**Форма № Н-9.02**

Національний університет «Полтавська політехніка імені Юрія Кондратюка»

(повне найменування вищого навчального закладу)

Навчально-науковий інститут інформаційних технологій та робототехніки

(повна назва факультету)

Кафедра комп’ютерних та інформаційних технологій і систем

(повна назва кафедри)

**Пояснювальна записка**

**до дипломного проекту (роботи)**

бакалавра

(рівень вищої освіти)

на тему

Використання 4D графіки при створенні системи контролю

знань учнів навчальних закладів

Виконала: студентка 4 курсу, групи 401-ТН спеціальності

122 Комп’ютерні науки

(шифр і назва спеціальності)

Колесник Д.Д.

(прізвище та ініціали)

Керівник \_\_\_\_ \_\_ Гайтан О.М.

(прізвище та ініціали)

Рецензент \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(прізвище та ініціали)

Полтава –2021року

**МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ**

**НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ «ПОЛТАВСЬКА ПОЛІТЕХНІКА ІМЕНІ ЮРІЯ КОНДРАТЮКА»**

**НАВЧАЛЬНО-НАУКОВИЙ ІНСТИТУТ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ ТА РОБОТОТЕХНІКИ**

**КАФЕДРА КОМП’ЮТЕРНИХ ТА ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ І СИСТЕМ**

**КВАЛІФІКАЦІЙНА РОБОТА БАКАЛАВРА**

**Спеціальність 122 «Комп’ютерні науки»**

**на тему**

**«Використання 4Dграфіки при створенні системи контролю знань учнів навчальних закладів»**

**Студентки групи 401-ТН Колесник Дар’ї Дмитрівни**

Керівник роботи старший викладач, Гайтан О.М.

Завідувач кафедри

кандидат технічних наук, доцент Головко Г.В.

Полтава–2021

**РЕФЕРАТ**

Кваліфікаційна робота бакалавра: 74 с., 53 рисунки, 2 таблиці, 2 додатки 15 джерел.

**Об’єкт дослідження**: системи контролю знань, використання комп’ютерної графіки для збільшення загальної зацікавленості навчанням учнів середніх навчальних закладів.

**Мета роботи**: створення гри жанру тест з використанням 4Dграфіки та анімації для перевірки знань учнів. У результаті буде створено самостійний програмний продукт для безкоштовного користування.

**Методи**:3D моделювання, методи мозкового штурму, системного аналізу, теорії ігор та експертних оцінок.

**Ключові слова:** вікторина, 3D-графіка, Cinema 4D, Unity 3D, комп’ютерна гра.

Результати виконання кваліфікаційної роботи бакалавра були апробовані на 73-й науково-практичній конференції Національного університету «Полтавська політехніка імені Юрія Кондратюка».

**ABSTRACT**

Bachelor's qualification work: 74 pages, 53 drawings, 2 tables,2 appendix, 15 sources.

**Object of research:** knowledge control systems, the use of computer graphics to increase the general interest in teaching students of secondary schools.

**Purpose:** creating a test genre game using 4D graphics and animation to test students' knowledge. As a result, a standalone software product will be created for free use.

**Methods:** 3D modeling, brainstorming, systems analysis, game theory and expert evaluations.

Keywords: quiz, 3D graphics, Cinema 4D, Unity 3D, computer game.

The results of the bachelor's qualification work were tested at the 73rd scientific-practical conference of the National University "Poltava Polytechnic named after Yuri Kondratyuk".

**ЗМІСТ**

[ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СКОРОЧЕНЬ І ТЕРМІНІВ 6](#_Toc74859129)

[РОЗДІЛ 1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ ТА ПОСТАНОВКА ЗАДАЧІ 8](#_Toc74859130)

[1.1.Огляд предметної області 8](#_Toc74859131)

[1.2.Системи контролю знань та приклади їх використання 10](#_Toc74859132)

[1.3.Постановка задачі 17](#_Toc74859133)

[РОЗДІЛ 2 ТЕХНОЛОГІЇ 3D МОДЕЛЮВАННЯ ТА СТВОРЕННЯ КОМП’ЮТЕРНИХ ІГОР 18](#_Toc74859134)

[2.1.Поняття 4D простору 18](#_Toc74859135)

[2.2.3D графіка та принципи її створення 19](#_Toc74859136)

[2.3.Етапи розробки комп’ютерної гри 26](#_Toc74859137)

[РОЗДІЛ 3 ВИБІР РІШЕНЬ І ЇХ ПРАКТИЧНА РЕАЛІЗАЦІЯ 32](#_Toc74859138)

[3.1.Створення ігроладу 32](#_Toc74859139)

[3.2.Вибір програмного забезпечення для створення 3D-моделей 33](#_Toc74859140)

[3.3.Вибір двигуну для розробки гри 38](#_Toc74859141)

[3.4.Розробка ігрового оточення 40](#_Toc74859142)

[3.4.1.Моделі ландшафту 41](#_Toc74859143)

[3.4.2.Моделі початкової сцени 41](#_Toc74859144)

[3.4.3.Моделі другої сцени 44](#_Toc74859145)

[3.4.4.Моделі третьої сцени 47](#_Toc74859146)

[3.4.5.Моделі четвертої сцени 49](#_Toc74859147)

[3.4.6.Моделі фінальної сцени при правильних відповідях 50](#_Toc74859148)

[3.4.7.Моделі фінальної сцени при неправильних відповідях 52](#_Toc74859149)

[3.5.Розробка механіки та інтерфейсу 54](#_Toc74859150)

[РОЗДІЛ 4 ТЕСТУВАННЯ 57](#_Toc74859151)

[ВИСНОВКИ 59](#_Toc74859152)

[СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ 60](#_Toc74859153)

[ДОДАТОК А ОСНОВНІ ЧАСТИНИ ПРОГРАМНОГО КОДУ 62](#_Toc74859154)

[ДОДАТОК В УЧАСТЬ У КОНФЕРЕНЦІЇ 71](#_Toc74859155)

# ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СКОРОЧЕНЬ І ТЕРМІНІВ

**3D**– тривимірний.

**ЗМІ** – засоби масової інформації.

**ПЗ** – програмне забезпечення.

**2D** – двовимірний.

**4D** – чотиривимірний.

**VFX** – візуальні ефекти.

**ВСТУП**

У наш час існує тенденція до втрати зацікавленості навчанням серед учнів і тому важливо знайти спосіб для урізноманітнення способів передачі інформації та перевірки її засвоєння. Логічним є використання об’єктів, що мають високий рівень популярності серед даної групи людей та впровадження його у навчальний процес, тобто використання комп’ютерних ігор з ціллю надання знань.

У роботі розглядається розроблення системи контролю знань для учнів навчальних закладів з використанням 4Dграфіки. Для розробки графіки використано програмне забезпечення Cinema 4D, що є одним з найпростіших у освоєнні серед аналогів та має великий вибір необхідних функціональних можливостей, а також безкоштовну ліцензію. У якості ігрового двигуну використовувався Unity 3D – комплексна система програмного забезпечення, що має повний набір функцій для створення інтерфейсу та ігрової механіки.

Об’єктом дослідження є розробка систем контролю знань та використання комп’ютерної графіки для збільшення загальної зацікавленості навчанням у учнів середніх навчальних закладів.

Предметом досліджень є засоби розробки інформаційних систем з використанням 3D графіки.

Мета кваліфікаційної роботи є створення гри жанру тест з використанням 4Dграфіки та анімації для перевірки знань учнів. У результаті буде створено самостійний програмний продукт для безкоштовного користування.

Цільовою платформою програмного продукту, що розроблюється, є персональні комп’ютери з операційними системами Windows, Linuxта Mac. Результуюче програмне забезпечення повинне бути максимально простим у встановленні, за можливості – не потребувати встановлення зовсім або ж з мінімальним набором сторонніх операцій.

# РОЗДІЛ 1

**АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ ТА ПОСТАНОВКА ЗАДАЧІ**

У суспільстві, де більшість галузей стають комп'ютеризованими, освіта не може вічно стояти осторонь і не підлягати таким змінам. Комп’ютерні технології та електронні мережі повільно, але впевнено починають проникати у школи. Через широко розповсюджене використання таких технологій виникає необхідність у запроваджені систем контролю знань для покращення рівня зацікавленості навчанням серед учнів.

# Огляд предметної області

Система контролю знань має досить широке поняття і під цей опис підходить велика кількість сучасних програм. У даній роботі використовується комп’ютерна гра як система контролю знань. Основною задачею такої системи є перевірка знань гравця шляхом задання питань та отримання відповідей на них.

Існує два види освіти: дидактичний погляд (або передача інформації) і конструктивістський погляд. Серед широкого загалу панує дидактичний погляд. Він стверджує, що викладачі повинні бути майстрами певних сфер знань, а їхня робота полягає в тому, щоб передавати свої знання щодо цих сфер студентам за допомогою лекцій. Студенти повинні запам’ятовувати факти та концепції домену та відпрацьовувати його навички, поки не засвоїть їх, і вони повинні мати можливість продемонструвати цю майстерність на відповідних тестах.

Конструктивістська точка зору стверджує, що викладачі повинні виступати в ролі посередників, які допомагають студентам побудувати власні розуміння та можливості у виконанні складних завдань. Цей погляд робить акцент на діяльності студента, а не на діяльності вчителя. Незважаючи на переважання у провідних школах освіти, конструктивістський погляд мало просунувся у проникненні в державну освіту в Україні або, загальніше, у цілому світі [11].

У останні роки все частіше помічається тенденція до втрати ефективності методів, що застосовуються у навчанні. Досить жорсткий контроль учнів на заняттях, а також необхідність виконання деяких навчальних процедур призводить до нерозуміння студентами здійснюваних ними дій та до низького рівню усвідомленості отриманого матеріалу. Тому у учнів знижується рівень мотивації до навчання, а також вміння планувати свою діяльність.

Тому, найочевиднішим способом покращення мотивації стає впровадження комп’ютерних технологій в освітній процес. Їх можна використовувати в класі трьома способами:

1) як інструменти, наприклад – текстові процесори, електронні таблиці, мови програмування та електронні мережеві системи;

2) як інтегровані навчальні системи, що представляють вправи для індивідуальної роботи студентів та ведуть записи про успіхи учнів для звітування перед викладачем;

3) як симуляції та ігри, що залучають учнів до комп’ютерної діяльності, призначеної для мотивації та виховання.

Найефективнішим та найперспективнішим підходом є використання в навчанні комп'ютерних ігор – DigitalGameBasedLearning (DGBL). Ідея використання ігор для підвищення зацікавленості учнів такаж стара, як і ігри в цілому. Оскільки, загальновідомо, що інформація засвоюється краще, якщо ми можемо її використати на практиці. Адаптація ігор під навчання вийшла на новий рівень з появою цифрових ЗМІ. Цей новий цифровий спосіб засвоєння інформації, побудований на отриманні задоволення від процесу навчання за допомогою гри. Але також існує думка, що ставить під сумнів можливість використання ігор для підвищення ефективності навчання [12].

Причини, за якими учням подобаються комп'ютерні ігри:

* комп'ютерні ігри являють фантазії і слідують простим принципом: виграш чи програш з миттєвим результатом;
* ігри використовують естетичне моделювання і впізнавані риси для залучення до себе уваги учня з візуальної зворотним зв'язком;
* ігри являють собою інтерактивну середу і забезпечують повне занурення в неї;
* ігри відкривають різні способи вирішення проблем.

Навчальні ігри виконують 3 основні функції:

* інструментальна: формування певних навичок і вмінь;
* гностична: формування знань і розвиток мислення учнів;
* соціально-психологічна: розвиток комунікативних навичок.

Кожній функції відповідає певний тип гри: інструментальна функція може виражатися в ігрових вправах, гностична – в дидактичних, остання – в рольових іграх. Слід зазначити, що в навчальних іграх використовується не тільки ігровий метод як такий. В процесі гри можна застосовувати групову та індивідуальну роботу, спільне обговорення, проводити тестування і опитування, створювати рольові ситуації. Іншими словами, гра органічно поєднує і дозволяє використовувати різні методи – «Мозкового штурму», метод розвитку критичного мислення через читання та письмо, анкетування, соціометрії та ін. [13].

Разом з тим, в педагогіці ігровий метод має деяку специфіку. В процесі навчання гра часто використовується як допоміжний елемент, доповнення до теоретичного матеріалу і не може виступати в якості основного методу навчання. Але з іншого боку про результати застосування навчальних ігор в цілому свідчать численні дослідження світових фахівців, які відзначають, що ця технологія дозволяє підвищити ефективність навчання в середньому в три рази.

# Системи контролю знань та приклади їх використання

Ігри в навчанні можуть використовуватися для полегшення засвоєння інформації, а також для перевірки знань учня в системах контролю знань. Перевірка засвоєного матеріалу є невід’ємною частиною освітнього процесу.

Існує два основних напрямків у організації контролю знань:

1. Дії студента оцінуються за допомогою систем, що ґрунтуються на знаннях, тобто в експертно-навчальних системах. Рівень знань тут визначаються у ході діалогу між викладачем та студентом і покладається на знання з предметної області та правил оцінки.
2. Даний підхід полягає у оцінці знань студента за допомогою відповідей на спеціалізовану вибірку завдань. Саме з цих відповідей виносяться судження про його знання. Обов’язковим у такому підході є систематична перевірка знань учнів. Такий метод використовують доволі часто, оскільки він має ряд певних переваг:

* результати отримуються у числовій формі, що дозволяє проводити їх математичну обробку;
* невеликий обсяг витраченого часу для перевірки.

Ефективним методом перевірки знань при стандартизовану контролю є тестування. Системи тестування можуть бути схожі на тести, вікторини, опитування, а також можуть мати різний вигляд: звичайний текстовий, інтерактивний, с використання різної графіки та анімації [14].

При створені інтерфейсу потрібно враховувати декілька факторів:

* вік аудиторії;
* специфіку дисципліни;
* можливості обраної базової системи 3D моделювання.

У даній роботі розглядається створення системи контролю знань з використанням ігрової форми для школярів молодших класів.

Використання ігрової форми навчання дозволяє поєднувати теоретичні та практичні навички, а також застосовувати індивідуальну та колективну роботу, тим самим забезпечуючи комплексний підхід. У процесі гри запам’ятовування інформації відбувається на підсвідомому рівні, тому засвоєння теоретичного матеріалу та отримання практичних навичок відбувається одночасно. Було проведено дослідження в американських школах з використанням ігрової комп’ютерної програми Reasoning Mind [8].



Рисунок 1.1 – Інтерфейс комп’ютерної програми Reasoning Mind

Reasoning Mind являє собою комп’ютерну систему розроблену для навчання математики школярів. У системі використовуються різні анімовані персонажі, які потрапляють у різні пригоди та історії. Існує головний герой, що є одночасно наставником для користувачів і саме він допомагає школярам вивчати математику. За результатами дослідження виявилось, що рівень успішності в експериментальних групах виріс на 16 – 19%, розвиток уміння думати (на думку 100% опитаних викладачів) та значне покращення ставлення учнів до вивчення математики (на думку 94% опитаних викладачів). Опитування учнів показало, що більшість дітей (75%) надає перевагу саме такому виду навчання.

Також досить популярним є застосування програми об’єктно-орієнтованого середовища Alice (Аліса), що використовує анімацію та ігрові сюжети для навчання основам програмування.

Середовище Alice дозволяє управляти тривимірними об'єктами, створювати програми (комп'ютерні відеоігри і анімаційні програми-фільми), що генерують анімацію в віртуальних світах на основі використання умов, циклів, функцій / методів, обробки масивів даних, списків, об'єктів, успадкування, інкапсуляції, поліморфізму тощо.

Вона також має у своїй бібліотеці персонажів та інші об’єкти з Sims ™ – однієї з найбільш відомих та продаваних комп'ютерних відеоігор у світі.

У середовищі є велика бібліотека об'ємних об'єктів з реального світу (природа, тварини, інструменти побуту тощо). Їх можна рухати, обертати, змінювати колір і розмір за допомогою миші, а на основі отриманого віртуального світу програмно описувати анімацію і створювати ігрові моделі.

Вікна Alice дуже схожі на ті, що зустрічаються в професійних сучасних інтегрованих середовищах розробки візуальних додатків типу VisualStudio. Учневі не потрібно запам'ятовувати синтаксис будь-якої конструкції, він завжди може скористатися набором наявних процедур і функцій даного об'єкту або спливаючі підказки. Саме це дозволяє школярам в подальшому сконцентрувати свою увагу на сценарії гри, сцени, які використовуються об'єктах, їх властивості та методи, а не турбуватися про синтаксичних помилках своїх додатків.

Програмування в Alice частково нагадує роботу в середовищі Scratch. Процес написання гри нагадує гру «Пазли», коли необхідно зістикувати між собою програмні елементи, з яких збирається вся програма. Від учня лише іноді потрібно ввести з клавіатури константи або назва власних ідентифікаторів. Для наочності представленого коду блоки програмних елементів виділяються певним кольором залежно від їх типу. Ці блоки можна легко перетягувати, змінюючи порядок і вкладеність.

У програму можна підключати різні мультимедійні елементи, що містять відео, графіку і звук (\* .mov, \* .gif, \* .jpg, \* .bmp, \* .png, \* .tif, \* .mp3, \* .wav). Також допустимий імпорт готових моделей з інших програм, в тому числі з графічної програми «3DS MAX» зі збереженням доступу до окремих частин моделі[9].

Віртуальний світ Alice має три виміри простору (і часу), і кожен об'єкт має властивості: колір, розмір, позиція, напрям. В Alice є також камера, яка дозволяє побачити віртуальний світ на екрані комп'ютера в різних положеннях. Змінити властивості можна на етапі проектування інтерфейсу на вкладці properties і під час виконання програми. Для зміни значення властивості об'єкт вибирається в дереві об'єктів. Потім навпроти назви властивості зі списку можливих значень вибирається бажане.

