МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ХЕРСОНСЬКИЙ ДЕРЖАВНИЙ УНІВЕРСИТЕТ
Факультет комп'ютерних наук, фізики та математики
Кафедра інформатики, програмної інженерії та економічної кібернетики

ГЕЙМІФІКОВАНИЙ ПІДХІД В ОСВІТІ: ТЕХНОЛОГІЇ СТВОРЕННЯ МОБІЛЬНИХ ДОДАТКІВ НАВЧАЛЬНОГО ПРИЗНАЧЕННЯ

Кваліфікаційна робота (проєкт)

на здобуття ступеня вищої освіти «магістр»

Виконав: студент 2 курсу
Спеціальності 122 Комп'ютерні науки
Освітньо-професійної програми
«Комп'ютерні науки» другого
(магістерського) рівня вищої освіти
Шапарьов Олексій Юрійович
Керівник: доктор педагогічних наук, професор Шерман Михайло Ісаакович
Рецензент: кандидатка педагогічних наук, доцентка Гончаренко Тетяна Леонідівна

Херсон – 2020

ЗМІСТ

ВСТУП

Актуальність:

Гейміфікація або ігрофікація розповсюджується в усі сфери життя. Перевагою гейміфікації освітнього процесу є щира зацікавленість абітурієнта, його залученість до навчання. Потрібно відзначити той факт, що гейміфікація – це не заглиблення у віртуальний світ, не ігри під час освітнього процесу, а допоміжний інструмент, який дозволяє підвищити пізнавальну активність та мотивацію. Ігри – це засоби виховання волі, що містять компонент розважального характеру, на відміну від традиційних форм навчання.

На жаль, достатньо часто виникає ситуація, коли в абітурієнта виникають сумніви щодо вибору майбутнього вищого навчального закладу. Також, останнім часом у закладах вищої освіти спостерігається невтішна тенденція, коли абітурієнти обирають для майбутнього навчання спеціальність, яка не вимагає складання екзамену (ЗНО) з математики або фізики.

Наслідком цього є зменшення контингенту студентів на факультетах природничо-математичного напрямку, аж до кризового рівня. З огляду на зазначене вище, проблема пошуку шляхів привернення студентів на ці спеціальності є актуальною. Тому одним із важливих напрямків діяльності університету є стимулювання мотивації абітурієнтів, що мають схильність та відповідну базову підготовку, щодо вибору місця для навчання та допомога у прийнятті рішення щодо вибору пріоритетного вищого навчального закладу в Україні.

Характерною ознакою нашого часу став стрімкий розвиток мобільних ігор, які день за днем набувають популярності. Розвиток індустрії мобільних ігор не стоїть на місці. День за днем простори Google Play Marker та Apps Store заповнюються різними цікавими додатками.

Щодня різні компанії випускають багато цікавих ігор, які своєю графікою та сюжетом заворожують та гіпнотизують гравців. А тому, кожен хто має смартфон може зіграти в різні ігри. Одні тренують розум та пам’ять, логіку та увагу, інші надають корисну інформацію.

У цьому зв’язку, на нашу думку, як один із способів формування позитивної мотивації майбутніх студентів щодо вибору певного напряму професійної підготовки та відповідного навчального закладу, є доцільним використання методів гейміфікації. Мобільний додаток "KSU_Inside" було розроблено за допомогою ігрового рушія Unity3D. Він покликаний ознайомити абітурієнтів із внутрішньою структурою Херсонського державного університету за допомогою методів гейміфікації, і, шляхом використання ігрових технологій, створювати привабливий імідж ХДУ та формувати позитивну мотивацію щодо перспектив вступу до закладу вищої освіти.

Об’єкт дослідження: ігрові мобільні додатки.

Предмет дослідження: технології проектування та розробки ігрових мобільних додатків.

Мета дослідження: спроектувати та розробити інформаційно-довідковий мобільний додаток “KSU_Inside”.

Завдання дослідження:

1. Уточнити на основі критичного аналізу фахових традиційних та електронних джерел дефініції «гейміфікація» та «гейміфікований підхід».
2. Визначити структуру та функціонал додатку.
3. На основі визначеного функціоналу спроектувати інформаційно-довідковий мобільний додаток “KSU_Inside”.
4. Обґрунтувати та обрати відповідно до структури та функціоналу програмне забезпечення.
5. Розробити інформаційно-довідковий мобільний додаток “KSU_Inside”.

Методи дослідження:

Для реалізації мети та вирішення поставлених завдань були використані як теоретичні, так і емпіричні методи дослідження.

Для з'ясування стану досліджуваної проблеми у сучасній теорії та практиці використовувались такі теоретичні методи дослідження:

- аналіз,
- синтез,
- порівняння,
- узагальнення,
- моделювання.

З-посеред емпіричних були використані такі методи, як дослідження ролі інформаційних технологій в освітньому середовищі, порівняння серверних мов програмування, порівняння систем керування вмістом, моделювання структури розроблюваної системи, опис результатів дослідження.

Наукова новизна дослідження:

1. Уточнено на основі критичного аналізу фахових традиційних та електронних джерел дефініції «гейміфікація» та «гейміфікований підхід».
2. Визначено структури та функціоналу розроблюваного мобільного інформаційно-довідкового додатку "KSU_Inside".

Практичне значення отриманих результатів:

1. На основі визначеного функціоналу спроектовано інформаційно-довідковий мобільний додаток "KSU_Inside".
2. Обґрунтовано та обрано, відповідно до структури та функціоналу, програмне забезпечення.
3. Розроблено інформаційно-довідковий мобільний додаток "KSU_Inside".

Апробація результатів роботи та публікації:

1. Шапарьов О.Ю. Гейміфікований підхід в освіті: технології створення мобільних додатків навчального призначення. Магістерські студії. Альманах. Вип. 19. – Херсон. ХДУ, 2019. С.693-696.
2. Sherman M., Samchynska Y., Shaparyov O. Information system of educational appointment. Тези доповідей IX Міжнародної науково-практичної конференції «Математика. Інформаційні технології. Освіта». Луцьк – Світязь, 1-3 червня 2020р. С.100-103.
3. Шапарьов О.Ю. Інформаційно-довідковий мобільний додаток “KSU_Inside”. Опубліковано в магазині додатків AppGallery. Режим доступу: https://appgallery.huawei.com/#/app/C103001865

Структура роботи. Наукова робота складається з наступних частин: вступу, трьох розділів, висновків, списку використаних джерел та додатків.

У першому розділі наукового проекту розглянуто категоріальний апарат гейміфікації та гейміфікований підхід в освіті, класифікацію та особливості мобільних ігор.

У другому розділі визначено структуру та функціонал мобільного додатку, продемонстровано модель мережевої взаємодії клієнта з сервером гри, спроектовано інтерфейс додатку.

У третьому розділі на основі порівняльного аналізу здійснюється вибір технологій розробки мобільних ігор, 3D моделювання об’єктів гри, а також описується процес розробки мобільного додатку «KSU_Inside».

РОЗДІЛ 1
ГЕЙМІФІКОВАНИЙ ПІДХІД ДО СТВОРЕННЯ МОБІЛЬНИХ ІГОР

1.1 Порівняльний аналіз підходів розробки мобільних ігор

Гейміфікація – підхід до навчання, який стає все більш поширеним та популярним в освіті, також являється альтернативою для багатьох існуючих методів навчання. Використання даного методу подовжує термін зацікавленості для вирішення проблеми, підвищує мотивацію та збільшує ймовірність досягнення своєї мети[43].

Таблиця 1.1

Принципи гейміфікації

Принципи гейміфікації	**Опис**
Мотивація	Найсильнішими мотиваторами до дії вважаються прагнення уникати дискомфорту та бажання задовольнитися. У першому випадку рекомендується якомога точніше визначити, що саме відчує та отримає людина, коли зуміє перемогти. Коли вона приміряє на себе роль переможця в грі, то захоче досягти того ж самого рівня і в реальному житті. А в другому - можна застосовувати будь-яку винагороду, від реального призу до завоювання визнання та поваги людей
Несподівані відкриття та заохочення	Цінується будь-який несподіваний контент: нові рівні, нова система нарахування бонусів, нові персонажі і т.д.

Продовження табл. 1.2

Принципи гейміфікації	**Опис**
Статус	Кожен гравець намагається бути кращим за інших, потребує того, щоб показати власну значущість. А для цього потрібно, щоб всі бачили його результати
Винагорода	Винагорода є основним принципом геймифікації. Адже без винагороди гравець не буде зацікавлений грою. Також винагорода має бути цікавою для гравця, бо інакше у гравця пропаде бажання грати

- **Unreal Engine** — ігровий рушій, який написаний на C++, дозволяє створювати ігри на більшість операційних систем та платформ : консолі Xbox, Mac OS та Mac OS X, Microsoft Windows, Linux, Wii, PlayStation 2 та PlayStation 3, Nintendo GameCube, Xbox 360, Dreamcast та PlayStation Portable. У березні 2010 роботу рушія було продемонстровано на комунікаторі Palm Pre, який базується на мобільній платформі webOS.

Рисунок 1.1 - Середовище розробки ігор – Unreal Engine

Для мережевої гри підтримуються технології GameSpy, Xbox Live та Windows Live, включаючи одночасно до 64 гравців.

Для описання логіки гри використовується C++ подібний UnrealScript. Всі елементи ігрового рушія представлено у вигляді об'єктів, які мають набір характеристик, та клас, що визначає характеристики, які доступні. Будь-який клас, у свою чергу, є «дочірнім» класом object. Серед основних класів та об'єктів можна виділити наступні:

Актор (actor) — базовий клас, який містить всі об'єкти, що мають відношення до ігрового процесу та мають просторові координати.

Пішак (pawn) — фізична модель гравця або об'єкта, який керований штучним інтелектом.

Світ, рівень (world, game level) — об'єкт, який характеризує загальні властивості «простору», наприклад, туман та сила тяжіння, у якому розташовані всі актори.

Топ 10 ігор створених за допомогою Unreal Engine 4:

1. Unreal Tournament
2. BioShock: Infinite
3. Mass Effect
4. Gears of War
5. Batman: Arkham Asylum
6. Mortal Kombat 11
7. Borderlands
8. XCOM: Enemy Unknown
9. Fortnite
10. A Way Out [1]

- **ShiVa3D —** тривимірний ігровий рушій, який має графічний редактор та призначений для створення ігор та додатків для мобільних пристроїв, консолей та веб.

Рисунок 1.2 - Середовище розробки ігор – ShiVa3D

За допомогою ShiVa3D можна розробляти ігри та програми для Android, Linux, Palm OS, Windows, iOS, Mac OS та Wii. Також є плагін для перегляду 3D прямо в браузері. ShiVa3D складається з чотирьох частин: Ігровий рушій, Редактор, Інструмент розробника (програма для складання проектів) і Сервер. Ігровий рушій Шиви базується на DirectX або Open GL графіці та фізиці ODE. Для програмування в основному використовується Lua, але можна писати оптимізувати скрипти на C ++. Редактор має 4 редакції: PLE (free), Basic (€ 169), Advanced (€ 1499), Educational (free). [7]

- **Unity —** багатоплатформовий інструмент для розробки дво- та тривимірних додатків та ігор, який працює на операційних системах Windows та OS X. Застосування, які створено за допомогою Unity працюють на системах Android, Windows, Linux, OS X, Apple iOS, а також на гральних консолях PlayStation 3, Wii та XBox 360.

Цей рушій задовольняє ряду поставлених до нього вимог, а саме:

– кінцевий продукт є мультимедійним 3D об'єктом, вбудованим в HTMLсторінку;

- кінцевий продукт є об'єктом високого рівня абстракції прототипів об'єктів;
- забезпечення високої якості графічного представлення інформації;
- бібліотека 3D об'єктів має можливість працювати з сучасними форматами тривимірної графіки - * .3ds, * .dae, * .fbx, * .flt;
- підтримка мови високого рівня (C#) для забезпечення ефективного процесу розробки;
- наявність ліцензії для вільного використання в некомерційних цілях;
- наявність редактора програмної і графічної розробки об'єктів;
- можливість підключення сторонніх бібліотек об'єктів (бібліотеки для обробки даних, веб-сервіси, драйвери баз даних і т.д.).

А ще Unity 3D:

- Широкий в розробці ігор та програм (друге місце в рейтингу за 2017 рік).
- Має велику область застосування: створення ігор та програм для пк, Android, iPhone, Xbox, та ін.
- Має велику популярність серед розробників та студентів.

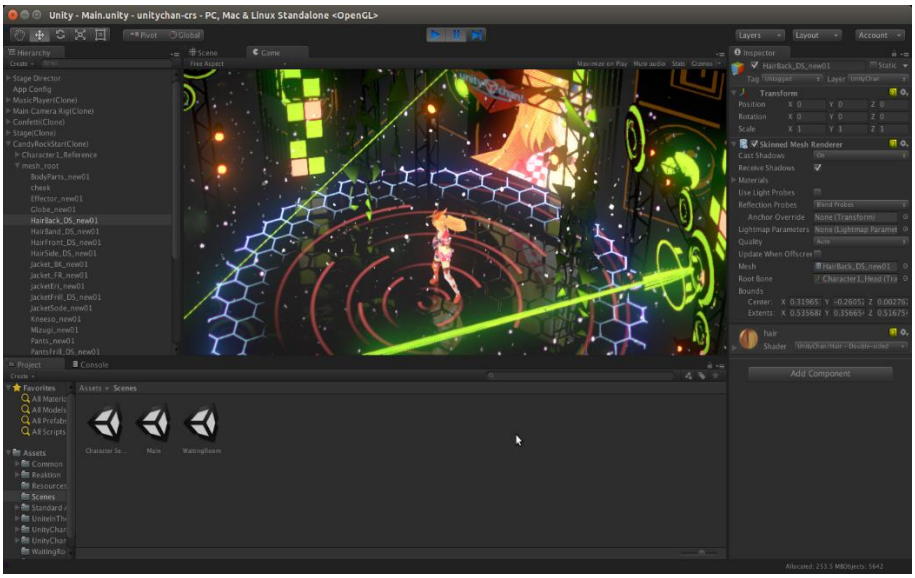


Рисунок 1.3 - Середовище розробки ігор – Unity

Топ 10 ігор створених за допомогою Unity 3D:

1. Endless Legend
2. Kerbal Space Program
3. Inside
4. Rust
5. Tyranny
6. Firewatch
7. Pokemon Go
8. Guns of Icarus Online
9. Sunless Sea
10. SuperHot [2]

Таблиця 1.2

Порівняльна характеристика ігрових рушіїв

Критерії/ Альтернативи	**VR**	**Зручність UI**	**Фізичний двигун**	**Мова програмування**	**Додаткові сервіси**	**Плат фор ми**	**WWW сервіси**
Unreal Engine	+(Google SDK)	3.5	+	C++, Blueprint system	Asset Store	Cross	+
Shiva 3D	Android VR	4	+	LUA	-	Cross	+
Unity	+(Cardboard SDK	5	+	C#/JS	UNET, CLOUD BUILD	Cross	UNET +.NET 3.5

Для розробки гри, використаємо такий ігровий рушій, як Unity 3D. Так як він зручний у використанні, не вимогливий до характеристик комп’ютера, на ньому можна програмувати такою мовою, як C#.

1.2 Класифікація мобільних ігор

Гра — це ідеалізована математична модель колективної поведінки кількох осіб (гравців), інтереси яких різні, що і породжує конфлікт.

У теорії ігор передбачено, що гра складається з ходів, які виконують гравці одночасно або послідовно: «Ходи бувають особистими та випадковими. Хід називається особистим, якщо гравець свідомо вибирає його з сукупності можливих варіантів дій і здійснює його (наприклад, будь-який хід у шаховій грі). Хід називається випадковим, якщо його вибір здійснюється не гравцем, а яким-небудь механізмом випадкового вибору» [3, с. 7].

Зазвичай мобільні ігри класифікують наступним чином:

- **За кількістю гравців:** гри 1,2, n гравців.
- **За кількістю стратегій:** кінцеві і нескінченні гри. Якщо у всіх гравців кінцеве число стратегій, то така гра кінцева, інакше — гра нескінченна.
- **За характером взаємовідносин між гравцями:** безкоаліційні та кооперативні ігри. Гра називається безкоаліційної, якщо гравці не укладають між собою ніяких угод. Кінцева безкоаліційна гра двох гравців називається біматричною грою. У кооперативній грі гравці можуть укладати угоди з метою збільшити свої виграші.
- **За властивостями функцій виграшів:** безперервні, опуклі, сепарабельні і т. д. Якщо сума виграшів всіх гравців в кожній партії дорівнює нулю, то це - гра з нульовою сумою. Гра яка має двох акторів с нульовою сумою називається антагоністичною. У такій грі один гравець виграє за рахунок іншого. Кінцева антагоністична гра називається матричною грою. В іграх з ненульовою сумою всі гравці в сумі можуть отримати менше їх сумарного внеску. Наприклад, в лотереї її організатори завжди

у виграші, а учасники в сумі отримують менше їх сумарного внеску.

– **За кількістю ходів:** одноходові і багатоходові. Серед багатоходових ігор виділимо позиційні ігри, в яких кілька гравців послідовно роблять ходи; виграші гравців залежать від стратегії вибору ходів (приклад - шашки, шахи, карткові ігри, ігрові автомати, динамічні економічні системи і т. д.).
– **За інформованістю гравців:** ігри з досконалою і недосконалою інформацією. У грі з досконалою інформацією на кожному кроці гравцям відомо, які ходи були зроблені раніше (наприклад, шашки та шахи). У грі з недосконалою інформацією гравці можуть не знати, в якій позиції вони знаходяться (деякі стохастичні ігри, зокрема, карткові ігри).” [4, с. 7].

