МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
СУМСЬКИЙ ДЕРЖАВНИЙ УНІВЕРСИТЕТ
ФАКУЛЬТЕТ ЕЛЕКТРОНІКИ ТА ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ
КАФЕДРА КОМП'ЮТЕРНИХ НАУК
СЕКЦІЯ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ ПРОЕКТУВАННЯ

КВАЛІФІКАЦІЙНА РОБОТА БАКАЛАВРА

на тему: «Віртуальна екскурсія парком "Казка" м.Суми**»**
за спеціальністю 122 «Комп'ютерні науки»,
освітньо-професійна програма «Інформаційні технології проектування»

Виконавець роботи: студент групи ІТ -62 Литвиненко Євгеній Сергійович

Кваліфікаційну роботу
захищено на засіданні ЕК
з оцінкою _____ « » травня 2020 р.

Науковий керівник _____ к.т.н., доц., Федотова Н.А.
(підпис)

Голова комісії _____ Шифрін Д. М.
(підпис)

Засвідчую, що у цій дипломній роботі немає
запозичень з праць інших авторів
без відповідних посилань.
Студент _____
(підпис)

Суми-2020

Сумський державний університет
Факультет електроніки та інформаційних технологій
Кафедра комп'ютерних наук
Секція інформаційних технологій проектування
Спеціальність 122 «Комп'ютерні науки»
Освітньо-професійна програма «Інформаційні технології проектування»

ЗАТВЕРДЖУЮ
Зав. секцією ІТП

_____
В. В. Шендрик
«__» _____ 2020 р.

З А В Д А Н Н Я
НА КВАЛІФІКАЦІЙНУ РОБОТУ БАКАЛАВРА СТУДЕНТУ

Литвиненко Євгенію Сергійовичу

1 Тема роботи *Віртуальна екскурсія парком "Казка" м.Суми*

керівник роботи *Федотова Наталія Анатоліївна, к.т.н.,доцент*,

затверджені наказом по університету від «*14* » травня 2020 р. № 0576-III
2 Строк подання студентом роботи «1» червня 2020 р.
3 Вхідні дані до роботи фото матеріали

4 Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити) Аналіз предметної області, проектування віртуальної екскурсії, реалізація віртуальної екскурсії

5 Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень)

6. Консультанти розділів роботи:

Розділ	**Консультант**	**Підпис, дата**	
		Завдання видав	**Завдання прийняв**

7.Дата видачі завдання_____

КАЛЕНДАРНИЙ ПЛАН

№ п/п	Назва етапів кваліфікаційної роботи	Строк виконання етапів роботи	Примітка
1	Аналіз останніх досліджень	До 12.12.19	
2	Розробка технічного завдання	До 15.03.20	
3	Планування робіт	До 22.03.20	
4	Вибір засобів реалізації	До 10.04.20	
5	Проектування та моделювання об’єктів	До 5.05.20	
6	Розробка додатку	До 15.05.20	
7	Тестування додатку	До 17.05.20	
8	Оформлення та здача пояснювальної записки та файлів розробленого проекту	До 01.06.20	

Студент _____
(підпис)

Керівник роботи _____
(підпис) к.т.н., доц. Федотова Н.А..

РЕФЕРАТ

Тема кваліфікаційної роботи бакалавра «Віртуальна екскурсія парком “Казка” м.Суми».

Пояснювальна записка складається зі вступу, 3 розділів, висновку, списку літератури із 21 найменування та додатків. Загальний обсяг роботи – 85 сторінок.

Кваліфікаційну роботу бакалавра присвячено створенню програмного додатка симулятора ходьби з локацією парку «Казка» м.Суми.

У першому розділі проведено дослідження технологій покращення якості продукту. Проведено аналіз програмного забезпечення та ігрових рушіїв. На основі порівнянь створено правила та вимоги до майбутнього додатку, а також було додано потрібний функціонал управління.

В другому розділі описано було проаналізовано вимоги до програмного додатка, побудовано контекстну діаграму, діаграму декомпозиції та діаграму декомпозиції розробки продукту.

В третьому розділі наведено дії з практичної реалізації віртуальної екскурсії. Результатом роботи є ігровий windows-додаток для екскурсії парком.

Практичне значення роботи полягає у створенні програмного додатка «Віртуальна екскурсія парком казка м.Суми» з простим функціоналом та може повністю виконувати поставлені задачі.

Ключові слова: ВІРТУАЛЬНА ЕКСКУРСІЯ, ДОДАТОК, UNITY, ДІАГРАМА, 3D-МОДЕЛЬ.

ЗМІСТ

ВСТУП

Сьогодні наявна проблема з відвідуванням різних міст та країн. Для економії власного часу та коштів люди потребують присутність, різного роду, розважального контенту. Саме використання спеціалізованого програмного забезпечення значно сильно полегшує подорожування, що є запорукою економії часу та коштів.

Багато компаній починають активно випробовувати демонстрацію своїх офісів інформаційними технологіями. Основною запорукою цих демонстрацій є висока деталізація та повний реалізм. Компанії та забудовники ставлять собі за мету – зацікавити клієнта, показати свої найкращі якості та приховати недоліки. Забудовники можуть показати на скільки сучасні будинки ті можуть побудувати, а офісні компанії можуть показати комфортабельність своїх місць роботи. Але екскурсію можна провести не лиже різними офісами та котеджами, а й історичними пам’ятками та парками. Саме тут на дизайнера лягає важка робота по створенню точних моделей та ідентичне перенесення атмосфери місця, яке потрібно відтворити в програмі та надати максимальної ідентичності з реальним місцем.

Основною проблемою таких екскурсій являється погана інтерактивність, адже саме так відсутнє відчуття, що ти знаходишся в цьому місці. Ця проблема може бути виправлена, але шляхом важкої роботи, фінансових витрат та збільшення часу яке відводиться на розробку та тестування програмного продукту.

Тому, метою роботи є реалізація додатку для екскурсії парком.

Для досягнення поставленої мети необхідно:

- Провести аналіз предметної області, зібрати інформацію про забудови, споруди, розважальні майданчики, об’єкти декору, рослинність та природні особливості ландшафту які наявні на території парку.

- Порівняти можливості програмних продуктів, які дають інструменти для вирішення завдання, але не підходять із-за ряду функціональних особливостей.
- Провести моделювання об’єктів та ландшафту.
- Інтегрувати створені об’єкти в інформаційну систему ігрового рушія.
- Виконати програмну реалізацію скриптів інформаційної системи.

Практичне значення програмного продукту полягає в тому, що створена інформаційна система буде мати унікальний функціонал, притаманний конкретному програмному продукту та використовувати моделі й ландшафт створені саме під цей програмний продукт.

1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

1.1 Огляд останніх досліджень і публікацій

Інформаційні системи постійно поповнюються новими технологіями, які одразу знаходять свою нішу серед програмних продуктів.

Потрібно постійно слідкувати за розвитком інформаційних технологій та технологій проектування, приділяти увагу дослідженням та розробкам. Час не стоїть на місці, а розвиток не зупиняється [1].

Трасування променів являється ще досить новою та непопулярною технологією відображення картинки, а особливо освітлення.

При всій непопулярності технології, саме цей спосіб побудови та рендеру моделей являється одним із найпрогресивніших та перспективних де головна частина моделювання зображення лягає на алгоритми обробки інформації [2].

Суть технології полягає в тому, що об’єкти отримують набагато більше специфікацій, а картинка майорить насиченими кольорами. При такій технології метали починають реалістично не лише відбивати світло, а й за допомогою алгоритму можуть відбивати зображення картинки, як дзеркало, але при строгих умовах. Тепер освітлення, тіні та відблиски отримують динамічний режим і обробляються алгоритмом в реальному часі. Вода тепер має не лише прозорість та відтінок, а й отримує спеціальну уявну сітку. І якщо, наприклад, уявне море має хвилі, то промені світла будуть відображатися не лише на поверхні водойму, а й цей ефект тепер буде ображуватися під водою. Насиченість картинки збільшується, помітна зміна глибини кольору. Об’єкти тепер відображуються як справжні, що надає картинці максимального реалізму.

Компанія Nvidia, турбуючись про дизайнерів, намагається оптимізувати обробку променів та зробити технологію більш доступною, додаючи все більше інструментів та спрощуючи рендер. [3].

Алгоритм цієї технології ще має вади. Важким каменем із-за якого важко донести цю технологію до звичайного користувача є обмежена кількість пристроїв які підтримують цю технологію. Щоб реалізувати алгоритм обробки променів потрібно спеціальне обладнання, яке, на жаль, являється ще досить дорогим та недоступним звичайному користувачеві.

У зв'язку з цим виникає необхідність заміни цієї технології на досить простий та доступний варіант, який можна реалізувати та використовувати не маючи особливих навичок в цій сфері.

Занурення у віртуальний світ стає все більш глибоким. Напевно, ще зовсім недавно потрібно було фантазувати та домальовувати картинку в голові, а зараз комп'ютерну графіку важко відрізнити від реальності [4].

Віртуальна реальність – це різновид реальності, який почали використовувати розробники програмного забезпечення не так давно.

Особливістю цієї технології являється повне занурення в створену розробником атмосферу. Ілюзія дійсності, саме так розробник вводить в оману мозок і заставляє повірити користувача, що той знаходиться у віртуальному світі, як в реальному.

Технологія віртуальної реальності має досить високу інтерактивність, адже дає змогу взаємодіяти з різними об'єктами, та реальностями, а також маніпулювати ними. Цю технологію ще розвивають, але вже помітна неймовірна реалізація в різних програмних продуктах, які забезпечують людей не лише розважальним контентом, а й пізнавальним та навчальним.

Процес використання віртуальної реальності може декілька відрізнятися між собою, але основними критеріями є наявність спеціального обладнання, до складу якого входять важкі окуляри з вбудованою гарнітурою, маніпулятори з відстежувачами, які кріпляться на руки, ноги та пояс та спеціальних приймачів, які відстежують місцезнаходження

користувача в просторі. А також важливо і велике приміщення для комфортного пересування в додатках та програмних продуктах, що використовують дану технологію

Останні дослідження показали, що віртуальна реальність все сильніше впливає на життя людини. Навчання, яке потребує спеціальної підготовки вже переходить на використання віртуальної реальності. Авжеж це не замінить повністю навчання в реальному світі, але саме програмно можна змоделювати багато специфічних ситуацій, де потрібно перевіряти навички користувача.

