкті реалізовано як окремі класи, що успадковуються від базового класу Level. Кожен рівень містить власний скінченний автомат (State Machine), який керує послідовністю сюжетних подій на цьому рівні.

Розглянемо, наприклад, рівень клубу (Level_1). Його скінченний автомат складається з таких станів: Start → InventoryChecked → FirstClient → FocusTold → DogMeeted → DogPrepared → DogSuccess → KassetTaken → Finish. Переходи між станами відбуваються у відповідь на певні події: гравець оглянув інвентар, обслужив перший замовлення, поговорив з персонажем Focus тощо.

Діаграма станів першого рівня гри наведена на рис. 8.

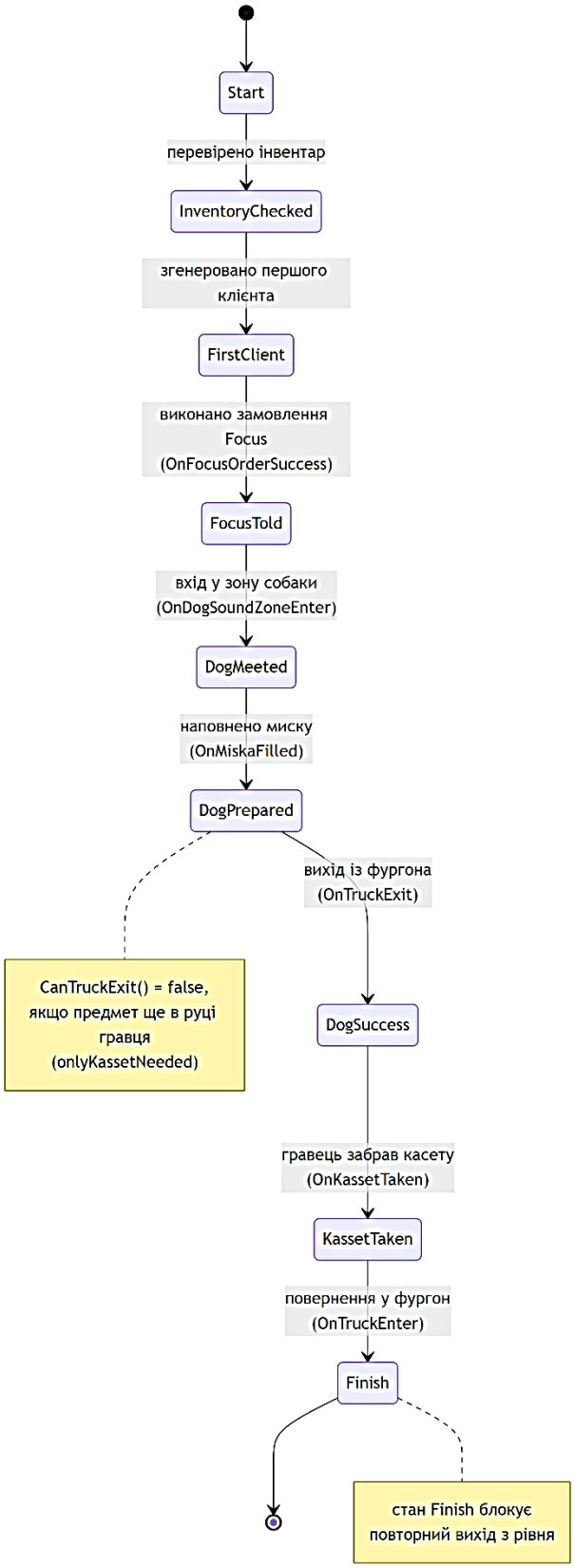


Рисунок *8 – Діаграма станів першого ігрового рівня*

Лістинг 4 містить визначення перерахування станів та фрагмент логіки переходів між ними на прикладі рівня клубу.

Лістинг 4 – Стани та логіка переходів рівня Level_1

```
public class Level_1 : Level {
 public enum Stage {
 Start,
 InventoryChecked,
 FirstClient,
 FocusTold,
 DogMeeted,
 DogPrepared,
 DogSuccess,
 KassetTaken,
 Finish,
 }
 public Stage stage;
 public void OnFocusOrderSuccess() {
 Dialog.focusTold.Run(null, () => {
 stage = Stage.FocusTold;
 SYSTEM.CLIENTS.HideClient();
 });
 }
 public override bool CanTruckExit() {
 if (!base.CanTruckExit()) return false;
 if (stage == Stage.DogPrepared) {
 if (SYSTEM.PLAYER.grabber.itemInHand) {
 Dialog.onlyKassetNeeded.Run();
 return false;
 }
 }
 if (stage == Stage.Finish) return false;
```

```
 return true;
 }
}
```

3.3 Функціональні вимоги системи

На основі аналізу предметної області та проведеного в попередньому розділі аналізу існуючих аналогів, сформульовано функціональні вимоги до розроблюваної системи:

1. Система повинна забезпечувати інтерактивну взаємодію гравця з тривимірним ігровим середовищем від першої особи.
2. Система повинна реалізовувати таку функціональність для роботи з ігровими об'єктами:
    - підіймання та перенесення об'єктів (пляшки, келихи, шейкер тощо);
    - розміщення об'єктів на поверхнях (барна стійка, полиці);
    - використання об'єктів (відкривання дверей, наливання рідини);
    - комбінування об'єктів (наливання рідини з пляшки у келих).
3. Система повинна реалізовувати механіку змішування коктейлів:
    - підтримку щонайменше п'яти базових типів інгредієнтів;
    - реалістичну фізичну поведінку рідини при нахилі контейнера;
    - плавне змішування кольорів при поєднанні інгредієнтів;
    - механіку додавання льоду до коктейлю;
    - механіку збовтування (шейк) через відстеження руху миші.
4. Система повинна реалізовувати систему обслуговування клієнтів:
    - появу клієнтів на сцені зі звуковою та анімаційною підтримкою;
    - формування клієнтом замовлення (набір інгредієнтів, лід, збовтування);

- автоматичну перевірку приготованого напою на відповідність замовленню;
- різні реакції клієнта на вдале чи невдале виконання замовлення.

5. Система повинна надавати діалогову систему з підтримкою:
   - множини реплік з розподілом на сторони діалогу (гравець/клієнт);
   - локалізації на декілька мов;
   - скриптових подій, пов'язаних з певними репліками;
   - можливості пропускання непропускних та пропускних реплік.
6. Система повинна підтримувати переміщення гравця між рівнями з анімацією переходу (водіння фургона).
7. Система повинна мати головне меню, з якого можна розпочати нову гру, продовжити гру, змінити налаштування або вийти з програми.
8. Система повинна забезпечувати збереження налаштувань гравця (гучність, мова) між сесіями.
9. Система повинна надавати аудіосупровід: фонову музику, звуки оточення, звукові ефекти дій гравця, голосові репліки клієнтів (за наявності).
10. Система повинна бути реалізована як готова збірка для операційної системи Windows 64-bit.

3.4 Функціонал системи

У цьому підрозділі детально розглянуто ключові функціональні можливості системи та наведено фрагменти коду, що реалізують ці можливості. Оскільки відеогра «Last Night Last Call» містить значну кількість скриптів, у роботі показано лише найбільш важливі для розуміння архітектури та алгоритмів частини коду.