Основні переваги Alice:

1. Високорівнева анімація, що дозволяє користувачам програмувати соціальні взаємодії між персонажами.

2. Підручник на основі розповіді знайомить користувачів з програмуванням через створення сюжету.

3. Галерея 3D-персонажів і декорації з призначеною для користувача анімацією дозволяють «оживляти» ідеї історії[9].



Рисунок 1.2 – Інтерфейс середовища Alice

Таким чином, за допомогою даного середовища студенти вчаться програмувати, одночасно отримуючи задоволення від створення 3D-анімації, історій і відеоігор.

За результатами досліджень в коледжі Ітаки і університеті Св. Йосипа з вивчення курсу «Комп'ютерні науки» студентами без досвіду програмування середня успішність покращилася з C до B, засвоєння інформації збільшилася з 47% до 88%. Також підвищився інтерес до програмування без будь-яких змін в основних програмних завданнях. Збільшення часу заняття програмуванням склало 42%[8].

З історичної точки зору проект Alice отримав досить широке поширення в США. Сьогодні за даними розробників середу використовують уже близько 10% американських коледжів. Проект створений і підтримується університетом Карнегі Меллона (CarnegieMellonUniversity) (CMU, cmu.edu). Середовище програмування Alice випускається для Windows, Mac OS і Linux в двох редакціях: основний (для вузів) і спрощеної (для шкіл).

CodeCombat– досить популярна гра для навчання програмуванню (Python, основи створення інтерактивних веб-сайтів з використанням HTML, CSS і JavaScript, розробка ігор у стиліCodeCombat). Програму використовують у навчальних закладах для отримання навичок з розробки комп’ютерних ігор. Має багато різних курсів, що призначені для різних вікових груп.

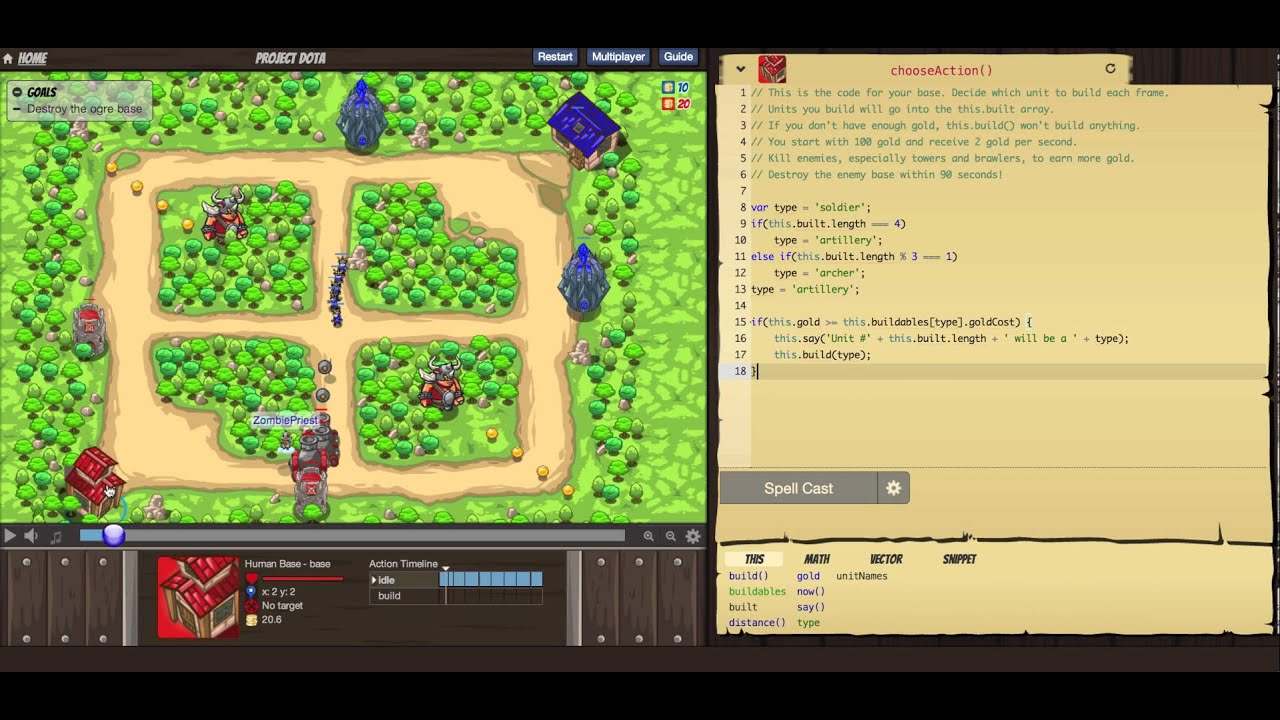


Рисунок 1.3 – Інтерфейс гриCodeCombat

CodeMonkey**–** це проста онлайнова гра, в процесі проходження якої діти можуть познайомитися з основами програмування. Гравець повинен скласти ланцюжок команд в правій текстової панелі, а потім запустити їх на виконання. Перед кожним рівнем даються підказки, так що впоратися із завданням зможуть навіть школярі молодших класів.



Рисунок 1.4 – Інтерфейс гри CodeMonkey

Scratch - це візуальне об'єктно-орієнтоване середовище програмування. Призначене для навчання школярів молодших і середніх класів основам створення власних ігор, анімацій і спільної роботи над проектами. Крім цих застосувань, Scratch можна використовувати для освітніх цілей і створювати в програмі ілюстративні матеріали для уроків не тільки по програмуванню, а й з історії, біології, фізики та інших предметів.

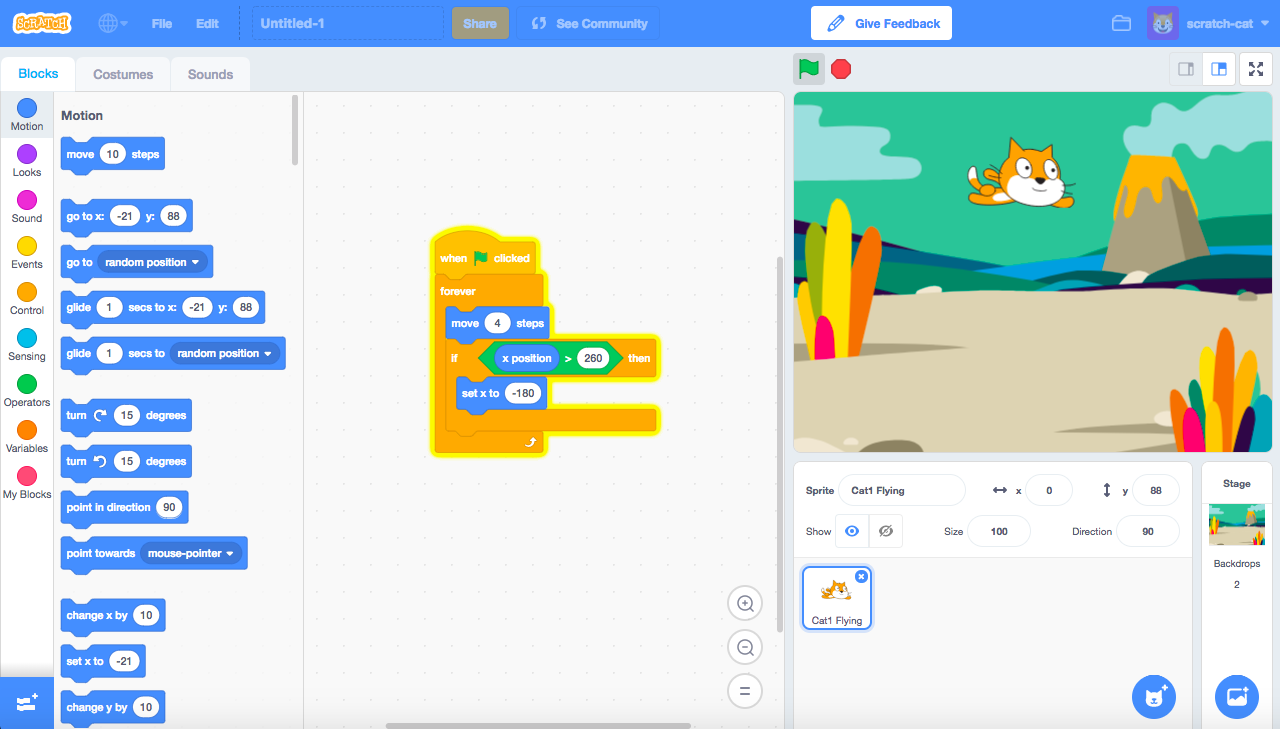


Рисунок 1.5 – Інтерфейс гри Scratch

# Постановка задачі

Спроектувати та програмно реалізувати систему контролю знань у вигляді комп’ютерної гри жанру тест з використанням 3Dграфіки та анімації для збільшення загальної зацікавленості навчанням учнів середніх навчальних закладів.

Функціональні вимоги до програмного забезпечення:

* можливість додавати питання;
* підтримка одиничних та множинних відповідей, можливість застосування таймеру;
* різні сценарії при правильній та неправильній відповіді;
* реалізація ландшафтів (сеттинг) в різних стилях: місто, пустеля, арктичний пейзаж, пляж, гірський пейзаж, болото.

Мінімальними вимоги до апаратної складової персонального комп’ютера є:

1. відео-картка, що потребує як мінімум 512 мегабайт оперативної пам’яті;
2. наявність на цільовому персональному комп’ютері вільної оперативної пам’яті в обсязі щонайменше 500 мегабайт;
3. об’єм оперативної пам’яті не менше 4 гігабайт.

Основні етапи процесу розробки:

1. Проектування гри.
2. Створення ігрового оточення.
3. Компонування сцен.
4. Програмування ігроладу.
5. Створення інтерфейсу.
6. Тестування та виправлення дефектів.

# РОЗДІЛ 2

**ТЕХНОЛОГІЇ 3D МОДЕЛЮВАННЯ ТА СТВОРЕННЯ КОМП’ЮТЕРНИХ ІГОР**

# Поняття 4Dпростору

4Dпростір досить складний для розуміння людей, оскільки ми живемо у тривимірному світі і людському розуму важко уявити об’єкт з додатковою перемінною. Перш ніж вникати в деталі чотирьох вимірів, нам слід зрозуміти, що таке перші кілька вимірів. Для початку візьміть точку, яка не має просторового масштабу – ми скажемо, що це 0D простір. Розтягування цієї точки створює перший вимір, який являє собою пряму лінію з шириною 0 і лише довжиною. Ви можете пересуватися лише двома способами – вперед або назад. 2D простір – це набір нескінченного 1D простору, розкинутого вздовж або вшир. Прикладом 2D фігури може бути квадрат. Є більше способів, за допомогою яких можна подорожувати у двох вимірах – вперед, назад, вліво та вправо. Тривимірний простір насправді є нескінченною купою двовимірного простору, складених один на одного. У тривимірному просторі є три координатні осі – зазвичай позначеніx, yта z. Шість напрямків у цьому просторі називаються: вгору, вниз, вліво, вправо, вперед і назад. Довжини, виміряні вздовж цих осей, можна назвати довжиною, шириною та висотою[15]. 4Dпростір має декілька варіантів представлення. Основними є:

1. З математичної точки зору чотиривимірним об’єктом може бути будь який об’єкт, що представлений чотирма перемінними. Теж саме місце зустрічі можливо задати за чотирма параметрами: координати, хвилини, секунди.
2. Також існує теорія, що вважає, 3D-об’єкти – це зріз 4D-об’єктів. Наприклад, якщо взяти куб та представити його у двовимірному просторі, то залишиться лише його зріз, тобто квадрат. Саме від цього відштовхується дана теорія.
3. Найвідомішою теорією є те, що четверта координата – час. Також прийнято вважати, що такий простір називається чотиривимірним простором-часом. Але саме ця теорія є найважчою для розуміння, оскільки тут просторові та часові координати відіграють зовсім різні ролі і існує досить багато розбіжностей між вченими щодо цієї думки.

У даній роботі чотиривимірний простір використовується з математичного погляду и задається чотирма координатами: x, y, zта переміщенням у часі моделі потягу.

# 3D графіка та принципи її створення

У3D комп'ютерній графіці, 3Dмоделювання являє собою процес розробки математичного уявлення будь-якої [поверхні](https://en.wikipedia.org/wiki/Surface_(mathematics)) об'єкта (неживої або живої) в[трьох вимірах](https://en.wikipedia.org/wiki/Three-dimensional_space) за допомогою спеціалізованого програмного забезпечення. Продукт називається3D-моделлю.Вона може бути представлена у вигляді програмного коду або відображена у вікні перегляду, як 3D-модель, а також за допомогою двовимірного зображення, що створюється за допомогою процесу [рендерингу](https://uk.wikipedia.org/wiki/%D0%A0%D0%B5%D0%BD%D0%B4%D0%B5%D1%80%D0%B8%D0%BD%D0%B3). 3D модель може бути фізично створена з використанням 3D-принтерів, які формують 2D-шари моделі із тривимірним матеріалом, по одному шару, що накладаються один на одного. З точки зору розробки ігор,3D-моделювання– це лише один з етапів у всьому процесі розробки.3D-моделі можуть створюватися автоматично або вручну.

[Програмне забезпечення для 3D-моделювання](https://en.wikipedia.org/wiki/3D_modeling_software) – це клас програм для 3D-комп’ютерної графіки, що використовується для створення 3D-моделей.Окремі програми цього класу називаються додатками моделювання.

3D-моделі, в даному випадку, є цифровими зображеннями фізичних об'єктів. Тривимірні моделі, як правило, складаються з полігональної сітки та текстури поверхні. Полігонна сітка – це «оболонка», що складається з різних поверхонь тривимірного об’єкта. З цієї оболонки складаються три основні компоненти: вершини (точки), ребра (лінії) та грані (площини).Що повинно бути зрозуміло з попереднього речення, багатокутні сітки часто називають просто «геометрією». Існують також інші технічні терміни, пов’язані із полігональними сітками [1].

Текстура поверхні, в основному, є зображенням, нанесеним на поверхню полігональної сітки. Текстура може бути простою, як однотонний колір, або настільки складною, як фотографія з високою роздільною здатністю. Текстура буде нанесена на поверхню геометрії за допомогою певного набору інструкцій. Ці текстури можуть мати фізичні властивості, які взаємодіють зі світлом, створюючи такі ефекти, як прозорість, відбиття, тіні тощо.

3D-моделі широко використовуються, і тому існують різні типи такого їх створення. Найбільш важливими є:

1. Полігональне моделювання. Це моделювання дозволяє виконувати різні маніпуляції з сіткою 3D-об'єктів на рівні під-об'єкта: вершини, ребра, грані. Само полігон являє собою набір граней, але в системах, що підтримують багатосторонні межі, полігони та межі будуть рівними.

Це перший та основний тип моделювання, оскільки за його допомогою можна створити об’єкт будь-якої складності, об’єднуючи різні групи полігонів. Полігональне моделювання поділяється на три типи:

• низькополігональні;

• середньополігональні;

• високополігональні.

Низькополігональне (Low-Poly) моделювання використовується для створення невеликої кількості дрібних об’єктів, як правило, для економії ресурсів, коли великі деталі не потрібні, та для створення низькополігональних ілюстрацій, які стають все більш популярними в останні роки;

Середньополігональне (Mid-Poly) моделювання зазвичай фокусується лише на бажаному результаті при рендерингу, тобто при моделюванні бажаної геометрії, наприклад, за допомогою логічних або булевих операцій; над полігональною сіткою не проводиться оптимізація або вона мінімальна;

Високополігональне (High Poly) моделювання зазвичай робить точну копію об’єкта з використанням великою кількості полігонів.

2. Сплайнове моделювання. Цей тип моделювання створює 3D-об’єкти за допомогою сплайну, тобто кривих ліній. Вони можуть мати різну форму: коло, прямокутник, дугу та багато інших. Об'єкти виходять гладкої форми, тому цей метод можна використовувати на органічних моделях, таких як рослини, люди, тварини тощо. Перевага цього методу полягає в тому, що форму сплайну можна гнучко змінювати.

Таке моделювання часто порівнюють з полігональним, як растрову з векторною графікою. Перевага векторної графіки полягає в тому, що при збільшенні об’єкта, його якість не змінюється, на відміну від растру, на якому видно пікселі. Подібним чином, якщо збільшити об’єкт, створений сплайнами, його якість залишається незмінною, а, моделюючи полігональним, полігони вже видно.

3. NURBS Моделювання. NURBS розшифровується як "Неоднорідний раціональний B-сплайн" і є технологією для створення 3D-об'єктів за допомогою спеціальних кривих, званих B-сплайнами. Деякі експерти виділяють цей тип моделювання, а інші –вважають його підтипомсплайнового моделювання. Принцип моделювання такий: за допомогою B-сплайнів, розташованих вертикально і горизонтально, створюється потрібна форма об'єкта, а потім все це з'єднується полігонами.

Існує два типи цього моделювання:

• Використання P-кривих (Point), форму яких можна змінити за допомогою вершин, розміщених на прямій;

• Використання CV-кривих (Control Vertex), форму яких можна змінювати за допомогою вершин поза лінією.

4. 3D-скульптинг, також відомий як «цифрова скульптура», є імітацією процесу «скульптури» 3D-моделі, тобто деформацією її багатокутної сітки за допомогою спеціальних інструментів - пензлів, щіток. Можна провести аналогію з різьбленням вручну фігур із пластиліну або глини. Тільки в програмах 3D-моделювання замініть пальці інструментом «пензлик», а «пластилін» полігональною сіткою.