З великими плюсами цієї технології маємо і великі недоліки. Вже зараз проводиться дослідження про вплив віртуальної реальності на мозок людини. Не дивлячись на погану активність розробників в цій сфері, людина не хоче повертатися в нецікавий реальний світ та більше надає перевагу повернутися в віртуальний світ, який наповнений різними пригодами. Часто ігри з використанням цієї технології створюються розробником, який не має особливої мети створити програмний продукт високої якості. Такі додатки мають лише мету зібрати якомога більше коштів при продажу, пропонуючи неймовірну особливість саме цього додатку, а насправді ніяких особливостей не мають взагалі.

До цієї технології належить технологія доповненої реальності. Технологія доповненої реальності використовується вже більше для навчальної діяльності. Технологія потребує вже менше обладнання. Такі компанії як Google та Apple покращують свої технології в цьому напрямку, щоб зробити технологію доповненої реальності більш масовою та доступною звичайному користувачеві.

Окуляри доповненої реальності з кожним роком стають все менші та отримують все більше функціоналу, що не може не радувати звичайних користувачів [5].

Особливістю доповненої реальності є поєднання в собі декількох реальностей та маніпулювання об’єктами в інформаційному полі. Саме

доповнена реальність може перевернути інформаційну сферу та надати нову можливість пізнання світу в новому форматі.

Ця технологія являється досить новою та дослідження про вплив її на зір на сприйняття реального світу ще ведуться.

Метою даного проекту є створення програмного продукту, який буде використовуватися, як екскурсія парком. Додаток має бути доступним для різних користувачів, а також бути універсальним у використанні. Аналізуючи ці технології та досліджень, що ведуться по ним, можна сказати, що не обов'язково додавати їх у цей програмний продукт. Технології являються сирими та надають надлишкові можливості.

Потрібно і розуміти, що деякий функціонал можливо замінити більш простими алгоритмами, або взагалі краще від них відмовитися.

Саме заміна непотрібних функцій або повна відмова від них, спрощує задачу розробника та знімає додатковий тиск на програмне забезпечення користувача [6].

Екскурсія не потребує повного занурення в атмосферу. Достатньо створити настрій користувачеві, при якому клієнт захоче відвідати цей парк уже в реальному світі.

Аналогічно цьому, достатньо створити імітацію реальності, а не повна їй відповідність. Потрібно створити дуже простий та приємний програмний продукт з використанням універсальних та легких інструментів.

1.2 Аналіз програмних продуктів

У цьому розділі будуть розглянуті програмні продукти, які дають можливість створення додатку. Переваги та недоліки.

Спочатку необхідно виділити пункти для порівняння. Ці пункти повинні відповідати вимогам.

Інформаційні системи мають:

- Написання на мовах C++/C#;
- кросплатформеність готового продукту;
- операційна система Microsoft Windows.

1.2.1 X-Ray (ігровий рушій)

Цей ігровий рушій використовує особливу технологію освітлення та затемнення. Дозволяє отримати високу достовірність в рендерінгу освітлення, маючи високу геометричну складність сцени. Багаторазово допрацьовувався та зараз має підтримку DirectX11. Наявна система динамічної зміни дня та ночі, а також підтримка різних погодних умов. Написаний на C++, має обмежену підтримку.

Цей ігровий рушій створює відчуття плину часу та занурює в атмосферу створеної гри. Але навіть для звичайного розробника, цей рушій може показатися складним, бо він не розрахований для використання на малі ігрові локації [7].

Рисунок 1.1 – X-Ray

1.2.2 4A Engine

Ігровий рушій, розроблений компанією 4AGames для внутрішнього використання в своїй комп’ютерній грі Metro 2033. Двигун використовує API

DirectX12 та OptnGL 4. Має технології Nvidia: PhysX та Nvidia RTX. Рушій використовує багатопоточність та використовує задану модель без будь-якої попередньої перевірки станів, тим самим дозволяючи завданням виконуватися паралельно. Має симуляцію вигинів в одязі, хвилі на воді та інші елементи повністю підвладні впливу навколишнього середовища. Повна підтримка на Microsoft Windows.

Рисунок 1.2 – 4A Engine

1.2.3 Creation Engine

Ігровий рушій розроблений компанією Bethesda Game Studios для внутрішнього використання. Рушій передбачає велику ігрову локацію, відкритою для вільного пересування. Присутній штучний інтелект якому відповідає власна система. Система управління сюжетом дозволяє розробникам змішувати завдання, які створені вручну та які згенеровані випадково. Мова програмування C++, має підтримку Microsoft Windows.

Досить амбіційний ігровий рушій. Має велику та добре розроблену систему штучного інтелекту яке може імітувати життя міста. Реалізація деяких функцій неможлива, але це і не потрібно. Проблемою є лише те, що цей ігровий рушій створювався для внутрішнього використання компанії [8].

Рисунок 1.3 - Creation Engine

1.2.4 Unity

Кросплатформене середовище розробки комп’ютерних ігор. Розроблена компанією Technologies. Має візуальне середовище розробки, підтримку різними платформами та модульну систему компонентів. Підтримує безліч популярних форматів, має систему адаптації та оптимізації. Рушій дає можливість передавати моделі або програмну частину в особливому форматі. Мова інтерфейсу – англійська. Підтримка Microsoft Windows. Мова програмування C#.

Цей ігровий рушій також дає можливість створювати не лише 3D ігри, а й 2D, вражаючи набором функціоналу та інструментів для створення проекту [9].

Рисунок 1.4 – Unity

Проаналізувавши вищенаведені програмні продукти можна дійти до висновку, що краще за все, для реалізації проекту підходить Unity. Саме ця

платформа з великим відривом лідирує серед інших зазначених ігрових рушіїв.

Unity вражає своєю універсальністю та простотою. В цьому ігровому рушії реалізовано не лише можливість підключати плагіни, а й імпортувати різні інструменти і контент прямо з офіційного сайту [10].

1.3 Постановка задачі

Мета дипломного проекту – створення програмного продукту, віртуальної екскурсії парком з можливістю пересування та відвідування різних локацій.

Для досягнення мети необхідно слідувати таким пунктам:

- Проаналізувати будівлі парку та ландшафт парку, прослідити за хронологією реставрації парку;
- Описати необхідний мінімум функціоналу, який повинен мати програмний продукт;
- Розробити інтерфейс та управління для користувача;
- Оптимізувати програмне забезпечення.

Додаток буде використовуватися великим колом осіб, які зацікавлені даним типом програмного забезпечення. Вони повинні бачити деталізовану картинку при низьких системних вимогах. Всі об’єкти повинні мати чіткий деталізований контур та мають бути легко пізнаваними відвідувачами парку. Моделі та ландшафт повинні бути добре пророблеими та мати приємний вигляд. Освітлення повинне бути м’яким та естетичним, яке не нав’язує об’єктам особливі відтінки. Потрібно провести детальну роботу з дрібницями: дерева та кущі повинні мати легке похитування, вода мати легкі контури, а земля та інші поверхні повинні відповідати матеріалам.

Інтерфейс додатка повинен бути інтуїтивно зрозумілим, не нагромоджати головний екран та не навантажувати користувача зайвою інформацією. Управління повинне бути простим та логічним, щоб при роботі в програмі, у користувача не виникли питання стосовно управління. Для полегшення навчання у використанні програми, реалізувати вбудовану інструкцію у вигляді текстового документа з коментаріями до кожного елементу.

2 ПРОЕКТУВАННЯ ВІРТУАЛЬНОЇ ЕКСКУРСІЇ

2.1 Проектування програмного додатку

Для створення додатку було обрано ігровий рушій Unity 5.6.4 PRO та мову програмування C#.

Ігровий рушій має покращену систему підтримки підключаємих модулів при стандартному налаштуванні, наявна функція оптимізації дальності відтворення [11].

При створенні ігрової локації, потрібно змінити деякі налаштування Unity для надання більш привабливого вигляду відображенню, а також додати можливість адаптивного розміру вікна [12].

C# використовується для написання скриптів, описання логіки поведінки об’єктів та анімації для них. Ця мова програмування надає можливість описувати дії в програмному продукту. Вбудовані графічні інструменти Unity дозволяють будувати інтерфейс користувача, дають можливість створювати та маніпулювати об’єктами в сцені, а також одразу мати змогу перевірити вигляд продукту.

C# найбільш близький до Java та C++, підтримує поліморфізм та перевантаження операторів, а також має статично типізацію. Був розроблений для Microsoft.NET Framework [13].

Для зручного користування додатком, вікно програми матиме назву, яка відповідає назві додатку. Всі кнопки будуть підписані та виконуватимуть лише описувані ними функції.

Основні елементи, що будуть використовуватися є головне меню додатку, зміна мови та головний екран демонстрації.

Головне меню додатку дає можливість почати роботу з програмою, змінити мову інтерфейсу та закрити вікно додатку.

Зміна мови інтерфейсу дозволяє змінити мову інтерфейсу.

Головний екран демонстрації дає користувачеві управління аватаром у віртуальному світі додатку.Для реалізації всіх функцій необхідно чітко описати структуру додатку. Враховуючи, що вікна матимуть різний функціонал, то вони будуть розділені на категорії та реалізовуватимуться тільки з урахуванням описаної функціональності.

Процес розробки дизайну постійно привертає увагу, адже саме в цих схематичних малюнках починають зароджуватися ідеї, реалізовуватися мрії та саме тут починають реалізуватися прагнення [14].

Перша форма, яка буде створена – головне меню додатку. Воно має надати користувачеві вибір мови, а також можливість почати роботу додатку або завершити роботу. Макет вікна показано на рисунку 2.1.

Рисунок 2.1– Головне меню додатку

Після успішного початку роботи додатку, користувачеві буде надано управління аватаром у віртуальному світі, де той зможе пересуватися по заданій локації.

Рисунок 2.2 – Робота додатку

Для завершення роботи додатку достатньо закрити вікно програми. Змінення розширення вікна додатку – не передбачено через відсутність необхідності в даній можливості. Інтерфейс спрощений до максимальної ступені.

2.2 Побудова контекстної діаграми IDEF0

IDEF0 – методологія функціонального моделювання та графічний опис процесів, що призначене для формалізації та опису бізнес процесів. IDEF0 має особливість – це увага прикута до ієрархічного представлення об’єктів, що особливо полегшує розуміння предметної області. В IDEF0 розглядаються логічні зв’язки між роботами, а не послідовність їх виконання в часі [15].

Для створення IDEF0 потрібно відокремити задачі та роботи, які виконуватимуться. Побудова проходить з чітких графічних блоків зі стійкою структурою.