3.4.1 Система взаємодії з інтерактивними об'єктами

Взаємодія гравця з ігровим світом реалізована через клас PlayerGrabber, який відповідає за виявлення інтерактивних об'єктів перед гравцем, підіймання та перенесення предметів, а також за виконання дій над ними. Клас працює у двох режимах: режим наведення (Hovering), коли гравець виявляє об'єкт у полі зору, та режим утримання (Grabbing), коли предмет підібрано у руку.

Для визначення інтерактивного об'єкта в полі зору гравця застосовано техніку трасування променя (Raycast). З позиції миші у тривимірний простір сцени випускається промінь, і якщо він перетинає колайдер об'єкта на шарі «Interactive», цей об'єкт отримує статус «наведеного». Після цього гравець може взаємодіяти з ним клацанням миші. Лістинг 5 містить фрагмент методу обробки наведення..

Лістинг 5 – Метод Hovering класу PlayerGrabber

```
public void Hovering() {
 if (!isGrabMode) {
 var ray =
_player.camera.ScreenPointToRay(Input.mousePosition);
RaycastHit hit;
 if (Physics.Raycast(ray, out hit, interactiveRayDistance,
Layer.Mask.interactives)) {
 if (_hoveredCollider != hit.collider) {
 _hoveredCollider = hit.collider;
 _hoveredInteractive =
hit.collider.GetComponentInParent<Interactive>();
 if (_hoveredInteractive == null) {
 SYSTEM.UI.SetTargetState(UI.TargetState.Light);  return;
 }
 }
 if (_hoveredInteractive) {
 if (_hoveredInteractive.CanInteract(_player,
interactiveDistance)
 && (_hoveredInteractive.CanUse(_player)  ||
_hoveredInteractive.CanCombine(_player))) {



 SYSTEM.UI.SetTargetState(UI.TargetState.Full);  } else {
```

```
 SYSTEM.UI.SetTargetState(UI.TargetState.Light);  }
 }
 return;
 }
 }
 if (_hoveredCollider) {
 _hoveredCollider = null;
 _hoveredInteractive = null;
 SYSTEM.UI.SetTargetState(UI.TargetState.Hide);
 }
}
```

Діаграма послідовності процесу приготування коктейлю, що демонструє взаємодію між класами PlayerGrabber, BottleItem, GlassItem та LiquidItem, наведена на рис. 9

.



Рисунок *9 – Діаграма послідовності «Приготування коктейлю»*

Коли гравець утримує предмет у руці та наводиться на інший інтерактивний об'єкт, система викликає метод CanCombine цього об'єкта. Якщо комбінування можливе (наприклад, гравець тримає пляшку і націлюється на келих), активується режим затиску. Після відпускання кнопки миші викликається метод Combine, який виконує передачу інгредієнта з пляшки у келих.

3.4.2 Механіка змішування коктейлів

Найскладнішою з технічної точки зору є реалізація механіки змішування рідин, яка включає два основних алгоритми: процедурну деформацію меша рідини при нахилі контейнера та плавне змішування кольорів при додаванні нового інгредієнта.

Процедурна деформація меша рідини реалізована у методі LiquidRotation класу LiquidItem. Основна ідея алгоритму полягає в обчисленні площини, яка відповідає поверхні рідини у світовому просторі, та коригуванні

позицій вершин меша таким чином, щоб вершини вище цієї площини «опускалися» до її рівня. Це забезпечує ефект того, що рідина завжди залишається горизонтальною, незалежно від нахилу контейнера. Лістинг 6 наведено повну реалізацію цього методу.

Лістинг 6 – Процедурна деформація меша рідини

```
protected virtual void LiquidRotation() {
 _liquidNormal =
transform.InverseTransformVector(Vector3.up);
_liquidNormal = _liquidNormal.ClampAngle(Vector3.up,
_liquidAngleMax);
 _liquidNormal = Quaternion.AngleAxis(90, Vector3.right) *
_liquidNormal;
 var verts = liquidMeshOrigin.vertices;
 _fillCurrent = Mathf.MoveTowards(_fillCurrent, fill,
_liquidSpeed * Time.deltaTime);
 var clampedFill = Mathf.Lerp(_liquidFillMinMax.x,
_liquidFillMinMax.y, _fillCurrent);
 _fillLocalPoint = new Vector3(0, 0, Mathf.Lerp(minHeight,
maxHeight, clampedFill));
 if (_fillCurrent < .1f) {
 for (int i = 0; i < verts.Length; i++) {
 verts[i] = _fillLocalPoint;
 }
 } else {
 for (int i = 0; i < verts.Length; i++) {
 var distance = SignedDistancePlanePoint(_liquidNormal,
_fillLocalPoint, verts[i]);
 if (distance > 0) {
 if (distance > errorHeight) {
 verts[i] = _fillLocalPoint;
 } else {
 verts[i].z = _fillLocalPoint.z;
 verts[i].z -=
SignedDistancePlanePoint(_liquidNormal, _fillLocalPoint,
verts[i]);  }
 }
 }
 }
 liquidMesh.vertices = verts;
 liquidMesh.RecalculateNormals();
}
public static float SignedDistancePlanePoint(Vector3
planeNormal,  Vector3 planePoint, Vector3 point) {
 return Vector3.Dot(planeNormal, (point - planePoint)); }
```

Алгоритм працює так: на кожному кадрі обчислюється нормаль до поверхні рідини шляхом перетворення глобального вектора «вгору» у локальну систему координат об'єкта. Щоб уникнути нефізичної поведінки при сильному нахилі контейнера, максимальний кут рідини обмежується значенням _liquidAngleMax (зазвичай 30 градусів). Після цього для кожної вершини меша обчислюється її знакова відстань до площини рідини за допомогою скалярного добутку. Вершини, що знаходяться вище поверхні, зміщуються до рівня поверхні.

Другий важливий алгоритм – плавне змішування кольорів. При доданні інгредієнта новий колір рідини у келиху не встановлюється різко, а інтерполюється від поточного кольору до цільового протягом заданого часу. Це реалізовано у методі ChangingColor, що викликається щокадрово, доки таймер _colorTime більше нуля. Лістинг 7 містить код цього методу.

Лістинг *7 – Плавна зміна кольору рідини*

```
protected virtual void ChangingColor() {
 _colorTime -= Time.deltaTime;
 var factor = _colorTime / _colorTimeLimit;
 tempColor = Color.Lerp(color, tempColor, factor);
 _liquidViewRenderer.material.SetColor("_BaseColor",
tempColor);
_liquidViewRenderer.material.SetColor("_SpecColor",
tempColor); }
public void SetColor(Color color, float time = 1f) {
 this.color = color;
 _colorTimeLimit = time;
 _colorTime = time;
}
```

3.4.3 Система обслуговування клієнтів

Система клієнтів побудована навколо двох ключових класів: Client (клієнт як ігровий об'єкт на сцені) та ClientOrder (опис замовлення клієнта). Кожен клієнт має поле order типу ClientOrder, у якому зберігається набір

інгредієнтів, що він хоче бачити у своєму коктейлі, а також додаткові вимоги щодо наявності льоду та збовтування.

Загальна структура класу Client та його нащадків показана на рис. 10.