5. Промислове моделювання. Системи автоматизованого проектування (САПР) або англійською CAD (Computer-Aided Design) в основному використовуються для створення 3D-моделей в промислових цілях. Вони призначені для точних копій реальних об’єктів. Цей тип моделювання враховує не тільки найдрібніші недоліки, а й матеріальні властивості об’єкта моделі. У зв'язку з цим даний вид моделювання широко використовується в техніці. Особливістю моделювання є створення моделей не з полігонів, а цілісних фігур. Промислове моделювання можна розділити на такі підтипи:

• Параметричний (математична модель створюється шляхом визначення необхідних параметрів);

• Твердий (внутрішня частина моделі не буде порожньою і повинна формуватися зразу на всій мембрані, а не на окремих поверхнях);

• Поверхневий (бажана форма моделі складається з декількох поверхонь, які були попередньо задані у бажаній формі)[2].

Також, потрібно враховувати, що процес створення 3D-графіки є досить складним і тому окремо поділяється на певні етапи. Залежно від того, яка робота виконується і що ми хочемо бачити наприкінці, кількість етапів може змінюватися. Загалом, існує 10 найпоширеніших етапів створення проекту 3D-анімації(Рисунок 2.1).

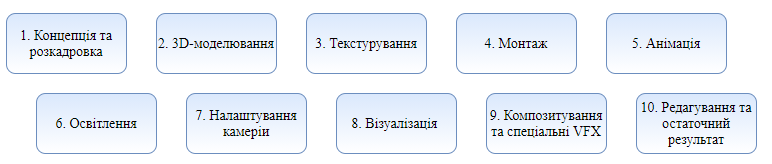


Рисунок 2.1 – Етапи створення 3D-анімації

Розглянемо детально кожний етап створення 3D-анімації:

1. Концепція та розкадровка.

Найпершим кроком у виробництві тривимірного 3D-проекту є концептуалізація ідей та створення розкадрувань, які перетворюють ці ідеї у візуальну форму.

Розкадровка – це послідовність ілюстрацій, що демонструють вашу цифрову історію у двох вимірах. Перший вимір - це час: що відбувається першим, наступним і останнім. Другий – взаємодія: як ваша історія взаємодіє із зображеннями, як візуальні переходи та ефекти допомагають пов’язати зображення? Будь-який елемент може взаємодіяти з будь-яким іншим, і розкадровка – це місце для планування історії.

2. 3D-моделювання.

Після того, як розкадровки закінчені та затверджені, починається завдання створення реквізиту, середовища та персонажів. Власний термін називається "моделювання".

Моделювання – це процес прийняття форми та формування її у завершену тривимірну сітку. Найбільш типовим способом створення тривимірної моделі є взяти простий об’єкт, який називається примітивним, і розширити або «виростити» його у форму, яку можна уточнити та деталізувати. Первісними можуть бути будь-що: від однієї точки (званої вершиною), двовимірної лінії (ребра), кривої (сплайну) до тривимірних об’єктів (граней або багатокутників). Використовуючи специфічні особливості обраного вами програмного забезпечення 3D, кожним із цих примітивів можна маніпулювати, щоб створити об’єкт. При створенні моделі у форматі 3D, як правило, вивчається один із способів створення моделі, який використовується кожен раз, коли необхідно створити нові моделі. Існує три основних методи, якими можна скористатися для створення тривимірної моделі.

3. Текстурування.

Коли створюється 3D-модель, на неї можна накласти 2D-зображення, щоб додати кольори, дизайн та текстури. Це називається картографуванням, і часто від цього походить весь колір моделі. Ці карти можна створювати в таких програмах, як Photoshop, і ілюзії текстур можна наносити на моделі так само просто, як якщо б ви їх намалювали самі; деякі аніматори навіть використовують реальні фотографії текстур, які вони намагаються створити, просто беруть, а потім змінюють, щоб зробити безшовні повторювані візерунки. Саме так створюється більшість текстур волосся.

4. Скелет моделі та покриття шкірою.

Налаштування персонажа на ходьбу та розмову є останнім етапом перед початком процесу анімації персонажів. Ця стадія називається "скелет моделі та покриття шкірою" і є основною системою, яка керує рухами персонажа, щоб оживити його.

Створення скелету (ріггінг) – це процес встановлення керованого скелета для персонажа, призначеного для анімації. Залежно від предмету, кожна установка унікальна, як і відповідний набір елементів управління.

Покриття шкіри (скіннінг) – це процес прикріплення 3D-моделі (шкіри) до сформованого каркаса, щоб 3D-моделлю можна було маніпулювати за допомогою елементів управління скелетоном[3].

5. Анімація.

Анімація – це процес зйомки тривимірного об’єкта та його переміщення. Анімація має кілька різних видів. Існує анімація ключових кадрів, де аніматор маніпулює об’єктами кадром за кадром, подібно до старих мальованих мультфільмів. Інші методи анімації включають розміщення об’єктів на сплайнах та налаштовують їх на шлях кривої, або імпортують дані захоплення руху та застосовують їх до скелетону. Ще один спосіб анімації – це використання вбудованих фізичних механізмів вашого 3D-додатку, наприклад, коли ваша сцена вимагає падіння об’єктів.

6. Освітлення.

Освітлення (у поєднанні з текстурами, кутом нахилу камери тощо) – це те місце, де сцена може потенційно ожити. При неправильному використанні світло може змити сцену, зробити предмети твердими або рівними та зруйнувати всю важку роботу. Але вміло застосоване освітлення може зробити сцену переконливою, або якщо метою є реалізм, створити (у поєднанні з матеріалами та геометрією) сцену, яку практично неможливо відрізнити від реального життя.

У 3D світло насправді не є таким, як у реальному світі. Світло в 3D – це об’єкти, призначені для імітації того, як працює освітлення в реальному житті, але для того, щоб отримати результати, який вам потрібні, вам слід застосувати ряд налаштувань не лише до освітлення, а й до матеріалів.

7. Налаштування камери.

У 3D, на відміну від реального світу, фізичні обмеження не існують. Ви можете створити сцену, в якій камера веде вас у подорож по кровоносних судинах людського тіла, або спостерігати за всім у сцені зі сторони небес, її можна використовувати для створення неможливих перспектив, збільшення та багато іншого. Ось декілька основ, щоб розпочати.

По-перше, розглянемо відмінності між 3D-камерами та реальними камерами. У 3D, на відміну від реального життя, немає необхідності в об'єктиві, елементах фокусування, плівці, діафрагмі тощо. Усі ці функції контролюються за допомогою програмного забезпечення. Єдина схожість – це саме використовування камери. У 3D можна створити одну або кілька камер, розташувати їх точно так, як це потрібно в 3D-просторі, та використовувати налаштування для імітації фокусної відстані, глибини різкості панорамування тощо.

Крім того, програмні камери не мають обмежень щодо розміру та ваги. Можна перемістити камеру в будь-яке місце, навіть всередині найдрібніших об’єктів. Можна анімувати камери таким чином, щоб відбулося кілька операцій одночасно, наприклад масштабування сцени під час зміни глибини різкості. Після створення камери в тривимірному режимі можна вибрати вид і призначити камері вигляд у цьому поданні, що означає, що сцену буде видно з точки зору камери.

8.Візуалізація.

Візуалізація зображення, як правило, є останнім кроком у виробничому конвеєрі 3D (але не останнім кроком у загальному конвеєрі), і є, мабуть, найважливішою частиною. Це крок, який початківці часто пропускають, більше зосереджуючись на створенні моделей та їх анімації. Існує багато аспектів створення хорошого кінцевого рендеру сцени, включаючи увагу до розміщення камери, вибору освітлення, яке може вплинути на настрій і тіні, відбиття та прозорість, а також використання спеціальних ефектів, таких як рідини або гази.

9. Композиція.

Візуалізація додається до композиційних програм для редагування, доопрацювання та додавання спецефектів. Тут кінцеві рендери вносяться до композиційних програм для редагування, доопрацювання та додавання спецефектів.

Композиція включає все, від того, що ви, напевно, зазвичай вважаєте спецефектами (де речі вибухають, випаровуються, перетворюються і т. д.), сюди також входять розширення сцени, до створення навколишнього середовища (від будівель до цілих світів), до заміни синього або зеленого екрану (зйомка перед синім або зеленим екраном, а потім заміна фону на цифрові кадри або кадри, зняті в іншому місці). В основному, мистецтво знімати живі кадри та поєднувати їх із створеними комп’ютером кадрами можна вважати композицією.

10. Редагування та остаточний результат.

На цьому все закінчується. Тут переконуються у синхронізації всіх об’єктів та коректності їх відображення. Після перевірки продукт експортується в один із багатьох форматів і відправляться далі[6].

# Етапи розробки комп’ютерної гри

Кожна гра проходить досить довгий шлях розробки. Досить часто в цьому процесі залучена ціла команда людей, в якій кожен відповідає за конкретний етап роботи. Кількість учасників залежить від обсягів та складності роботи.

Майже всі сучасні ігри проходять однакові етапи розробки, маючи відмінності в залежності від жанру та складності гри (Рисунок 2.2).

Рисунок 2.2–Етапи розробки комп’ютерних ігор

1. Мета (визначення цілей проекту, його жанру та тематики).

На цьому етапі потрібно спланувати гру – чітко обрати жанр і, відштовхуючись від цього, створити її концепт. Жанр визначає основні дії, що виконуватиме гравець, а за тематику середовища відповідає – сеттинг (англ.*Setting*«обстановка»). Тобто характеризувати належність гри до якоїсь сюжетної теми або до конкретного віртуального світу. Найпопулярніші сеттинги: фентезі, постапокаліпсис, наукова фантастика, середньовіччя, друга світова війна, комікси та аніме.

1. Засоби реалізації (вибір матеріалів та інструментів).

Програмний код – це каркас гри, який тримає на собі усі подальші етапи розробки. Тому тут обирається мова програмування та пишеться програмний код, який зможе оперувати 2D або 3D-вимірними об’єктами у просторі та під’єднати зображення та звуки. В наш час існують програмні модулі (ігрові движки), в яких уже реалізовані базові функції, що здатні поєднати графіку, звук, об’єкти та їх переміщення.

1. Ігрова механіка (розробка правил функціонування гри, об’єктів та управління ними; створення штучного інтелекту).

Ігрова механіка являє собою набір правил, за якими буде функціонувати гра. Тут повинні бути прописані відповіді на всі можливі запитання, що виникають у процесі гри. Що трапиться, якщо два об’єкті взаємодіятимуть між собою? Чи може гравець впливати на той чи інший предмет? Кожна дія має своє правило и саме це забезпечую логіку гри. Також у цей етап входить розробка об’єктів та управління ними, призначення клавіш, якими буде керуватися головний герой або об’єкт та дій, що відбуватимуться після натискання певних кнопок. Також потрібно враховувати і фізичні властивості об’єктів і надати їм характеристики відповідні до фізичних законів реального світу. В залежності від жанру гри є необхідність розробити штучний інтелект, що відповідатиме за поведінку другорядних персонажів, наприклад, ворогів або союзників.

1. Рівні (компонування створених об’єктів на локаціях).

Формування середовищ, на яких розташовуються моделі та застосовуються створені раніше правила.

1. Графіка (створення 2D або 3D моделей, анімації, фонів та спецефектів; оформлення ігрового меню та екрану).

Спочатку розроблюються образи героїв, ворогів, ігрових предметів та задніх фонів. На основі цих образів створюються моделі та анімації до них, малюються фони. Додаються спецефекти, що за принципом роботи схожі на анімацію, але замість переміщення об’єктів в них використовується переміщення частинок та світлофільтрів.

1. Сюжет (складання історії в грі).

Як правило, рідко коли сучасна гра не побудована навколо якогось сценарію. Особливо сильна роль сценаріїв в рольових, пригодницьких іграх і в іграх змішаних жанрів

1. Звук (створення звукових ефектів та музики).

Створення звукового супроводу є не менш важливим у розробці ігор. Фонова мелодія задає основний настрій гри, а звукові ефекти додають процесу реальності.

1. Тестування.

На цьому етапі відбувається повна збірка проекту та перевіряється коректність відображення компонентів гри.

1. Випуск.

Останнім етапом є випуск гри та подальша її підтримка[5].

Перед початком роботи також потрібно визначити, в якому саме жанрі буде створена гра. Спектр існуючих жанрів досить великий, тому для початку можна обрати напрямок, а далі у процесі розробки чітко визначити, що саме ми хочемо бачити наприкінці.

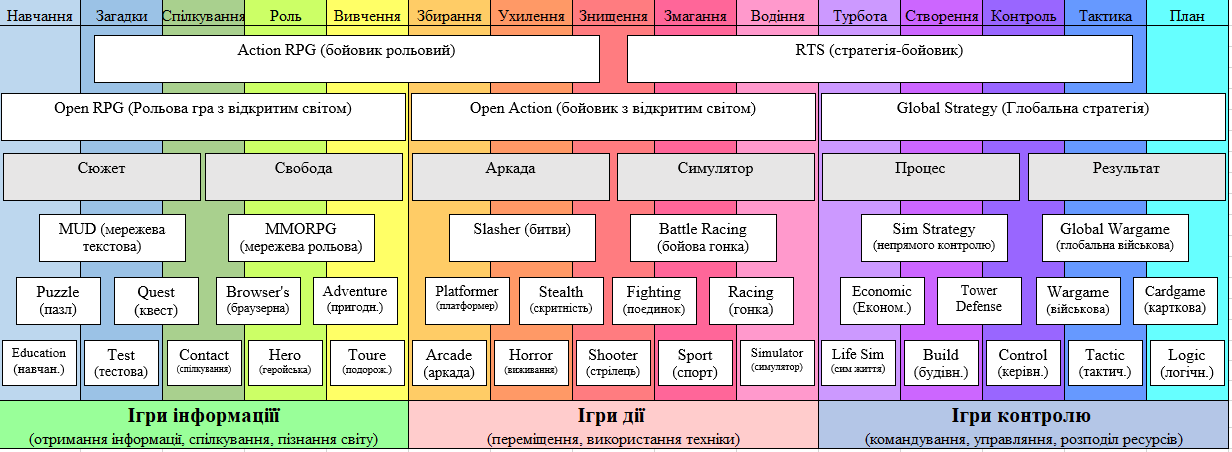


Рисунок 2.3 – Жанри комп’ютерних ігор

На рис. 2.3 зображений повний перелік існуючих жанрів в залежності від кількості елементів, що вони вміщають. Існують ігри, що складаються з одного, двох, трьох, п’яти елементів, а також гібридні ігри.

За основний критерій розподілу жанрів обираємо дії, які найчастіше виконуються в іграх даного жанру та більше нічого[7].

Ігри між собою діляться на три основних жанри:

* Ігри дії;
* Ігри інформації (рольові);
* Ігри контролю (стратегії).

Головним в іграх дії є саме переміщення тіла, яким керує гравець. При чому не важливо, яким саме тілом, – це може модель людини, тварини, техніки тощо. Такі ігри відомі своєю динамікою та допомагають гравцям розвивати свою швидкість реакції. Найкращим прикладом будуть ігри жанрі бойовик. Вони займають найбільший сегмент світового ринку ігор і є найпопулярнішими. Стандартний сценарій такої гри: управління персонажем, що може володіти різними спеціальними прийомами, для знищення ворогів. Через свою популярність, такі ігри є найвидовишнішіми і саме тут використовуються найновіші прийоми спецефектів та технологій. Такі ігри завжди використовують 3D-графіку, тому є дуже гарними, але й ш видко застарівають через велику конкуренцію.

Ігри інформації відповідають за передачу інформації гравцеві у будь-яких її проявах. Сюди входить: пізнання світу через оточуюче середовище, засвоєння навчального матеріалу або отримання навичок через спілкування. Діапазон варіацій таких ігор досить великий, але золотою серединною вважаються рольові ігри. Тобто це ігри в яких можливо вживатися в роль героя и проживати його життя. Такі ігри цінуються за атмосферу, сюжет та деталізований ігровий світ.

Ігри, спрямовані на розвиток навичок управління та адміністрування ресурсами, називаються іграми контролю. Суть гри – планування та управління подіями, ресурсами для отримання переваг у майбутньому. Сюди відносяться всі ігри типу стратегій, економічних, логічних та тактичних жанрів. Часто вважається, що деякі жанри ігор інформації та контролю схожі між собою, але потрібно пам’ятати про суттєву відмінність: ігри інформації потрібно вирішувати покладаючись на отриманні у грі знання, на відміну від ігор контролю. Тут потрібно прийти до результату покладаючись лише на вміння детально розплановувати кожен свій подальший крок.

Всі ігри всередині кожної з груп дуже схожі між собою, але мають певні відмінності. Існує 15 основних елементів, із яких складається будь-яка гра, а саме: навчання, загадки, спілкування, роль, вивчення, збирання, ухилення, знищення, змагання, водіння, турбота, створення, контроль, тактика, план.

Створена мною гра відноситься до жанру ігор інформації і несе в собі певну загадку, яку гравець повинен розгадати, тобто є елементарним жанром загадки – тестом. Вона не несе освітньої цілі, а слугує лише для перевірки раніше отриманих знань. І учасник має свободу дій, якої не має в іграх жанру навчання, оскільки вони слугують тільки для передачі інформації.

Це свого роду вікторина, що за принципом схожа на відому гру «Хто хоче стати міліонером?», але має абсолютно різний інтерфейс. Задається питання і гравець має обрати один єдиний правильний варіант відповіді. В залежності від зроблених виборів, результат наприкінці буде різний.

# РОЗДІЛ 3

**ВИБІР РІШЕНЬ ТА ЇХ ПРАКТИЧНА РЕАЛІЗАЦІЯ**

# Створення ігроладу

Найпершим етапом у складані ігрового процесу є визначення жанру гри. Оскільки жанр створеної гри – тест, то вона повинна відповідати його критеріям та обов’язково містити елемент вікторини з питаннями та можливістю надання відповідей на них. У грі повинні використовуються одиночні та множинні варіанти відповідей.