Рисунок 1.4 показує які ігрові елементи активують різні моделі поведінки:

Рисунок 1.4 - Механіка гри

Рисунок 1.5 - Класифікація ігор

- **Класифікація за платформою:**
  - **Одноплатформенна (ексклюзивна) гра** — розроблена для однієї платформи.
  - **Мультиплатформенна гра** — розроблена відразу для декількох платформ.
- **Класифікація за віртуальною інтернет платформою:**
  - **Флеш-гра** — різновид програми, створеної з використанням flash-технології. Флеш-ігри не потрібно завантажувати і встановлювати, вони запускаються прямо в браузері.
  - **Браузерна гра** — гра, яка запускається в браузері, з обов'язковим створенням профілю, що дозволяє грати разом з іншими гравцями. Це різновид ММО-ігор, в яких відсутня повноцінний режим відео, замість нього використовується графіка, намальована на інтернет-сторінках, з використанням флеш-відеовставок.
  - **Клієнтська гра** — тип гри, в якій необхідна додаткова програма-клієнт. Це форма поширення гри, яку потрібно

встановити (іноді досить переписати) на ПК. Обсяг дистрибутива коливається від десятка мегабайт до гігабайтів [51].

- **Класифікація за жанрами:**

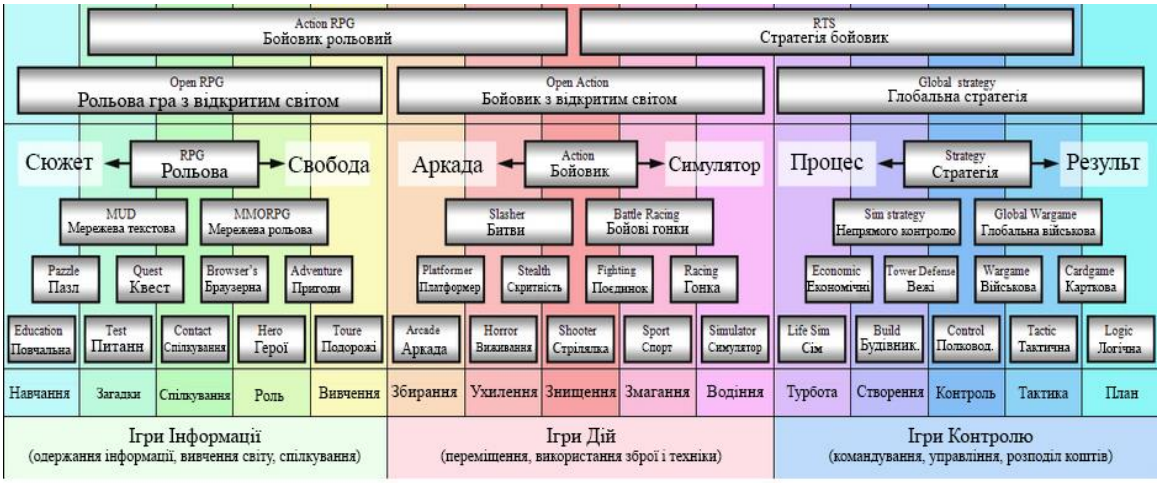


Рисунок 1.6 - Жанри мобільних ігор

- **Класифікація по розташуванню ігрової камери:**
  - **Вид від 1-ї особи** (вид з очей). Використовується в жанрах: екшен, шутер, РПГ, симулятор, гонки.
  - **Вид від 3-ї особи** (вид ззаду). Використовується в жанрах: екшен, шутер, РПГ, симулятор, гонки, слешерів, файтинг, 3D-платформер.
  - **Двомірний вигляд збоку** (2D вид збоку). Використовується в жанрах: платформер, головоломка, файтинг, 2D екшен.
  - **Тривимірний вид збоку** (3D вид збоку, псевдотривимірного). Використовується в жанрах: платформер, квест, головоломка, файтинг, 2D екшен.
  - **Двомірний вигляд зверху** (2D TopDown). Використовується в жанрах: стратегія, РПГ, тактика, головоломка, логічні ігри.
  - **Тривимірний вид зверху** (3D TopDown, изометрия). Використовується в жанрах: стратегія, РПГ, тактика, головоломка, логічні ігри [52].

РОЗДІЛ 2
ПРОЕКТУВАННЯ МОБІЛЬНОГО ДОДАТКУ «KSU_INSIDE»

2.1 Визначення структури та функціоналу додатку

Мобільний додаток «KSU_INSIDE» має на меті ознайомити абітурієнтів із Херсонським державним університетом у вигляді квесту.

При проектуванні гри було розроблено такі UML діаграми:

- **Діаграма варіантів (Use Case)**

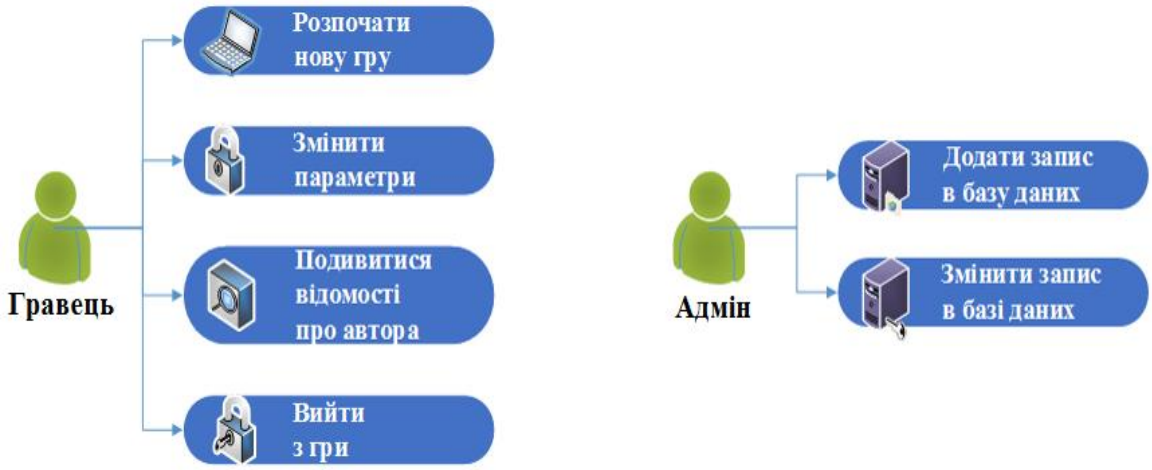


Рисунок 2.1 - Діаграма варіантів (Use Case)

Діаграма варіантів описує можливості які будуть надаватися системою для кінцевого користувача, яка інформація необхідна для обробки запиту користувача.

В даній діаграмі описано дії, які може здійснювати:

- **Гравець** (в головному меню він може розпочати гру, змінити налаштування, проглянути інформацію про автора гри, та вийти з гри)
- **Адміністратор** (додати та змінити запитання в базі даних)

- **Діаграма послідовностей**

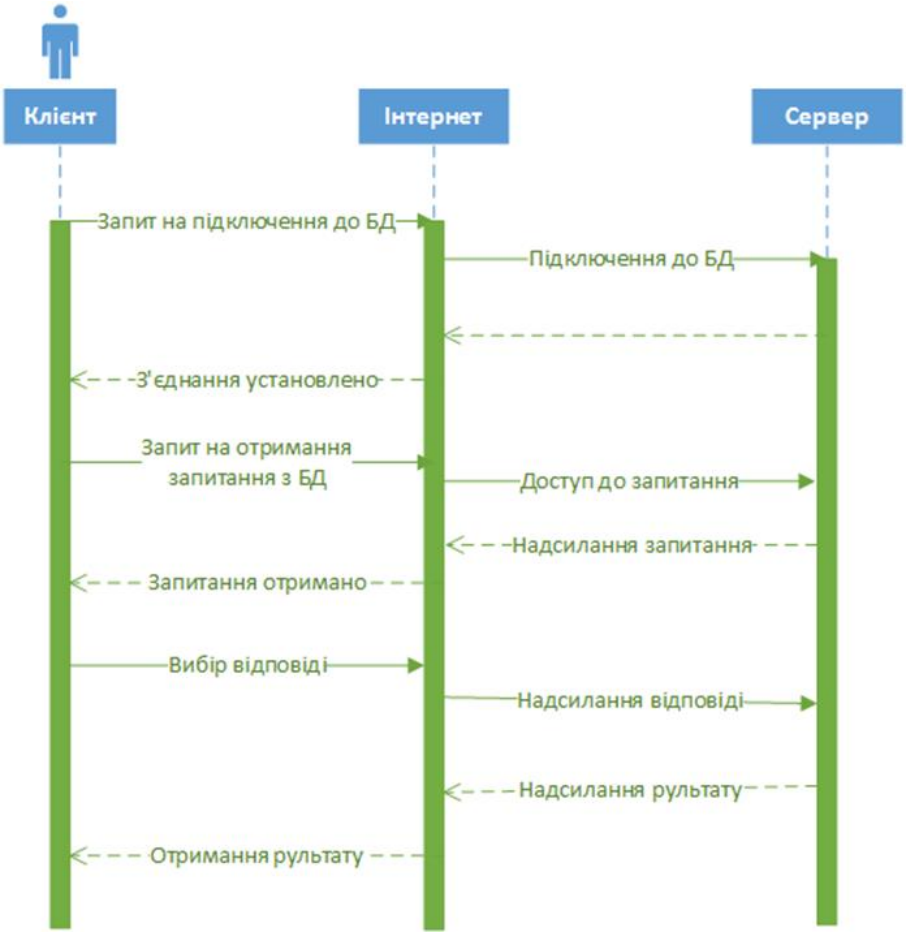


Рисунок 2.2 - Діаграма послідовностей

UML-діаграма, на якій для деякого набору об'єктів на єдиній тимчасової осі показаний життєвий цикл будь-якого певного об'єкта і взаємодія дійових осіб інформаційної системи в рамках якого-небудь певного прецеденту (відправка запитів та отримання відповідей).

На цій діаграмі показано життєвий цикл трьох дійових осіб: клієнта, інтернету та серверу. Клієнт через інтернет звертається до бази даних для отримання запитання та варіантів відповідей, а база даних, в свою чергу, дає клієнтові цю інформацію. Коли гравець відповідає на запитання правильно, то йому нараховуються бонуси. Після кожної неправильної відповіді, у гравця згорає одна спроба продовжити гру, та поки є можливість, продовжує гру на збереженому місці, інакше розпочинає гру заново.

- **Діаграма стану**

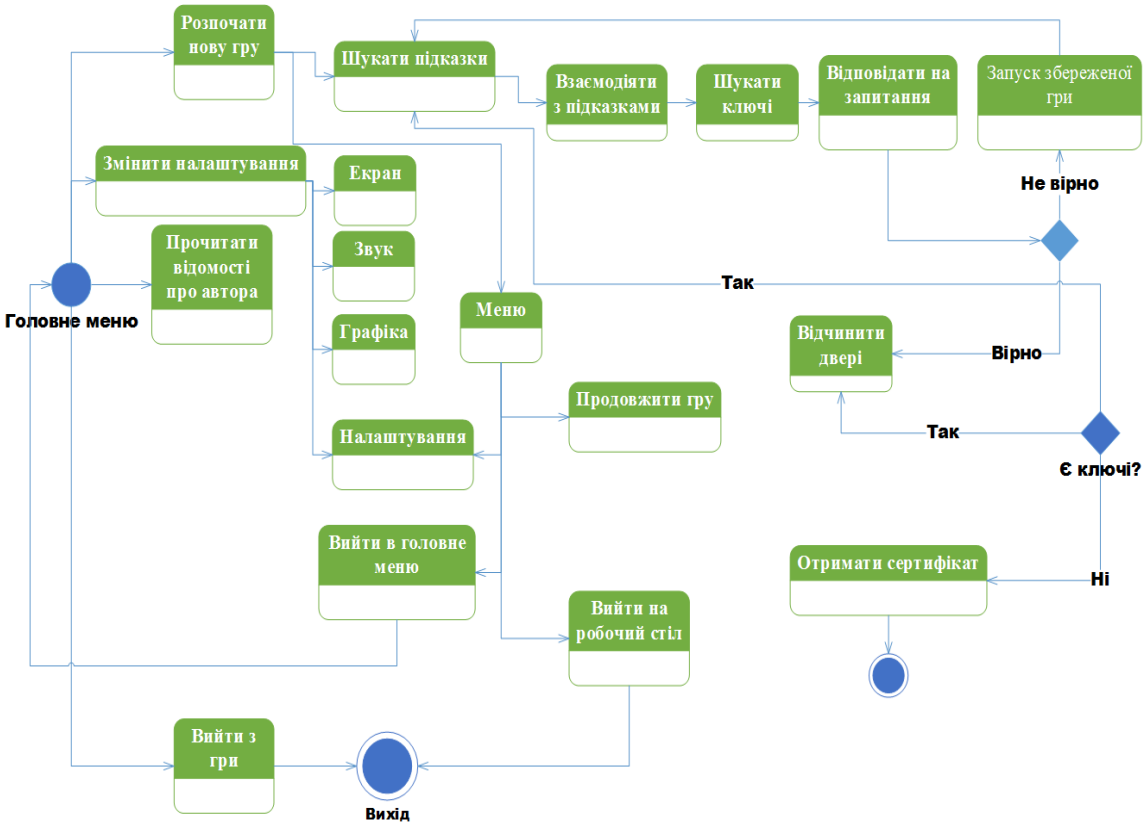


Рисунок 2.3 - Діаграма стану

UML-діаграма, що визначає зміну станів об'єкта у часі. Тобто, дії нашого об’єкта змінюються залежно від пройдених моментів. А саме, спочатку гравець повинен розгадати загадки кімнат, знайти ключі від дверей, потім відповісти на поставлене йому запитання, а вже потім перейти до іншої кімнати та розгадувати нову загадку нової кімнати. Після того як гравець завершить гру без жодної помилки, він отримає сертифікат, який підтверджує той факт, що гравець віртуально ознайомився зі стінами університету та приблизним матеріалом, що вивчатиме впродовж навчального року.

- **Діаграма класів**

Діаграма класів — статичне представлення структури моделі. Відображає статичні елементи, такі як: класи, типи даних, їх зміст та відношення.

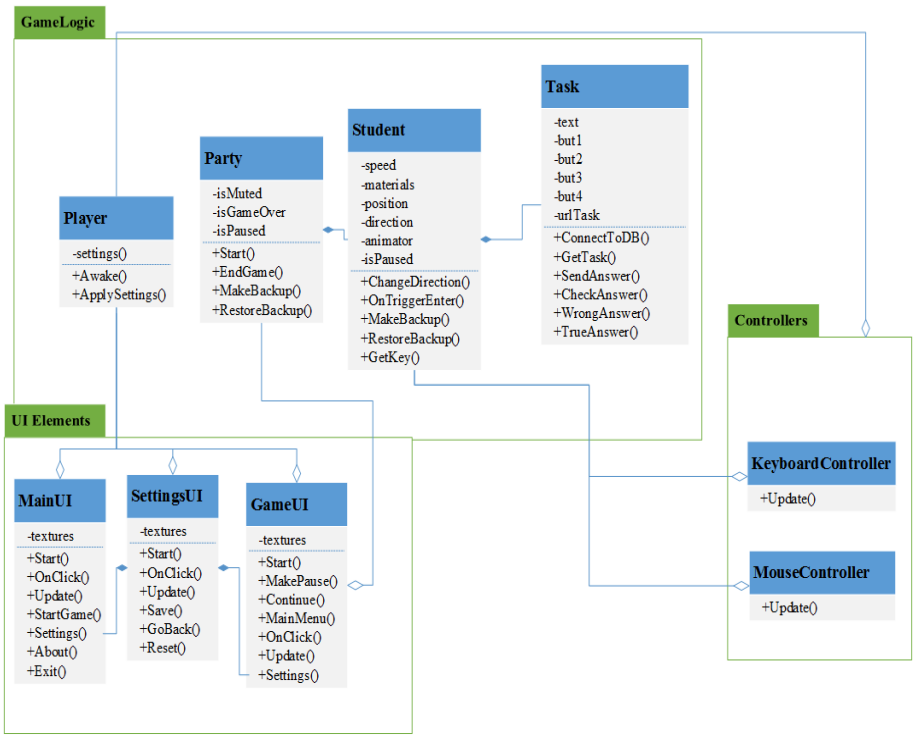


Рисунок 2.4 - Діаграма класів

Всі класи розділено на три пакети:

GameLogic – пакет в якому предоставлено класи ігрових об’єктів сцени.

UI Elements – пакет з класами, які відповідають за побудову інтерфейсу користувача: головне меню, меню гри, налаштування і т.д.

Controllers – пакет з класами, які відповідають за управління головним героєм гри.

В пакеті ігрової логіки центральне місце займає клас Party, який відповідає за початок та кінець гри, за збереження та відновлення гри та інших допоміжних функцій.

Клас Student відповідає за рух персонажа, його анімацію, управління залежними об’єктами класу Task, який в свою чергу через інтернет зв’язується з базою даних та отримує конкретне завдання з варіантами відповідей. Відповівши на запитання, ця відповідь зрівнюється з правильною, що міститься в базі даних, якщо дані

співпадають, то гравець продовжує гру та отримує бонус, інакше віднімається життя, та запускається остання контрольна точка.

Пакет з інтерфейсами має окремі класи, які відповідають за відображення та обробку інтерфейсу користувача.

Пакет з контроллерами зберігає класи управління персонажом (управління клавіатурою та мишою).