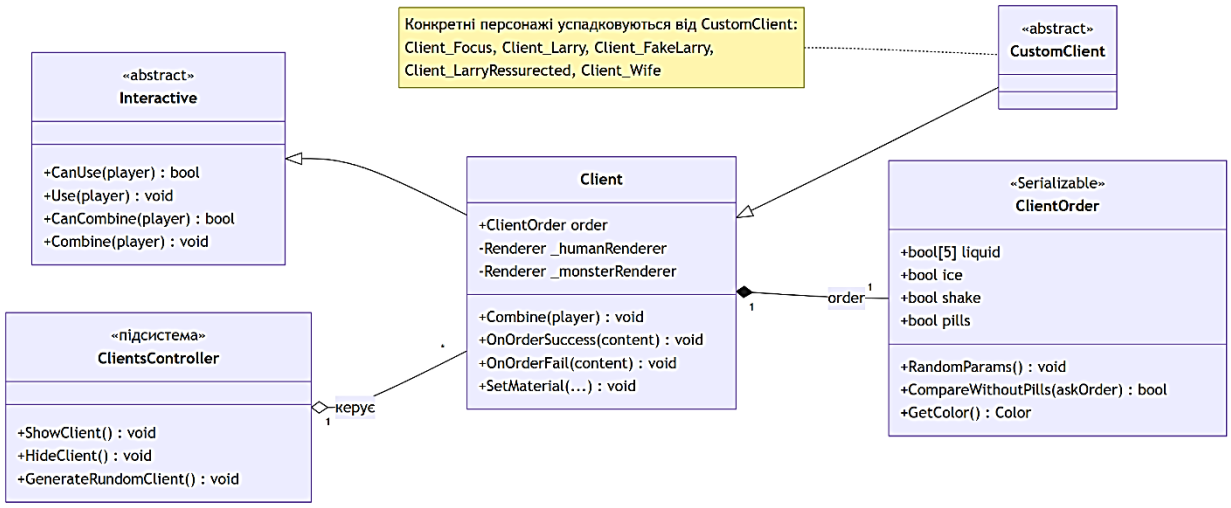


Рисунок *10 – Діаграма класів клієнтів та їх замовлень*

Замовлення представлене класом ClientOrder, що містить булевий масив бажаних інгредієнтів, а також прапорці ice (чи потрібен лід), shake (чи потрібно збовтування) та pills (чи було додано таблетки – спеціальна сюжетна механіка). Лістинг 8 наведено визначення класу ClientOrder разом з методом порівняння замовлення з приготованим напоєм.

Лістинг 8 – Клас ClientOrder з методами порівняння та обчислення кольору

```
[System.Serializable]
public class ClientOrder {
 public bool[] liquid = new bool[5];
 public bool ice;
 public bool pills;
 public bool shake;
 public void RandomParams() {
 var minIndex = Random.Range(0, 5);
 for (int i = 0; i < liquid.Length; i++) {
 if (minIndex == i) {
 liquid[i] = true;
 } else {
```

```
 liquid[i] = Random.Range(0, 100) < 50;
 }
 }
 ice = true;
 shake = true;
 }
 public bool CompareWithoutPills(ClientOrder askOrder) {
for (int i = 0; i < liquid.Length; i++) {
 if (liquid[i] != askOrder.liquid[i]) {
 return false;
 }
 }
 return ice == askOrder.ice && shake == askOrder.shake;  }
 public Color GetColor() {
 var color = new Color();



 var isFirstColor = true;
 for (int i = 0; i < liquid.Length; i++) {
 if (liquid[i]) {
 if (isFirstColor) {
 color = SYSTEM.COLORS.liquid[i];
 isFirstColor = false;
 } else {
 color = Color.Lerp(color,
SYSTEM.COLORS.liquid[i], .5f);
 }
 }
 }
 return color;
 }
}
```

Метод CompareWithoutPills виконує порівняння двох замовлень за всіма полями крім наявності таблеток. Якщо всі поля збігаються, метод повертає true, що означає – напій приготовано правильно. У іншому випадку клієнт висловлює незадоволення та чекає на нову спробу.

Діаграма послідовності процесу обслуговування клієнта наведена на рис. 11.

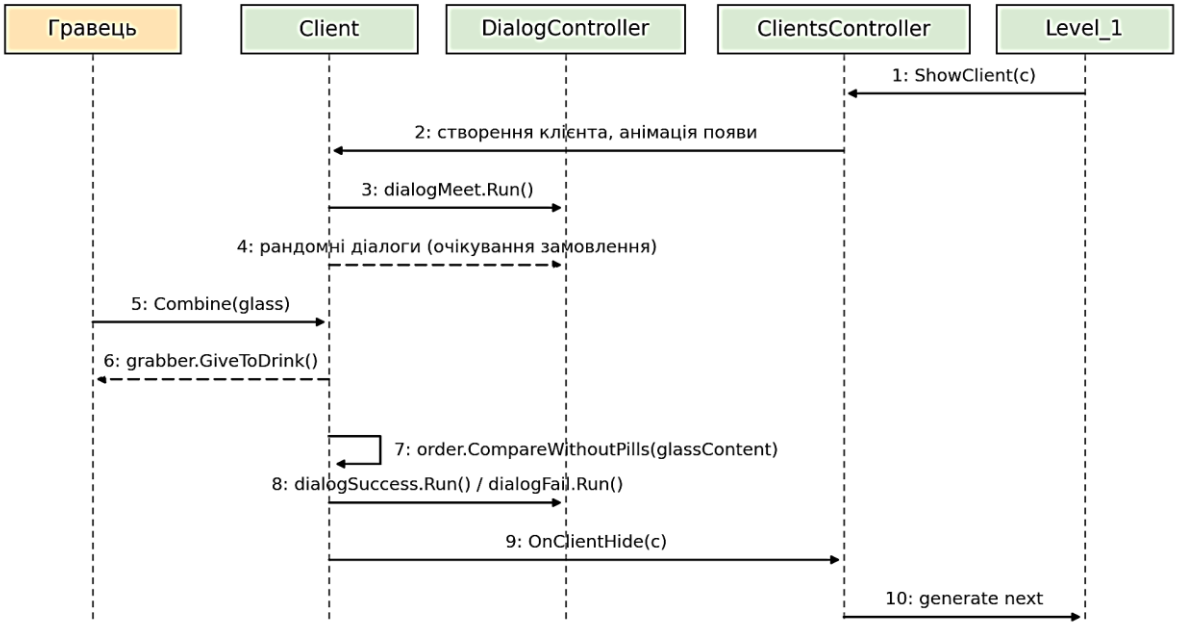


Рисунок *11 – Діаграма послідовності «Обслуговування клієнта»*

Процес обслуговування клієнта починається з методу ShowClient контролера ClientsController, який створює клієнта на сцені та запускає його анімацію появи. Після завершення анімації викликається вітальний діалог, після якого клієнт чекає на замовлення. Гравець готує напій і передає його клієнту, після чого викликається метод Combine класу Client, що порівнює вміст келиха із замовленням. Лістинг 9 демонструє ключовий метод Combine клієнта, який реалізує цю логіку.