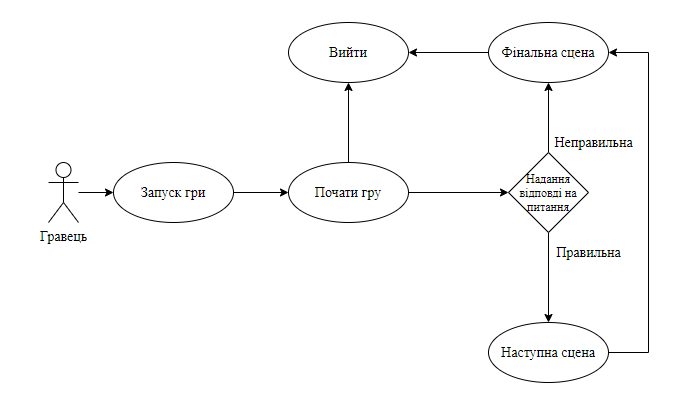


Рисунок 3.1 – Взаємодії гравця під час гри

Далі складаємо сценарії взаємодії користувача з грою, яка складена наступним чином:

1. Користувач заходить у гру. На екрані відображається вікно запуску гри, де гравець має можливість почати гру або вийти з неї.
2. Після початку гри з’являється перша сцена з відправною точкою, головного ігрового об’єкту, потяга.
3. Далі перед користувачем з’являється вікно з питанням та варіантом відповідей. Кожне питання обмежене у часі і гравець може слідкувати за ним за допомогою таймера, що розташований у верхньому правому куті.
4. В залежності від наданої відповіді перед гравцем з’являється екран, що сповіщає про правильність наданої відповіді. Також від цього залежить, по якому маршруту буде далі рухатись потяг і яке саме анімація буде програватись.
5. У випадку надання неправильної відповіді, наступною перед гравцем з’являється фінальна сцена з вікном, дозволяє користувачу почати гру спочатку або вийти з неї.

Для оформлення зовнішнього вигляду використовуються 3D-моделі і для збереження плавності гри був обраний мінімалістичний низькополігональний стиль моделювання. Для початку, потрібно визначити які саме моделі необхідні та розподілити їх за категоріями. Знадобляться моделі:

* ландшафту (terrains);
* природного оточення (nature);
* будівлі (buildings);
* залізничного потяга та колій (train, railways);
* персонажів (people).

У підсумку, ігролад налаштований на створення середовища для оцінки знань користувача за допомогою тестів, з використанням обмеження у часі та доданням графічної складової для покращення візуальної частини гри.

# Вибір програмного забезпечення для створення 3D-моделей

Не менш важливим у створені 3D-графіки є вибір необхідного програмного забезпечення. Сучасний ринок таких програм має широкий вибір, враховуючи різні аспекти, кожен знайде програму до вподоби.

1. Autodesk Maya

Ціна ліцензії: 245 доларів на місяць, безкоштовна версія на 3 роки для студентів.

При опитуванні професійних спеціалістів з 3D-моделей у галузі, яку програму вони використовують найбільше, найпоширенішою відповіддю є AutodeskMaya. Більшість провідних анімаційних студій використовують його (включно з Pixar), частково завдяки масивному набору потужних інструментів, пропонованих у пакеті. За останні кілька років програма ввела декілька особливо нових функції, що відразу потрапили в центр уваги, наприклад, по-справжньому жива візуалізація.

Але є і недоліки: Mayaнедешева і вимагає ґрунтовного оволодіння нею для створення серйозних проєктів.

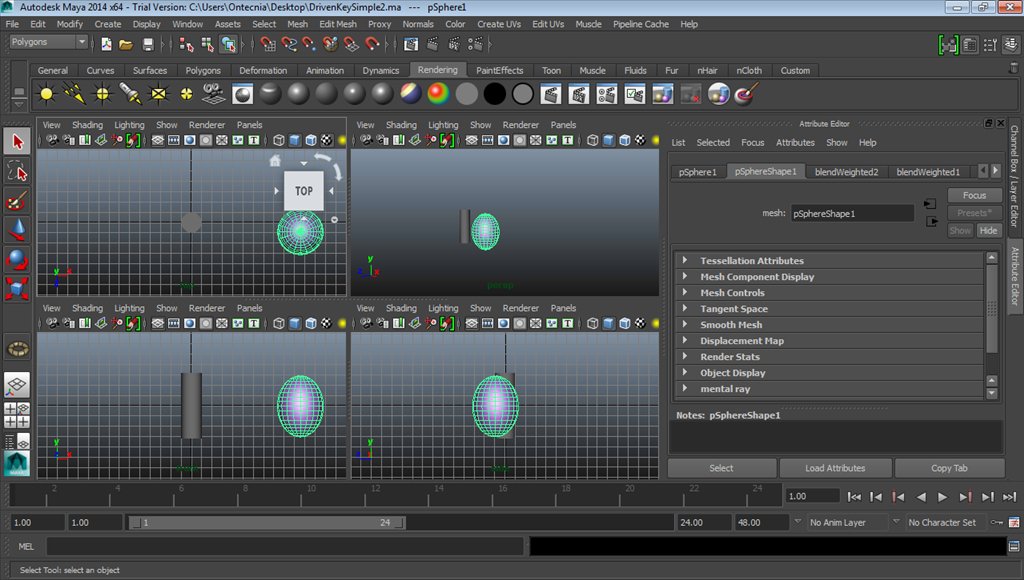


Рисунок 3.2 – Інтерфейс програми AutodeskMaya

2. Houdini

Ціна ліцензії : 1995, 4495 або 499 доларів на рік.

Houdini – ще один стандартний у галузі інструмент, який є одним із найкращих програмних пакетів 3D-моделювання. Він використовує іншу методологію, ніж AutodeskMaya, використовуючи процедурний стиль виробництва на основі вузлів, який надає художникам величезний контроль. Саме в Houdini збираються сцени фільмів або відео, а все тому, що програма має потужні можливості для створення спецефектів. В цій програмі також існує велика кількість різних прийомів, необхідних для створення професійних моделей. На освоєння програми та розкриття всього її потенціалу потрібно буде витратити певний час, тому ви не зможете створювати шедеври з першого використання.

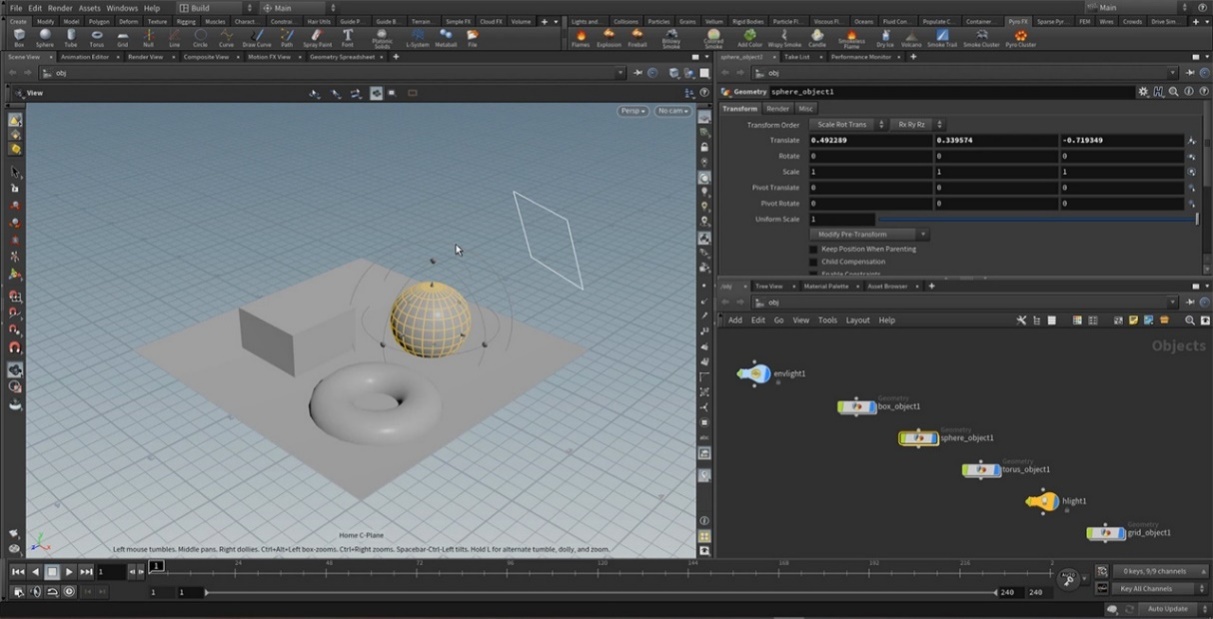


Рисунок 3.3 – Інтерфейс програми Houdini

3. Cinema 4D

Ціна ліцензії: 480-2850 доларів на рік.

Cinema 4D від Maxon легко потрапляє в число найкращих програм для 3D-моделювання. Спочатку був розроблений для створення моушн-дизайну (графіки руху), але зараз цей потужний інструмент складає конкуренцію вищезазначеним програмам та наділений усім необхідним функціоналом. Ви можете зробити моделі неймовірної якості та анімацію до них, але найсильніша перевага над конкурентами - це набагато простіший у використанні інтерфейс. Саме завдяки легкому та інтуїтивно зрозумілому інтерфейсу Cinema 4D являється однією з найкращих програм для новачків.

Головний мінус – вартість, але розробники програми також пропонують безкоштовну пробну версію, яка триває трохи довше місяця, щоб допомогти прийняти рішення, та пропонують безкоштовні студентські ліцензії.

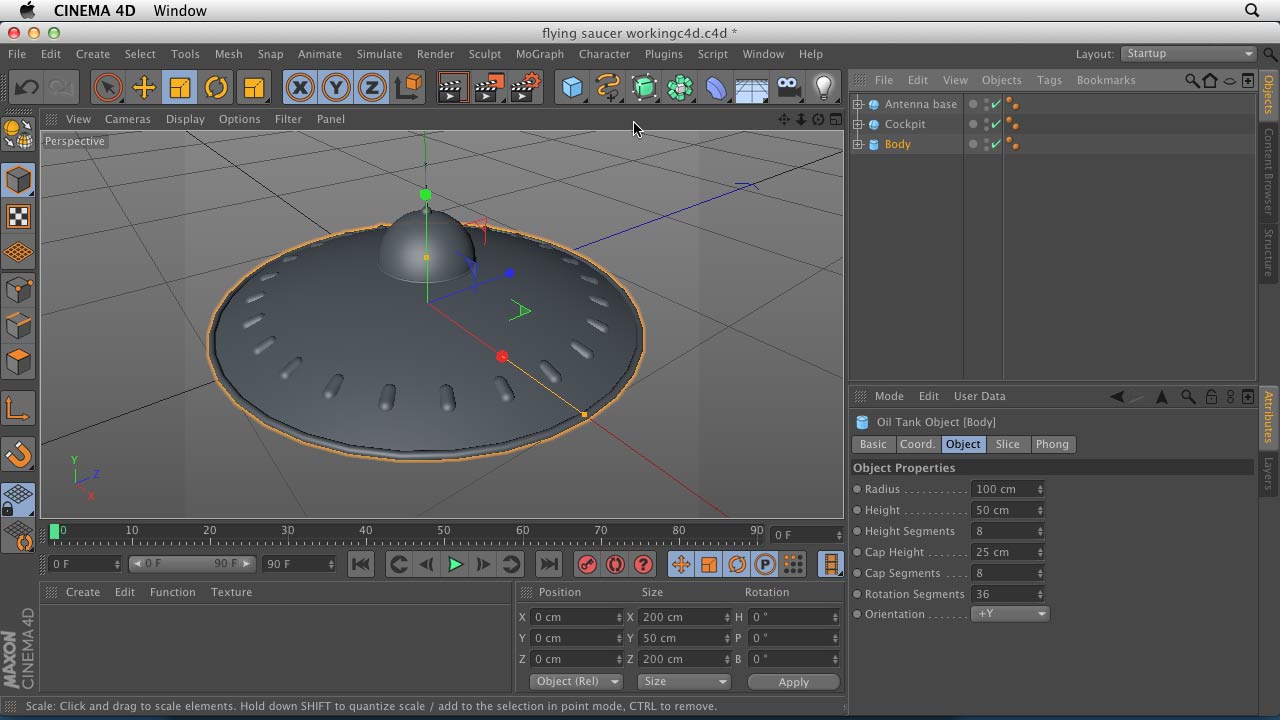


Рисунок 3.4 – Інтерфейс програми Cinema 4D

4. Autodesk 3Ds Маx

Ціна підписки: 216 доларів на місяць або 1740 доларів на рік.

3Ds Max існує вже досить давно і є однією із найстаріших та найпопулярніших програм для моделювання. Завдяки цьому він має найбільшу кількість відео уроків, курсів, а також додаткових розширень. Варто згадати, що дане програмне забезпечення має гігантську доступну бібліотеку, що забезпечує доступ до незліченних функцій, які можуть зробити процес моделювання більш цікавим.

Він має більш високу ціну, але пропонує безкоштовні студентські ліцензії та пробну версію, яка дозволяє отримати доступ до всіх функцій, які додаток може запропонувати протягом 30 днів.

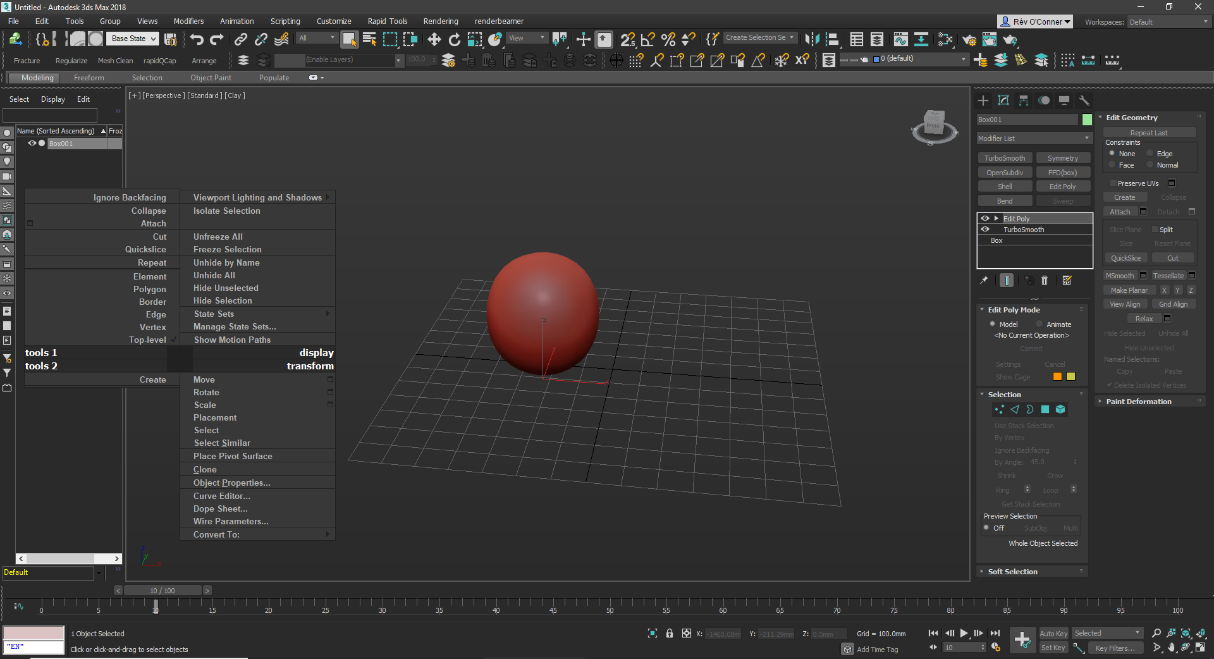


Рисунок 3.5 – Інтерфейс програми Autodesk 3Ds Маx

5. ZBrush

Ціна ліцензії: 895 доларів.

Коли справа доходить до ліплення дивовижних істот, людей та місць, немає конкурентів, які б приблизно були схожі на ZBrush.У цій програмі можливо створювати об’єкти за допомогою віртуальної глини. Це основний продукт візуальних ефектів для фільмів та телебачення, і існує лише декілька програм 3D-моделювання, які можуть порівнятися з ним в галузі дизайну відеоігор . ZBrush кращий варіант для створення вінілових об’єктів та іграшок, що друкуються на 3D-принтерах.

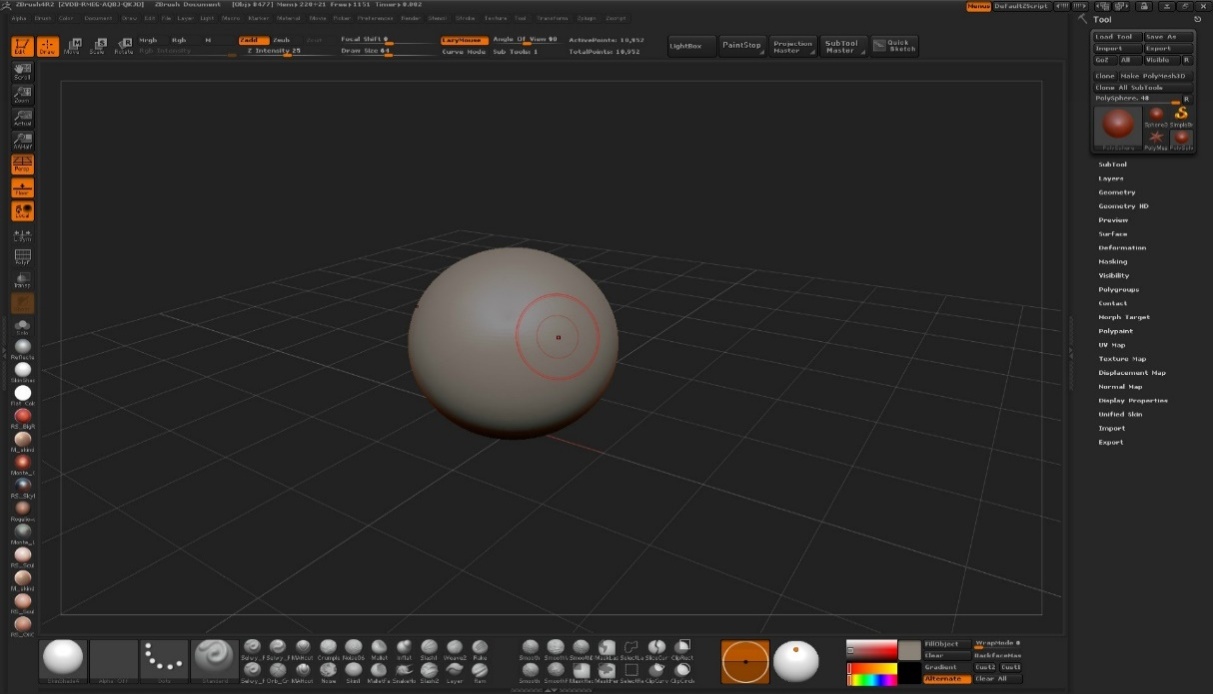


Рисунок 3.6 – Інтерфейс програми ZBrush

6.Blender

Blender – найкращий вибір серед безкоштовних програм для 3D-моделювання. Працює на всіх основних операційних системах, і надає доступ до усіх необхідних інструментів, таких як: текстура, ліплення, такелаж, ультрафіолетове випромінювання та анімація. Завдяки надійному двигуну візуалізації, він ані трохи не здає позиції у порівнянні з дорогими програмами.