Інформаційна система, що описується, має конкретну задачу – віртуальна екскурсія парком «Казка» м. Суми. Щоб побудувати діаграму, необхідно визначити вхідні дані, вихідні дані, механізми та управління. Саме вони будуть використані для її побудови.

Вхідними даними є: власне спостереження, інформація про парк.

Вихідними даними є: комп’ютерна програма.

Механізмами є: програмні продукти, розробник, апаратні засоби.

Управліннями є: нормативні акти, технічне завдання, правила.

Рівень контекстної діаграми А0, який є найвищим рівнем абстракції.

Діаграма будується з точки зору архітектора проекту. Мета – віртуальна екскурсія парком «Казка» м. Суми.

Контекстна діаграма представлена на рисунку 2.3.

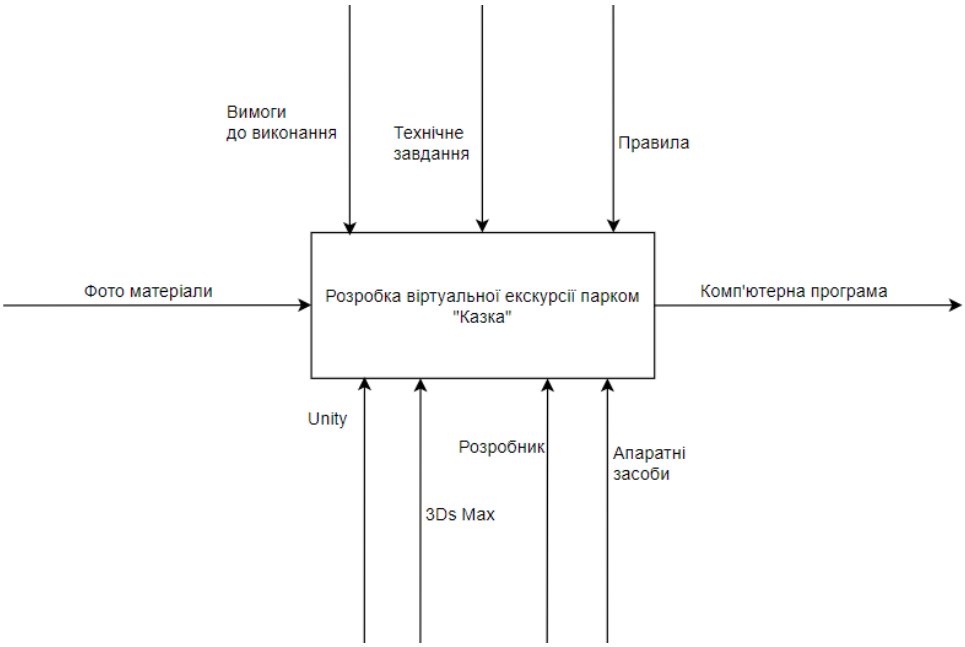


Рисунок 2.3 – Контекстна діаграма

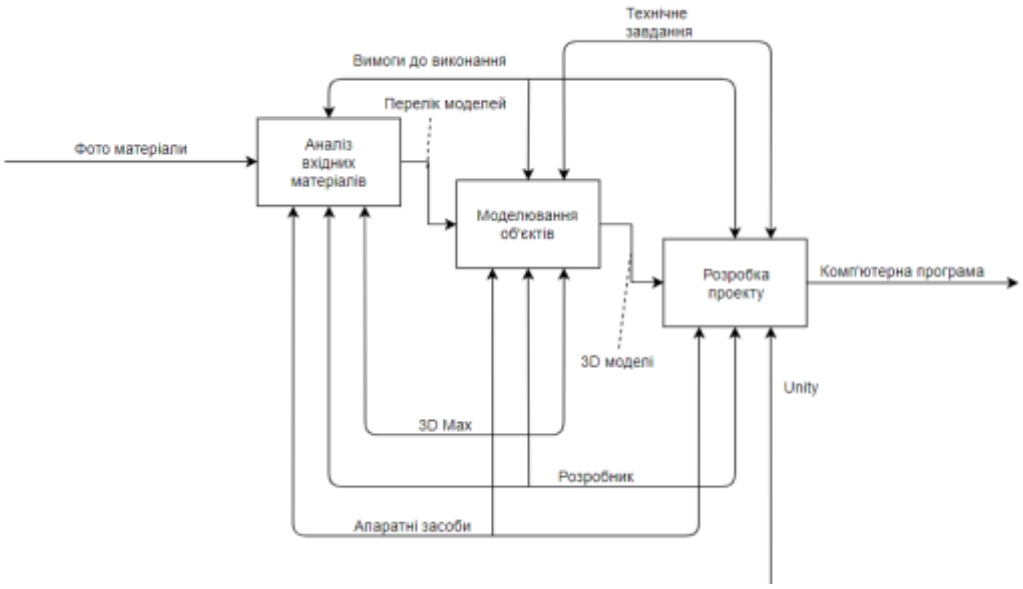


Рисунок 2.4 – Діаграма декомпозиції

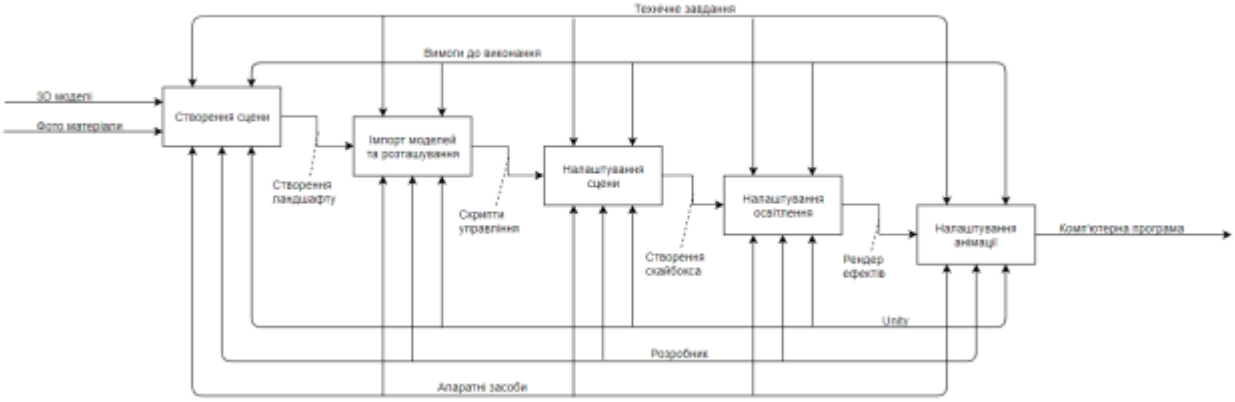


Рисунок 2.5 – Діаграма декомпозиції розробки

Варіанти використання – це графічне представлення функцій, які надаються системою користувачеві. Надається набір функцій для кожного варіанту використання, які виконуються системою під час роботи з актором.

На рисунку 2.5 представлено діаграму використання

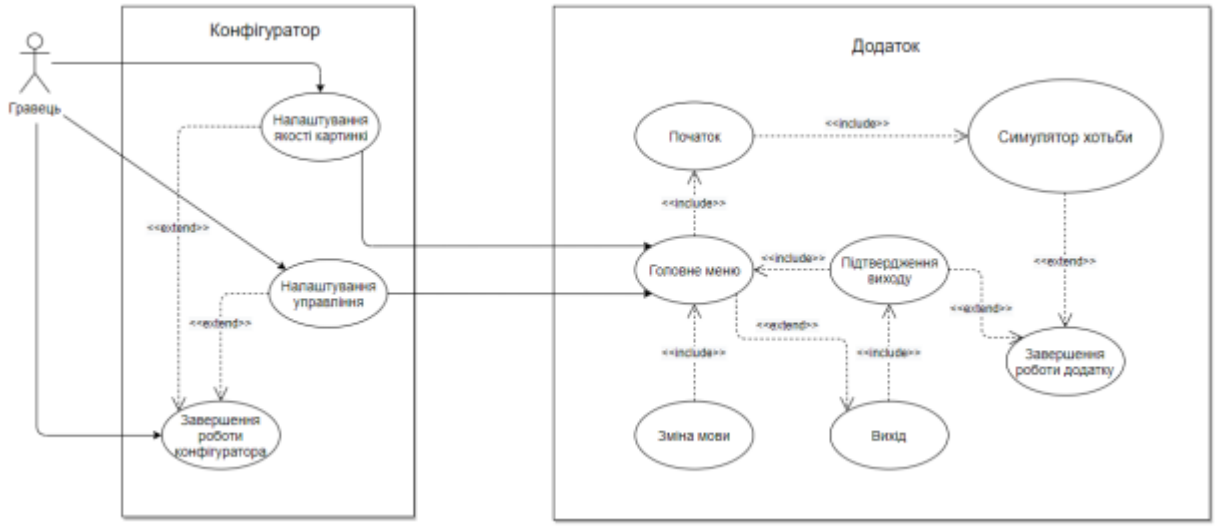


Рисунок 2.6 – UML діаграма

3 РЕАЛІЗАЦІЯ ВІРТУАЛЬНОЇ ЕКСКУРСІЇ

3.1 Процес розробки 3D-моделі

Створення починається з моделювання сплайну кола, яке потім потрібно витягувати вверх враховуючи всі вирізи для дверей та вікон. Потрібно дотримуватися охайності сітки та не допускати випадкових зміщень вершин полігонів. Після видаляння полігонів для вікон та дверей, об’єкту придасться об’єм за допомогою модифікатора Shell.