Лістинг 9 – Метод Combine класу Client

```
public override void Combine(Player player) {
 var glass = (GlassItem)player.grabber.itemInHand;
 if (SYSTEM.SCRIPT.activeLevel is Level_1) {
 if (glass.content.ice == false) {
 Dialog.iceNeeded.Run();
 return;



 }
 if (glass.content.shake == false) {
 Dialog.shakeNeeded.Run();
 return;
```

```
 }
 }
 player.grabber.GiveToDrink(transform);
 var glassContent = glass.content;
 if (order.CompareWithoutPills(glassContent)) {
 this.WaitForSeconds(2f, () => {
 OnOrderSuccess(glassContent);
 });
 glass.content = new ClientOrder();
 glass.UpdateLiquidStats();
 } else {
 this.WaitForSeconds(2f, () => {
 OnOrderFail(glassContent);
 });
 }
}
```

3.4.4 Діалогова система та локалізація

Діалогова система в грі реалізована через клас DialogPreset, який представляє окремий блок діалогу, що складається з послідовності реплік (DialogPhrase). Кожна репліка містить ідентифікатор сторони (гравець, клієнт, оповідач), текст та опціональний номер скрипта, який запускається при показі цієї репліки (наприклад, поворот камери до певного об'єкта).

Локалізація реалізована через текстові файли у директорії locale: файл en.txt для англійської мови та ru.txt – для російської. Кожна репліка записується у форматі «сторона|текст|скрипт», а блоки діалогів розділені порожніми рядками. Під час ініціалізації гри відповідний файл зчитується та парситься методом ReloadPhrases. Лістинг 10 наведено цей метод разом з допоміжним класом DialogPhrase.

Лістинг 10 – Класи DialogPreset та DialogPhrase

```
public class DialogPreset {
 public int phraseIndex;
 public DialogPhrase[] phrases;
 public void ReloadPhrases(string[] lines, ref int
startIndex) {  while (lines[startIndex] == string.Empty) {
 startIndex++;
 }
 int length = 1;
```

```
 if (lines.Length > startIndex + length) {


 while (lines[startIndex + length] != string.Empty) {
length++;
 if (lines.Length == startIndex + length) {  length--;
 break;
 }
 }
 }
 int index = 0;
 phrases = new DialogPhrase[length];
 for (int i = startIndex; i < startIndex + length; i++) {
phrases[index] = new DialogPhrase(lines[i]);
 index++;
 }
 startIndex += length;
 }
 public void Run(Action<DialogPreset> callback = null,
Action  endCallback = null) {
 SYSTEM.DIALOG.Run(this, callback, endCallback);
 }
}
public class DialogPhrase {
 public int side;
 public string text;
 public int script;
 public bool canSkip = true;
 public DialogPhrase(string localeLine) {
 var substr = localeLine.Split('|');
 side = int.Parse(substr[0]);
 text = substr[1];
 script = (substr.Length > 2) ? int.Parse(substr[2]) : -1;
}
}
```

Такий формат зберігання діалогів дозволяє легко додавати нові мови локалізації без модифікації коду програми – достатньо створити новий файл перекладу з тією ж структурою. Гравець може обрати мову в налаштуваннях гри, після чого система автоматично перезавантажить всі діалоги.

3.4.5 Механіка шейкера (збовтування)

Однією з оригінальних механік гри є збовтування коктейлю у шейкері або безпосередньо у келиху. Ця механіка реалізована через відстеження вертикальних рухів миші: якщо гравець швидко переміщує мишу вгору-вниз, гра інтерпретує це як спробу збовтати напій.

Алгоритм роботи механіки наведено у лістингу 11.

Лістинг 11 – Механіка відстеження руху шейкера

```
public void Grabing() {
 var mouseLocalPos = Input.mousePosition;
 mouseLocalPos.z = distanceFromCamera.z;
 mouseLocalPos =
_player.camera.ScreenToWorldPoint(mouseLocalPos);  var
containerPosition =
_handItemContainer.transform.position.y;
 if (_shakeLastUp) {
 if (containerPosition - mouseLocalPos.y > _shakeDistance)
{  _shakePosition = mouseLocalPos.y;
 _handItemContainer.OnShake();
 _shakeLastUp = false;
 }
 } else {
 if (mouseLocalPos.y - containerPosition > _shakeDistance)
{  _shakePosition = mouseLocalPos.y;
 _handItemContainer.OnShake();
 _shakeLastUp = true;
 }
 }
 _handItemContainer.MoveToPosition(mouseLocalPos);
}
```

Алгоритм зберігає останній напрямок руху (вгору чи вниз) у полі _shakeLastUp. Коли миша переміщується на задану відстань у протилежному напрямку порівняно з попереднім, система фіксує це як один «шейк-рух» та викликає подію OnShake. Після певної кількості таких рухів вміст контейнера позначається як збовтаний, що відображається у структурі ClientOrder через прапорець shake.

3.4.6 Система ігрових рівнів та переміщення

Переміщення між рівнями реалізовано через механіку водіння фургона. Гравець заходить до кабіни фургона, сідає на водійське сидіння та натискає кнопку «Їхати». Після цього викликається метод DriveStart класу MainScript, який запускає послідовність анімацій та подій: затемнення екрану, звук двигуна, очікування, освітлення екрану, завантаження нового рівня. Лістинг 12 наведено метод DriveStart.

Лістинг 12 – Метод DriveStart класу MainScript

```
public void DriveStart() {
 if (isDriving) {
 isDriving = false;
 SYSTEM.BeforeDriveStart();
 SYSTEM.OnDriveStart();
 SYSTEM.AfterDriveStart();
 isDriving = true;



 return;
 }
 isDriving = true;
 SYSTEM.BeforeDriveStart();
 SYSTEM.UI.FadeIn(() => {
 SYSTEM.OnDriveStart();
 SYSTEM.WaitForSeconds(3f, () => {
 SYSTEM.UI.FadeOut(() => {
 SYSTEM.AfterDriveStart();
 });
 });
 });
}
```

Важливою особливістю архітектури є те, що сам процес переміщення розподілено на послідовні етапи (Before → On → After), кожен з яких диспетчеризується до всіх підсистем через паттерн Template Method. Це дозволяє кожній підсистемі виконати необхідні підготовчі та завершальні дії самостійно: наприклад, система клієнтів приховує активного клієнта перед початком переміщення, а система UI затемнює екран.

3.4.7 Механіка просвічування клієнтів крізь пляшку

Однією з ключових особливостей відеогри «Last Night Last Call», яка вирізняє її з-поміж інших симуляторів бармена, є механіка просвічування клієнтів крізь спеціальну пляшку. Ця механіка поєднує ігрову функцію з нарративною та виконує роль елемента психологічного хорору, що надає грі унікальної атмосфери та поглиблює сюжетне наповнення.

Сюжетно механіка побудована на тому, що головний герой Джо, який переживає втрату дружини Сари, виявляє серед клієнтів бару істот, прихованих під людською зовнішністю. Відсилання до класичного фільму «Invasion of the Body Snatchers», плакат якого розміщено в фургоні бармена, підкреслює мотив двоїстості реальності та параноїдального сприйняття світу, характерний для стану емоційної травми. Гравець отримує спеціальну пляшку з позначкою «Look through», через яку він може побачити справжню сутність відвідувача.