Як додатковий бонус, програма має відкритий код. Отже, програма постійно вдосконалюється, а створені доповнення для нових функціональних можливостей не тільки загальні, але й завжди безкоштовні [10].

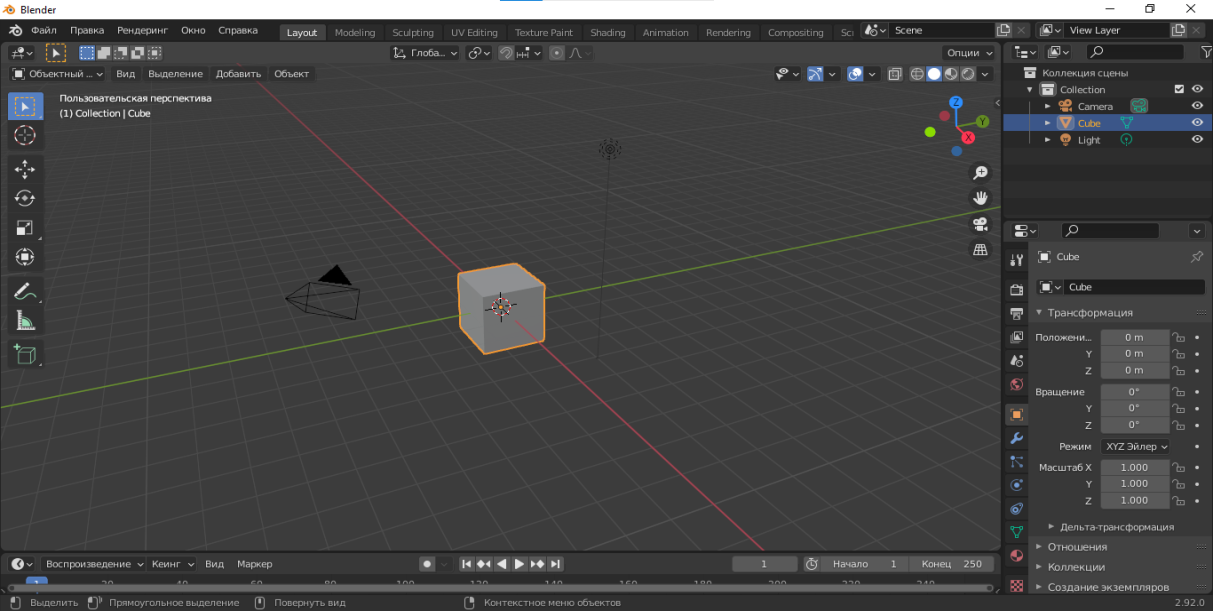


Рисунок 3.7 – Інтерфейс програми Blender

Провівши оцінку представлених програм, мною була обрана – Cinema 4D, оскільки вони є простою у використанні та має безкоштовну ліцензовану підписку для студентів.

# Вибір двигуну для розробки гри

Потрібно обрати інструментарій, що відповідатиме за забезпечення відображення графічних об’єктів на екрані та дозволить користувачеві взаємодіяти із створеним середовищем. Отже, необхідно обрати ігровий двигун для реалізації поставлених завдань.

Ігровий двигун –це ядро гри, тобто базове програмне забезпечення, на базі якого будуються всі інші складові гри. Існує два види використання ігрових двигунів при розробці гри:

1. використання готового програмного забезпечення;
2. власна реалізація.

Використання першого передбачає значне скорочення часу для розробки, оскільки здійснення більшості завдань низького рівня складності вже прописані авторами. Це дозволяє значно спростити процес і надає можливість залучитися до розробки людям з нижчою кваліфікацією у програмуванні.

При виборі ігрового двигуна потрібно враховувати мову програмуванню, що його підтримує та жанр, у якому планується створюватися гра. Оскільки ігрові двигуни призначені для створення рольових ігор не матимуть необхідного забезпечення для реалізації, наприклад, інді-ігор.Отже, потрібне ПЗ, що буде підтримувати використання 3D-графіки та одну з таких мов програмування:

1. С++;
2. Java;
3. C# та .Net.

Після огляду представлених ринкових рішень, я обрала Unity 3Dна базі C# з ряду певних причин:

1. Мова програмування C# була в пріоритеті при виборі ігрового двигуна, оскільки знання з неї значно вищі в порівнянні з іншими мовами. Також він досить простий у використанні та зручний у розробці.
2. Безкоштовна ліцензія. Двигун Unity 3D надає користувачам безкоштовну ліцензію на розробку та випуск ігор у випадку, якщо їх річний дохід не перевищує 100 тисяч доларів США та на розробку гри було зібрано менше ніж 100 тисяч доларів США. Інші програми мають більш жорсткі умови ліцензування.
3. Наявність великої бібліотеки та магазину готових компонентів Unity Asset Store. Дає можливість використовувати у своїх проектах різні готові компоненти – аудіо-файли, елементи інтерфейсу, програмний код, графіку та інше.

Графіка, освітлення, фізика, інтерфейс – основні системи у структурі Unity 3D. Ці компоненти цілковито відрізняються один від одного за принципом роботи та кінцевим результатом, але всі вони мають однаковий принцип роботи.

Головним компонентом Unity 3D– є ігровий об’єкт (англ. – Gameobject). Цей об'єкт є абстракцією високого рівня, в якій є функції, які виконуються регулярно, в певному порядку і безперервно. Безперервність програмної логіки характеризується тим, що вона лежить в основі створення ігор. Цикл повного виконання призначених функцій ігрових об'єктів називається фреймом або кадром. Циклічність стосується будь-якої функції движка: для розрахунку просторового положення гравця, ефектів, що його оточують, графіки і так далі. Звичайно, швидкість обчислень надзвичайно важлива, тому що візуально результатом є плавність зображення, яке з'являється перед користувачем, як швидко та м’яко реагує на дії гравця. Отже, вимога для будь-якої гри – забезпечити високу частоту кадрів. Існує стандарт в 60 кадрів на секунду, але можна використовувати і більше, проте людське око фізично не може відстежувати частоту кадрів більшу за 60 кадрів. Більшу частоту використовують для того, щоб внутрішній стан гри оновлювався швидше і передавав останні зміни як найшвидше, що особливо важливо в іграх з серверною підтримкою.

Ігровий об’єкт є центральним елементом, оскільки він є контейнером для всієї іншої ігрової логіки, задаючи їй фізичний стан в ігровому просторі і взаємодіючі з системами кодових викликів. Отже, він є фундаментом в усіх іграх створених на Unity 3D. За реалізацію функціоналу та логіки відповідають скрипти – модулі, що задаються за допомогою програмного коду.

# Розробка ігрового оточення

Було прийнято рішення, що гра матиме 6 різних сцен, між якими буде пересуватися потяг. Кожна з цих сцен була створена з дотриманням конкретної тематики та має різний набір моделей. Основним ігровим об’єктом є потяг та залізничні колії, по яких він пересувається. На рис. 3.8 наведена модель потягу.

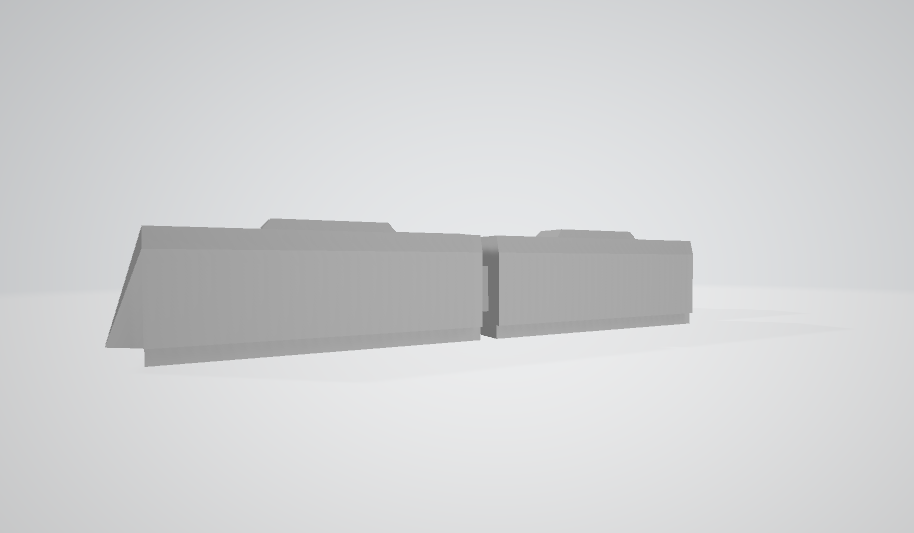


Рисунок 3.8 – Модель потяга без застосування текстур

# Моделі ландшафту. Моделі ландшафту створюються у вигляді квадратів розміром 30 на 30.Кожен створений відповідно до необхідного вигляду ландшафту. Так, були створені моделі схилів, рівнин, долин та міжгір’я. На рис. 3.9 представлена одна із таких моделей.

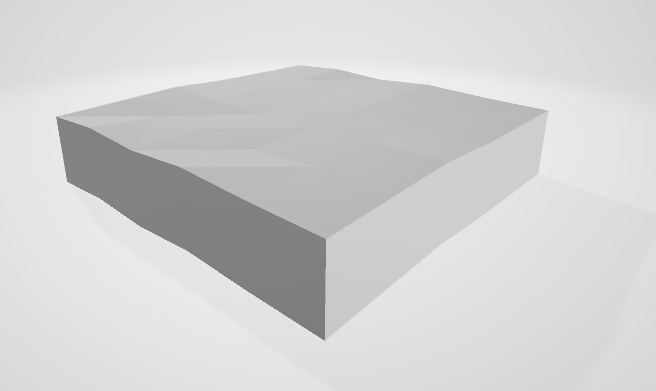


Рисунок 3.9 – Модель рівнини без застосування текстур

# Моделі початкової сцени. Перша сцена, яку бачить гравець, виконана у тематиці міста. Тут використані моделі інфраструктури, жителів та різні варіації моделей природи, для надання гри більшої реалістичності. На рис. 3.10 – 3.15представлені моделі, що використовуються у першій сцені.



Рисунок 3.10– Модель будинку без застосування текстур

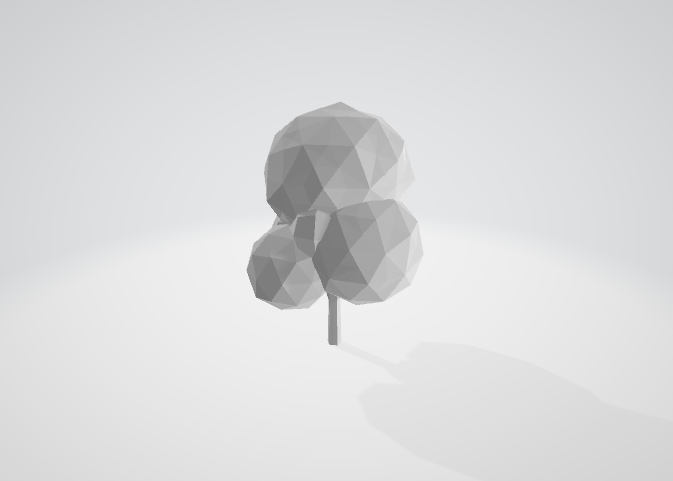


Рисунок 3.11 – Модель паркового дерева без використання текстур

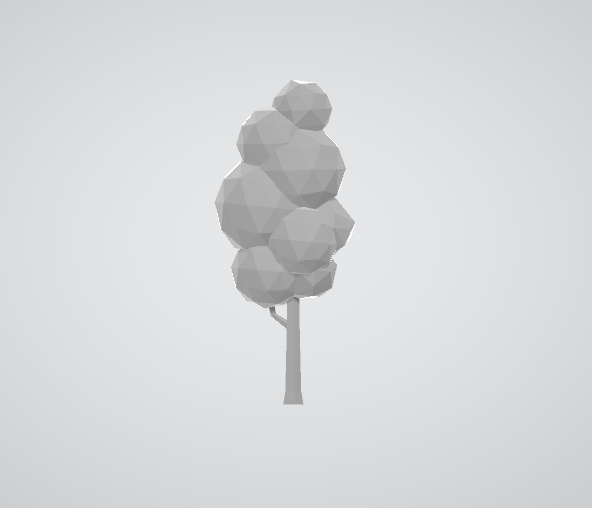


Рисунок 3.12 – Модель дерева без використання текстур

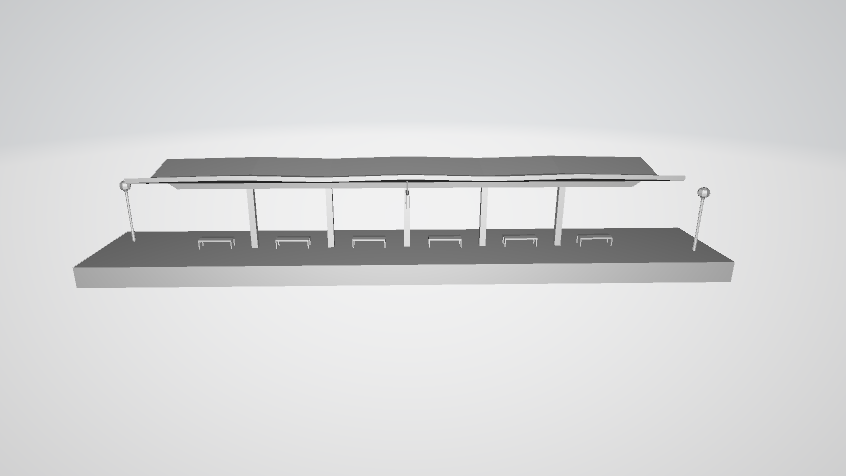


Рисунок 3.13 – Модель станції без використання текстур

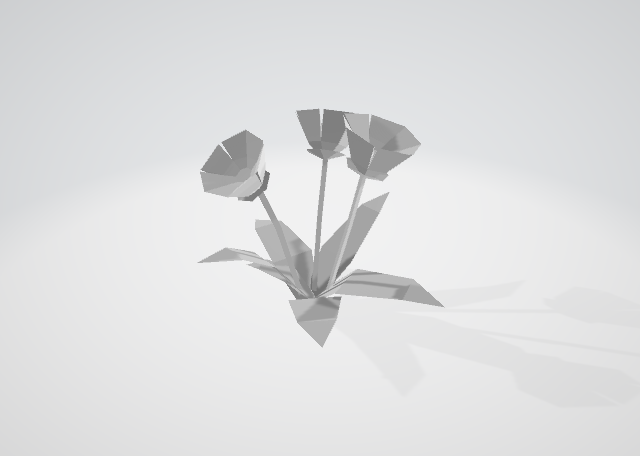


Рисунок 3.14 – Модель квітів без використання текстур



Рисунок 3.15 – Модель місцевого жителя без використання текстур

Далі всі моделі розміщуються на сцені, до них використовуються різні види текстур та матеріалів. Остаточний вигляд сцени представлений на рис. 3.16.

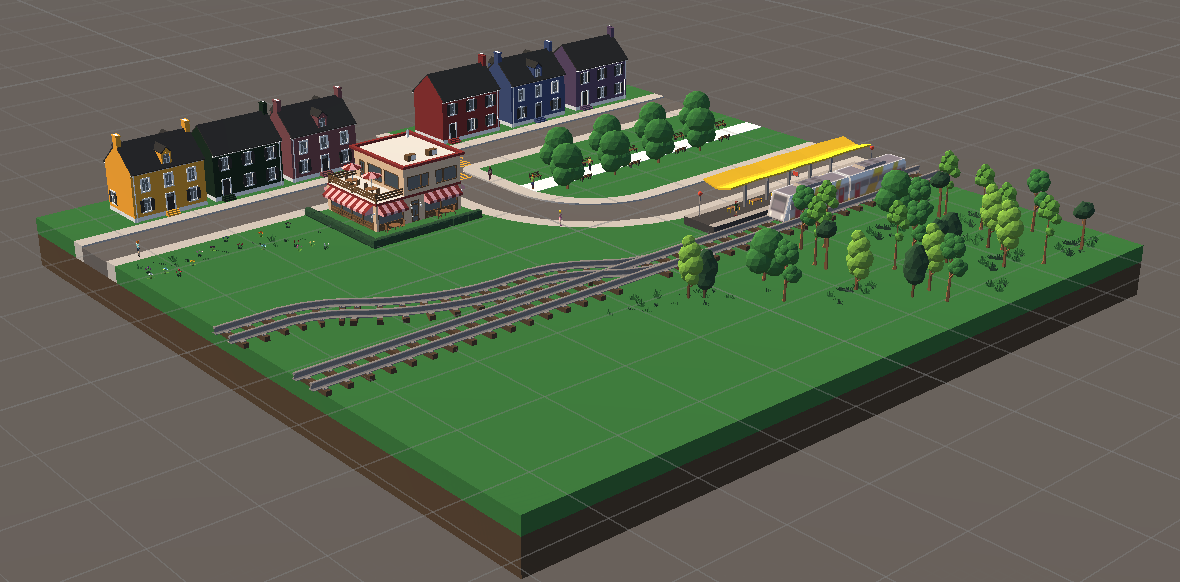


Рисунок 3.16 – Фінальний вигляд сцени 1

# Моделі другої сцени.Друга сцена виконана у вигляді гірського пейзажу з озерами та місцями для кемпінгу. Тут використані моделі гір, озер, хвойних дерев та відповідної атрибутики для відпочинку в лісі. На рис. 3.17 – 3.23 представлені моделі, що були застосовані у другій сцені.