Структура та функціонал додатку:

- **Клієнтська частина гравця**

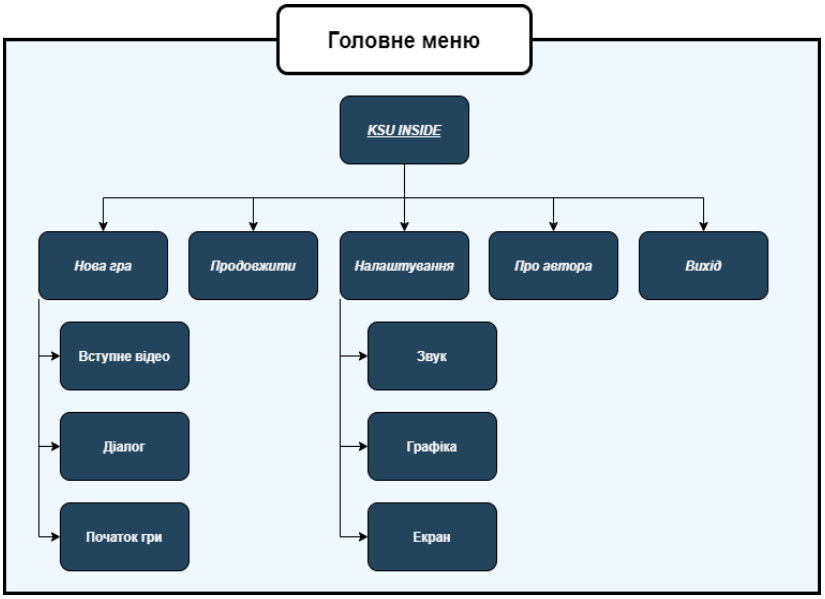


Рисунок 2.5 - Клієнтська частина гравця. Головне меню

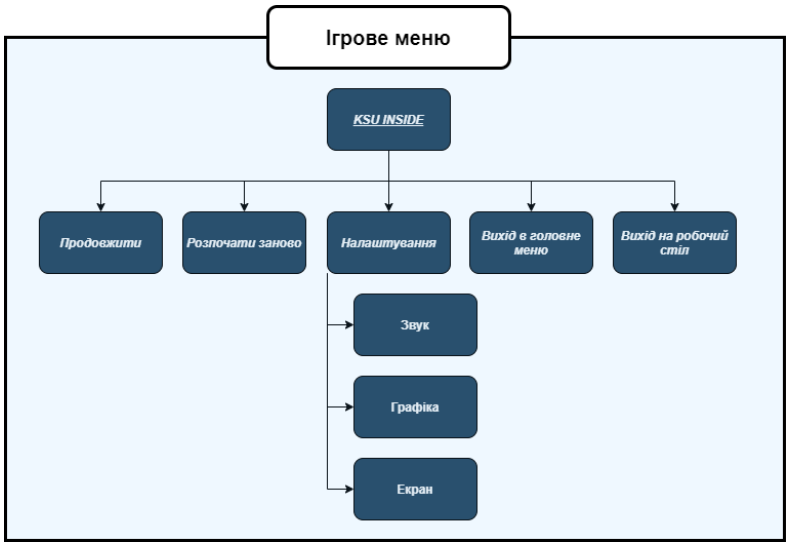


Рисунок 2.6 - Клієнтська частина гравця. Ігрове меню

- **Клієнтська частина адміністратора**

Рисунок 2.7 - Клієнтська частина адміністратора.

- **Серверна частина**

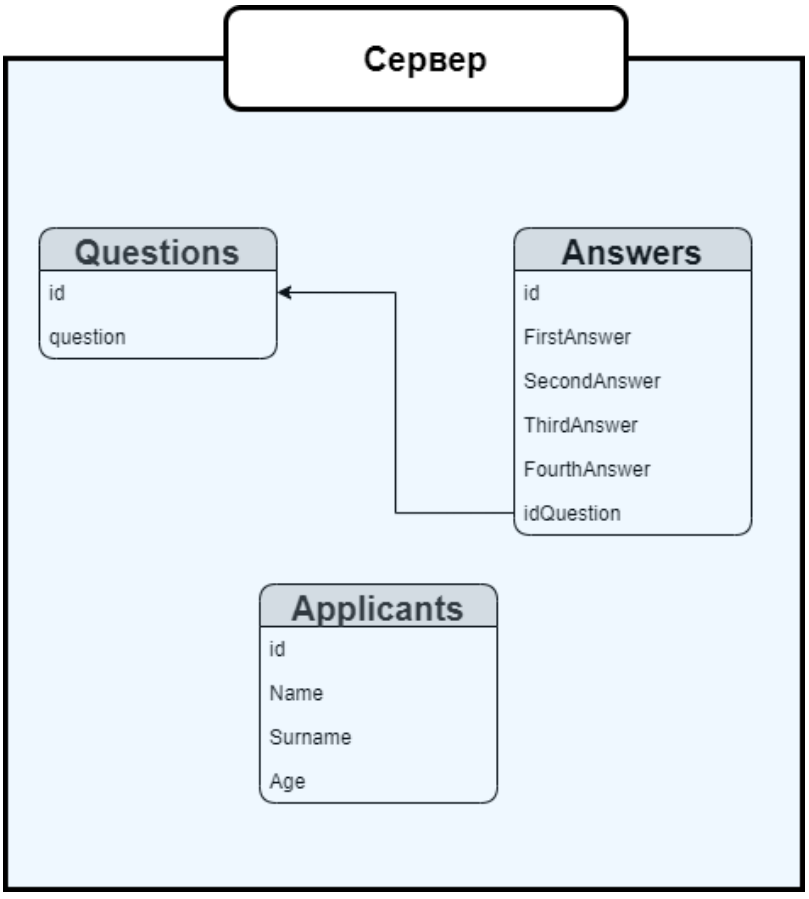


Рисунок 2.8 - Серверна частина

Коротко про гру.

Абітурієнт, який ще не визначився із вступом у вищий навчальний заклад, потрапляє до Херсонського державного університету. В холі він зустрічає робота-путівника, який проведе абітурієнту екскурсію університетом, а саме на факультеті комп'ютерних наук, фізики та математики. Покаже що саме він зможе робити після закінчення університету, продемонструє винаходи студентів та максимально зацікавить, щоб той обрав саме цей університет, після закінчення якого зможе претендувати на вакансії престижних компаній, які існують повсякчас.

2.2 Модель мережевої взаємодії

Модель мережевої взаємодії або мережева модель — теоретичний опис принципів роботи набору мережевих протоколів, які взаємодіють один з одним.

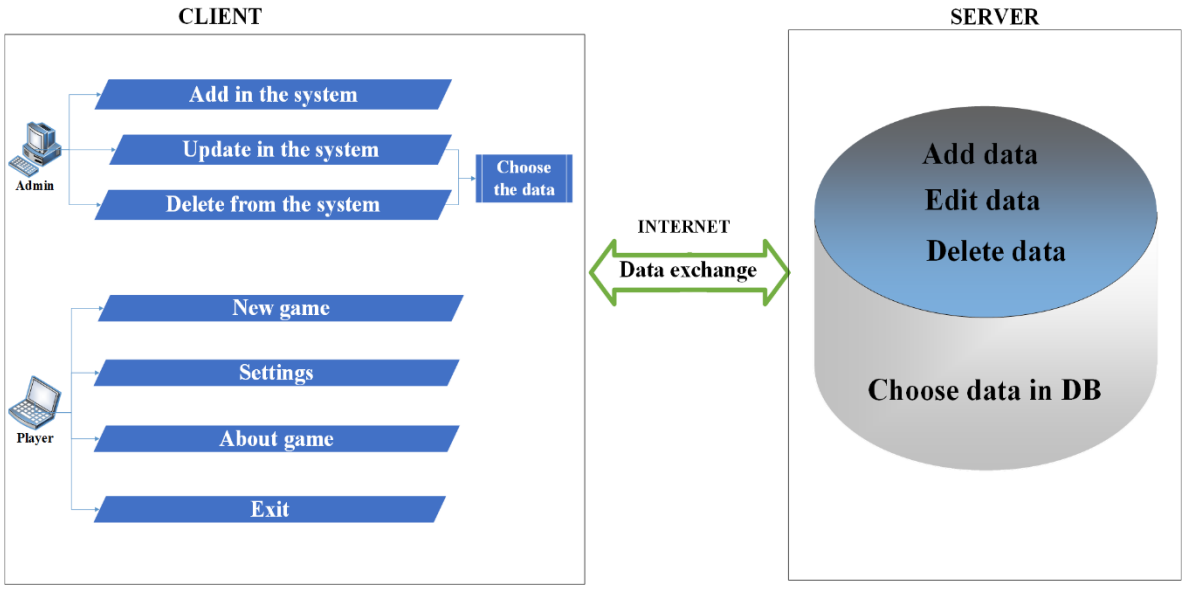


Рисунок 2.9 - Мережева взаємодія

На рис. 2.1 зображено мережеву взаємодію, яка використовується у грі для того щоб клієнт зміг підключитися до серверу для отримання запитань та варіантів відповідей. А потім, коли гравець відповідає, то його

відповідь зрівнюється з відповіддю що зберігається на сервері, залежно від відповіді вирішується, грати далі, чи запустити контрольну точку.

2.3 Проектування інтерфейсу додатку

Спочатку було розроблено прототипи клієнтських частин:

- **клієнтська частина гравця:**

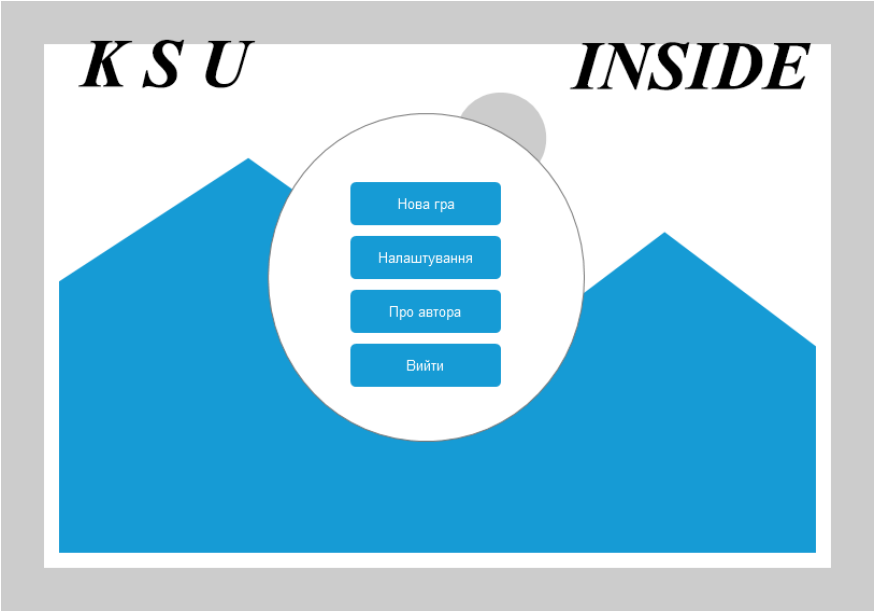


Рисунок 2.10 - Клієнтська частина гравця. Прототип головного екрану

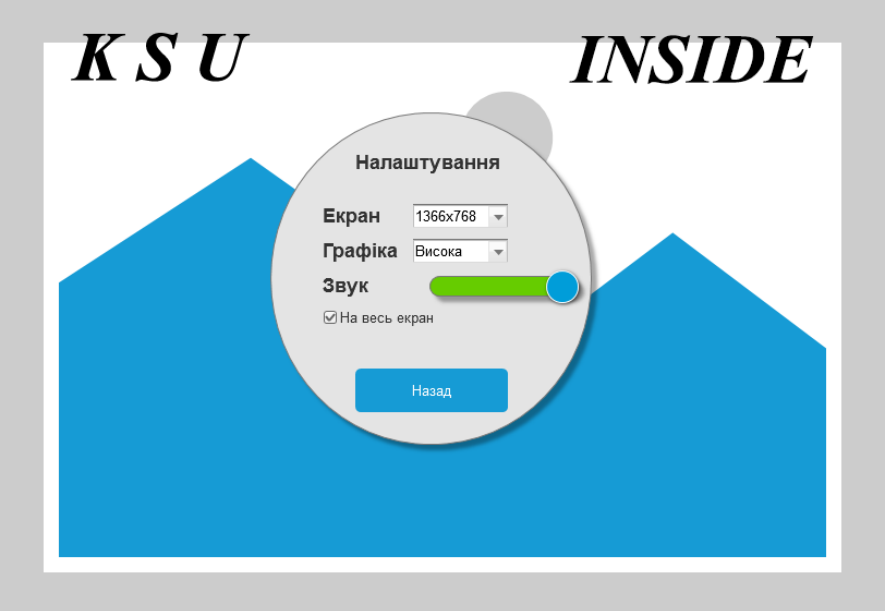


Рисунок 2.11 - Клієнтська частина гравця. Прототип екрану «Налаштування»

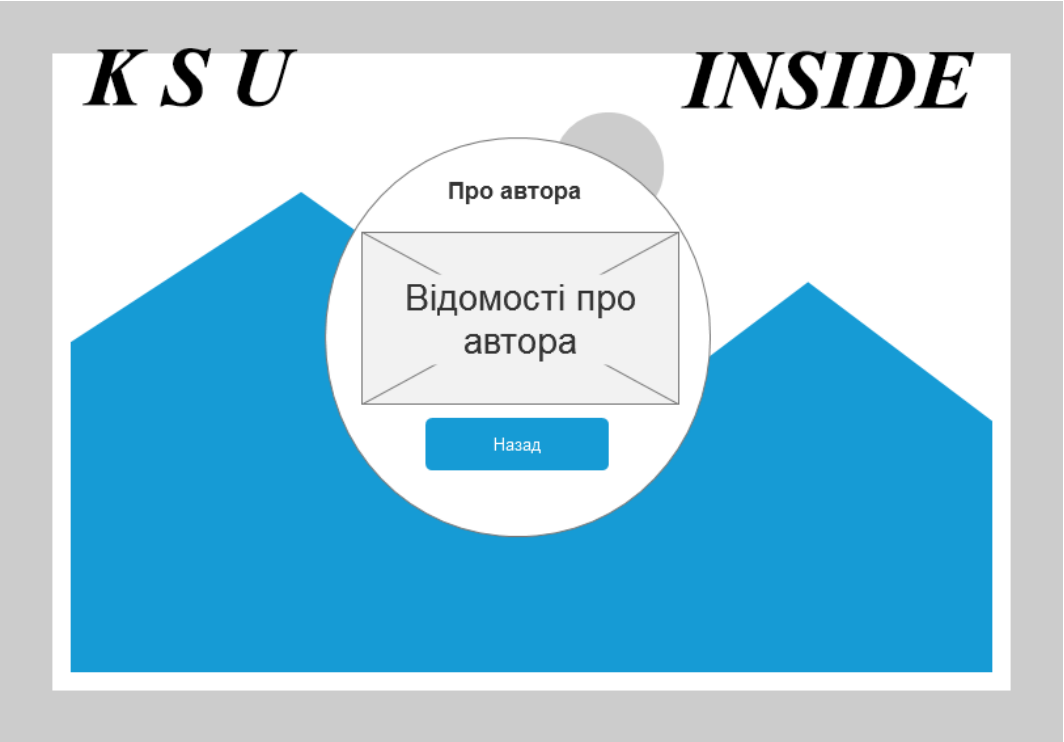


Рисунок 2.12 - Клієнтська частина гравця. Прототип екрану «Про автора»

- **Клієнтська частина адміністратора:**

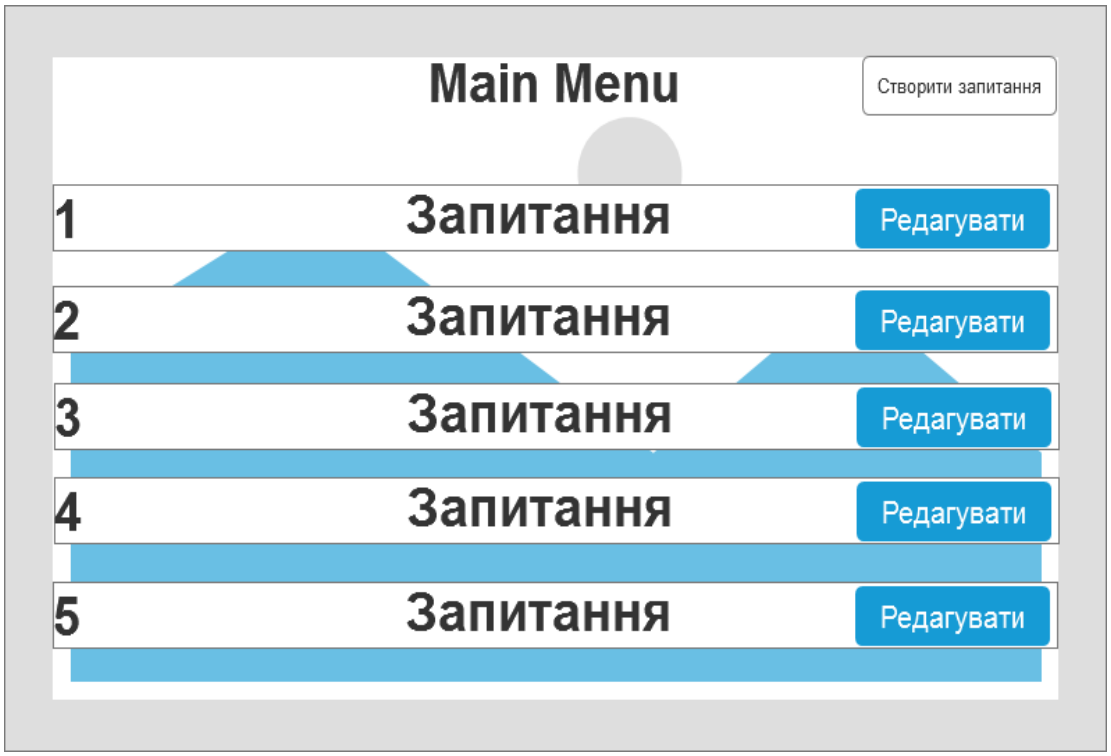


Рисунок 2.13 - Клієнтська частина адміністратора. Прототип головного екрану

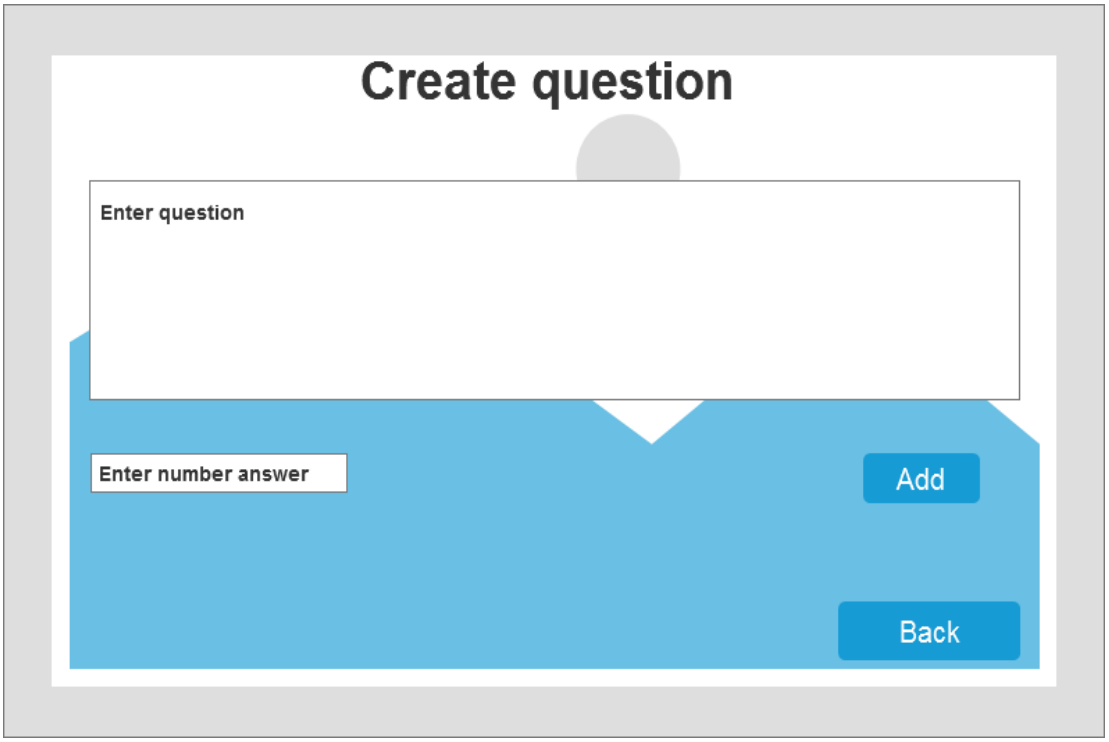


Рисунок 2.14 - Клієнтська частина адміністратора. Прототип екрану «Створення завдання»