З технічної точки зору кожен клієнт має два візуальні представлення, кожне з яких відображається власним рендерером: людську форму (поле _humanRenderer) та форму істоти (поле _monsterRenderer з окремим матеріалом). Обидва представлення налаштовуються під час генерації клієнта: контролер ClientsController методом SetMaterial призначає кожному рендереру випадковий матеріал із заздалегідь підготовлених наборів randomClients та randomMonsters. Завдяки цьому кожен відвідувач отримує унікальну зовнішність як у людській, так і у прихованій формі.

Форма істоти проявляється лише при погляді крізь спеціальну пляшку з позначкою «Look through»: меш істоти винесено в окремий шар рендерингу зі спеціальним матеріалом, який стає помітним саме крізь скло цієї пляшки, тоді як у звичайному погляді залишається невидимим. Такий підхід реалізує ефект просвічування засобами графічного конвеєра URP, без додаткової ігрової логіки. Призначення матеріалів обом представленням клієнта виконує метод SetMaterial класу Client, наведений у лістингу 13.

Лістинг 13 містить фрагмент класу Client з двома рендерерами та методом SetMaterial, а також метод GenerateRundomClient контролера ClientsController, який призначає матеріали обом представленням клієнта.

Лістинг 13 – Налаштування двох візуальних форм клієнта (метод SetMaterial)

```
public class Client : Interactive {
    [SerializeField] protected Renderer _monsterRenderer;
    [SerializeField] protected Renderer _humanRenderer;

    public void SetMaterial(Material material, bool
monster) {
        if (monster) {
            _monsterRenderer.material = material;
        } else {
            _humanRenderer.material = material;
        }
    }
}

public class ClientsController : SystemComponent {
    public Material[] randomClients;
    public Material[] randomMonsters;

    public Client GenerateRundomClient() {
        var client = Instantiate(clientPrefab, transform);
        client.SetMaterial(
            randomClients.GetRundom(ref
_lastRandomClientIndex), false);
        client.SetMaterial(
            randomMonsters.GetRundom(ref
_lastRandomMonsterIndex), true);
        return client;
    }
}
```

Окрім візуальної складової, механіка впливає на сюжетні події: якщо гравець ідентифікує істоту та обслуговує її особливим чином (наприклад, додає у коктейль спеціальну таблетку замість звичайного інгредієнта), це призводить до альтернативного розвитку подій на рівні. Таким чином механіка

просвічування виконує не лише естетичну, а й суто ігрову функцію, впливаючи на результат проходження.

Цікавим технічним аспектом є те, що обидві візуальні форми клієнта — людська та монстроподібна — існують одночасно як окремі меші з власними рендерерами (_humanRenderer та _monsterRenderer). Метод SetMaterial призначає кожному рендереру відповідний матеріал, а перемикання між формами реалізується суто засобами графічного конвеєра URP, без зміни ігрової логіки чи повторного створення об'єктів, що дозволяє зберегти продуктивність.

3.4.8 Аудіосистема

Звуковий супровід є одним з найважливіших засобів створення атмосфери в іграх жанру симулятор та психологічного хорору, що детально обґрунтовано у фундаментальній праці К. Коллінз [21]. Технічна реалізація аудіо в середовищі Unity описана в офіційній документації [22]. У відеогрі «Last Night Last Call» аудіосистема реалізована як окрема підсистема Sound, що успадковується від базового класу SystemComponent та підпорядковується загальній архітектурі життєвого циклу. Це забезпечує уніфіковане управління звуком на всіх етапах роботи гри: ініціалізації, переходах між рівнями, в момент паузи тощо.

Гучність звуку розподілено на окремі канали аудіомікшера (AudioMixer): загальний (Master), музика (Music) та звукові ефекти (SFX). Метод SetVolume класу Sound переводить лінійне значення повзунка налаштувань у децибели за логарифмічною шкалою (Mathf.Log10) і застосовує його до відповідного параметра мікшера. Завдяки цьому гравець може незалежно регулювати гучність музики, ефектів та загальну гучність у меню налаштувань, що є стандартною практикою сучасних ігор.

Короткі звукові ефекти відтворюються методом PlayInstance: для кожного звуку у точці події створюється тимчасовий екземпляр AudioSource з префаба, який після завершення відтворення автоматично знищується

корутиною IEPlayInstance. Такий підхід дозволяє відтворювати декілька звуків одночасно та розміщувати їх у тривимірному просторі (просторовий звук через spatialBlend), що підсилює відчуття присутності у замкненому просторі фургона при частому відтворенні звуків (наприклад, при наливанні рідини у склянку).

Окрім точкових ефектів, у грі застосовано просторову реверберацію через компонент AudioReverbZone (поле reverbZone класу Sound), яка вмикається та вимикається залежно від того, чи перебуває гравець усередині фургона. Для плавного завершення звуків, що відтворюються циклічно (наприклад, стуку у двері на рівні VHS-магазину), використовуються методи розширення FadeStop та FadePause, які поступово зменшують гучність джерела перед його зупинкою чи паузою.

Звукові ефекти гри включають: звуки наливання рідини різної інтенсивності залежно від об'єму, звук збовтування шейкера, звук падіння кубиків льоду, звуки кроків гравця по різних поверхнях, звук відкриття пляшки, звук двигуна фургона під час переміщення між локаціями, а також спеціальні звуки для механіки просвічування – шепіт та спотворення, що супроводжують появу форми істоти. Усі звукові файли оброблені в аудіоредакторі Audacity з нормалізацією гучності та застосуванням реверберації для відповідності акустиці замкненого простору фургона.

Лістинг 14 наведено фрагмент класу Sound з основними методами аудіосистеми.

Лістинг 14 – Фрагмент класу Sound

```
public class Sound : SystemComponent {
    [SerializeField] private AudioMixer _volumeMixer;
    [SerializeField] private AudioSource
_audioSourcePrefab;

    public ClipPreset stepWood;
    public ClipPreset dogAlert;
    public ClipPreset bottleProbka;
    public ClipPreset waterDrain;
    public ClipPreset waterShake1;
```

```
    public ClipPreset pills;

    public void SetVolume(string key, float value) {
        if (value > 0.0001f) {
            value = Mathf.Log10(value) * 20f;
        }
        _volumeMixer.SetFloat(key, value);
    }

    public void PlayInstance(ClipPreset preset, Vector3
point,
                             System.Action callback = null)
{
        var instance = Instantiate(_audioSourcePrefab,
point,
                                   Quaternion.identity);
        instance.Play(preset);
        StartCoroutine(IEPlayInstance(preset.clip.length,
                                      instance, callback));
    }

    IEnumerator IEPlayInstance(float delay,
                     AudioSource instance, System.Action
callback) {
        yield return new WaitForSeconds(delay);
        Destroy(instance.gameObject);
        callback?.Invoke();
    }
}

public static class AudioSourceExtension {
    public static void Play(this AudioSource source,
                     ClipPreset preset, System.Action
callback = null) {
        source.Stop();
        source.clip = preset.clip;
        source.volume = preset.volume;
        source.spatialBlend = preset._3d;
        source.loop = preset.loop;
        source.pitch = Random.Range(preset.pitchMin,
preset.pitchMax);
        source.Play();
    }
}
```

Система Sound також забезпечує збереження налаштувань гучності між сесіями гри: значення гучності зчитуються зі сховища PlayerPrefs під час

ініціалізації та зберігаються при зміні. Це забезпечує комфортний досвід користувача, оскільки налаштовані одного разу параметри звуку застосовуються автоматично при кожному наступному запуску.