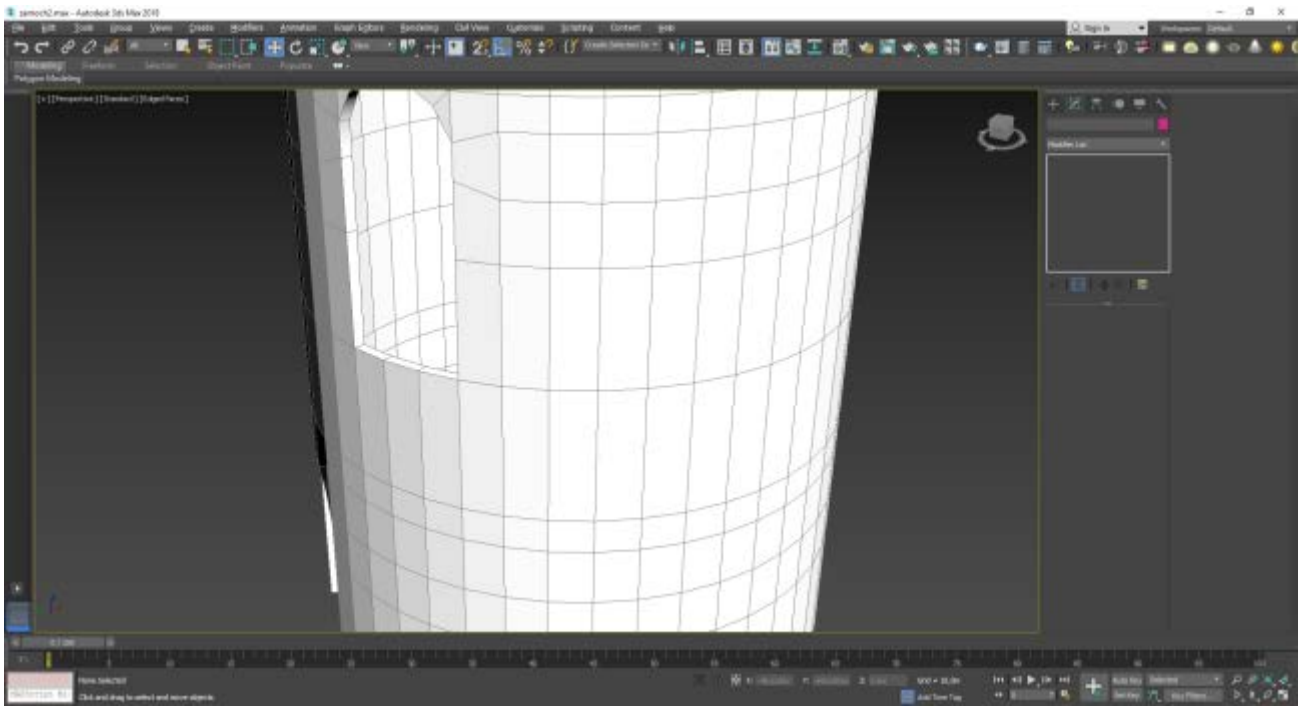


Рисунок 3.1 – Початок створення основної частини замку

Основні частини забудови лише на перший погляд схожі, сітку вони мають різну. Отже, потрібно моделювати ці частини замку з початку, використовуючи модифікатори, які були використані при створенні першою частини замку.

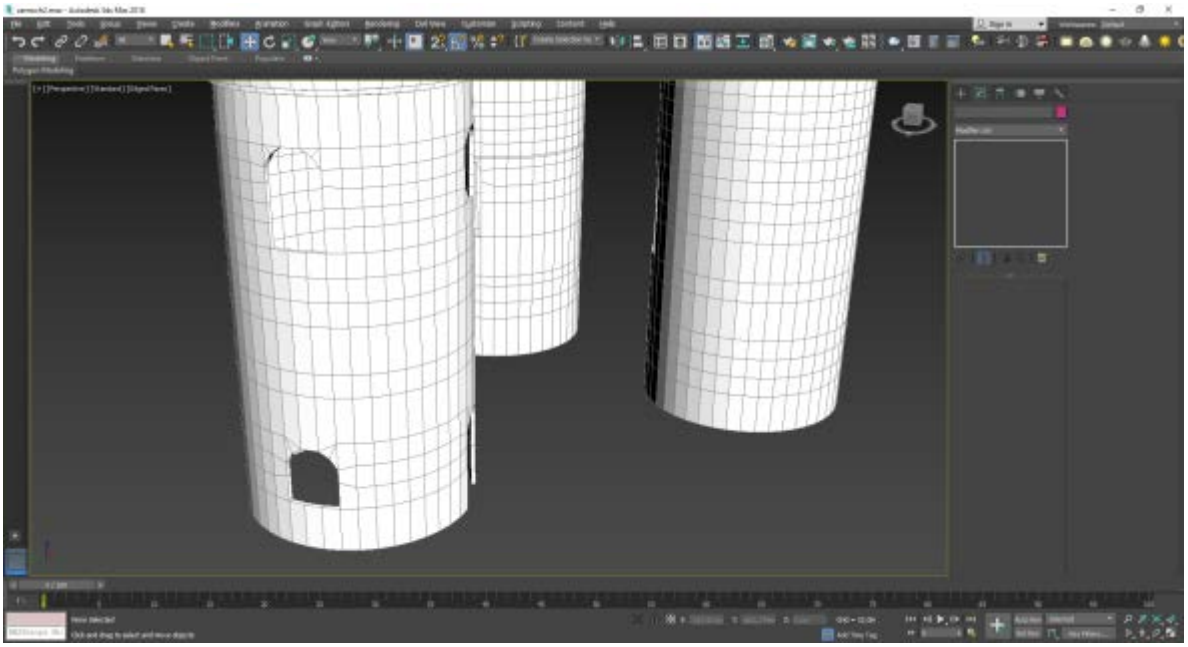


Рисунок 3.2 – Створення двох основних деталей замку

Накладати текстури потрібно поетапно, присвоюючи різним рівням замку – різні види текстур. Робиться це шляхом виділення потрібних полігонів модифікатором Edit Mesh, та присвоєння їм потрібного матеріалу. Якщо матеріал потрібно підігнати під модель, то краще використати UVW Map.

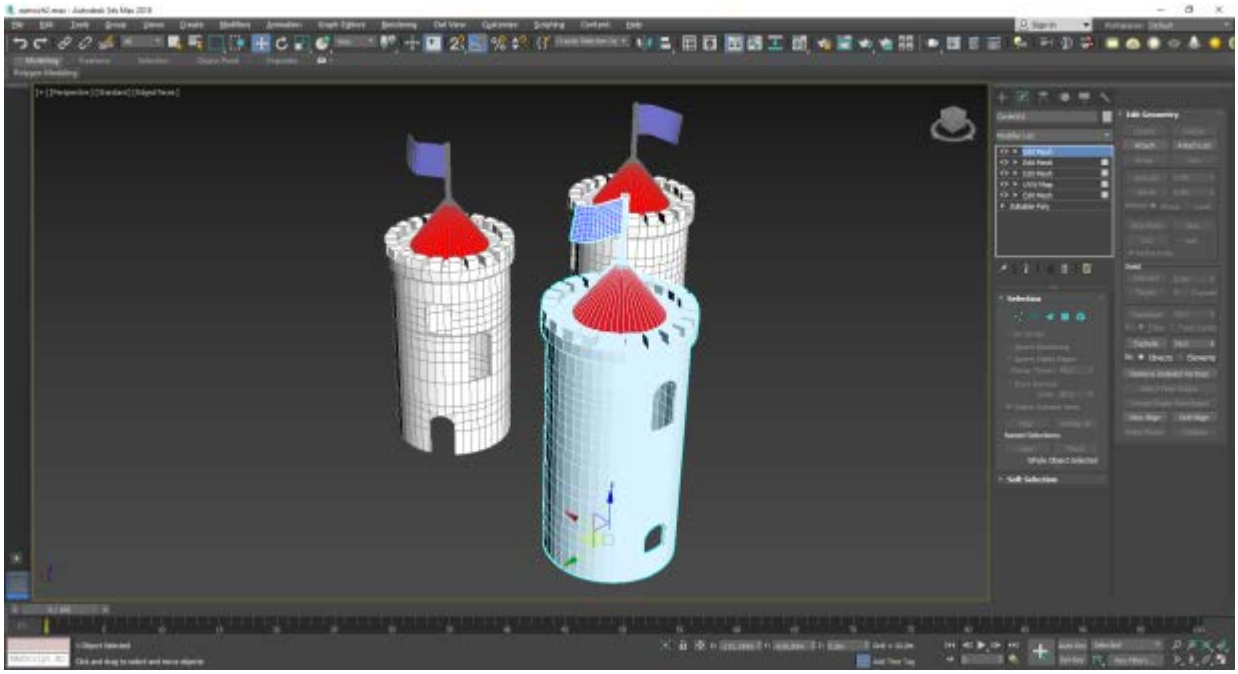


Рисунок 3.3 – Накладання текстур на об’єкти

Підлога на другому поверсі між банями замку має в основі багатокутник, малювати потрібно за допомогою Splines, та надати потрібний об’єм модифікатором Editable Poly.

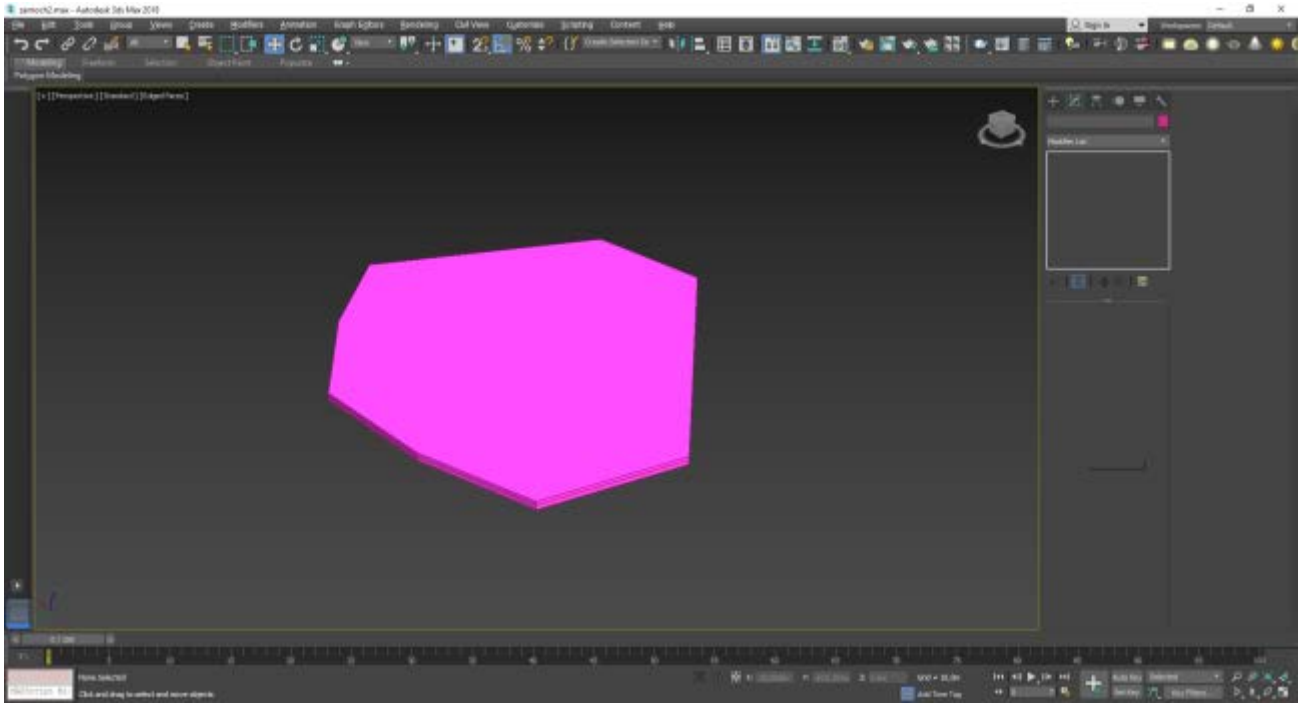


Рисунок 3.4 – Створення підлоги-перегородки
між основними вежами замку

В основі огорожі лежить той самий багатокутник, який було використано для створення підлоги та такий же багатокутник, але з меншим масштабом. Потрібно з’єднати ці сплакни, надати їм об’єм та додати ребер для сітки з подальшим витягуванням для надання форм огорожі. Огорожа майбутніх сходинок створюється за допомогою послідовних витягувань полігонів, вирізання непотрібних вершин та створення потрібних полігонів між ребрами.