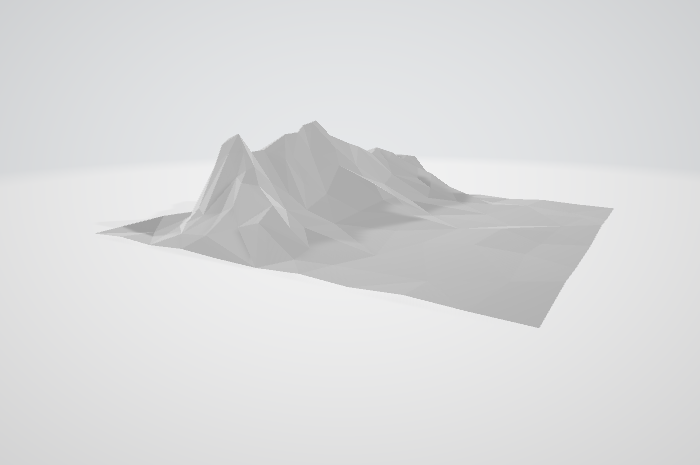


Рисунок 3.17 – Модель гір без використання текстур

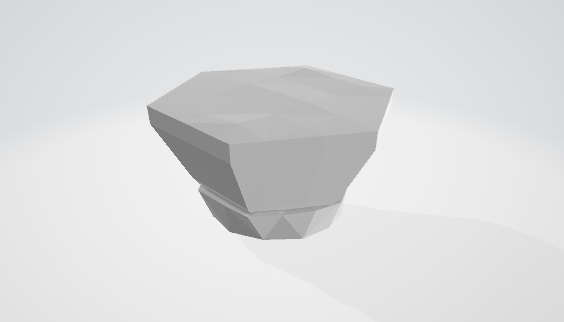


Рисунок 3.18 – Модель озера без застосування текстур



Рисунок 3.19 – Модель ялини без застосування текстур

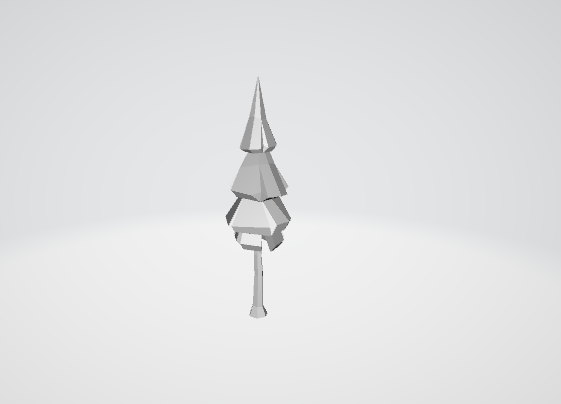


Рисунок 3.20 – Модель сосни без застосування текстур



Рисунок 3.21 – Модель палатки без застосування текстур

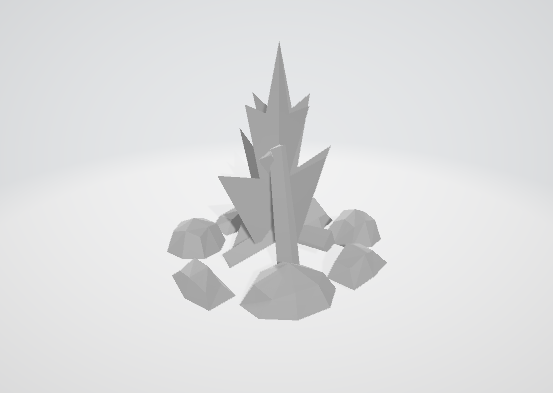


Рисунок 3.22 – Модель багаття без застосування текстур

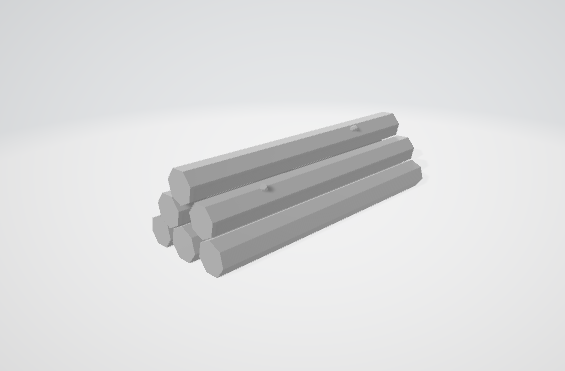


Рисунок 3.23 – Модель стосу дров без застосування текстур

Використовуючи необхідні матеріали створено кінцевий варіант сцени, що зображений на рис. 3.24.

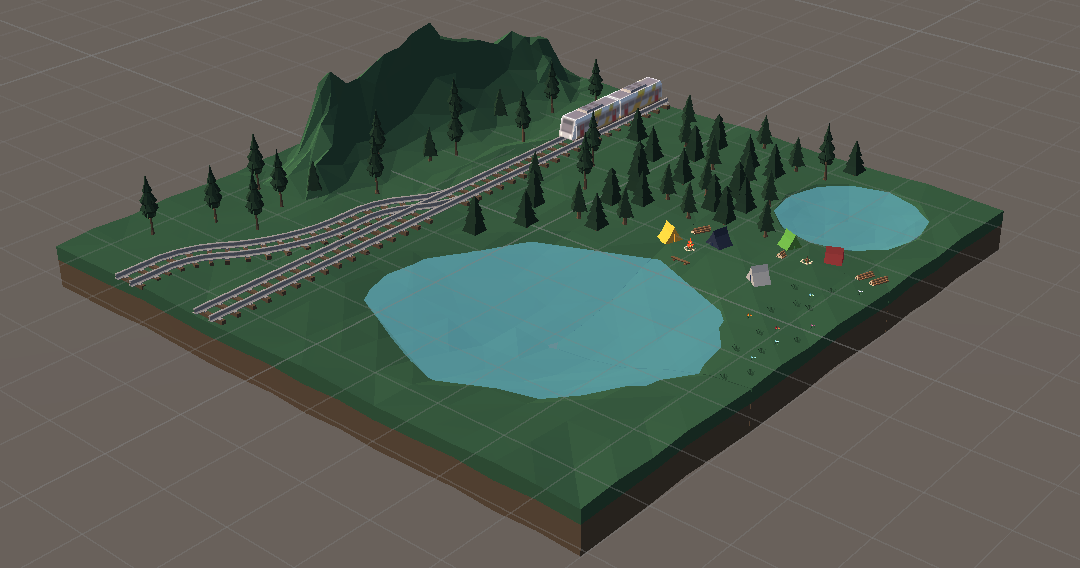


Рисунок 3.24 – Кінцевий вигляд сцени 2

# Моделі третьої сцени.Дана сцена виконана с дотриманням тематики холодних земель. Використовуються моделі засніжених дерев, льодовиків та різної зимової атрибутики, що представлені на рис. 3.25 – 3.28.

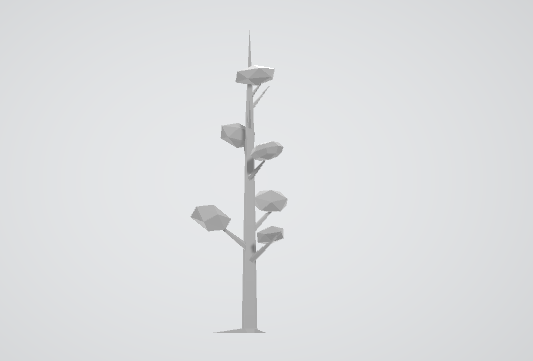


Рисунок 3.25 – Модель засніженого дерева без застосування текстур

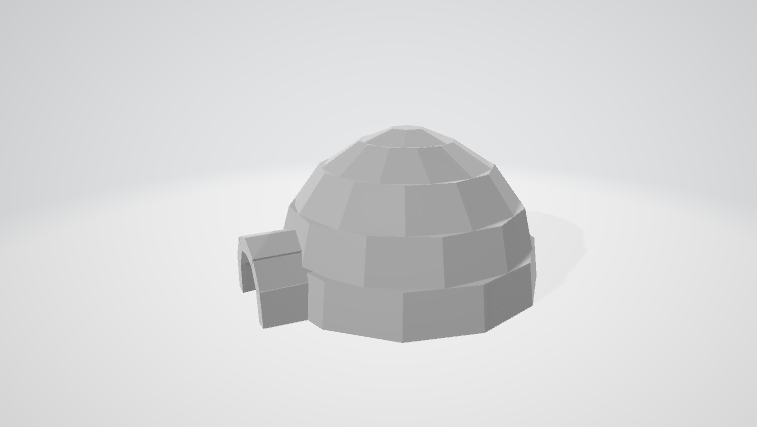


Рисунок 3.26 – Модель іглу без застосування текстур

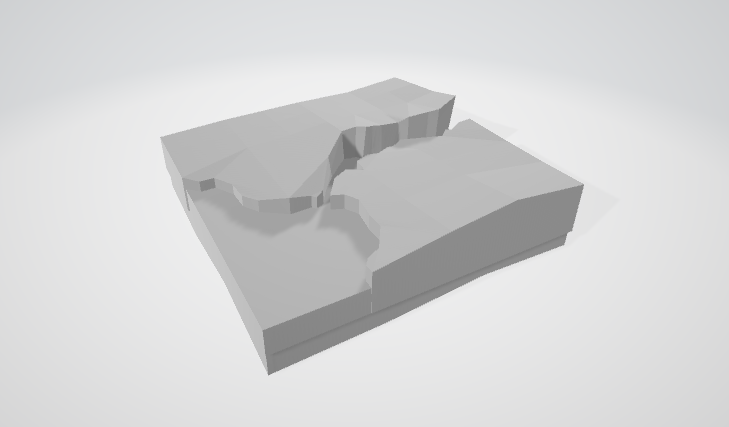


Рисунок 3.27 – Модель льодовикових розколів без застосування текстур



Рисунок 3.28 – Модель сніговика без застосування текстур

Після розміщення об’єктів сцена отримала остаточний вигляд, що зображений на рис. 3.29.

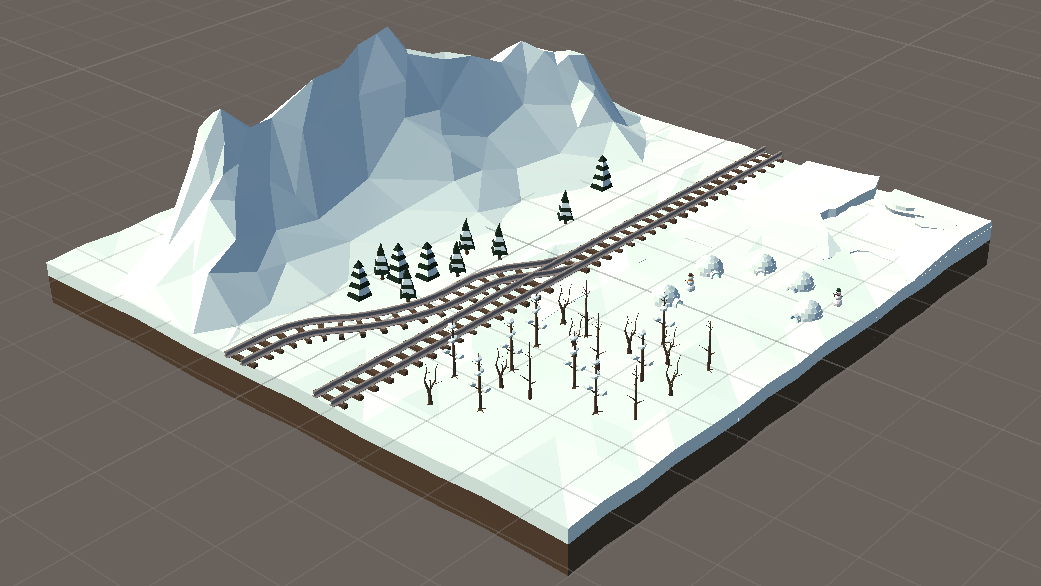


Рисунок 3.29 – Кінцевий вигляд сцени 3

# Моделі четвертої сцени.Тематика цієї сцени – пустеля. Тут використовуються пейзажі у стилі вестерну з горами характерного вигляду, кактусами та розтрісканою від засухи землею. Остаточний вигляд сцени зображений на рис. 3.30.

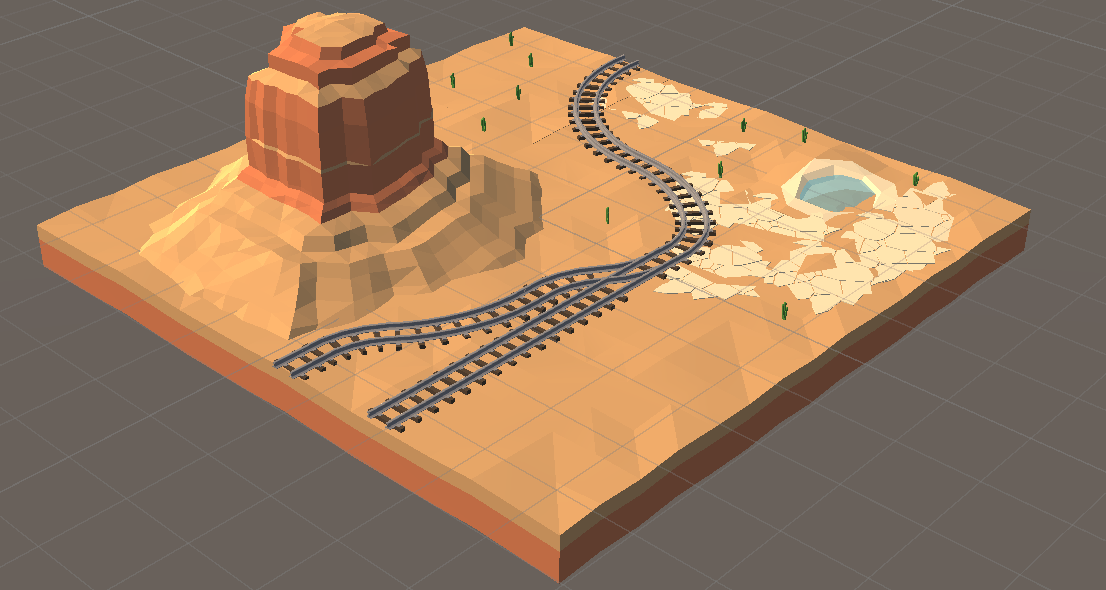


Рисунок 3.30 – Кінцевий вигляд сцени 4

Моделі, що були використані при створені сцени, зображені на рис. 3.31 – 3.33.

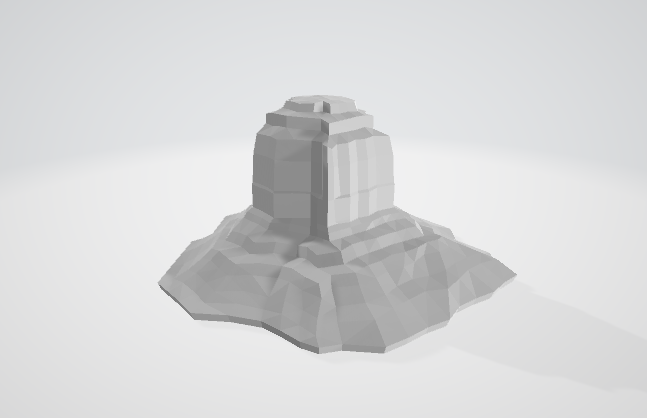


Рисунок 3.31– Модель гори без застосування текстур

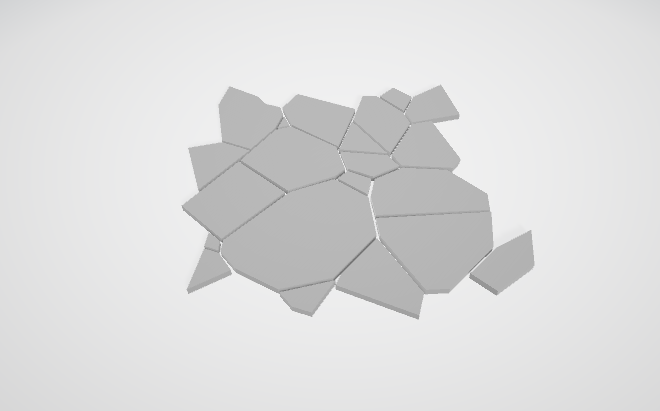


Рисунок 3.32 – Модель розтрісканої землі без застосування текстур

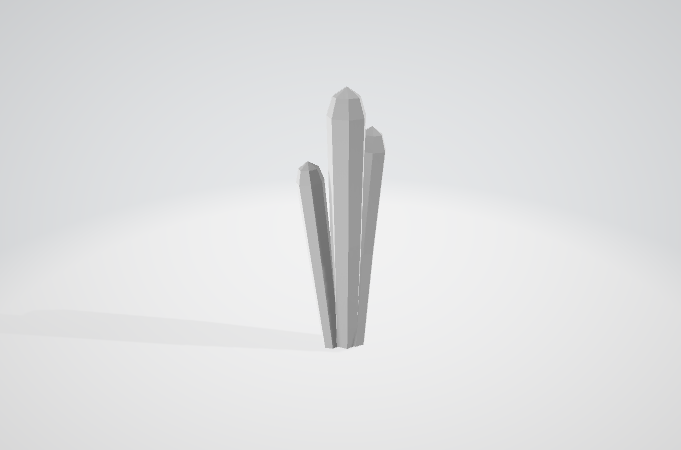


Рисунок 3.33 – Модель кактусу без застосування текстур

# Моделі фінальної сцени при правильних відповідях.Це фінальна сцена гри при всіх правильних відповідях. Вона виконана у вигляді морського острівного пейзажу з використанням пальмових дерев, тропічних споруд та іншої морської атрибутики. Моделі, що були використані у даній сцені, зображені на рис. 3.34 – 3.37.