3.4.9 Система налаштувань та збереження стану

Відповідно до сформульованих функціональних вимог, система повинна забезпечувати збереження налаштувань гравця між сесіями. Цю функцію реалізує клас Settings, який, як і інші підсистеми, успадковується від SystemComponent та керується через паттерн Service Locator.

Клас Settings зберігає такі параметри: загальна гучність (masterVolume), гучність музики (musicVolume), гучність ефектів (sfxVolume), поточна мова локалізації (locale) та чутливість миші (mouseSensivity). Зберігання виконується через стандартний механізм Unity PlayerPrefs, який записує дані у реєстр операційної системи Windows.

Такий підхід має перевагу простоти та надійності: PlayerPrefs автоматично обробляє питання серіалізації примітивних типів, коректно працює при оновленні версії гри та не потребує додаткової логіки ініціалізації при першому запуску – значення за замовчуванням задаються другим аргументом методів PlayerPrefs.GetFloat/GetString у методі Load. Лістинг 15 наведено фрагмент класу Settings.

Лістинг 15 – Фрагмент класу Settings

```
public class Settings : SystemComponent {
    public static float mouseSensivity;
    public static float masterVolume;
    public static float sfxVolume;
    public static float musicVolume;
    public static string locale;

    public const string VOLUME_MASTER = "volume-master";
    public const string VOLUME_SFX = "volume-sfx";
    public const string VOLUME_MUSIC = "volume-music";
    public const string MOUSE_SENS = "mouse-sens";
    public const string LOCALE = "locale";
```

```
    [SerializeField] private Slider _volumeMasterSlider;
    [SerializeField] private Dropdown _localeSelector;

    private string[] locales = new string[] { "en", "ru" };

    private void Load() {
        mouseSensivity = PlayerPrefs.GetFloat(MOUSE_SENS,
4f);
        masterVolume = PlayerPrefs.GetFloat(VOLUME_MASTER,
.5f);
        sfxVolume = PlayerPrefs.GetFloat(VOLUME_SFX, 1f);
        musicVolume = PlayerPrefs.GetFloat(VOLUME_MUSIC,
1f);
        locale = PlayerPrefs.GetString(LOCALE, locales[0]);
    }

    private void Save() {
        PlayerPrefs.SetFloat(MOUSE_SENS, mouseSensivity);
        PlayerPrefs.SetFloat(VOLUME_MASTER, masterVolume);
        PlayerPrefs.SetString(LOCALE, locale);
    }

    private void Apply() {
        SYSTEM.SOUND.SetVolume(VOLUME_MASTER,
masterVolume);
        SYSTEM.SOUND.SetVolume(VOLUME_SFX, sfxVolume);
        SYSTEM.SOUND.SetVolume(VOLUME_MUSIC, musicVolume);
    }
}
```

Важливою властивістю класу Settings є те, що зміна мови викликає метод Apply, який через SYSTEM.DIALOG.Init() повторно зчитує відповідний файл локалізації та оновлює всі репліки методом ReloadPhrases без необхідності перезапуску гри. Аналогічно, зміна параметрів гучності одразу застосовується до груп аудіомікшера через виклики SYSTEM.SOUND.SetVolume.

3.5 Вимоги до інтерфейсу

Інтерфейс користувача відеогри «Last Night Last Call» було спроєктовано з урахуванням вимог інтуїтивності, мінімалізму та відповідності жанру симулятора.

Загальні вимоги до інтерфейсу:

1. Інтерфейс повинен бути реалізований у вигляді накладеного на тривимірну сцену HUD (Heads-Up Display), а також вікон, що з'являються у потрібні моменти.
2. Інтерфейс повинен використовувати типографіку, яка відповідає стилю гри – ретро, з неоновим відтінком.
3. Інтерфейс повинен підтримувати локалізацію та динамічно відображати текст обраною мовою.
4. Керування грою повинно здійснюватися за допомогою клавіатури та миші.

Структура інтерфейсу містить такі елементи:

- Головне меню з кнопками «Почати нову гру», «Продовжити», «Налаштування», «Вийти».
- Прицільний маркер у центрі екрану, який змінює свій стан залежно від наявності інтерактивного об'єкта у полі зору (Hide – нема об'єкта, Light – є не-активний об'єкт, Full – є активний інтерактивний об'єкт). – Вікно діалогу, що відображається внизу екрану під час розмови з клієнтом або у сюжетних моментах.
- Меню паузи, що з'являється при натисненні клавіші Esc і містить кнопки «Продовжити», «Налаштування», «Вийти».
- Меню налаштувань з параметрами гучності (загальна, музика, ефекти) та мови інтерфейсу.

3.6 Реалізація і тестування

3.6.1 Реалізація

Фізичні характеристики поточної версії системи наведено нижче.

Обсяг коду:

- C# – близько 60 скриптів, близько 4000 рядків коду;
- Shader (HLSL/ShaderGraph) – 5 шейдерів;
- Локалізаційні файли – 2 мови, 201 рядок кожен.

Структурні складові проєкту:

- 4 ігрові рівні (головне меню, клуб, VHS-магазин, кінотеатр); – 5 типів інгредієнтів для коктейлів + лід + шейк;
- 6 типів клієнтів (базовий випадковий, Larry, FakeLarry, LarryRessurected, Wife, Focus) плюс CustomClient;
- 10+ інтерактивних об'єктів (пляшки, келих, шейкер, двері, кран, льодогенератор тощо);
- Близько 30 діалогових блоків та сюжетних вставок з репліками персонажів (201 рядок у кожному файлі локалізації).

3.6.2 Апробація

Для перевірки працездатності розробленої системи було проведено тестування на цільовій платформі (Windows 10 64-bit). У процесі тестування гра була запущена з білду, послідовно пройдено всі три ігрові рівні, проведено обслуговування різних типів клієнтів та перевірено коректність роботи всіх ігрових механік.

На рис. 12 зображено головне меню гри.