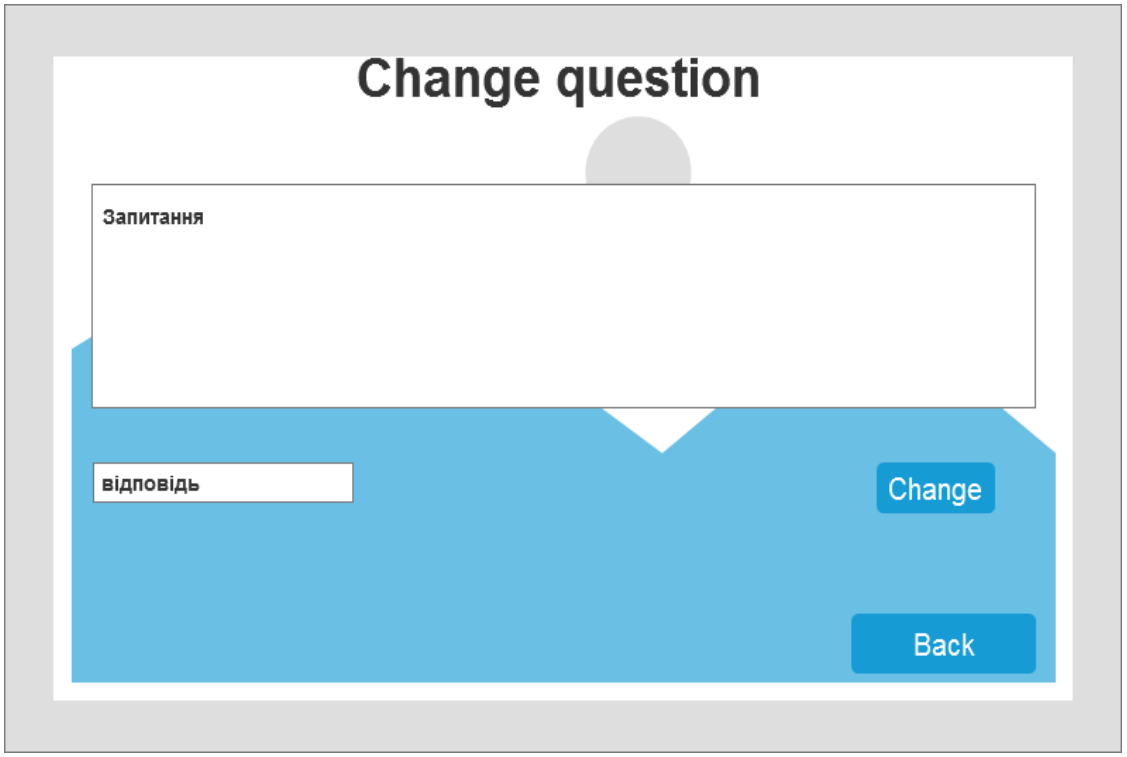


Рисунок 2.15 - Клієнтська частина адміністратора. Прототип екрану «Редагування завдання»

Потім, спираючись на ці прототипи, було розроблено дизайни інтерфейсів програм:

- **Дизайн інтерфейсу клієнтської частини гравця**

Рисунок 2.16 - Клієнтська частина гравця. Інтерфейс головного екрану

Рисунок 2.17 - Клієнтська частина гравця. Інтерфейс екрану «Налаштування»

Гравець може розпочати гру, змінити налаштування, подивитися відомості про автора, а також вийти з гри. А в самій грі може поділитися результатом в соціальних мережах.

- **Дизайн інтерфейсу клієнтської частини адміністратора**

Рисунок 2.18 - Клієнтська частина адміністратора. Інтерфейс головного екрану

Рисунок 2.19 - Клієнтська частина адміністратора. Інтерфейс екрану «Створення завдання»

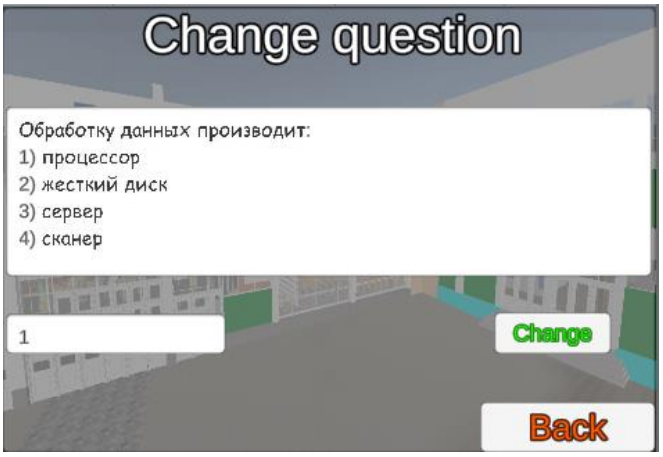
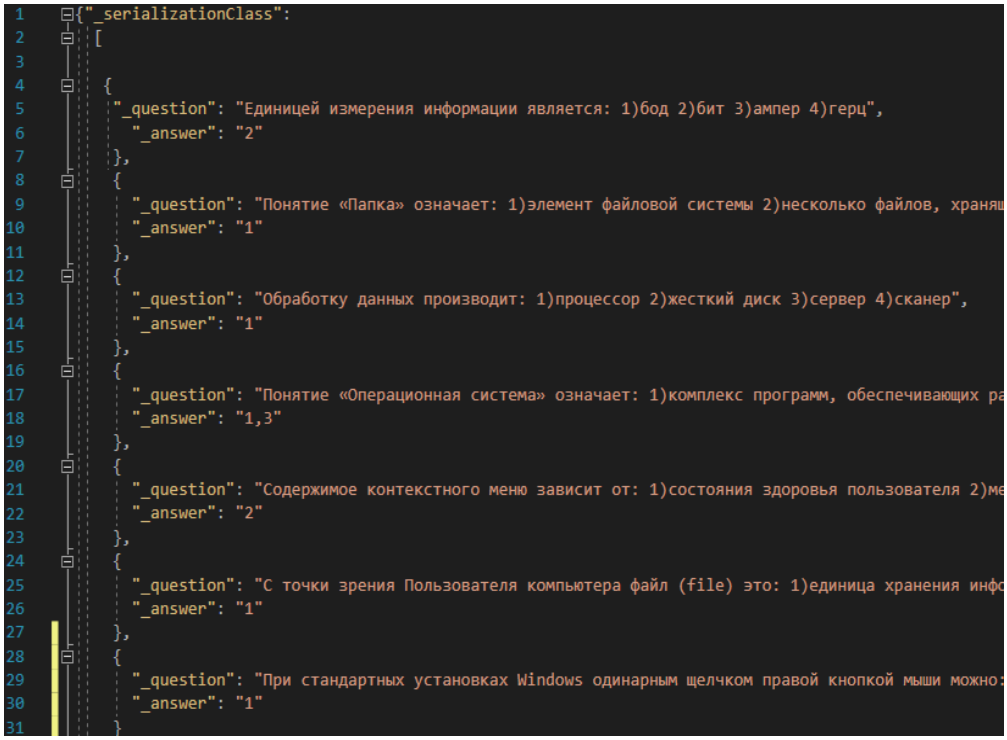


Рисунок 2.20 - Клієнтська частина адміністратора. Інтерфейс екрану «Редагування завдання»

Адміністратор гри може добавляти та редагувати завдання в базі даних. Сама база даних зберігається у форматі JSON, після зміни завдань, адміністратор зберігає цю базу даних та завантажує на сервер.



```
{"_serializationClass":
 [

  {
   "_question": "Единицей измерения информации является: 1)бод 2)бит 3)ампер 4)герц",
    "_answer": "2"
  },
   {
    "_question": "Понятие «Папка» означает: 1)элемент файловой системы 2)несколько файлов, храняш
    "_answer": "1"
   },
   {
    "_question": "Обработку данных производит: 1)процессор 2)жесткий диск 3)сервер 4)сканер",
    "_answer": "1"
   },
   {
    "_question": "Понятие «Операционная система» означает: 1)комплекс программ, обеспечивающих ра
    "_answer": "1,3"
   },
   {
    "_question": "Содержимое контекстного меню зависит от: 1)состояния здоровья пользователя 2)ме
    "_answer": "2"
   },
   {
    "_question": "С точки зрения Пользователя компьютера файл (file) это: 1)единица хранения инфо
    "_answer": "1"
   },
   {
    "_question": "При стандартных установках Windows одинарным щелчком правой кнопкой мыши можно:
    "_answer": "1"
   }
```


Рисунок 2.21 - База даних завдань у форматі JSON

Звідти вже клієнт користувача при першому запуску гри перевіряє на наявність актуальної версії бази даних. Якщо вона актуальна, то клієнт нічого не завантажує, інакше виконується завантаження актуальної версії даних.

РОЗДІЛ 3
МОДЕЛЮВАННЯ 3D ОБ'ЄКТІВ ТА РОЗРОБКА МОБІЛЬНОГО ДОДАТКУ «KSU_INSIDE»

3.1 Технології розробки мобільних ігор

Autodesk 3ds Max, раніше 3D Studio і 3D Studio Max, - це професійний інструмент архітекторів і дизайнерів інтер'єру. Він зручний в 3D моделюванні твердотільних об'єктів, має якісні модулі для фотореалістичної візуалізації та велику свободу в створенні моделей.

Основна функція програми - створення та редагування 3D графіки. 3Ds Max пропонує такі типи проектування тривимірних об'єктів:

- Полігональне моделювання. Може використовуватися для розробки моделей різної складності;
- Моделювання на основі примітивів. 3Ds Max містить вбудовану бібліотеку стандартних об'єктів, так званих примітивів (куб, сфера, циліндр і т.д.).
- Моделювання на основі сплайнів. Полягає в побудові каркаса вироби з тривимірних кривих (сплайнів). На його основі генерується сам 3D-об'єкт;
- Моделювання на основі NURBS-кривих. Ідеальний варіант для моделювання органіки та об'єктів, що мають гладку поверхню;
- Моделювання на основі поверхонь Безьє. Часто застосовується до окремих частин 3D моделі, для яких створюється мережа контрольних точок. З їх допомогою поверхню можна розтягувати в будь-якому напрямку.

3D візуалізація

Додаток надає можливість гнучкого управління налаштуваннями, включаючи експозицію, глибину різкості, та багато іншого. У числі

візуалізаторів програми такі модулі, як Arnold, V-Ray, Mental Ray, RenderMan, FinalRender, Luxrender та багато інших.

Анімація

Анімації піддаються як цілі об'єкти, так і окремі їх елементи. Присутні ефекти руху частинок (вогонь, дим, бризки, сніг), рідинні ефекти. Також є можливість детального моделювання траєкторій руху об'єктів.

Для гри було створено 3D модель університету за допомогою цієї програми та налаштовано матеріали.

C# - сучасна об'єктно-орієнтована та типобезпечна мова програмування. Вона відноситься до широко відомого сімейства мов C. Розробка сучасних додатків все більше тяжіє до створення програмних компонентів у формі автономних та самоописних пакетів, що реалізують окремі функціональні можливості. Головна особливість таких компонентів в тому, що вони являють собою модель програмування з властивостями, методами та подіями. У C # існує єдина система типів. Всі типи цієї мови програмування, включаючи типи-примітиви, такі як int і double, успадковують від одного кореневого типу object. Крім того, C # підтримує призначені для користувача посилальні типи та типи значень, дозволяючи як динамічно виділяти пам'ять для об'єктів, так і зберігати спрощені структури в стеці.

Для гри було написано скрипти на цій мові програмування

Unity3d є сучасним багатоплатформовим ігровим рушієм для створення ігор та додатків, розроблений компанією Unity Technologies. За допомогою цього рушія можна розробляти не тільки програми для комп'ютерів, а також ігрових приставок, мобільних пристроїв (наприклад, на базі Android) та інших девайсів.

в середовище розробки Unity інтегровано ігровий рушій, а це означає, що протестувати свою гру можна не виходячи з редактора. Написання скриптів здійснюється на мові програмування C #.

За допомогою цього рушія було розставлено моделі на сцені, налаштовано скрипти, шейдери, матеріали, світло та створено саму гру

3.2 Основи моделювання 3D об’єктів

Моделювання – це метод пізнання(дослідження) навколишнього світу, в якому деякого досліджуваного явища поставлено в відповідність модель у вигляді також об'єкта, явища, процесу, яке може замінити натуру в процесі досліджень. Практичне значення моделювання:

- Моделі зручні для дослідження, ніж вихідні об'єкти. Деякі об'єкти, явища або процеси можна вивчити тільки спираючись на моделі.
- Моделювання дозволяє виявити істотні фактори об'єкта або явища яке досліджується, також це інструмент для глибокого вивчення реальності.

Об'єкти в 3ds max розділяють на такі категорій:

- Geometry (Геометрія);
- Shapes (Форми);
- Lights (Джерела світла);
- Cameras (Камери);
- Helpers (Допоміжні об'єкти);
- Space Warps (Об'ємні деформації);
- Systems (Додаткові інструменти).

У програмі 3ds Max можна використовувати кілька різних типів тривимірного моделювання, які можна застосовувати в найрізноманітніших ситуаціях.

- Моделювання, засноване на примітивах. Примітиви – найпростіші параметричні форми, (куби, сфери, піраміди та ін).
- Моделювання, засноване на перетинах. Перетини (плоскі форми) - двовимірні об'єкти. При створення тривимірних об'єктів кілька форм розташовуються уздовж деякого шляху.

- Моделювання, засноване на використанні булевих операцій. Булеві об'єкти (Booleans) створюються додаванням, відніманням та перетином поверхонь які перекриваються[34].

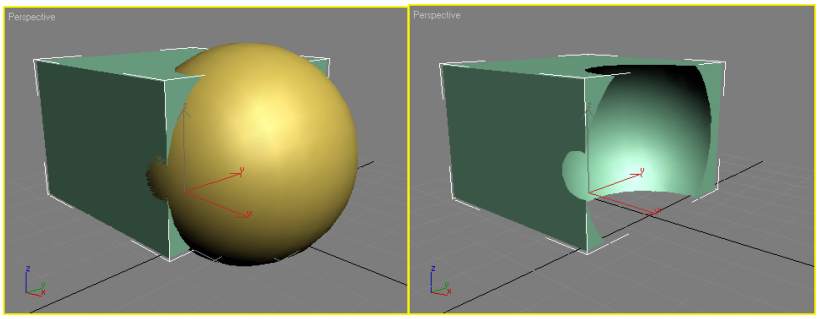


Рисунок 3.1 - Результат виконання булевих операцій

Поверхневе моделювання грунтується на створенні довільних поверхонь. При створенні поверхонь використовують різного роду математичні моделі та, відповідно, свої види моделювання:

- складні моделі будуються на основі сплайнів (гладких кривих) та можуть змінюватися за допомогою контрольних точок.
- змінюватися за допомогою контрольних точок.

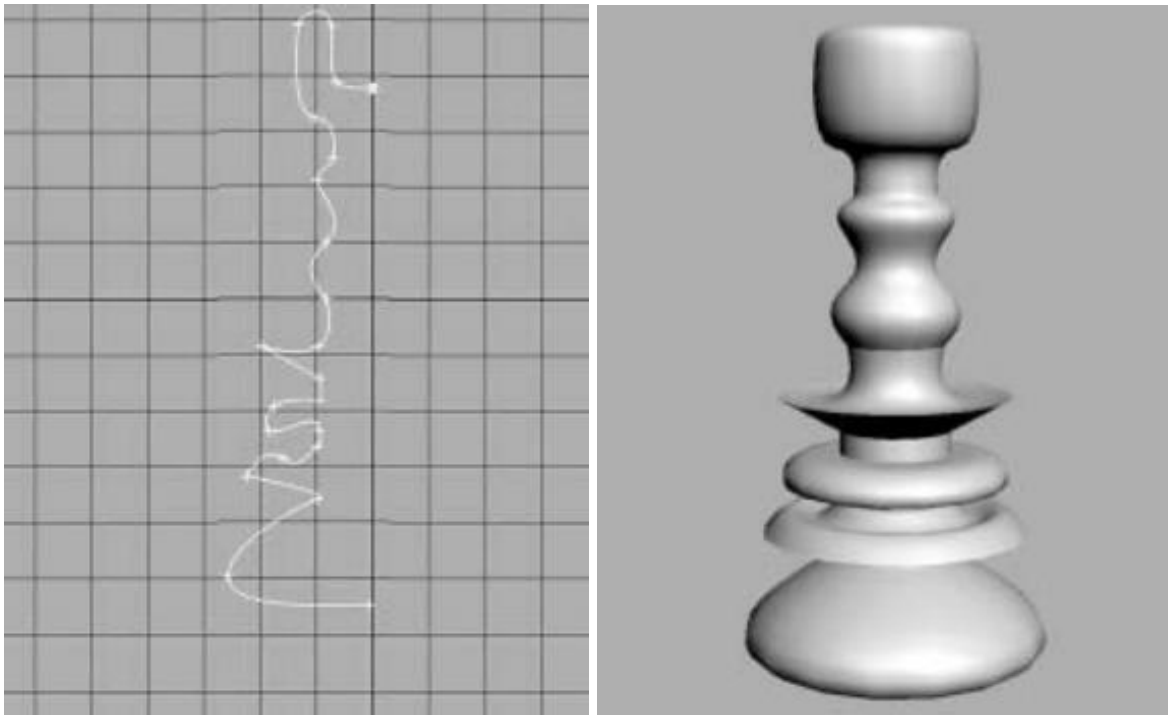


Рисунок 3.2 - Сплайнове моделювання

- **Неоднорідні раціональні B-сплайни (NURBS)** - технологія, призначена для створення плавних форм та моделей. Створюється сукупність сплайнів, своєрідний каркас, на основі якого формується поверхню (Surface).