Рисунок 3.34 – Модель бунгало без застосування текстур



Рисунок 3.35 – Модель пальми без застосування текстур

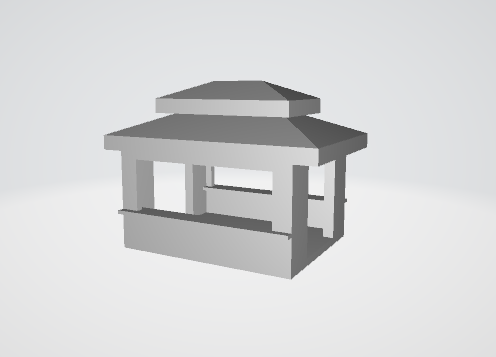


Рисунок 3.36 – Модель бару без застосування текстур

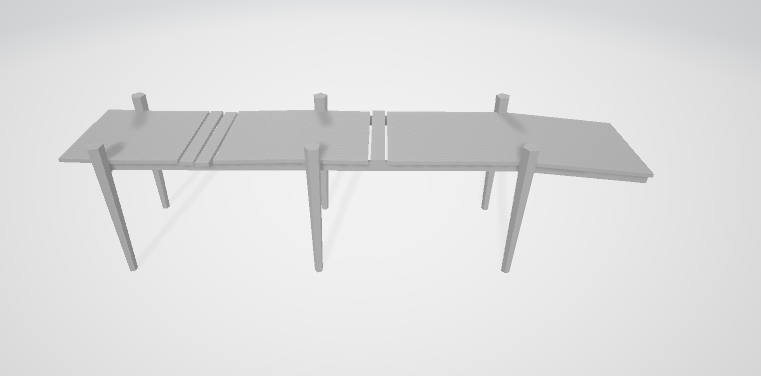


Рисунок 3.37 – Модель дерев’яного пірсу без застосування текстур

В даній сцені основним елементом є море, для якого була створена спеціальна текстура з певним рівнем прозорості для надання реалістичності поверхні води. До інших елементів також були застосовані матеріали та текстури. Остаточний вигляд сцени наведений на рис. 3.38.

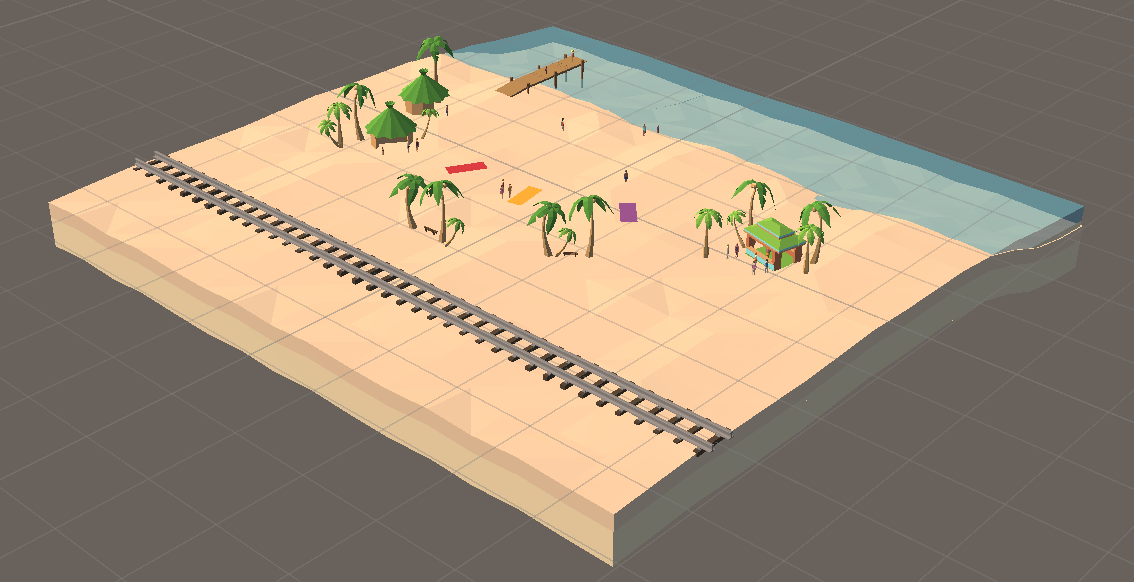


Рисунок 3.38 – Кінцевий вигляд фінальної сцени при правильних відповідях

# Моделі фінальної сцени при неправильних відповідях.Тематика останньої сцени при неправильних відповідях – болото. Тут використовуються темні кольори переважно коричневих відтінків. Використовуються моделі сухих дерев, трави, змій та інші. Використані моделі представлені на рис. 3.39 –3.41.



Рисунок 3.39 – Модель дерева без застосування текстур

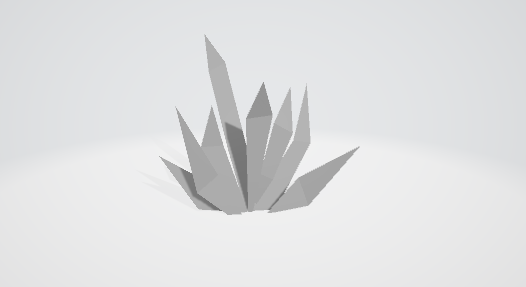


Рисунок 3.40 – Модель трави без застосування текстур

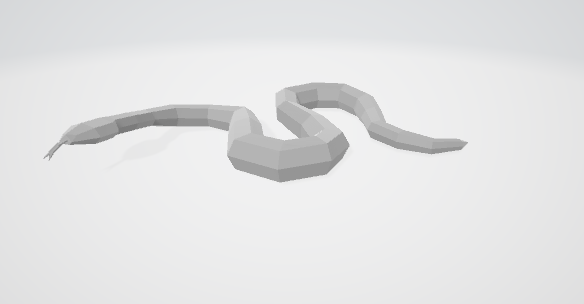


Рисунок 3.41 – Модель змії без застосування текстур

Після використання різних матеріалів та текстур сцена набула остаточного вигляду, що представлений на рис. 3.42.

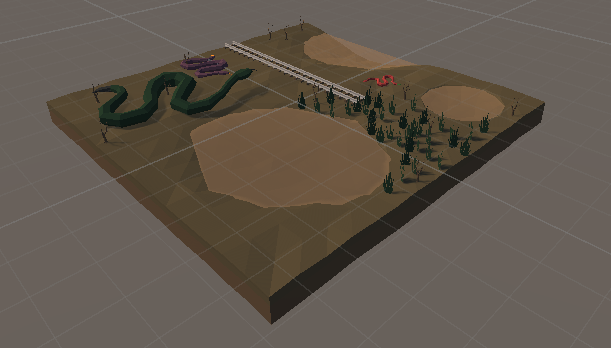


Рисунок 3.42 – Кінцевий вигляд фінальної сцени при неправильних відповідях

# Розробка механіки та інтерфейсу

Наступним етапом є створення інтерфейсу гри. Основними елементами інтерфейсу слугують різні кнопки, а також вікно для відображення вікторини. Потрібно враховувати, що від зручності використання функціоналу також залежить рівень якості продукту, тому розробці функціоналу придається багато уваги. Для реалізації функціоналу та механіки гри необхідно прописати скрипти на мові програмування C#. Кожна кнопка відповідає певній функції, що прописані у скрипті. Загалом у грі використано 4 види кнопок: початок гри, пауза, запуск гри заново та вихід.

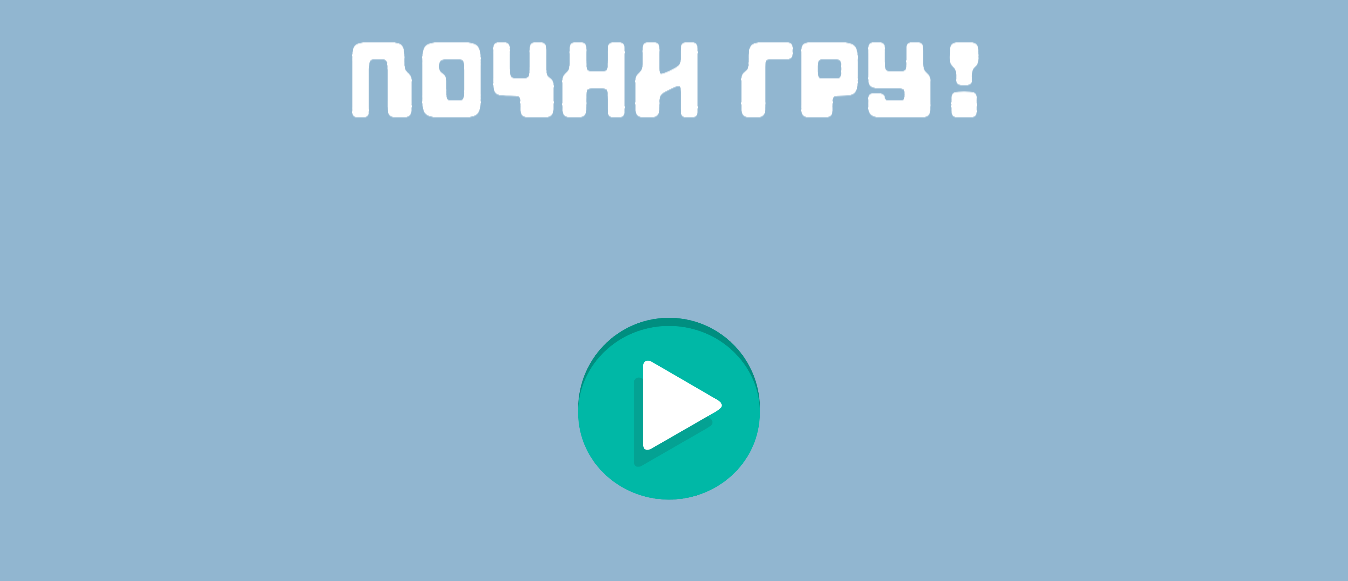


Рисунок 3.43 – Початковий екран гри

Основним етапом створення гри є реалізація вікторини. Спочатку створився інтерфейс вікна, далі за допомогою скриптів біла прописана можливість створення питань та відповідей до них.

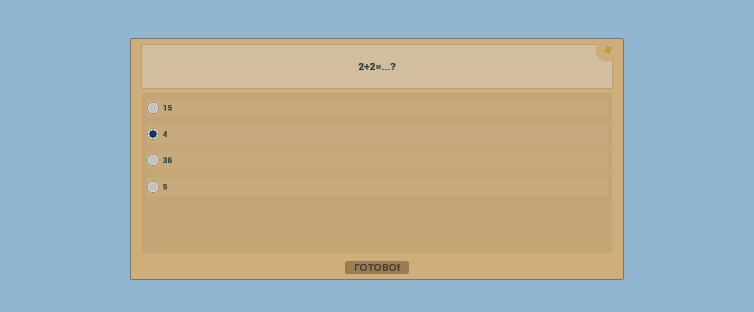


Рисунок 3.44 – Вікно вікторини

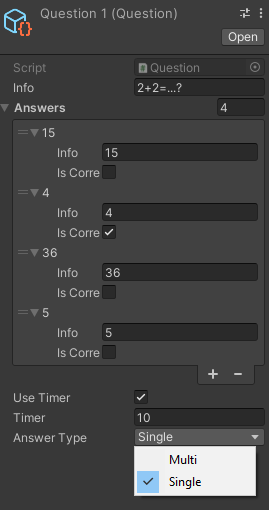


Рисунок 3.45 – Створення питання

Питання має два варіанти відповідей: одиночні та множині. Також можливо використовувати таймер для кожного питання окремо та визначити кількість відведеного часу, що вимірюється в секундах. Якщо гравець не встигає відповісти на питання до закінчення часу, то питання автоматично зараховується як невірне.

# РОЗДІЛ 4

**ТЕСТУВАННЯ**

Тестування є одним із головних компонентів у створені будь-якого проекту. Саме тестування визначає рівень якості продукту і для більшої його ефективності рекомендується проводити його за принципом ітерації, тобто на кожному етапі розробки. Такий підхід значно полегшить роботу під час проекту та зменшить час на перевірку перед випуском продукту. Також можливо проводити перевірку на коректність роботи певних функцій програми за допомогою тестових випадків.

Тестовий випадок (англ. «testcase») – це послідовність кроків, певних умов і параметрів, потрібних для перевірки здійснення функції, що тестується, або її сегменту. Тестові випадки можуть поєднуватися у групи, якщо тестується одна й та сама функція, але очікується різний результат. Такі об’єднання називаються тестовими наборами. В даному випадку тестується набір «Надання відповіді на питання» за такими сценаріями:

* Надання правильної відповіді на питання;
* Надання неправильної відповіді на питання.

Таблиця 4.1–Надання правильної відповіді на питання

|  |  |
| --- | --- |
| Тема: Надання правильної відповіді на питання | |
| Опис: | Надання правильної відповіді на питання |
| Передумова: | Відкрите вікно вікторини |
| Кроки: | Очікуваний результат: |
| 1.Обрати правильні відповіді на питання. | Змінюється прапорець, що ідентифікує обрану відповідь. |
| 2.Натиснути кнопку «Готово». | З’являється вікно, що повідомляє про правильність результату. |

Продовження таблиці 4.1

|  |  |
| --- | --- |
| Тема: Надання неправильної відповіді на питання | |
| Опис: | Надання неправильної відповіді на питання |
| Передумова: | Відкрите вікно вікторини |
| Кроки: | Очікуваний результат: |
| 1.Обрати неправильні відповіді на питання. | Змінюється прапорець, що ідентифікує обрану відповідь. |
| 2.Натиснути кнопку «Готово». | З’являється вікно, що повідомляє про неправильність результату. |

Також проводиться ігрове тестування, що відповідає за імітацію дій користувача та виконує перевірку дійсного результату з очікуваним. У результаті різних видів тестування було виявлено, що програма повністю відповідає очікуваному результату і є працездатною.

# ВИСНОВКИ

У кваліфікаційній роботі розроблено систему контролю знань для учнів навчальних закладів з використанням 4Dграфіки. Для розробки графіки використано програмне забезпечення Cinema 4D, що є одним з найпростіших у освоєнні серед аналогів та має великий вибір необхідних функціональних можливостей, а також безкоштовну ліцензію. У якості ігрового двигуну використовувався Unity 3D – комплексна система програмного забезпечення, що має повний набір функцій для створення інтерфейсу та ігрової механіки.

Розроблена система контролю знань є грою в жанрі тест та забезпечує такі можливості:

* можливість додавати питання;
* підтримку одиничних та множинних відповідей, можливість застосування таймеру;
* різні сценарії при правильній та неправильній відповіді;
* реалізацію ландшафтів (сеттинг) в різних стилях: місто, пустеля, арктичний пейзаж, пляж, гірський пейзаж, болото.

Розроблений програмний продукт є безкоштовним та кросплатформеним з можливістю встановлення на персональних комп’ютерах з операційними системами Windows, Linuxта Mac.

Система призначена для контролю знань для учнів середніх навчальних закладів.

Результати виконання кваліфікаційної роботи бакалавра були апробовані на 73-й науково-практичній конференції Національного університету «Полтавська політехніка імені Юрія Кондратюка».

Можливі перспективи розвитку програмного продукту: збільшення кількості сцен, додавання додаткових типів питань, класифікація питань за тематикою.

# СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Типи 3Dмоделювання [Електронний ресурс]: [Веб-сайт]. – Електронні дані. – Режим доступу: https://foyr.com/learn/types-of-3d-modeling/.
2. Види 3Dмоделювання [Електронний ресурс]: [Веб-сайт]. – Електронні дані. – Режим доступу:https://3dd-modeli.com/uroki-videokursi/3d-grafika/6175-vidy-3d-modelirovaniya.html.
3. Особливості 3Dмоделювання [Електронний ресурс]: [Веб-сайт]. – Електронні дані. – Режим доступу:https://www.bibliofond.ru/view.aspx?id=896988
4. Основные этапы разработки компьютерной обучающей системы и их особенности[Електронний ресурс]: [Веб-сайт]. – Електронні дані. – Режим доступу: https://novainfo.ru/article/12824.
5. Этапы создания компьютерной игры[Електронний ресурс]: [Веб-сайт]. – Електронні дані. – Режим доступу:https://gamesisart.ru/game\_dev-create.html#Game\_Create.
6. TheProcessof 3Danimation[Електронний ресурс]: [Веб-сайт]. – Електронні дані. – Режим доступу:https://www.media-freaks.com/the-process-of-3d-animation/
7. Жанры компьютерных игор[Електронний ресурс]: [Веб-сайт]. – Електронні дані. – Режим доступу:https://gamesisart.ru/TableJanr.html
8. Как объектно ориентированная среда для разработки компьютерных игр[Електронний ресурс]: [Веб-сайт]. – Електронні дані. – Режим доступу:https://cyberleninka.ru/article/n/alice-kak-obektno-orientirovannaya-sreda-dlya-razrabotki-kompyuternyh-igr/viewer.
9. Haitan O.M., Alyoshin S.P., Kolesnyk D.D. Information technologies of virtual and augmented reality in education / O.M. Haitan, S.P. Alyoshin, D.D. Kolesnyk // Тези 73-ї наукової конференції професорів, викладачів, наукових працівників, аспірантів та студентів Національного університету «Полтавська політехніка імені Юрія Кондратюка». Том 2. (Полтава, 21 квітня – 13 травня2021р.) – Полтава: Національний університет імені Юрія Кондратюка,2021. – С. 449 - 450.
10. Top 3Dmodelingsoftware[Електронний ресурс]: [Веб-сайт]. – Електронні дані. – Режим доступу:https://www.format.com/magazine-resources/design/3d-modeling-software
11. J.D.Raskin. Constructivism in psychology: personal construct psychology, radical constructivism, and social constructionism // Studies in meaning: exploring constructivist psychology. – New York: Pace University Press, 2002. – 125 с.
12. Prensky. M. Digital Game-Based Learning. McGraw-Hill – New York, 2011 – 23 с.
13. Использование игровых приемов при формировании элементарных математических представлений у детей старшего дошкольного возраста[Електронний ресурс]: [Веб-сайт]. – Електронні дані. – Режим доступу: https://nsportal.ru/detskii-sad/vospitatelnaya-rabota/2020/05/18/ispolzovanie-igrovyh-priemov-pri-formirovanii
14. Основні форми контролю знань студентів[Електронний ресурс]: [Веб-сайт]. – Електронні дані. – Режим доступу:https://ru.osvita.ua/vnz-reports/pedagog/14679/.
15. The 4th Dimension [Електронний ресурс]: [Веб-сайт]. – Електронні дані. – Режим доступу: https://ysjournal.com/the-4th-dimension/.

# ДОДАТОК А

**ОСНОВНІ ЧАСТИНИ ПРОГРАМНОГО КОДУ**

1. GameManager.cs

using System.Collections;

using System.Collections.Generic;

using UnityEngine;

using System.Linq;

using TMPro;

publicclassGameManager : MonoBehaviour

{

private Question[] \_questions = null;

public Question[] Questions { get { return \_questions; } }

[SerializeField] GameEvents events = null;

[SerializeField] Animator timerAnimator = null;

[SerializeField] TextMeshProUGUI timerText = null;

[SerializeField] Color timerHalfWayOutColor = Color.blue;

[SerializeField] Color timerAlmostOutColor = Color.red;

private Color timerDefaultColor = Color.black;

private List<AnswersData> PickedAnswers = new List<AnswersData>();

private List<int> FinishedQuestions = new List<int>();

privateint currentQuestion = 0;

private AnimationController animationController;

privateint timerStateParaHash = 0;

private IEnumerator IE\_WaitTillNextRound = null;

private IEnumerator IE\_StartTimer;

privatebool IsFinished

{

get

{

return (FinishedQuestions.Count < Questions.Length) ? false : true;

}

}

void OnEnable()

{

events.UpdateQuestionAnswer += UpdateAnswers;

}

void OnDisable()

{

events.UpdateQuestionAnswer -= UpdateAnswers;

}

void Start()

{

timerDefaultColor = timerText.color;

animationController = GetComponent<AnimationController>();

LoadQuestion();

timerStateParaHash = Animator.StringToHash("TimerState");

var seed = UnityEngine.Random.Range(int.MinValue, int.MaxValue);

UnityEngine.Random.InitState(seed);

Display();

}

publicvoid UpdateAnswers(AnswersData newAnswer)

{

if (Questions[currentQuestion].GetAnswerType == Question.AnswerType.Single)

{

foreach (var answer in PickedAnswers)

{

if (answer != newAnswer)

{

answer.Reset();

}

PickedAnswers.Clear();

PickedAnswers.Add(newAnswer);

}

}

else

{

bool alreadyPicked = PickedAnswers.Exists(x => x== newAnswer);

if (alreadyPicked)

{

PickedAnswers.Remove(newAnswer);

}

else

{

PickedAnswers.Add(newAnswer);

}

}

}

publicvoid EraseAnswers()

{

PickedAnswers = new List<AnswersData>();

}

void Display ()

{

EraseAnswers();

var question = GetRandomQuestion();

if (events.UpdateQuestionUI != null)

{

events.UpdateQuestionUI(question);

} else { Debug.LogWarning("Ups! Something went wrong!"); }

if (question.UseTimer)

{

UpdateTimer(question.UseTimer);

}

}

publicvoid Accept()

{

UpdateTimer(false);

bool isCorrect = CheckAnswers();

FinishedQuestions.Add(currentQuestion);

var type = (IsFinished) ? UIManager.ResolutionScreenType.Finish : (isCorrect) ? UIManager.ResolutionScreenType.Correct : UIManager.ResolutionScreenType.Incorrect;

if (events.DisplayResolutionScreen != null)

{

events.DisplayResolutionScreen(type);

}

if (type != UIManager.ResolutionScreenType.Finish)

{

if (IE\_WaitTillNextRound != null)

{

StopCoroutine(IE\_WaitTillNextRound);

}

IE\_WaitTillNextRound = WaitTillNextRound();

StartCoroutine(IE\_WaitTillNextRound);

}

}

IEnumerator WaitTillNextRound()

{

yieldreturnnew WaitForSeconds(GameUtility.ResolutionDelayTime);

Display();

}

void UpdateTimer (bool state)

{

switch (state)

{

casetrue:

IE\_StartTimer = StartTimer();

StartCoroutine(IE\_StartTimer);

timerAnimator.SetInteger(timerStateParaHash, 2);

break;

casefalse:

if (IE\_StartTimer != null)

{

StopCoroutine(IE\_StartTimer);

}

timerAnimator.SetInteger(timerStateParaHash, 1);

break;

}

}

IEnumerator StartTimer()

{

var totalTime = Questions[currentQuestion].Timer;

var timeLeft = totalTime;

timerText.color = timerDefaultColor;

while (timeLeft > 0)

{

timeLeft--;

if (timeLeft < totalTime / 2 && timeLeft > totalTime / 4)

{

timerText.color = timerHalfWayOutColor;

}

if (timeLeft < totalTime / 4)

{

timerText.color = timerAlmostOutColor;

}

timerText.text = timeLeft.ToString();

yieldreturnnew WaitForSeconds(1.0f);

}

Accept();

}

Question GetRandomQuestion()

{

var randomIndex = GetRandonQuestionIndex();

currentQuestion = randomIndex;

return Questions[currentQuestion];

}

int GetRandonQuestionIndex()

{

var random = 0;

if (FinishedQuestions.Count<Questions.Length)

{

do

{

random = UnityEngine.Random.Range(0, Questions.Length);

} while (FinishedQuestions.Contains(random) || random == currentQuestion);

}

return random;

}

bool CheckAnswers()

{

if (!CompareAnswers())

{

returnfalse;

}

returntrue;

}

bool CompareAnswers()

{

if (PickedAnswers.Count > 0)

{

List<int> c = Questions[currentQuestion].GetCorrectAnswers();

List<int> p = PickedAnswers.Select(x => x.AnswerIndex).ToList();

var f = c.Except(p).ToList();

var s = p.Except(c).ToList();

return !f.Any() && !s.Any();

}

returnfalse;

}

void LoadQuestion()

{

Object[] objs = Resources.LoadAll("Questions", typeof(Question));

\_questions = new Question[objs.Length];

for (int i = 0; i < objs.Length; i++)

{

\_questions[i] = (Question)objs[i];

}

}

publicvoid RestartGame()

{

}

publicvoid QuitGame()

{

Application.Quit();

}

}

1. Question.cs

using System.Collections;

using System.Collections.Generic;

using UnityEngine;

[System.Serializable]

publicstructAnswer

{

[SerializeField] privatestring \_info;

publicstring info { get { return \_info; } }

[SerializeField] privatebool \_isCorrect;

publicbool IsCorrect { get { return \_isCorrect; } }

}

[CreateAssetMenu(fileName = "Question", menuName = "Quiz/new Question")]

publicclassQuestion : ScriptableObject {

publicenumAnswerType { Multi, Single }

[SerializeField] privatestring \_info = string.Empty;

publicstring info { get { return \_info; } }

[SerializeField] Answer[] \_answers = null;

public Answer[] Answers { get { return \_answers; } }

[SerializeField] privatebool \_useTimer = false;

publicbool UseTimer { get { return \_useTimer; } }

[SerializeField] privateint \_timer = 0;

publicint Timer { get { return \_timer; } }

[SerializeField] private AnswerType \_answerType = AnswerType.Multi;

public AnswerType GetAnswerType { get { return \_answerType; } }

public List<int> GetCorrectAnswers ()

{

List<int> CorrectAnswers=new List<int>();

for (int i=0; i < Answers.Length; i++)

{

if (Answers[i].IsCorrect)

{

CorrectAnswers.Add(i);

}

}

return CorrectAnswers;

}

}

1. AnswersData.cs

using UnityEngine;

using UnityEngine.UI;

using TMPro;

publicclassAnswersData : MonoBehaviour

{

[Header("UI Elements")]

[SerializeField] TextMeshProUGUI infoTextObject;

[SerializeField] Image toggle;

[Header("Textures")]

[SerializeField] Sprite uncheckedToggle;

[SerializeField] Sprite checkedToggle;

[Header("References")]

[SerializeField] GameEvents events;

private RectTransform \_rect;

public RectTransform Rect

{

get

{

if (\_rect == null)

{

\_rect = GetComponent<RectTransform>() ?? gameObject.AddComponent<RectTransform>();

}

return \_rect;

}

}

privateint \_answerIndex = -1;

publicint AnswerIndex { get { return \_answerIndex; } }

privatebool Checked = false;

publicvoid UpdateData(string info, int index)

{

infoTextObject.text = info;

\_answerIndex = index;

}

publicvoid Reset()

{

Checked = false;

UpdateUI();

}

publicvoid SwitchState()

{

Checked = !Checked;

UpdateUI();

if (events.UpdateQuestionAnswer != null)

{

events.UpdateQuestionAnswer(this);

}

}

void UpdateUI()

{

toggle.sprite = (Checked) ? checkedToggle : uncheckedToggle;

}

}

1. UIManager.cs

using System.Collections;

using System.Collections.Generic;

using UnityEngine;

using UnityEngine.UI;

using System;

using TMPro;

[Serializable()]

publicstructUIManagerParameters

{

[Header("Answers Options")]

[SerializeField] float margins;

publicfloat Margins { get { return margins; } }

[Header("Resolution Screen Options")]

[SerializeField] Color correctBGColor;

public Color CorrectBGColor { get { return correctBGColor; } }

[SerializeField] Color incorrectBGColor;

public Color IncorrectBGColor { get { return incorrectBGColor; } }

[SerializeField] Color finalBGColor;

public Color FinalBGColor { get { return finalBGColor; } }

}

[Serializable()]

publicstructUIElements

{

[SerializeField] RectTransform answersContentArea;

public RectTransform AnswersContantArea { get { return answersContentArea; } }

[SerializeField] TextMeshProUGUI questionIhfoTeatObject;

public TextMeshProUGUI QuestionInfoTeatObject { get { return questionIhfoTeatObject; } }

[Space]

[SerializeField] Animator resolutionScreenAnimator;

public Animator ResolutionScreenAnimator { get { return resolutionScreenAnimator; } }

[SerializeField] Image resolutionBG;

public Image ResolutionBG { get { return resolutionBG; } }

[SerializeField] TextMeshProUGUI resolutionStateInfoText;

public TextMeshProUGUI ResolutionStateInfoText { get { return resolutionStateInfoText; } }

[Space]

[SerializeField] CanvasGroup mainCanvasGroup;

public CanvasGroup MainCanvasGroup { get { return mainCanvasGroup; } }

[SerializeField] RectTransform finishUIElements;

public RectTransform FinishUIElements { get { return finishUIElements; } }

}

publicclassUIManager : MonoBehaviour

{

publicenumResolutionScreenType { Correct, Incorrect, Finish}

[Header("References")]

[SerializeField] GameEvents events;

[Header("UI Elements (Prefabs)")]

[SerializeField] AnswersData answerPrefab;

[SerializeField] UIElements uIElements;

[Space]

[SerializeField] UIManagerParameters parameters;

List<AnswersData> currentAnswers = new List<AnswersData>();

privateint resStateParaHash = 0;

private IEnumerator IE\_DisplayTimeResolution;

void OnEnable()

{

events.UpdateQuestionUI += UpdateQuestionUI;

events.DisplayResolutionScreen += DisplayResolution;

}

void OnDisable()

{

events.UpdateQuestionUI -= UpdateQuestionUI;

events.DisplayResolutionScreen -= DisplayResolution;

}

void Start()

{

resStateParaHash = Animator.StringToHash("ScreenState");

}

void UpdateQuestionUI (Question question)

{

uIElements.QuestionInfoTeatObject.text = question.info;

CreateAnswers(question);

}

void DisplayResolution(ResolutionScreenType type)

{

UpdateResUI(type);

uIElements.ResolutionScreenAnimator.SetInteger(resStateParaHash, 2);

uIElements.MainCanvasGroup.blocksRaycasts = false;

if (type != ResolutionScreenType.Finish)

{

if (IE\_DisplayTimeResolution != null)

{

StopCoroutine(IE\_DisplayTimeResolution);

}

IE\_DisplayTimeResolution = DisplayTimeResolution();

StartCoroutine(IE\_DisplayTimeResolution);

}

}

IEnumerator DisplayTimeResolution()

{

yieldreturnnew WaitForSeconds(GameUtility.ResolutionDelayTime);

uIElements.ResolutionScreenAnimator.SetInteger(resStateParaHash, 1);

uIElements.MainCanvasGroup.blocksRaycasts = true;

}

void UpdateResUI(ResolutionScreenType type)

{

switch (type)

{

case ResolutionScreenType.Correct:

uIElements.ResolutionBG.color = parameters.CorrectBGColor;

uIElements.ResolutionStateInfoText.text = "BIPHO!";

break;

case ResolutionScreenType.Incorrect:

uIElements.ResolutionBG.color = parameters.IncorrectBGColor;

uIElements.ResolutionStateInfoText.text = "HEBIPHO!";

break;

case ResolutionScreenType.Finish:

uIElements.ResolutionBG.color = parameters.FinalBGColor;

uIElements.ResolutionStateInfoText.text = "KIHEЦЬ!";

uIElements.FinishUIElements.gameObject.SetActive(true);

break;

}

}

void CreateAnswers(Question question)

{

EraseAnswers();

float offset = 0 - parameters.Margins;

for (int i=0; i<question.Answers.Length; i++)

{

AnswersData newAnswer = (AnswersData)Instantiate(answerPrefab, uIElements.AnswersContantArea);

newAnswer.UpdateData(question.Answers[i].info, i);

newAnswer.Rect.anchoredPosition = new Vector2(0, offset);

offset -= (newAnswer.Rect.sizeDelta.y + parameters.Margins);

uIElements.AnswersContantArea.sizeDelta = new Vector2(uIElements.AnswersContantArea.sizeDelta.x, offset \* -1);

currentAnswers.Add(newAnswer);

}

}

void EraseAnswers()

{

foreach (var answer in currentAnswers)

{

Destroy(answer.gameObject);

}

currentAnswers.Clear();

}

}

1. StartScreenFadeOut.cs

using System.Collections;

using System.Collections.Generic;

using UnityEngine;

using UnityEngine.UI;

publicclassStartScreenFadeOut : MonoBehaviour

{

[SerializeField] privatefloat timeToFadeOut;

[SerializeField] private List<Image> imagesList;

[SerializeField] private Text text;

// Start is called before the first frame update

publicvoid StartFadeOut()

{

StartCoroutine(FadeOutCoroutine());

}

IEnumerator FadeOutCoroutine()

{

float time = 0;

float alpha = 0;

Color tempColor = new Color(1, 1, 1, 1);

while (time <= timeToFadeOut)

{

time += Time.deltaTime;

alpha = (float)1 - ((float) time / timeToFadeOut);

Debug.Log(alpha);

foreach (var item in imagesList)

{

tempColor = item.color;

tempColor.a = alpha;

item.color = tempColor;

}

tempColor = text.color;

tempColor.a = alpha;

text.color = tempColor;

yieldreturnnull;

}

}

}

# ДОДАТОК В

**УЧАСТЬ У КОНФЕРЕНЦІЇ**



Рисунок В.1 – Титульна сторінка матеріалів конференції

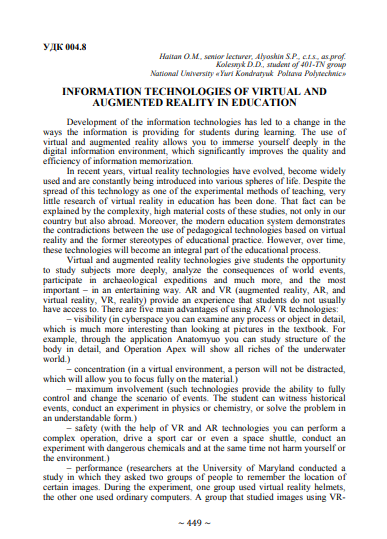


Рисунок В.2 – Перша частина тез

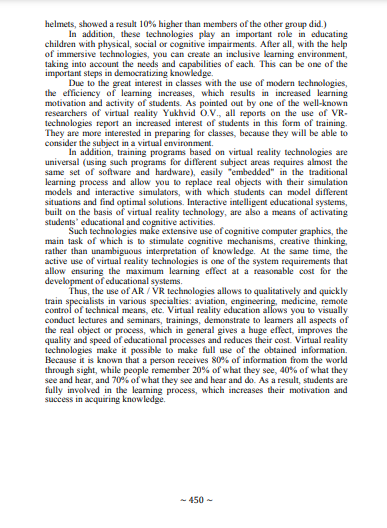


Рисунок В.3 – Друга частина тез