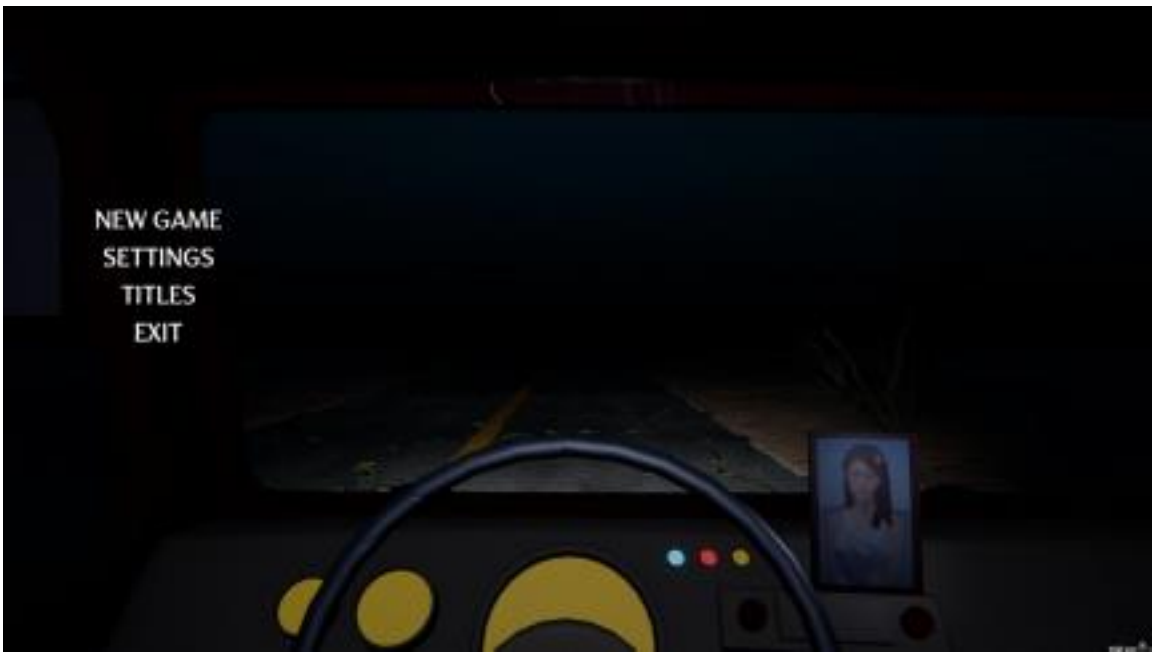


Рисунок *12 – Головне меню гри «Last Night Last Call»*

На рис. 13 зображено ігровий процес у локації клубу – барна стійка, пляшки з інгредієнтами, шейкер та льодогенератор.



Рисунок *13 – Ігрова локація «Клуб»*

На рис. 14 показано момент приготування коктейлю – гравець наливає один з інгредієнтів у келих з використанням пляшки.



Рисунок *14 – Процес приготування коктейлю*

На рис. 15 показано взаємодію з клієнтом: на екрані відображається діалогове вікно з репліками клієнта та гравця.



Рисунок *15 – Діалог з клієнтом*

На рис. 16 зображено другу ігрову локацію – VHS-магазин, який відвідує гравець після завершення подій у клубі.



Рисунок *16 – Ігрова локація «VHS-магазин»*

На рис. 17 зображено фінальну ігрову локацію – кінотеатр, на якому завершується сюжет гри.



Рисунок *17 – Ігрова локація «Кінотеатр»*

Продуктивність гри вимірювалась за допомогою вбудованого інструменту Unity Profiler та системного монітора операційної системи. Таблиця 2 відображає середні показники споживання ресурсів на тестовому комп'ютері (Intel Core i5-10400, 16 ГБ ОЗП, NVIDIA GeForce GTX 1660 Super).

Таблиця *2 – Продуктивність відеогри «Last Night Last Call»*

Стан гри	**CPU, %**	**RAM, МБ**	**GPU, %**	**FPS**
Головне меню	~8	~420	~15	~1800
Клуб (Level_1), очікування клієнта	~15	~780	~35	~280
Клуб, активне обслуговування	~18	~820	~42	~220
VHS-магазин (Level_2)	~14	~740	~30	~310
Кінотеатр (Level_Finish)	~12	~690	~25	~340
Перехід між рівнями (fade)	~10	~760	~20	~450

Як видно з таблиці 2, гра забезпечує стабільну продуктивність з частотою кадрів понад 200 FPS навіть на середньому за потужністю комп'ютері. Споживання оперативної пам'яті не перевищує 850 МБ, що дозволяє комфортно запускати гру на пристроях з 4 ГБ ОЗП.

У процесі тестування були виявлені та виправлені такі дефекти:

- некоректна поведінка рідини при різких рухах пляшки – вирішено шляхом обмеження максимального кута нахилу меша;
- проблема з повторним ініціалізуванням діалогів після смерті персонажа

– вирішено шляхом перевантаження стану рівня через метод ResetLevel;

- невідповідність кольору коктейлю при одночасному додаванні трьох і більше інгредієнтів – вирішено коригуванням коефіцієнта лінійної інтерполяції у методі GetColor;
- неправильне відтворення звукових ефектів у момент зміни рівнів – вирішено шляхом правильного розподілу звукових подій між етапами життєвого циклу.

3.7 Висновки до розділу 3

1. Описано архітектуру відеогри «Last Night Last Call», яка базується на компонентно-орієнтованій моделі Unity з застосуванням паттернів Service Locator (центральний клас SYSTEM) та Template Method (базовий клас SystemComponent). Розроблено діаграми класів, що ілюструють структуру системи та ієрархію інтерактивних об'єктів.
2. Сформульовано функціональні вимоги до системи та описано реалізацію ключових ігрових механік: системи взаємодії з предметами на основі трасування променя, механіки змішування коктейлів з процедурною деформацією меша рідини, системи обслуговування клієнтів із зіставленням замовлень, діалогової системи з підтримкою локалізації на двох мовах (російська та англійська), механіки збовтування, оригінальної механіки просвічування клієнтів крізь спеціальну пляшку як елемента психологічного хорору, аудіосистеми з пулом AudioSource та автоматичним переключенням музики при переходах між рівнями, а також системи налаштувань зі збереженням стану між сесіями гри.
3. Наведено повноцінні лістинги коду ключових компонентів системи (15 лістингів), що демонструють практичне застосування архітектурних рішень, алгоритмів тривимірної математики, роботу з пулом AudioSource та механізмом збереження стану PlayerPrefs.

4. Визначено вимоги до інтерфейсу користувача та описано структуру меню та HUD-елементів гри.
5. Проведено тестування розробленої системи на цільовій платформі. Продемонстровано коректну роботу всіх реалізованих механік, виміряно показники продуктивності, виявлено та виправлено низку дефектів.

ВИСНОВКИ

1. У процесі виконання кваліфікаційної роботи було проведено аналіз предметної області, що дозволив встановити високу актуальність розробки відеоігор у жанрі симулятор бармена. За даними аналітичної платформи Gamalytic, жанр Simulation на платформі Steam налічує понад 25 тисяч ігор із сукупним доходом, що перевищує 17,8 мільярда доларів США. Здійснено детальний порівняльний аналіз чотирьох провідних ігор-аналогів – VA-11 Hall-A: Cyberpunk Bartender Action, Coffee Talk, The Red Strings Club та Bartender: The Right Mix, – визначено їх сильні та слабкі сторони.

2. На основі проведеного аналізу обґрунтовано нішу розроблюваного продукту. Відеогра «Last Night Last Call» об'єднує переваги кращих аналогів: тривимірне середовище з тактильною взаємодією (у дусі Bartender: The Right Mix), глибокий наратив (як у VA-11 Hall-A та The Red Strings Club) та атмосферність нічного міста (як у Coffee Talk).

3. Сформульовано функціональні та нефункціональні вимоги до розроблюваної системи, обґрунтовано вибір технологічного стеку: ігровий рушій Unity версії 6, мова програмування C#, графічний конвеєр URP, додаткові інструменти Blender, Visual Studio, Git, Audacity. Визначено мінімальні та рекомендовані вимоги до апаратного забезпечення пристрою гравця.

4. Спроєктовано модульну архітектуру системи з застосуванням низки архітектурних паттернів: Service Locator (центральний клас SYSTEM), Template Method (базовий клас SystemComponent із життєвим циклом), Component (стандартна компонентна модель Unity). Розроблено ієрархію інтерактивних об'єктів з базовим класом Interactive та його нащадками Item, LiquidItem, BottleItem, GlassItem, ShakerItem, що забезпечує уніфікований інтерфейс взаємодії з різноманітними ігровими об'єктами.