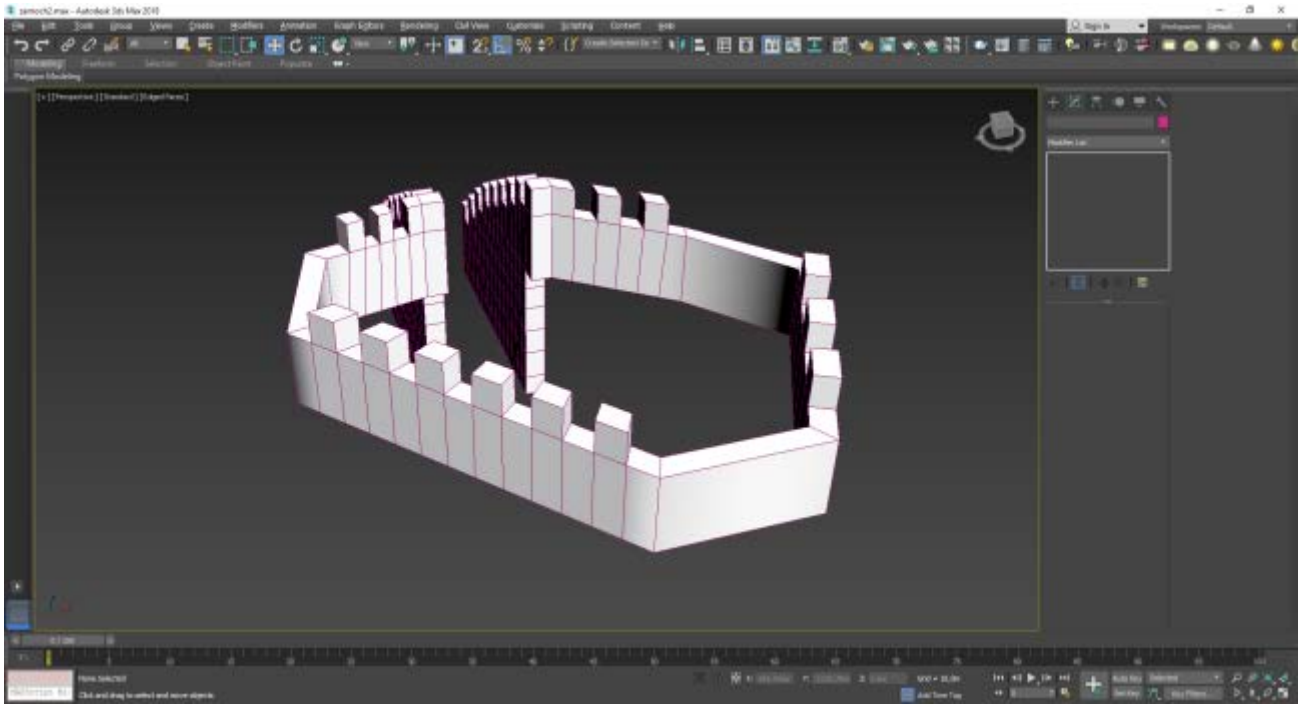


Рисунок 3.5 – Створення огорожі між вежами замку

Текстура для огорожі використовується одна, але вона потребує налаштування. Налаштування проводиться за допомогою модифікатора UVW Map.

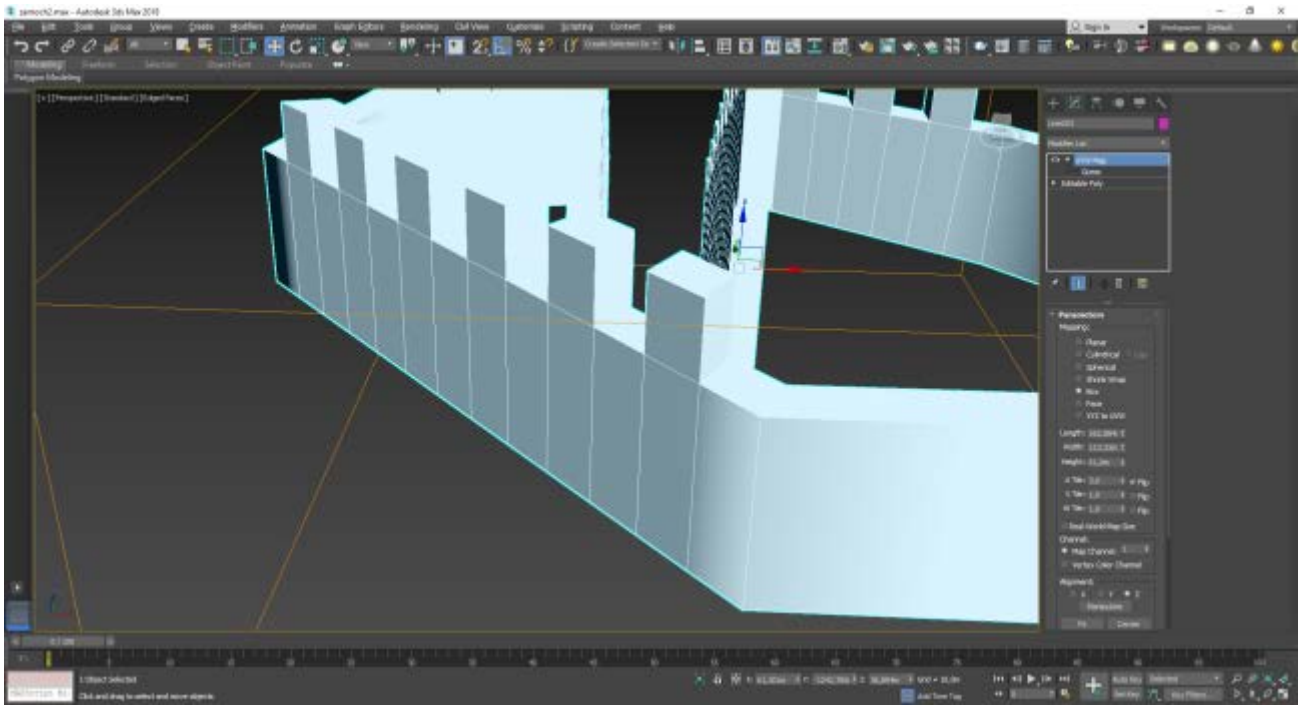


Рисунок 3.6 – Накладання та налаштування текстури огорожі

Основою дверей являється Tube, модифікований за допомогою модифікатора Editable Poly. Самі двері – це Box, а дверна ручка та замкова щілина намальовані за допомогою Splines. Арка над дверима потребує детального редагування, щоб матеріал не мав швів та відповідав дійсності. Налаштування проводиться за допомогою модифікатора UVW Map на полігони які виділили модифікатором Edit Mesh.

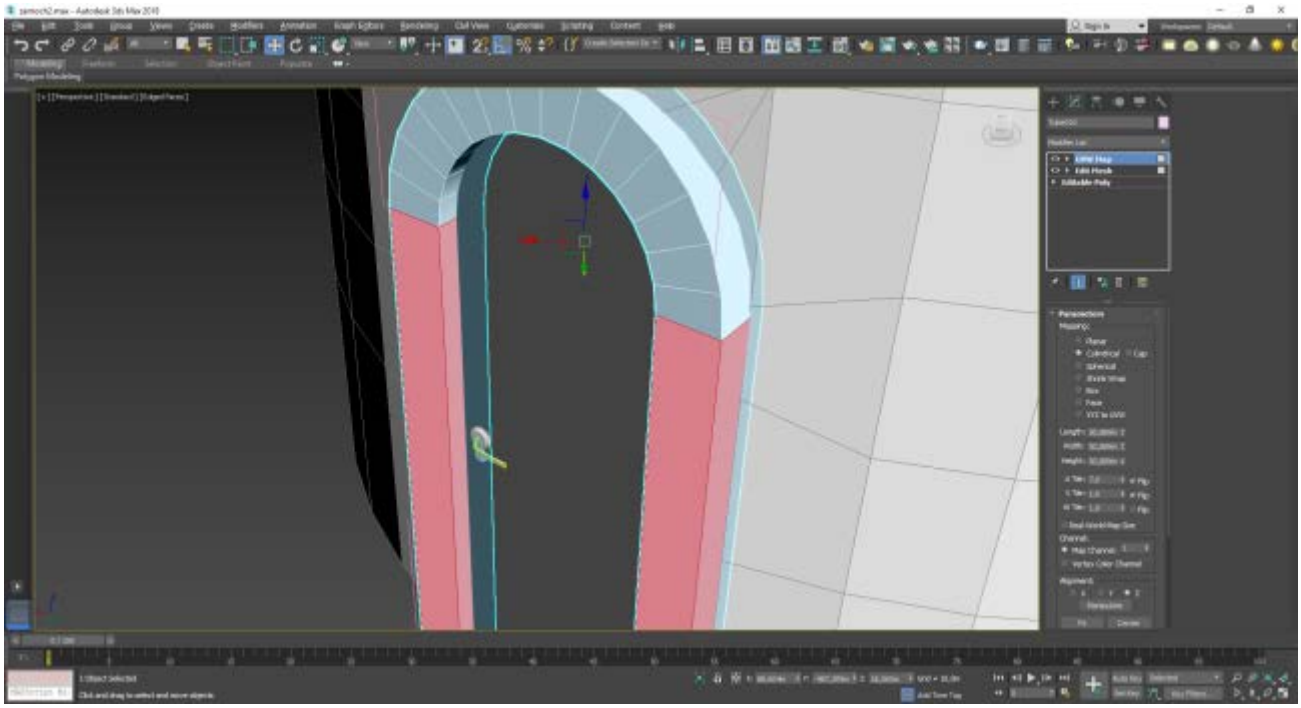


Рисунок 3.7 – Створення дверей та накладання текстур з подальшим налаштуванням

Основою маленького вікна являється два Tube, модифіковані за допомогою модифікатора Editable Poly. Перегородка вікна – це Box. Арка над та під вікном потребує детального редагування, щоб матеріал не мав швів та відповідав дійсності. Налаштування проводиться за допомогою модифікатора UVW Map на полігони які виділили модифікатором Edit Mesh.