Створення об’єктів та змінення їх параметрів

Для створення об’єктів, потрібно:

1. Перейти на вкладку Create (Створення) командної панелі.

2. Вибрати категорію, в якій знаходиться потрібний об'єкт, для примітивів це категорія Geometry (Геометрія).

3. Із списку вибрати групу, в якій знаходиться потрібний об'єкт. Для простих примітивів - це група Standard Primitives (Прості примітиви).

4. Натиснути кнопку з назвою об'єкта.

5. Натиснути в будь-якому місці вікна проекції і, не відпускаючи кнопку, навести курсор миші до тих пір, поки не зміниться розмір об'єкта до потрібного.

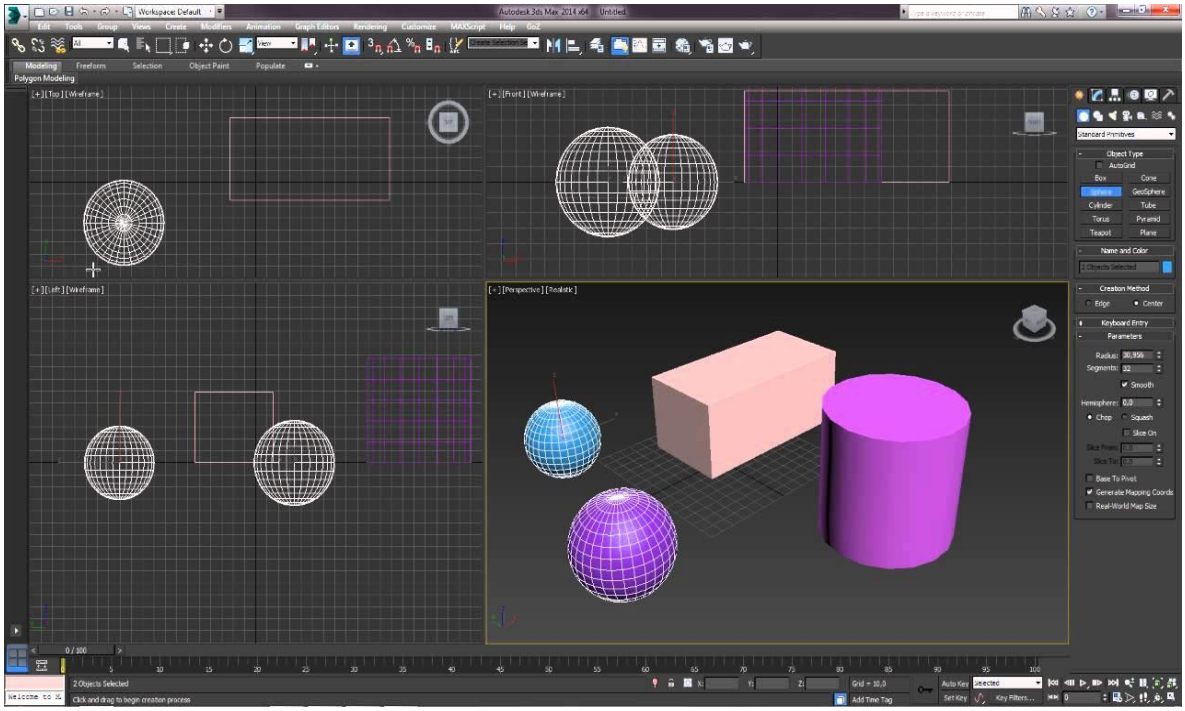


Рисунок 3.3 - Створення об’єктів в 3D’s Max

Клонування та створення масивів об'єктів

У 3ds max існують три види клонів:

- Copy (Копія).

- Instance (Зразок).
- Reference (Примірник).

Метод copy: створює повністю окремий клон з оригіналу. Змінення одного не впливає на інший. Під час копіювання об'єкта створюється новий, незалежний основний об'єкт і потік даних, в результаті чого з'являється новий, названий об'єкт. Копія дублює всі дані оригінального об'єкта під час копіювання. Копія не має зв'язку з оригінальним об'єктом.

Метод instance: створює повністю взаємозамінний клон оригіналу. Змінення об'єкта instanced аналогічно зміні оригіналу. Випадки схожі не тільки в геометрії, але й у будь-якому іншому випадку. Інсталяція об'єкта призводить до кількох іменованих об'єктів на основі головного об'єкта. Кожен об'єкт із назвою об'єкта має власний набір перетворень, зв'язків просторової деформації та властивостей об'єкта, але він поділяє модифікатори об'єкта та основний об'єкт з іншими екземплярами. Потік даних для гілок екземпляру відразу після оцінки модифікаторів об'єктів.

Коли змінюється один екземпляр, застосовується або налаштовується модифікатор, всі інші екземпляри також змінюються.

Метод reference: створює клон залежно від оригіналу до точки, коли об'єкт клонує. Зміна параметрів для модифікаторів, які були застосовані до об'єкта до того, як було спрямовано посилання на об'єкт, змінить обидва об'єкти. Однак новий модифікатор може бути застосований до одного з еталонних об'єктів, і це вплине тільки на об'єкт, до якого він застосовується. Список посилань базується на оригінальному об'єкті, як і на екземплярах, але також може мати свої унікальні модифікатори. Як і в instance, посилання поділяють, як мінімум, один основний об'єкт і, можливо, деякі модифікатори об'єктів [35].

3.3 Етапи розробки гри “KSU_Inside”

3.3.1 Обгрунтування вибору програмного забезпечення

Таблиця 3.1

Обгрунтування вибору програмного забезпечення

Назва ПЗ	**Обгрунтування**
Unity3D	Для розробки гри було використано саме цей інструмент, тому що він є багатоплатформовим (має змогу створювати додатки для різних платформ. Таких як Android, Windows, Linux, OS X, Apple iOS та ін.) для розробки 2D та 3D додатків. Рушій підтримує більшість форматів 3D об'єктів, такі як: .3ds, .fbx, .obj та ін. Також підтримує розробку на такій мові високого рівня як C#. Має безкоштовну ліцензію для створення додатків у власних цілях. Також має свою власний магазин моделей та готових сцен «Asset Store», які можна використовувати у власних цілях.
3D’s Max	Для розробки моделей для гри було використано саме цей професійний інструмент 3D моделювання. Він є зручним у використанні та підтримує експорт 3D моделей у більшість форматів, які підтримує

Продовження табл. 3.1

Назва ПЗ	Обгрунтування
3D’s Max	Unity3D для імпорту (.3ds, .fbx, .obj та ін.). Також 3Ds Max має підтримку таких типів проектування тривимірних об'єктів: полігональне моделювання, на основі сплайнів, NURBS-кривих та баго іншого.

3.3.2 Моделювання елементів гри

Моделювання університету

Університет було змодельовано за допомогою методу «знизу вверх». Тобто, спочатку було виявлено результат, який в кінці має бути.

Рисунок 3.4 - Очікуваний результат

Далі було створено стандартний примітив «куб»

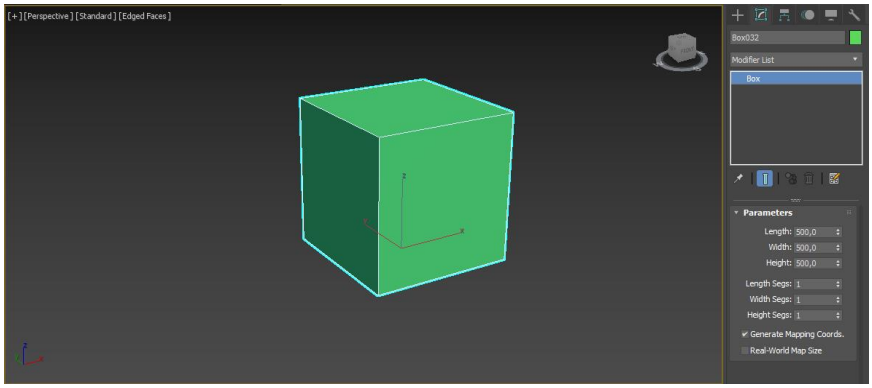


Рисунок 3.5 - Стандартний примітив «куб»

з якого потім за допомогою інстументів видавлення Extrude, Bevel та інструмента з'єднання Bridge було видавлено складну фігуру (стіни, двері, вікна).

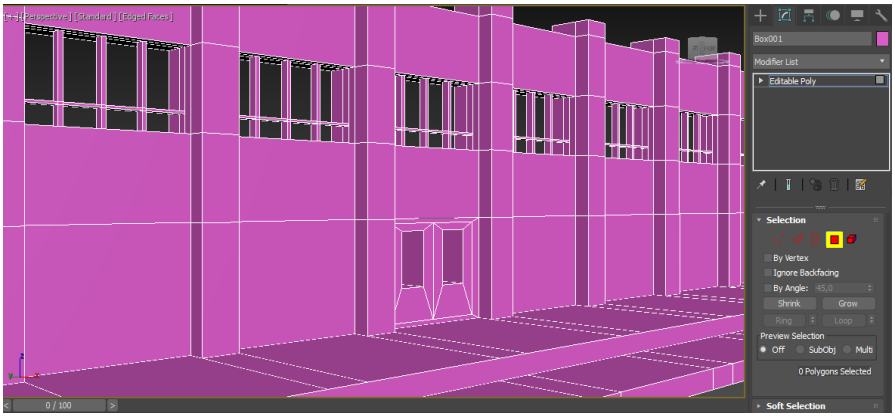


Рисунок 3.6 - Моделювання стіни, вікон та дверей

Потім створили матеріали та застосували їх до нашої моделі

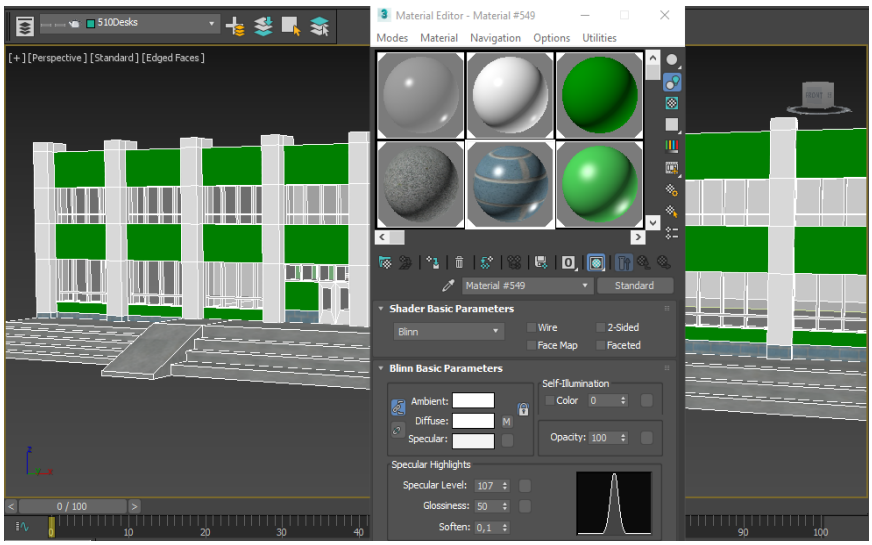


Рисунок 3.7 - Застосування матеріалів до моделі

Після імпорту моделі в ігровий рушій Unity 3D та налаштування її там, маємо повноцінну модель для гри.

Рисунок 3.8 - Модель університету

Моделювання діалогу

Діалог було змодельовано за допомогою онлайн діаграм «*Draw.io*». Спочатку головний персонаж зустрічає робота, який проведе його просторами університету, розповість про те, які можливості будуть у персонажа, коли той буде навчатися в університеті.

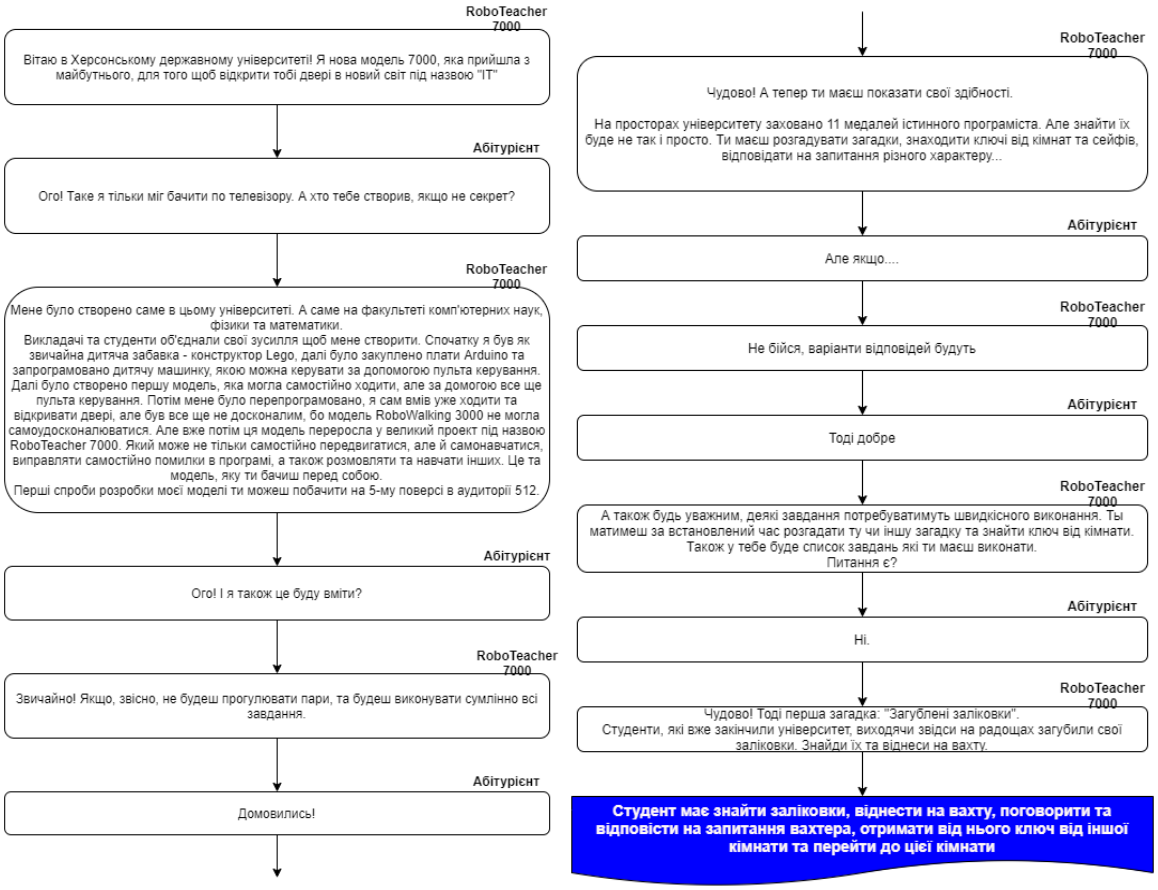


Рисунок 3.9 - Діалог

Створення ефектів

Ефект зникнення.

Цей ефект було зроблено в два етапи:

1) Редагування прозорості об’єкта при анімації зникнення та з’явлення.
2) Створення світлих країв поверх цієї анімації.

Спочатку створили пустий файл шейдеру (рис. 3.10).