5. Реалізовано ключові ігрові механіки:

- система взаємодії з інтерактивними об'єктами на основі трасування променя (Raycast);
- механіка змішування коктейлів з процедурною деформацією тривимірного меша рідини, побудованою на обчисленні знакової відстані від вершин меша до площини поверхні рідини;
- плавне змішування кольорів при доданні інгредієнтів через лінійну інтерполяцію;
- механіка збовтування (шейк) через відстеження вертикальних рухів миші; – система обслуговування клієнтів зі зіставленням замовлень та диференційованою реакцією клієнта;
- діалогова система з підтримкою локалізації на двох мовах;
- оригінальна механіка просвічування клієнтів крізь спеціальну пляшку, що виконує роль елемента психологічного хорору та впливає на сюжетні варіанти проходження;
- аудіосистема з розподілом звуків на категорії (музика, ефекти, інтерфейс), пулом AudioSource для ефективного відтворення та автоматичним переключенням фонової музики при переходах між рівнями;
- система налаштувань та збереження стану між сесіями гри на основі механізму PlayerPrefs;
- система переходів між ігровими рівнями з використанням скінченних автоматів для керування станом кожного рівня.

6. Розроблено відеогру «Last Night Last Call», яка містить три ігрові локації (нічний клуб, VHS-магазин, кінотеатр), шість типів клієнтів, понад 30 діалогових блоків та сюжетних вставок, п'ять базових інгредієнтів для коктейлів та оригінальну механіку просвічування монстрів. Загальний обсяг розробленого коду становить близько 60 скриптів C# (близько 4000 рядків), 5 шейдерів та 2 файли локалізації (російська та англійська, по 201 рядку кожен).

7. Проведено тестування розробленого продукту на цільовій платформі Windows 10 64-bit. Виміряно показники продуктивності: гра забезпечує стабільну частоту кадрів понад 200 FPS на середньому за потужністю комп'ютері, споживання оперативної пам'яті не перевищує 850 МБ. У процесі тестування виявлено та виправлено низку дефектів, що підтверджує працездатність системи.

8. Отримані результати підтверджують, що розроблений продукт задовольняє сформульованим на початку роботи вимогам. Відеогра «Last Night Last Call» може бути використана як референсне рішення для створення подібних ігор у жанрі симулятора бармена або опублікована на платформах цифрової дистрибуції. Реалізовані механіки (зокрема процедурна деформація меша рідини) можуть бути використані іншими розробниками як зразок технічного рішення.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Gamalytic: Video Game Analytics and Market Research. URL: https://gamalytic.com/ (дата звернення: 05.04.2026).

2. Schell J. The Art of Game Design: A Book of Lenses. 3rd ed. Boca Raton: CRC Press, 2019. 666 p.

3. Unity Manual. Unity Technologies. URL: https://docs.unity3d.com/Manual/index.html (дата звернення: 18.03.2026).

4. Gibson Bond J. Introduction to Game Design, Prototyping, and Development: From Concept to Playable Game with Unity and C#. 3rd ed. Boston: Addison-Wesley, 2022. 1040 p.

5. Nystrom R. Game Programming Patterns. Genever Benning, 2014. 354 p. URL: https://gameprogrammingpatterns.com/ (дата звернення: 15.03.2026).

6. Gamma E., Helm R., Johnson R., Vlissides J. Design Patterns: Elements of Reusable Object-Oriented Software. Boston: Addison-Wesley, 1994. 395 p.

7. Gregory J. Game Engine Architecture. 3rd ed. Boca Raton: CRC Press, 2018. 1240 p.

8. Lengyel E. Foundations of Game Engine Development, Volume 1: Mathematics. Lincoln: Terathon Software, 2016. 194 p.

9. Liquid Simulation in Unity Using Shader Graph. Gamasutra (Game Developer). URL: https://www.gamedeveloper.com/ (дата звернення: 28.03.2026).

10. Steam Store Page. VA-11 Hall-A: Cyberpunk Bartender Action. Valve Corporation. URL: https://store.steampowered.com/app/447530/ (дата звернення: 05.04.2026).

11. Steam Store Page. Coffee Talk. Valve Corporation. URL: https://store.steampowered.com/app/914800/ (дата звернення: 05.04.2026).

12. Steam Store Page. The Red Strings Club. Valve Corporation. URL: https://store.steampowered.com/app/589780/ (дата звернення: 05.04.2026).

13. C# Programming Guide. Microsoft Learn. URL: https://learn.microsoft.com/en-us/dotnet/csharp/programming-guide/ (дата звернення: 22.03.2026).

14. . Universal Render Pipeline Documentation. Unity Technologies. URL: https://docs.unity3d.com/Packages/com.unity.render-pipelines.universal@latest (дата звернення: 20.03.2026).

15. Unity Documentation. Shader Graph. Unity Technologies. URL: https://docs.unity3d.com/Packages/com.unity.shadergraph@latest (дата звернення: 25.03.2026).

16. . Blender Foundation. Blender Manual. URL: https://docs.blender.org/manual/en/latest/ (дата звернення: 25.03.2026).

17. Steam Hardware & Software Survey. Valve Corporation. URL: https://store.steampowered.com/hwsurvey/ (дата звернення: 10.04.2026).

18. Perron B. The World of Scary Video Games: A Study in Videoludic Horror. New York: Bloomsbury Academic, 2018. 336 p.

19. Millington I., Funge J. Artificial Intelligence for Games. 3rd ed. Boca Raton: CRC Press, 2019. 976 p.

20. Unity Scripting API. Unity Technologies. URL: https://docs.unity3d.com/ScriptReference/ (дата звернення: 18.03.2026).

21. Collins K. Game Sound: An Introduction to the History, Theory, and Practice of Video Game Music and Sound Design. Cambridge: MIT Press, 2008. 200 p.

22. Unity Documentation. Audio in Unity. Unity Technologies. URL: https://docs.unity3d.com/Manual/Audio.html (дата звернення: 01.04.2026).

Декларація

академічної доброчесності

здобувача ступеня вищої освіти ЗНУ

Я, Фролова Ірина Миколаївна, студентка 4 курсу, форми навчання денної, Інженерного навчально-наукового інституту ім. Ю.М. Потебні ЗНУ, спеціальність 121 Інженерія програмного забезпечення освітньої програми «Програмне забезпечення систем», адреса електронної пошти iriskafro123@gmail.com, – підтверджую, що написана мною кваліфікаційна робота на тему «Розробка інтерактивної відеогри жанру симулятор бармена з механікою змішування коктейлів на Unity» відповідає вимогам академічної доброчесності та не містить порушень, що визначені у ст.42 Закону України «Про освіту», зі змістом яких ознайомлена.

- заявляю, що надана мною для перевірки електронна версія роботи є ідентичною її друкованій версії;

- згодена на перевірку моєї роботи на відповідність критеріям академічної доброчесності у будь-який спосіб, у тому числі за допомогою інтернет-системи, а також на архівування моєї роботи в базі даних цієї системи.

Дата 06.06.2026 _____ Фролова Ірина Миколаївна
(студент)

Дата 06.06.2026 _____ Жуков Олександр Олексійович
(керівник)