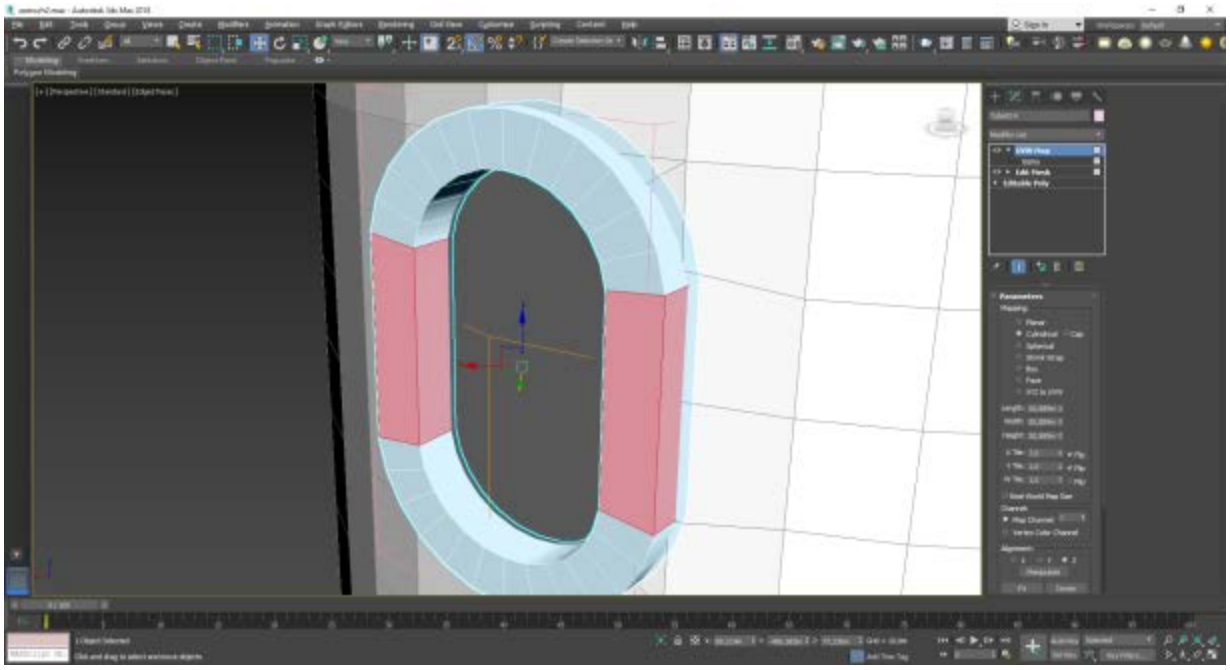


Рисунок 3.8 – Створення вікна та накладання текстур з подальшим налаштуванням

Велике вікно мало чим відрізняється від дверей, лише перегородкою та розмірами. Обов’язково потрібно прослідкувати за правильністю розмірів та рівним накладанням текстур.

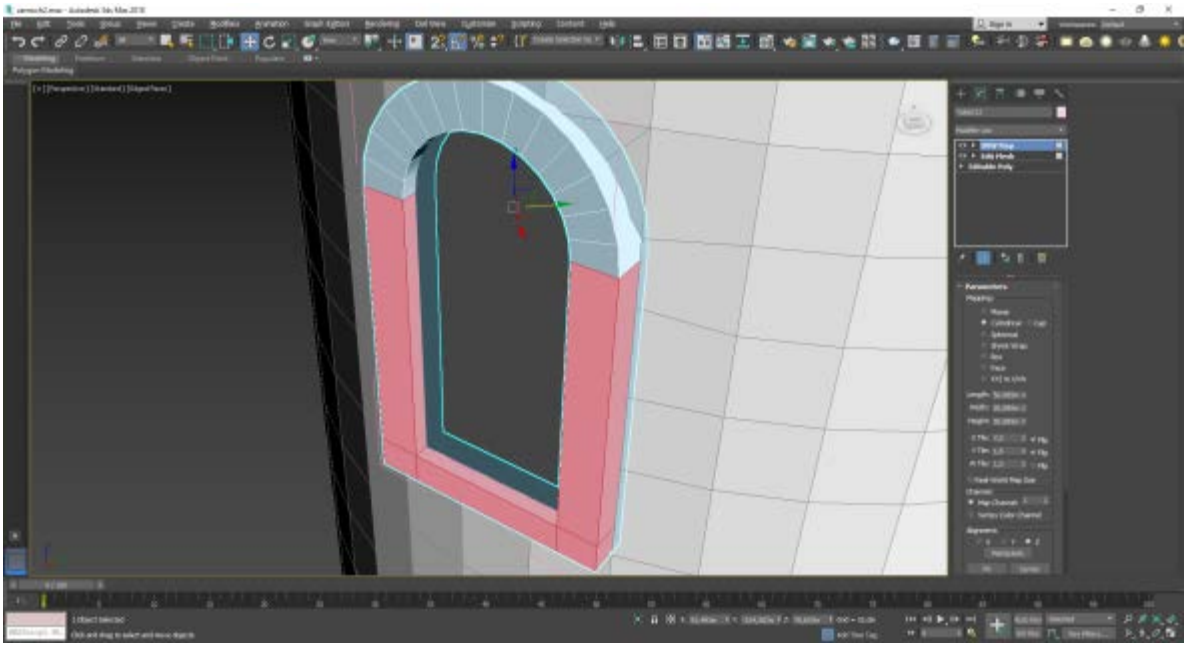


Рисунок 3.9 – Створення іншого вікна та накладання текстур з подальшим налаштуванням

Сходинки моделюємо звичайні за допомогою StraightStair. Накладання текстур рівномірне та втручання не потребує.

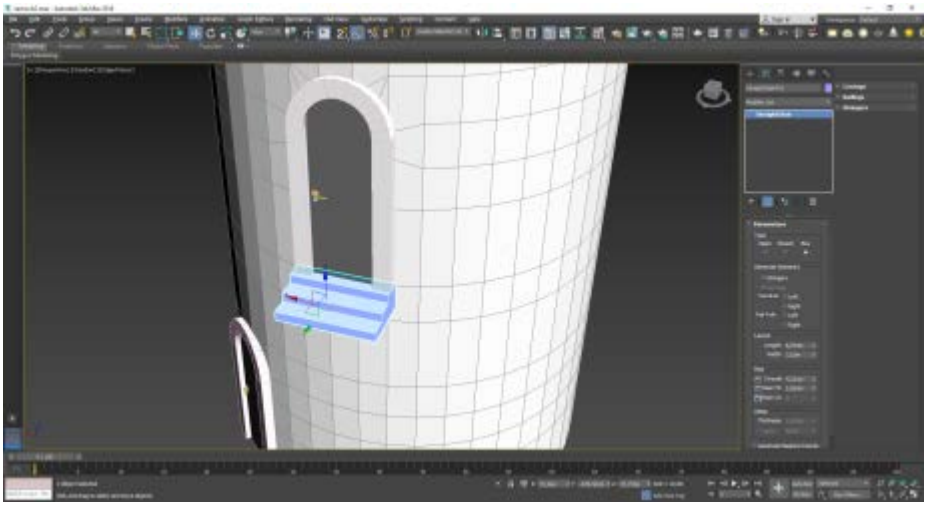


Рисунок 3.10 – Створення сходинок на другому поверсі

Сходинки моделюємо звичайні за допомогою StraightStair. Потрібно дотримуватися ергономічності та естетичності сходинок. Накладання текстур рівномірне та втручання не потребує.

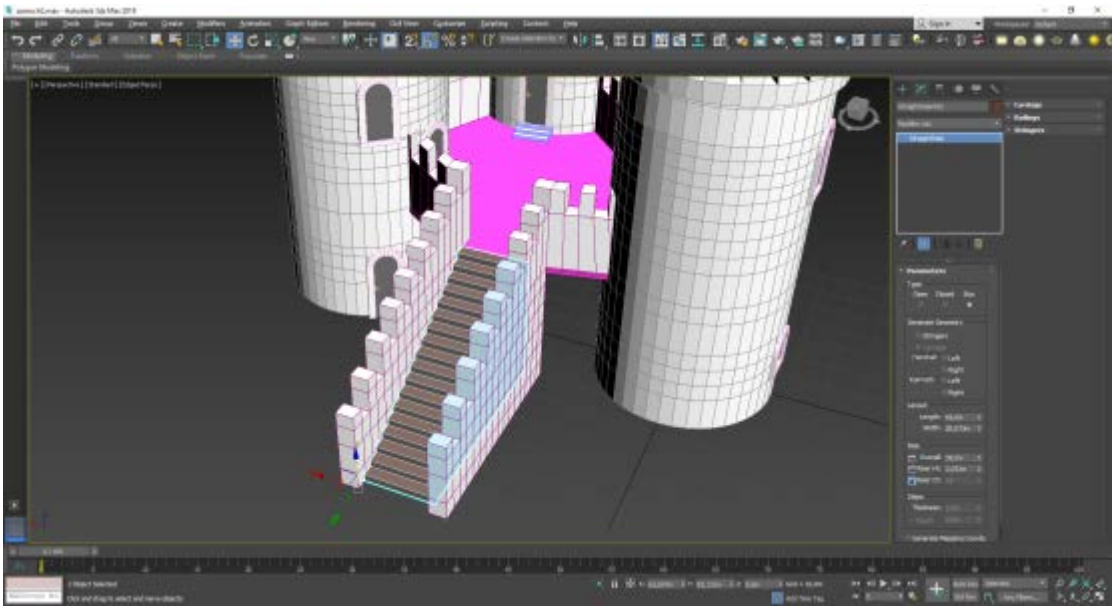


Рисунок 3.11 – Створення сходинок на другий поверх замку

Площину фундаменту малюємо за допомогою Splines. Об’єм неважливий тому лише створюємо полігон за допомогою Editable Poly. Накладання текстури рівномірне та жодних маніпуляцій не потребує.