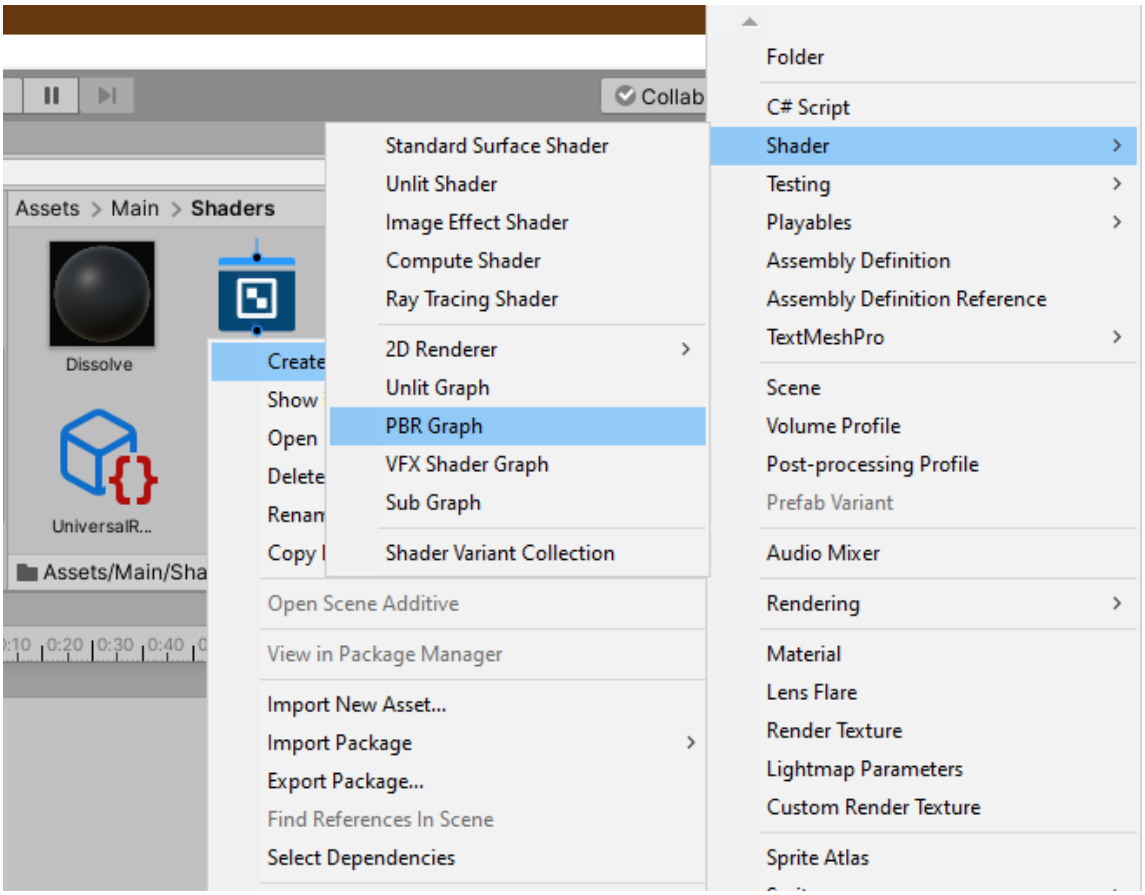


Рисунок 3.10 - Створення шейдеру

Для редагування прозорості об’єкта для анімації використали «Noise (Шум)» (рис. 3.11) та з’єднали його вихідну вузлову точку із альфа каналом

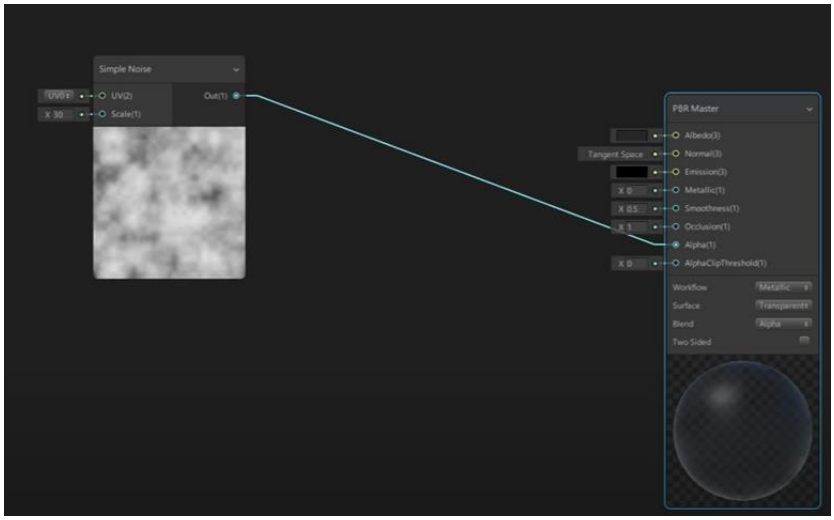


Рисунок 3.11 - Редагування прозорості об’єкта (Noise (Шум))

Далі створили вузли Time і Remap. У вузлі Time є параметр Sin який змінюється із -1 до 1, а вузол Remap переоформлює ці параметри із 0 до 1. Можемо спостерігати що Remap вузол змінює колір із чорного в білий, де чорний в даному випадку повністю непрозорий, а білий прозорий. З’єднали вузлову точку «Sin Time» із точкою «In» вузла Remap, а вихідну точку цього ж вузла з’єднали із «AlphaClip» точкою. Можемо спостерігати плавне зникнення та з’явлення об’єкта

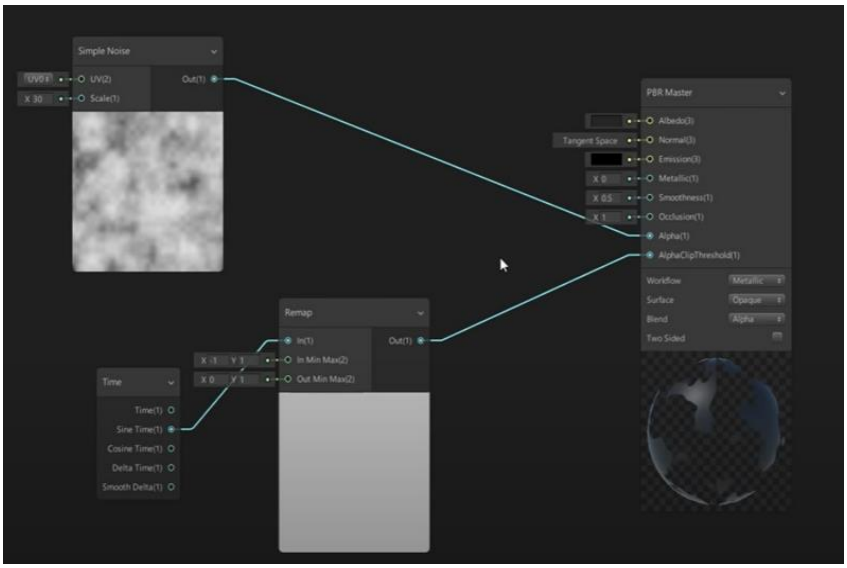


Рисунок 3.12 - Редагування прозорості об’єкта (Вузли Time і Remap)

Для створення ребер анімації використали вузол Step, та з’єднали його з вузлом Simple Noise. Також створили Add вузол (в ньому редагуємо

товщину ребер), через який з’єднали два вузли Remap та Step. А вузол Step з’єднали з Emission.

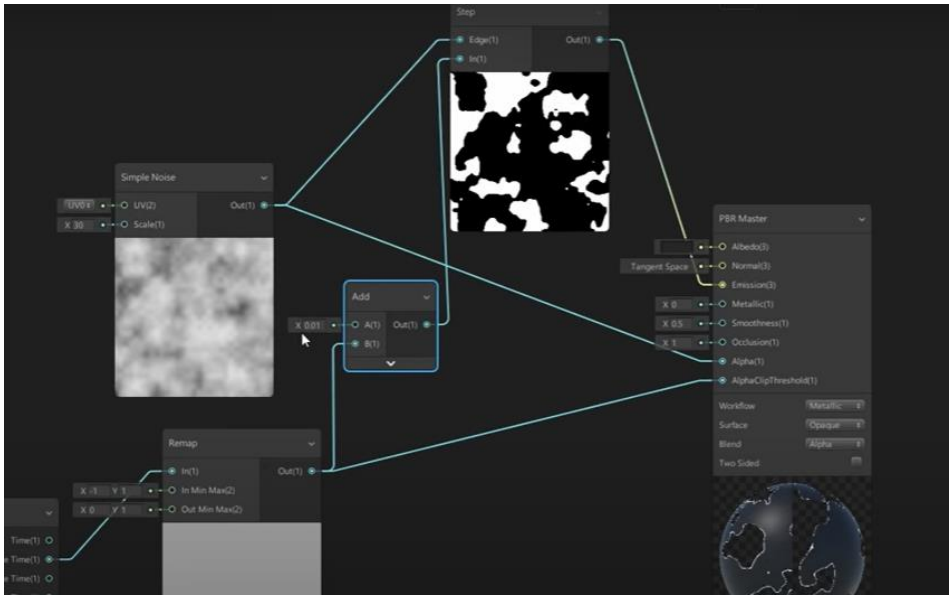


Рисунок 3.13 - Створення світлих країв (вузол Step)

Далі створили вузол Color, щоб змінити колір ребер, та через інший вузол Multiply з’єднали з вузолом Step, а вихідну точку Multiply вузолу з’єднали з точкою Emission.

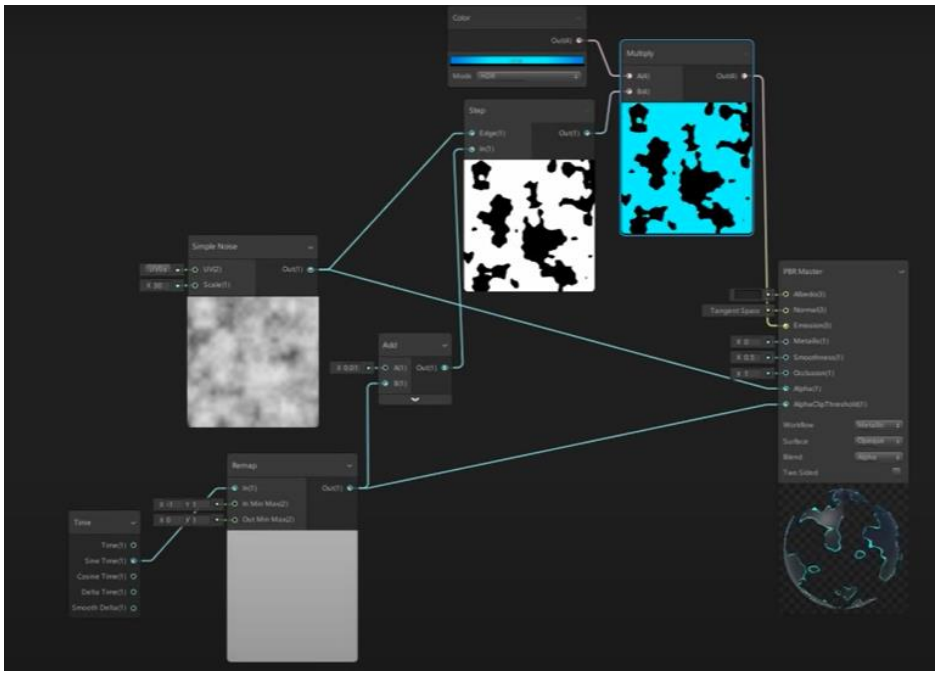


Рисунок 3.14 - Створення світлих країв (вузол Color)

Потім застосували цей шейдер на робота та отримали гарний ефект зникнення та з’явлення (рис. 3.15).

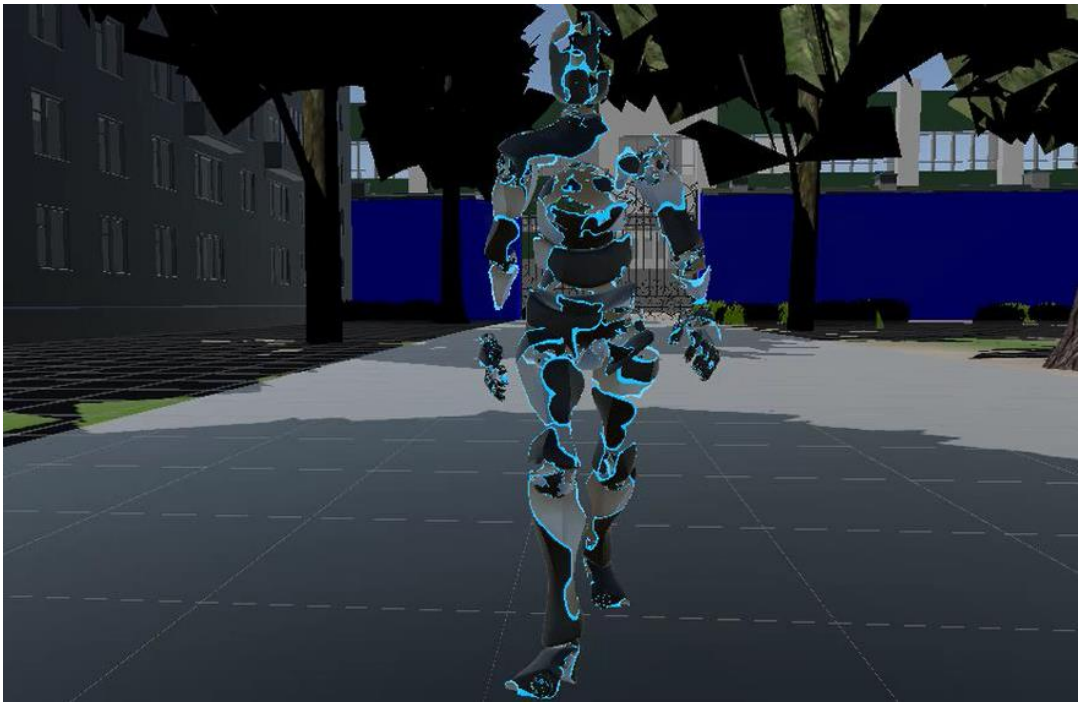


Рисунок 3.15 - Готовий ефект

3.3.3 Програмна реалізація додатку

Під час створення гри було написано безліч скриптів. Таких як:

- **Рух персонажа та поворот камери.** Для контролювання руху героя та його анімації створено клас *PlayerController* (рис. 3.16).

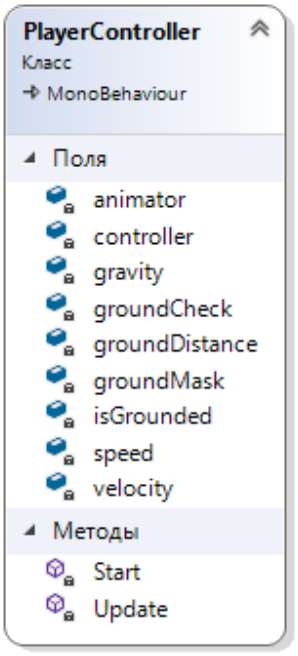


Рисунок 3.16 - Клас PlayerController

У цьому класі реалізовані такі методи як *Start()* (викликаються перед першим оновленням кадру) та *Update()* (викликається при кожному

оновленні кадру, тобто, якщо FPS гри 30 кадрів/секунду, то ця функція буде викликана 30 разів).

В полях класу зберігаються посилання на об'єкти аніматора (тут розташовані анімації), controller (типу *CharacterController* – компонент, за допомогою якого рухається персонаж), *groundCheck*, *groundDistance*, *groundMask*, *isGrounded* – змінні які впливають на гравітацію та перевірку «чи знаходиться гравець на землі?», та інші. Реалізація цього класу представлена в додатку А.

Для повороту камери персонажа відповідає клас *CameraLook* (рис. 3.17).

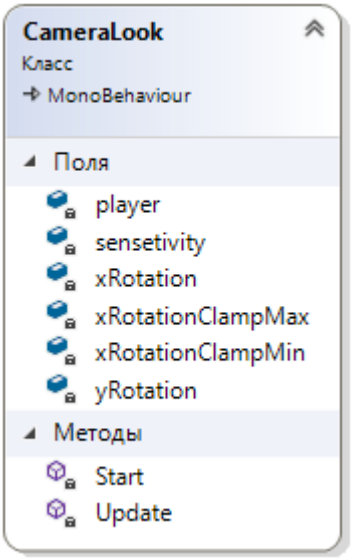


Рисунок 3.17 - Клас CameraLook

У цьому класі також реалізовані методи *Start()* та *Update()*. А в полях класу створено змінні, що відповідають за поворот камери (*xRotation*, *yRotation*), чутливість її повороту (*sensetivity*), гравець, який буде повертатися разом із камерою по осі *Y* (*player*), та змінні, які відповідають за обмеження повороту цієї камери (*xRotationClamMin*, *xRotationClamMax*), коли персонаж дивиться вверх, щоб вона не поверталась на 360 градусів. Реалізація цього класу представлена в додатку Б.

- **Рух дрона назустріч гравцю (штучний інтелект).** Для руху дрона було створено клас *Drone_Way*, в якому описано змінні, методи та лямбда-вирази, які використовуються для опису штучного інтелекту (рис. 3.18).

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;

public class Drone_Way : MonoBehaviour
{
    [SerializeField] private float _stopDistance = 2f;
    [SerializeField] private float speed;
    [SerializeField] private float _playerDetectionDistance = 20f;
    [SerializeField] private  float _playerAroundWaypointsMinDistance = 10f;
    [SerializeField] private Transform _player;
    [SerializeField] private List<Transform> _waypointsPatrol;
    [SerializeField] private List<Transform> _waypointsAroundPlayer;
    private float DistanceToPlayer => Vector3.Distance(transform.position, _player.position);
    private float MoveDelta => speed * Time.deltaTime;
    private int _waypointsIndexAroundPlayer;
    private int _waypointsIndex;
    private enum State { PATROL, CHASE }
    private State _state;

    private void Start()
    {
        _state = State.PATROL;
        StartCoroutine(DroneState());
    }
```

Рисунок 3.18 - Клас Drone_Way. Оголошення змінних

Для перевірки стану було створенно корутину DroneState. В ній у безкінечному циклі відбувається перевірка стану дрона: патрулювання (метод *Patrol*) або переслідування (метод *Chase*).

```
private IEnumerator DroneState()
{
    while (true)
    {
        switch (_state)
        {
            case State.PATROL:
                Patrol();
                break;
            case State.CHASE:
                Chase();
                break;
        }
        yield return null;
    }
}
```

Рисунок 3.19 - Клас Drone_Way. Перевірка стану

Метод *Patrol* відповідає за кружляє дрона по кімнаті. Для цього було створено та розміщено точки (waypoints), по яким він і

передвигається. В коді було створено список цих точок та змінна *_waypointIndex*, яка відповідає за індекс елемента в списку. Якщо дрон приблизився до цільової точки, то цей індекс збільшується на 1, тим самим цілью стає сусідня точка, а коли досягає кінцевої, то індекс обнуляється. Але якщо дистанція між гравцем, та дроном менша зазначеної, то стан змінюється на *Chase*.

```
private void Patrol()
{
    Vector3 waypointPosition = _waypointsPatrol[_waypointsIndex].position;

    float distanceToWaypoint = Vector3.Distance(transform.position, waypointPosition);

    if (distanceToWaypoint > _stopDistance)
        transform.position = Vector3.MoveTowards(transform.position, waypointPosition, MoveDelta);

    else if (distanceToWaypoint < _stopDistance)
    {
        _waypointsIndex++;
        if (_waypointsIndex == _waypointsPatrol.Count)
            _waypointsIndex = 0;
    }

    if (DistanceToPlayer < _playerDetectionDistance)
        _state = State.CHASE;
}
```

Рисунок 3.20 - Клас Drone_Way. Метод Patrol

Метод *Chase* відповідає за переслідування гравця. Коли стан змінюється на *Chase*, то дрон летить назустріч гравцю (рис. 3.22) та кружляє навколо нього. А якщо дистанція між гравцем та дроном збільшується, то стан змінюється на *Patrol* і дрон летить туди звідки прилетів до гравця.