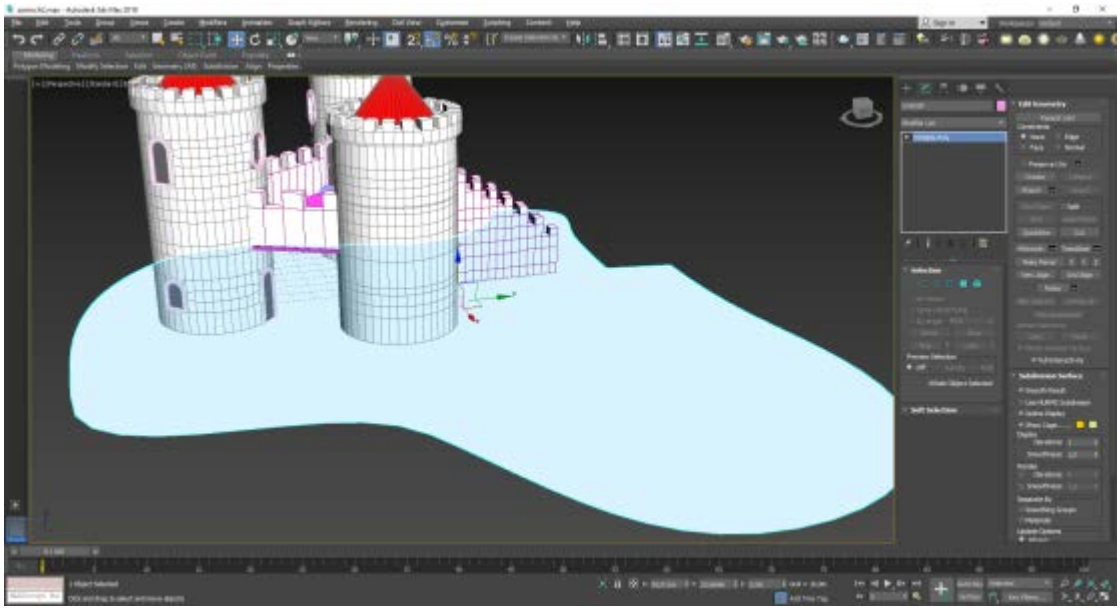


Рисунок 3.12 – Створення площини з фундаментом замку

Огорожа створюється за допомогою двох з’єднаних сплайнів, якім надається об’єм за допомогою Editable Poly.

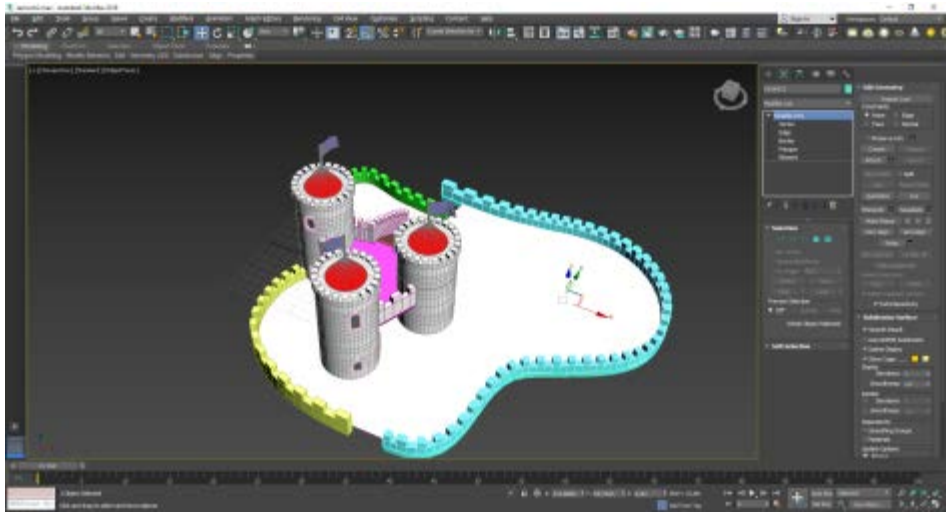


Рисунок 3.13 – Створення огорожі фундаменту замка

Площину двох інших рівнів ландшафту малюємо за допомогою Splines які знаходяться на різній висоті. Об’єм неважливий тому лише створюємо полігон за допомогою Editable Poly. Накладання текстури рівномірне та жодних маніпуляцій не потребує.

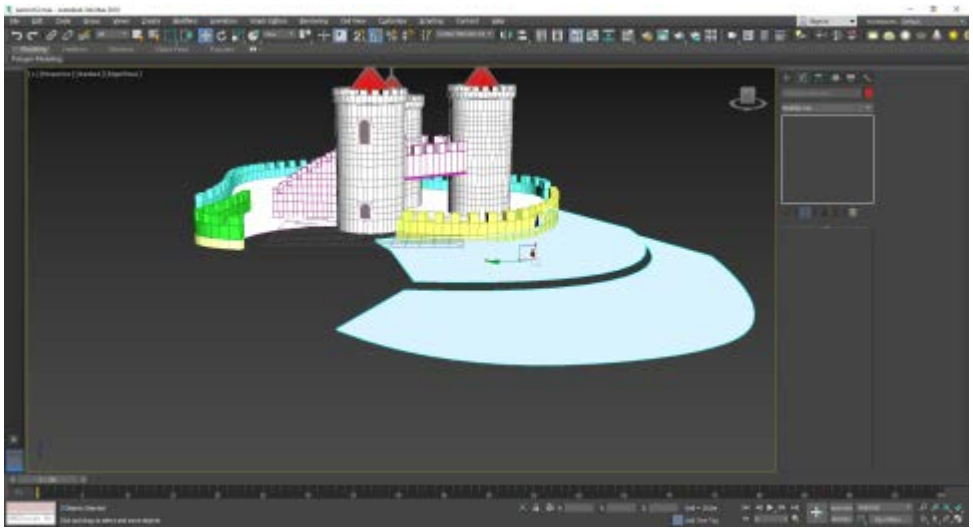


Рисунок 3.14 – Створення двох інших рівнів ландшафту

Дві інші огорожі, що прилягають до різних рівнів ландшафту створюється аналогічно до першої огорожі фундаменту, якім надається об’єм за допомогою Editable Poly. Накладання текстури рівномірне та жодних маніпуляцій не потребує.

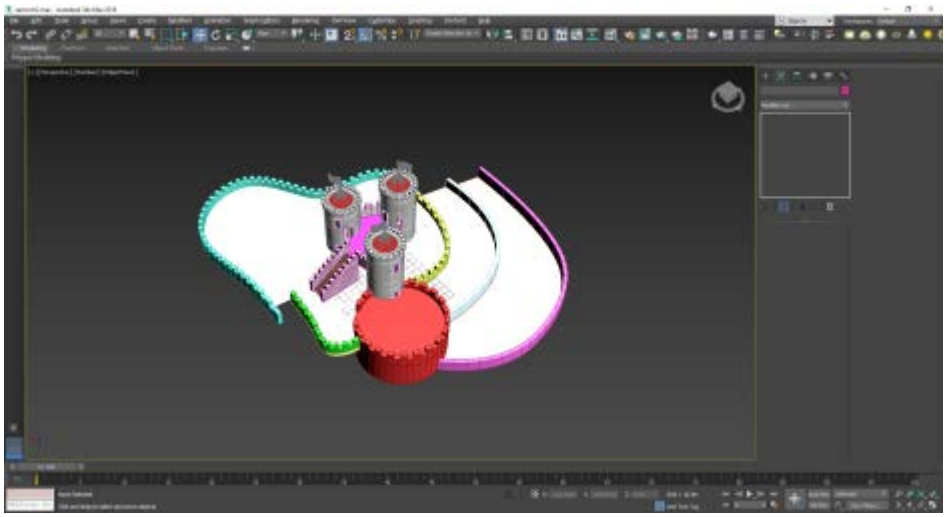


Рисунок 3.15 – Створення огорожі двох інших рівнів огорожі

Сходинки моделюємо звичайні за допомогою StraightStair. Потрібно дотримуватися ергономічності та естетичності сходинок. Сходинки, які ведуть через декілька рівнів мають додаткову модель з низькою основою. Накладання текстур рівномірне та втручання не потребує.

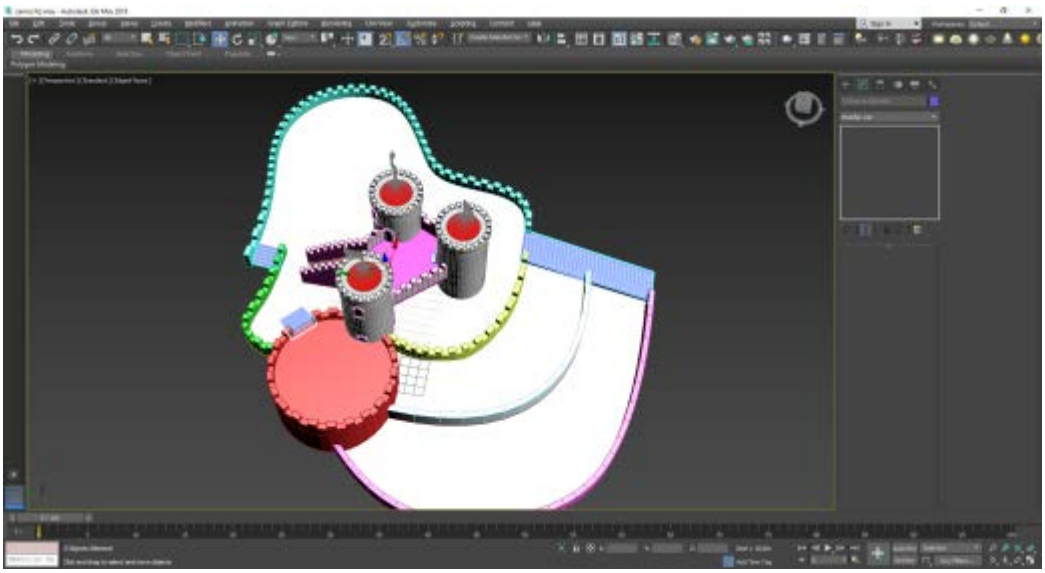


Рисунок 3.16 – Створення сходинок до різних рівнів ландшафту

3.2 Програмна реалізація

Розробка програмного продукту ділиться на два головних етапу, які створюються по черзі та мають безпосередній зв’язок між собою. Спочатку створюється головна сцена де користувач отримує управління над аватаром та має змогу вільно переміщуватися по головній локації.