```
    private void Chase()
    {
        if (DistanceToPlayer < _playerDetectionDistance && DistanceToPlayer > _playerAroundWaypointsMinDistance)
            transform.position = Vector3.MoveTowards(transform.position, _player.position, MoveDelta);
        else if (DistanceToPlayer <= _playerAroundWaypointsMinDistance)
        {
            Vector3 waypointPosition = _waypointsAroundPlayer[_waypointsIndexAroundPlayer].position;

            float distanceToPlayerWaypoint = Vector3.Distance(transform.position, waypointPosition);

            if (distanceToPlayerWaypoint > _stopDistance)
                transform.position = Vector3.MoveTowards(transform.position, waypointPosition, MoveDelta);
            else
            {
                _waypointsIndexAroundPlayer++;
                if (_waypointsIndexAroundPlayer == _waypointsAroundPlayer.Count)
                    _waypointsIndexAroundPlayer = 0;
            }
        }

        if (DistanceToPlayer > _playerDetectionDistance)
            _state = State.PATROL;
    }
}
```

Рисунок 3.21 - Клас Drone_Way. Метод Chase

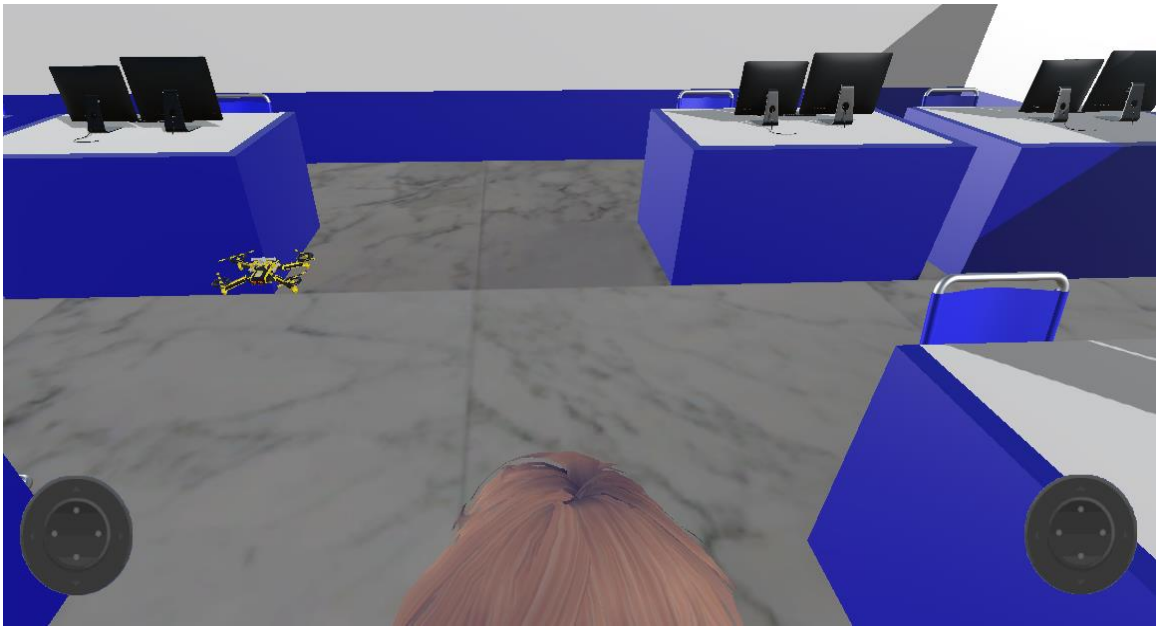


Рисунок 3.22 - Дрон летить назустріч гравцю

Також є й інші дрони, які допомагають розгадувати загадки кімнат.

- **Система отримання завдань із бази даних.** Під час розгадування головоломок, в кімнаті можуть траплятися різного роду задачі: знайти загублену річ та вернути на місце, знайти код та відкрити двері та багато іншого, але основною задачею це буде розв’язати завдання які будуть братися із бази даних. Це можуть бути завдання як із одним так і з декількома варіантами відповідей та завдання відкритого характеру (написати власноруч відповідь).

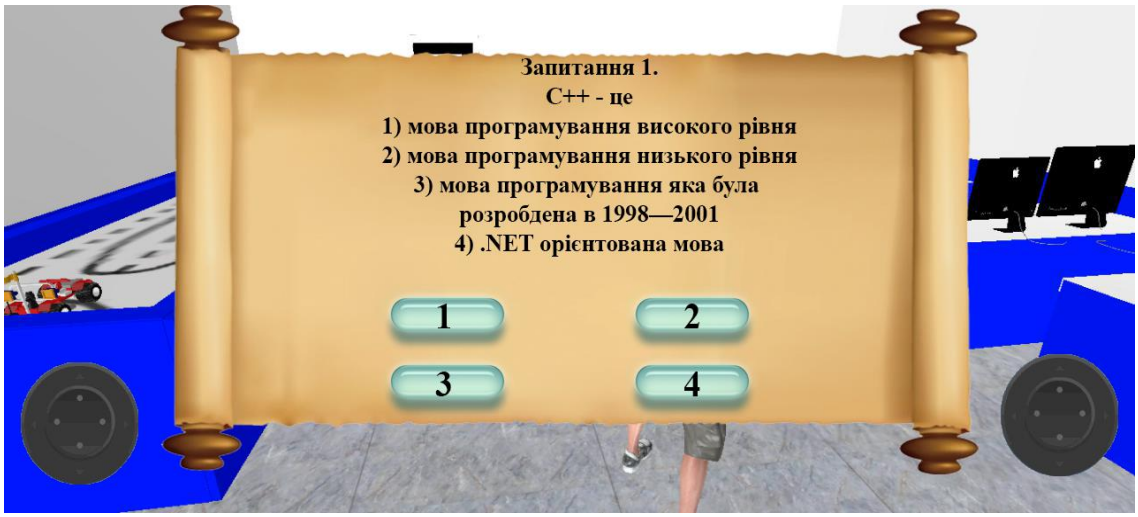


Рисунок 3.23 - Завдання з одним варіантом відповіді

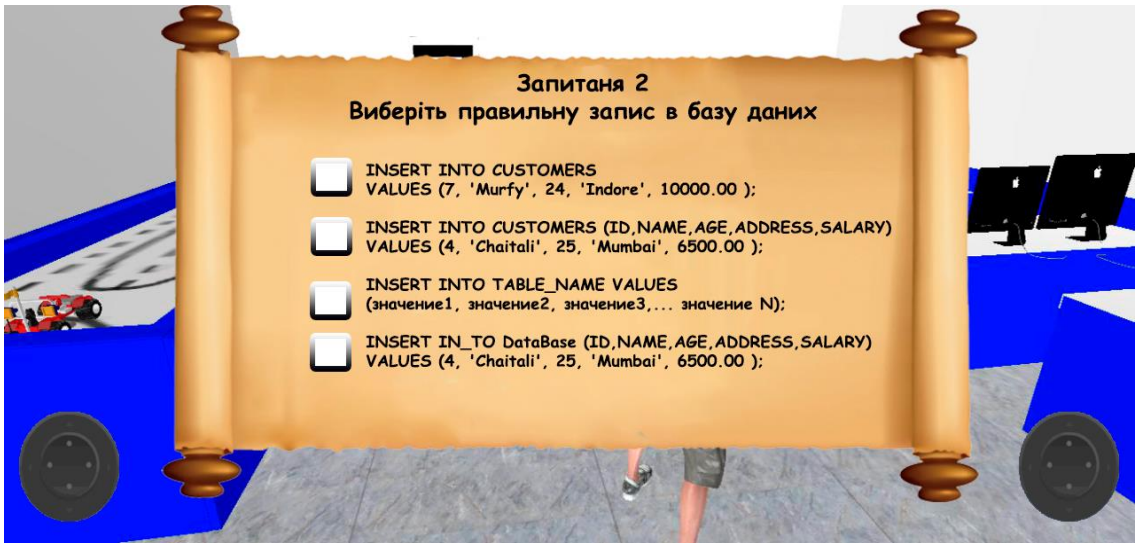


Рисунок 3.24 - Завдання з декількома варіантами відповідей

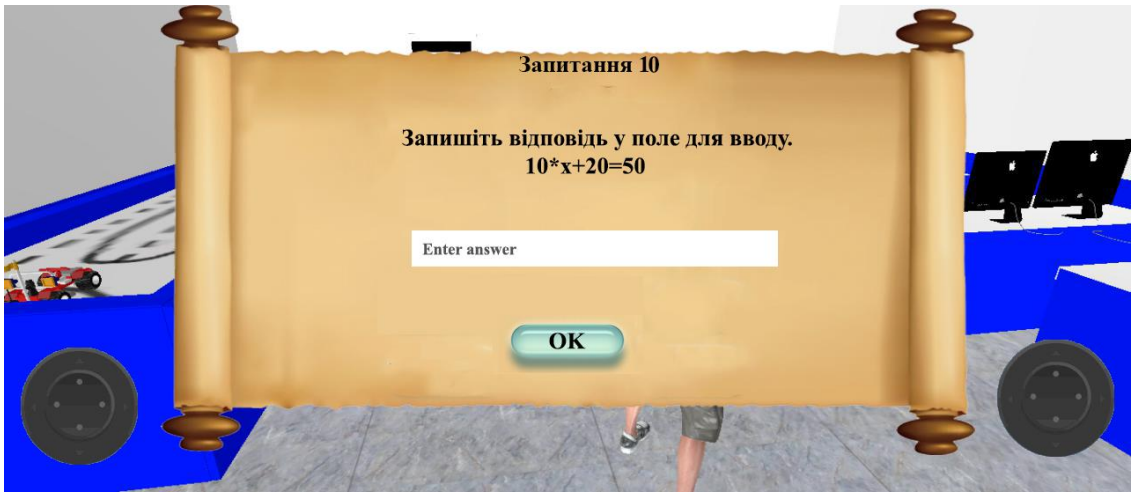


Рисунок 3.25 - Завдання відкритого характеру

Після кожної неправильної відповіді, у гравця згорає одна спроба продовжити гру, та поки є можливість, продовжує гру на збереженому місці, інакше розпочинає гру заново.

- **Система запису завданнь в базу даних та зчитування з неї (клієнтська частина адміністратора).** База даних зберігається у форматі JSON. Адміністратор несе відповідальність за створення та редагування завдань, які потім вносить до бази даних. Для зчитувати завдання з бази даних було описано метод *ReadQuestion*. В ньому визначено клас *StreamReader*, який і відповідає за

зчитування інформації, а змінна *_dataPATH* зберігає в собі шлях до файлу, який потрібно зчитати.

```
public void ReadQuestions()
{
    using (StreamReader sr = new StreamReader(_dataPATH))
    {
        string json = sr.ReadToEnd();
        _questions = JsonUtility.FromJson<Questions>(json);
    }
    _arrQuestion = _questions._serializationClass;
}

IEnumerator Read_Write_Questions()
{
    int a = 1;
    while (a<6) {
        yield return new WaitForSeconds(0.1f);
        GameObject _temp = Instantiate(_prefabPanelQuestion, _parentPanelQuestion);
        _temp.transform.GetChild(0).gameObject.GetComponent<TMP_Text>().text = a.ToString();
        _temp.transform.GetChild(1).gameObject.GetComponent<TMP_Text>().text = _questions._serializationClass[a].Question;
        a++;
    }
}
```

Рисунок 3.26 - Зчитування з бази даних

Для запису завдання в базу даних було описано метод *CreateQuestion*. В ньому було створено перевірку, якщо строки, які повинен заповнити адміністратор не пусті, то тільки тоді і записуємо ті дані, які користувач записав у поле для вводу. Інакше говоримо, що були заповнені не всі поля.

```
public void CreateQuestions()
{
    if (_questionINP.text !="" && _answerINP.text !="") {
        _intVal.i++;
        Array.Resize(ref _arrQuestion, _intVal.i);
        _arrQuestion[_intVal.i - 1] = new SerializationClass() { Question = _questionINP.text, Answer = _answerINP.text };
        _questions._serializationClass = _arrQuestion;
        WriteQuestions();
    }
    else
    {
        StartCoroutine(ErrorWait());
    }
}
IEnumerator ErrorWait()
{
    _errorPanel.SetActive(true);
    yield return new WaitForSeconds(1);
    _errorPanel.SetActive(false);
}
```

Рисунок 3.27 - Створення завдання

а в методі *CreateQuestion* визвали метод *WriteQuestion*, який саме і записує дані в базу даних. В цьому методі визначено клас *StreamWriter*, який і відповідає за запис інформації, а змінна *_dataPATH* зберігає в собі шлях до файлу, в який потрібно записати дані.

```
public void WriteQuestions()
{
    using (StreamWriter sw = new StreamWriter(_dataPATH))
    {
        string _data = JsonUtility.ToJson(_questions);
        sw.Write(_data);
    }
    _questionINP.text = _answerINP.text = null;
}
```

Рисунок 3.28 - Запис в базу даних

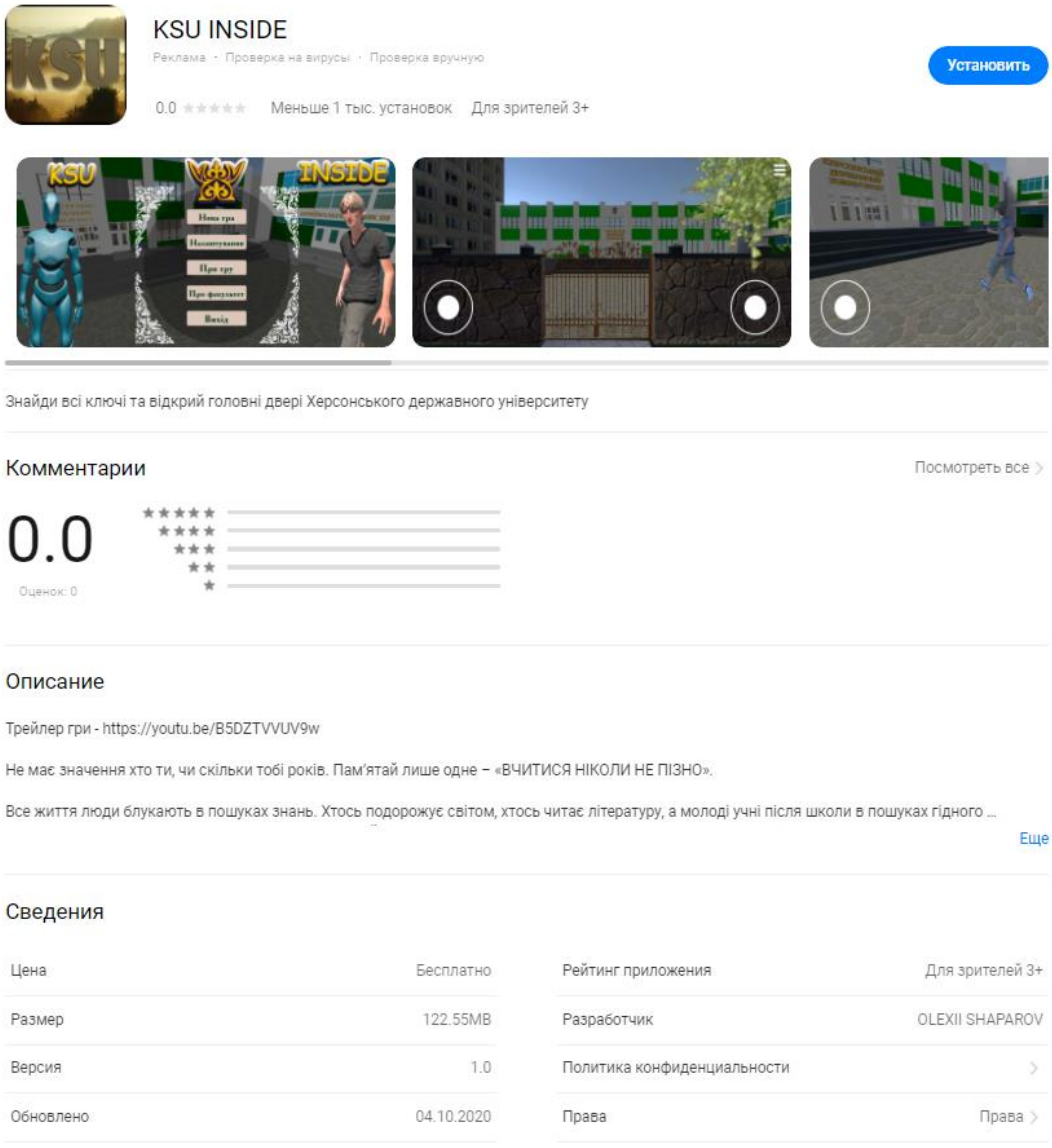


Рисунок 3.29 - Публікація гри в AppGallery

Гра була одобрена адміністрацією магазину додатків AppGallery та опублікована в цьому ж магазині.

ВИСНОВКИ

В ході дипломної роботи було:

1. Проаналізовано на основі критичного аналізу фахових традиційних та електронних джерел дефініції «гейміфікація» та «гейміфікований підхід».
2. Проведено порівняльний аналіз підходів розробки мобільних ігор та визначено середовище розробки проєкта.
3. Розглянуто та проаналізовано класифікації мобільних ігор:
    - Класифікація за кількістю гравців
    - Класифікація за кількістю стратегій
    - Класифікація за характером взаємовідносин між гравцями
    - Класифікація за властивостями функцій виграшів
    - Класифікація за кількістю ходів
    - Класифікація за інформованістю гравців
    - Класифікація за платформою
    - Класифікація за віртуальною інтернет платформою
    - Класифікація за жанрами
    - Класифікація по розташуванню ігрової камери
4. Визначено структуру та функціонал додатку
5. Побудовано UML діаграми:
    - Діаграма варіантів (Use Case)
    - Діаграма послідовностей
    - Діаграма стану
    - Діаграма класів
6. На основі визначеного функціоналу спроектовано та розроблено інформаційно-довідковий мобільний додаток “KSU_Inside”, який є засобом опосередкованого впливу на потенційних абітурієнтів з метою

формування у них позитивної мотивації щодо навчання у Херсонському державному університеті.

7. Опубліковано інформаційно-довідковий мобільний додаток “KSU_Inside” в магазині додатків AppGallery. Посилання на публікацію - https://appgallery.huawei.com/#/app/C103001865

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Unreal Engine – особливості, достатки та недоліки [Електронний ресурс]. – Режим доступу: https://cubiq.ru/dvizhok-unreal-engine/
2. Топ 10 кращих ігор зроблених за допомогою Unity [Електронний ресурс]. – Режим доступу: https://gamedata.club/article-top/1359-top-10-luchshih-igr-na-dvizhke-unity.html
3. Теория игр: Программа учебной дисциплины и методические указания к выполнению контрольной работы / Сост. Л. В. Березина; РГАТА. Рыбинск, 2011. с. (Заочная форма обучения / РГАТА).
4. Введення в теорію ігор / Н. Н. Писарук. — Минск : БДУ, 2015. — 256 с
5. Unity – Manual: Animator Component [Informational resource] – Access mode:https://docs.unity3d.com/Manual/class-Animator.html Мова:англ.
6. Joseph Hocking — Unity in Action. Multiplatform game development in C# with Unity 5 Chapter 1, 2015, p. 24
7. Shiva3D [Електронний ресурс]. – Режим доступу: https://shiva-engine.com/
8. Рихтер, Дж. CLR via C#. Программирование на платформе Microsoft .NET Framework 4.0 на языке C#/ Дж.Рихтер.– 3-е изд. – СПб.: Питер,2012. –928 с.
9. Беляков, А. Ю. Объектно-ориентированное программирование : учебное пособие / А.Ю. Беляков; М-во с.-х. РФ; федеральное гос. бюджетное об- разов. учреждение высшего образования «Пермская гос. с.-х. акад. им. акад. Д.Н. Прянишникова». – Пермь : ИПЦ «Прокрость», 2017. – 88 с.
10. Павловская Т.А. C#. Программирование на языке высокого уровня : учебник для вузов. – СПб. : Питер, 2014. – 432 с.

11. Агуров П.В. C#. Разработка компонентов в MS Visual Studio 2005/2008. СПб. : БХВ-Петербург, 2008. – 480 с.
12. Объектно-ориентированное программирование / Г.И. Радченко, Е.А. Захаров. – Челябинск: Издательский центр ЮУрГУ, 2013. – 167 с.
13. C# для начинающих. — СПб.: БХВ-Петербург, 2014. — 432 с.: ил.
14. Unity в действии. Мультиплатформенная разработка на C#. 2-е межд. изд. — СПб.: Питер, 2019. — 352 с.: ил. — (Серия «Для профессионалов»).
15. Ларман К. Застосування UML 2.0 та шаблонів проектування, 3-є видання / Пер. з англ. – М.: «вид-во Вільямс», 2007. – 736с.: іл.
16. Pender T.A. UML Weekend Crash Course. Wiley Publishing Inc., 2002. 358 p
17. Bell D. UML Basics: The class diagram. An introduction to structure diagrams in UML 2 [Informational resource] // IBM Developer Works // Access mode: http://www.ibm.com/developerworks/rational/library/content/RationalEdge/sep04/bell/
18. Макконнелл С. Совершенный код. Практическое руководство по разработке программного обеспечения. 2-е изд. / Пер. с англ.- М.: Русская редакция, СПб.: Питер, 2005.- 896 с.: ил.
19. Грейди Буч. Язык UML. Руководство пользователя / Грейди Буч, Джеймс Рамбо, Айвар Джекобсон. — СПб.: Питер, 2004. — 432 с.
20. Unity Game Engine [Informational resource] // Unity Game Engine. – 2018. Access mode: https://unity3d.com
21. Unity – Manual: Networking Overview [Informational resource] – Access mode:https://docs.unity3d.com/Manual/UNetOverview.html Мова:англ.
22. Огляд ігрового движка Unity3d [Електронний ресурс]. – Режим доступу: https://gamedevmania.ru/engines/unity3d

23. Кононюк А. Е. К65 Узагальнена теорія моделювання. Начала. К.1.Ч.1 К.4:"Освіта України", 2012. - 602 с.
24. Кенни Ламмерс Шейдеры и эффекты в Unity. Книга рецептов / пер. с англ. Шапочкин Е.А., под редакцией Симонова В. В. – М.: ДМК Пресс, 2016. – 274 с.: ил.
25. Kaner C., Bach J., Pettichord B. Lessons Learned in Software Testing. – USA: Wiley, 2001. – 320 p.
26. Це вам не іграшки: темна сторона гейміфікації [Електронний ресурс]. – Режим доступу: https://newtonew.com/discussions/gamificationdark-side
27. Миловская О. 3ds Max 2017. Дизайн интерьеров и архитектуры. – СПб.: Питер, 2017. – 416 с.: ил.
28. Малова Н. А. ArchiCAD 18 в примерах. Русская версия. – СПб.: БХВ-Петербург, 2015. – 480 с.: ил.
29. Габідулін В. М. Тривимірне моделювання в AutoCAD 2016. – М.: ДМК Пресс, 2016. – 2017 с.
30. Ларман К. Застосування UML 2.0 та шаблонів проектування. Вступ в об’єктно-орієнтований аналіз, проектування та ітеративну розробку. – М.: Вільямс, 2013. – 736 с.
31. Пацей, Н. В. Об'єктно-орієнтоване програмування: навч.-метод. посібник: в 2 ч./ Н. В. Пацей. – Мінськ: БДТУ, 2013.– Ч. 2.– 186 с
32. Меженин А.В. Технологии 3d моделирования для создания образовательных ресурсов. Учебное пособие. – СПб., 2008. - 112 с.
33. Шильдт, Герберт. C# 4.0: повне керівництво.: Пер. з англ. — М.: ООО " вид-во Вільямс", 2011. — 1056 с.: іл. — Парал. тит. англ
34. 3ds Max. Трехмерное моделирование и анимация на примерах / В. Т. Тозик, А. В. Меженін, К. А. Звягин. — СПб.: БХВ-Петербург, 2008. — 880 с.: ил. + Видеоуроки (на CD-ROM)
35. 3ds Max 2014. — СПб.: БХВ-Петербург, 2014. — 512 с.: ил. + Видеокурс — (В подлиннике)

36. І. Б. Аббасов. Основи тривимірногоо моделювання в графічній системі 3ds Max 2018: навчальний посібник. 3-є вид-во. Перероблене – М.: ДМК Пресс, 2017.-186 с.
37. Роллингз, Эндрю, Моррис, Дейв. Проектирование и архитектура игр, : Пер. с англ. – М.: Издательский дом «Вильямс», 2006. – 1040 с.: ил. – Парал. тит. Англ.
38. Новітні комп'ютерні технології. – Кривий Ріг : Видавничий центр ДВНЗ «Криворізький національний університет», 2017. – Том XV. – 281 с. : іл.
39. Євгеній Карасюк. Що таке гейміфікація [Електронний ресурс]. – Режим доступу: http://delo.ua/lifestyle/chto-takoe-gejmifikacija-i-kak-ona-pomogaetrasshevelit-sotrudni-202074
40. Принципи Гейміфікації [Електронний ресурс]. – Режим доступу: https://www.softservebs.com/uk/solutions-2/gejmifikatsiya-nematerialna-motivatsiya/
41. Unity – Manual: UnityWebRequest [Informational resource] – Access mode: https://docs.unity3d.com/Manual/UnityWebRequest.html Мова:англ.
42. Геймификация в бизнесе: как пробиться сквозь шум и завладеть вниманием сотрудников и клиентов / Гейб Зикерманн, Джоселин Линдер: Манн, Иванов и Фербер; Москва; 2014
43. Новітні комп'ютерні технології. – Кривий Ріг : Видавничий центр ДВНЗ «Криворізький національний університет», 2017. – Том XV. – 281 с. : іл.
44. Евплова Е. В. Геймификация как средство повышения мотивации к обучению // Одинцовские чтения. М., 2013. URL: http://evplova.ru/nauchnye-i-metodicheskie-stati/53-gejmifikatsiya-kak-sredstvo-povysheniya-motivatsii-k-obucheniya (дата обращения: 12.05.2015).

45. Herger M. Enterprise Gamifi cation. 2012. URL: http://enterprise-gamifi cation.com/index.php/de/blog/4-blog/79-the-gamifi cation-tipping-point (дата обращения: 12.05.2015).
46. Игрофикация // Википедия – свободная энциклопедия. URL: https://ru.wikipedia.org (дата обращения: 12.05.2015).
47. Кельберер Г. Р. Перспективы применения принципов игрофикации в подготовке педагогических кадров // Педагогическое образование и наука. 2014. № 4. С. 144–147.
48. Вестник Томского государственного педагогического университета. – Томск : Учреждение высшего профессионального образования «Томский государственный педагогический университет», 2015. - Выпуск 9 (162). — 328 с.
49. Бугайчук К.Л. Гейміфікація у навчанні: сутність, переваги, недоліки.—Харків : Харківський національний університет внутрішніх справ.— 5с. : іл.
50. Ігри в освітньому процесі [Електронний ресурс]. – Режим доступу: http://www.edtoday.ru/poleznye-stati/100-mozhet-li-gejmifikatsiyaspasti-obrazovanie
51. Геймифікація в освіті [Електронний ресурс]. – Режим доступу: http://gamemotiv.ru/top-10primerov-geymifikatsii-v-obrazovanii-kotoryie-izmenyatnashe-budushhee/
52. Гра як інструмент: що таке гейміфікація? [Електронний ресурс]. – Режим доступу: https://mistosite.org.ua/en/articles/hra-iak-instrument-shcho-take-heimifikatsiia
53. Кенни Ламмерс. Шейдеры и эффекты в Unity. Книга рецептов / пер. с англ. Шапочкин Е.А., под редакцией Симонова В.В. – М.: ДМК Пресс, 2016. – 274 с.: ил.
54. Вольф Д. OpenGL 4. Язык шейдеров. Книга рецептов / пер. с англ. А.Н. Киселева. – М.: ДМК Пресс, 2015. – 368 с.: ил.

55. Горнаков С.Г. Инструментальные средства програмирования и отладки шейдеров в DirectX и OpenGL – СПб.: БХВ-Петербург, 2005. – 256 с.: ил.
56. Боресков А.В. Расширения OpenGL. – СПб.: БХВ-Петербург, 2005. – 688 с.: ил.
57. Класифікація ігор [Електронний ресурс]. – Режим доступу: https://pidruchniki.com/11970524/ekonomika/klasifikatsiya_igor
58. Схема жанрів мобільних ігор [Електронний ресурс]. – Режим доступу: http://www.gamer.ru/everything/shema-zhanrov-kompyuternyh-igr
59. Шерман М.І. Удосконалення алгоритму оцінки теоретичних знань з інформатики при автоматизованому контролі. Збірник наукових праць міжнародної науково-технічної конференції ―СИЭТ-2000: Сучасні інформаційні та енергозберігаючі технології життєзабезпечення людини.―, випуск №8. К.:, ФАДА, ЛТД‖.- 2000. – с. 294 -297.
60. Шерман М.І. Методичні вимоги до тестово-контролюючих комп’ютерних програм. Збірник наукових праць міжнародної науково-технічної конференції ―СИЭТ9-11-02: Сучасні інформаційні та енергозберігаючі технології життєзабезпечення людини.―, випуск №11. К.:, МП ―Леся‖.- 2002. – с. 20 – 24.
61. Шерман М.І., Удовіченко І.Є. Інженерно–педагогічні вимоги до тестових комп’ютерних програм. // Педагогіка і психологія формування творчої особистості: проблеми і пошуки: Збірник наукових праць. Вип.25. Київ-Запоріжжя, 2002 . – С. 311-315.

ДОДАТКИ

Додаток А

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;

public class PlayerController : MonoBehaviour
{
        [SerializeField] private CharacterController controller;
        [SerializeField] private float speed;
        [SerializeField] private float gravity;
        [SerializeField] private Transform groundCheck;
        [SerializeField] private LayerMask groundMask;
        [SerializeField] private Animator animator;

        private Vector3 velocity;
        private float groundDistance = 0.5f;
        private bool isGrounded;
        void Start()
        {
                speed=10;
        }

    void Update()
    {
                isGrounded = Physics.CheckSphere(groundCheck.position,
groundDistance, groundMask);
                if (isGrounded && velocity.y < 0)
```

```
        {
            velocity.y = -0.5f;
        }
        float xAxis = Input.GetAxis("Horizontal");
        float yAxis = Input.GetAxis("Vertical");
        Vector3 move = transform.right * xAxis + transform.forward *
yAxis;
        controller.Move(move * speed * Time.deltaTime);
        velocity.y += gravity * Time.deltaTime;
        controller.Move(velocity * Time.deltaTime);


        if (yAxis == 0 || xAxis==0)
        {
            animator.SetBool("running", false);
            animator.SetBool("backRun", false);
            animator.SetBool("rightStrafe", false);
            animator.SetBool("leftStrafe", false);
        }
        if (yAxis > 0)
        {
            animator.SetBool("running", true);
        }

        if (yAxis < 0)
        {
            animator.SetBool("backRun", true);
        }

        if (xAxis > 0)
```

```
            {
                animator.SetBool("rightStrafe", true);
            }

            if (xAxis < 0)
            {
                animator.SetBool("leftStrafe", true);
            }

        }
    }
```

Додаток Б

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;

public class CameraLook : MonoBehaviour
{
    [SerializeField] private Transform player;
    [SerializeField] private float sensetivity;
    float xRotation = 0;
    float yRotation = 0;
    float xRotationClampMin = -60;
    float xRotationClampMax = 60;

    void Start()
    {
        sensetivity = 10;
        Cursor.lockState = CursorLockMode.Locked;
    }

    void Update()
    {
        float mouseX = Input.GetAxis("Mouse X") * sensetivity *
Time.deltaTime;
        float mouseY = Input.GetAxis("Mouse Y") * sensetivity *
Time.deltaTime;

        xRotation -= mouseY;
```

```
        xRotation = Mathf.Clamp(xRotation, xRotationClampMin,
xRotationClampMax);

        transform.localRotation = Quaternion.Euler(xRotation, 0, 0);
        player.Rotate(Vector3.up * mouseX);
    }
}
```

Додаток В

КОДЕКС АКАДЕМІЧНОЇ ДОБРОЧЕСНОСТІ
ЗДОБУВАЧА ВИЩОЇ ОСВІТИ ХЕРСОНСЬКОГО
ДЕРЖАВНОГО УНІВЕРСИТЕТУ

Я, Шапарьов Олексій Юрійович, учасник освітнього процесу Херсонського державного університету, **УСВІДОМЛЮЮ**, що академічна доброчесність – це фундаментальна етична цінність усієї академічної спільноти світу.

ЗАЯВЛЯЮ, що у своїй освітній і науковій діяльності **ЗОБОВ'ЯЗУЮСЯ**:

– дотримуватися:

- вимог законодавства України та внутрішніх нормативних документів університету, зокрема Статуту Університету;
- принципів та правил академічної доброчесності;
- нульової толерантності до академічного плагіату;
- моральних норм та правил етичної поведінки;
- толерантного ставлення до інших;
- дотримуватися високого рівня культури спілкування;

– надавати згоду на:

- безпосередню перевірку курсових, кваліфікаційних робіт тощо на ознаки наявності академічного плагіату за допомогою спеціалізованих програмних продуктів;
- оброблення, збереження й розміщення кваліфікаційних робіт у відкритому доступі в інституційному репозитарії;
- використання робіт для перевірки на ознаки наявності академічного плагіату в інших роботах виключно з метою виявлення можливих ознак академічного плагіату;

– самостійно виконувати навчальні завдання, завдання поточного й підсумкового контролю результатів навчання;

– надавати достовірну інформацію щодо результатів власної навчальної (наукової, творчої) діяльності, використаних методик досліджень та джерел інформації;

– не використовувати результати досліджень інших авторів без використання покликань на їхню роботу;

– своєю діяльністю сприяти збереженню та примноженню традицій університету, формуванню його позитивного іміджу;

– не чинити правопорушень і не сприяти їхньому скоєнню іншими особами;

– підтримувати атмосферу довіри, взаємної відповідальності та співпраці в освітньому середовищі;

– поважати честь, гідність та особисту недоторканність особи, незважаючи на її стать, вік, матеріальний стан, соціальне становище, расову належність, релігійні й політичні переконання;

– не дискримінувати людей на підставі академічного статусу, а також за національною, расовою, статевою чи іншою належністю;

– відповідально ставитися до своїх обов'язків, вчасно та сумлінно виконувати необхідні навчальні та науково-дослідницькі завдання;

– запобігати виникненню у своїй діяльності конфлікту інтересів, зокрема не використовувати службових і родинних зв'язків з метою отримання нечесної переваги в навчальній, науковій і трудовій діяльності;

– не брати участі в будь-якій діяльності, пов'язаній із обманом, нечесністю, списуванням, фабрикацією;

– не підроблювати документи;

– не поширювати неправдиву та компрометуючу інформацію про інших здобувачів вищої освіти, викладачів і співробітників;

– не отримувати і не пропонувати винагород за несправедливе отримання будь-яких переваг або здійснення впливу на зміну отриманої академічної оцінки;

– не залякувати й не проявляти агресії та насильства проти інших, сексуальні домагання;

– не завдавати шкоди матеріальним цінностям, матеріально-технічній базі університету та особистій власності інших студентів та/або працівників;

– не використовувати без дозволу ректорату (деканату) символіки університету в заходах, не пов'язаних з діяльністю університету;

– не здійснювати і не заохочувати будь-яких спроб, спрямованих на те, щоб за допомогою нечесних і негідних методів досягати власних корисних цілей;

– не завдавати загрози власному здоров'ю або безпеці іншим студентам та/або працівникам.

УСВІДОМЛЮЮ, що відповідно до чинного законодавства у разі недотримання Кодексу академічної доброчесності буду нести академічну та/або інші види відповідальності й до мене можуть бути застосовані заходи дисциплінарного характеру за порушення принципів академічної доброчесності.

05.05.2020		Олексій Шапарьов
(дата)	(підпис)	(ім'я, прізвище)