На рисунку 3.17 представлено створений Terrain та декілька типів лан шафу.

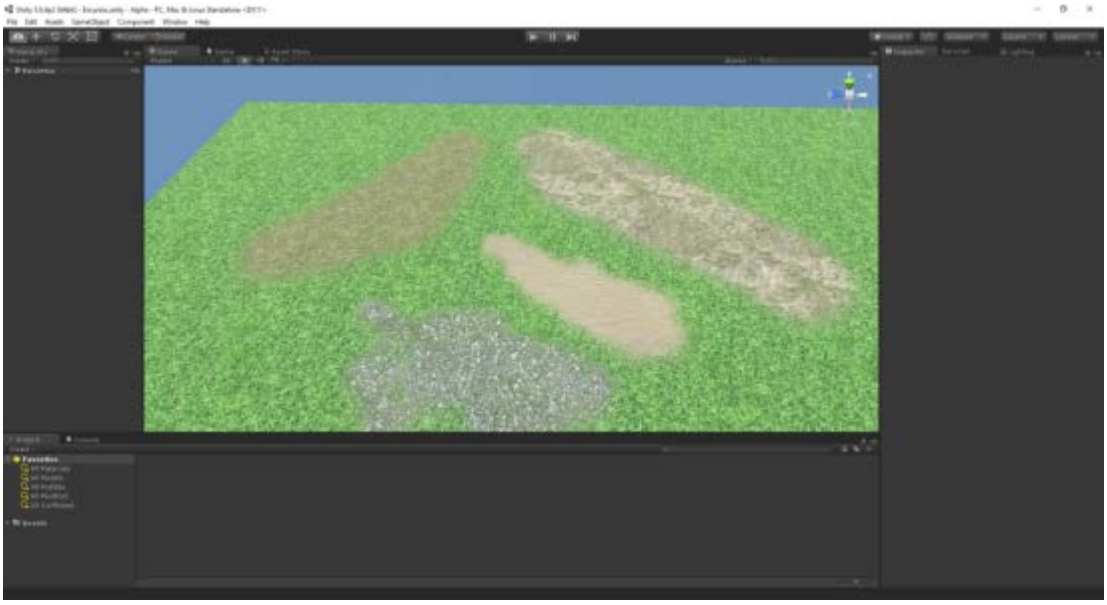


Рисунок 3.17 – Демонстрація площини

Одним із головних типів поверхонь являється вода. Потребується особлива увага для правильного та реалістичного відображення водної поверхні.

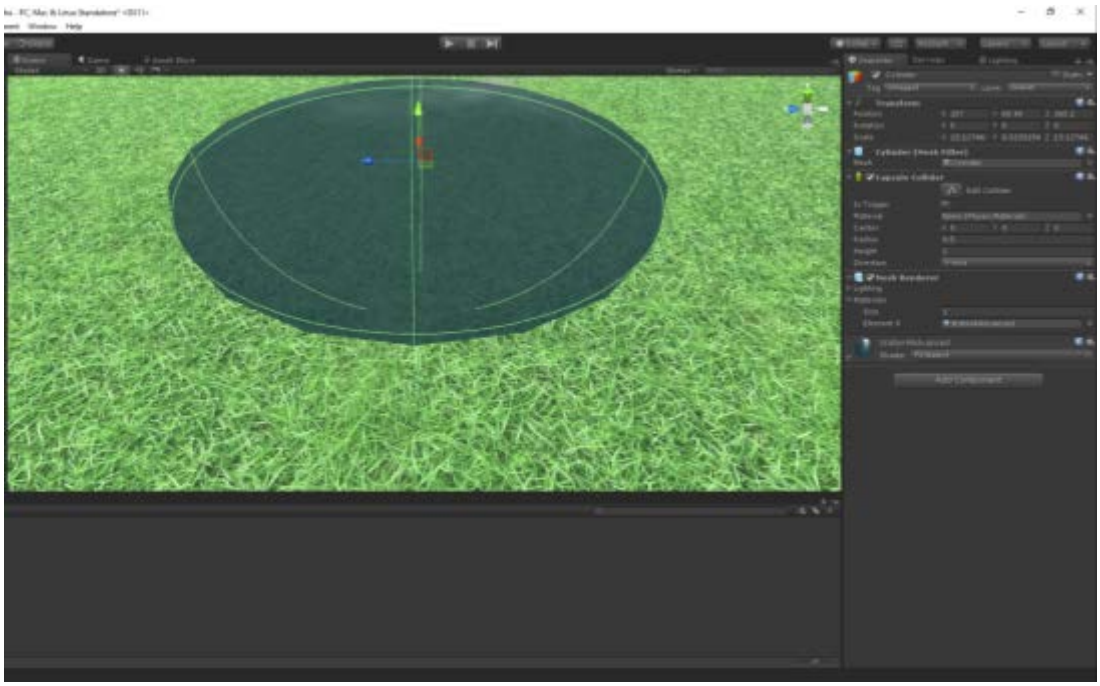


Рисунок 3.18 – Демонстрація води

Створенню рослинності потрібно приділяти досить багато уваги. Обов’язково включити методи оптимізації для відображення моделі користувачеві. Дерево має 4 види відображення для оптимального навантаження на пристрій відображення.

Рисунок 3.19 – Демонстрація дерева

Дерево повинне мати декілька видів для створення відчуття різноманітності рослинності.

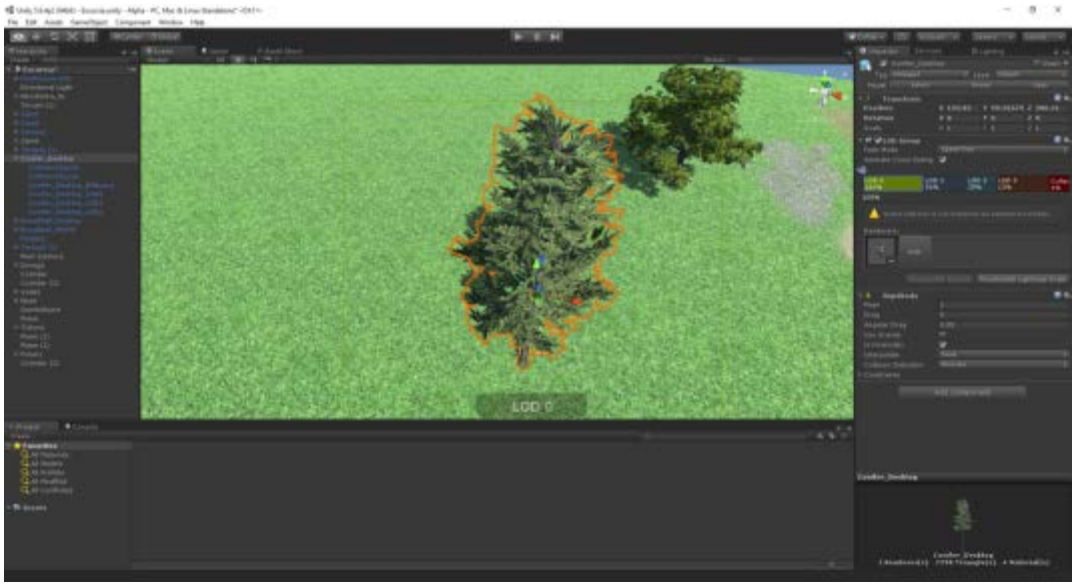


Рисунок 3.20 – Демонстрація високого дерева

Достатньо мати всього 3 види дерева різних розмірів та відтінку рослинності.

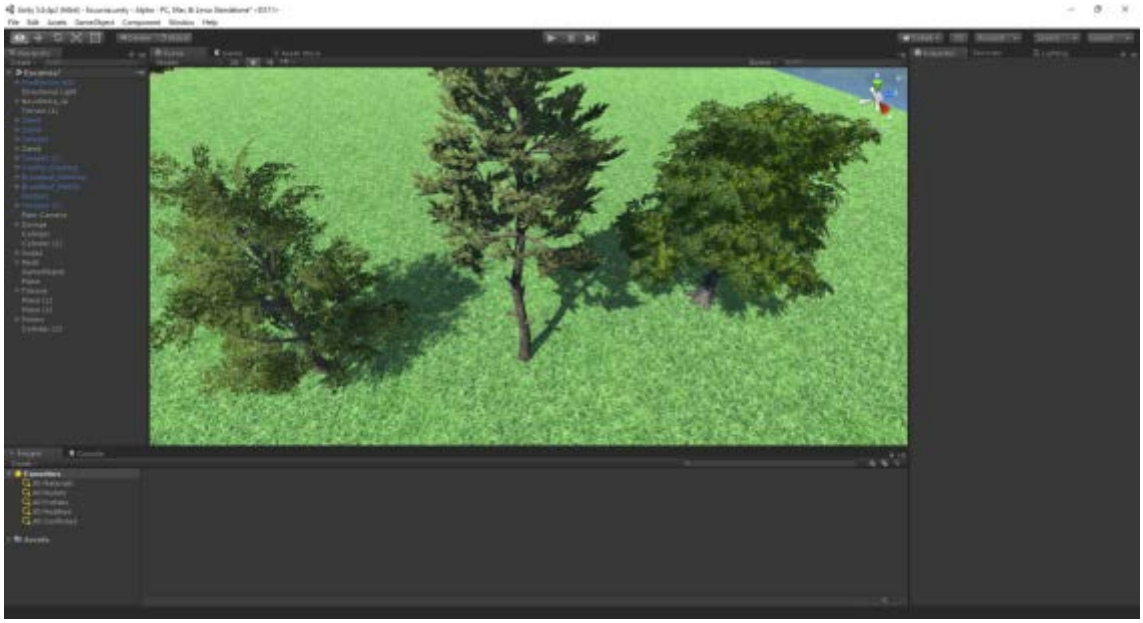


Рисунок 3.21 – Демонстрація всіх видів дерев

При розміщенні об’єктів можливі неточності. Допустимо зменшити деталізацію об’єктів для плавності картинки при відображенні.

Рисунок 3.22 – Демонстрація зменшення деталізації об’єкта

Потрібно звернути увагу на моделювання ландшафту. Підвищення рівня грунту повинне мати цільове направлення, а водойма – мати плавний перехід висоти.

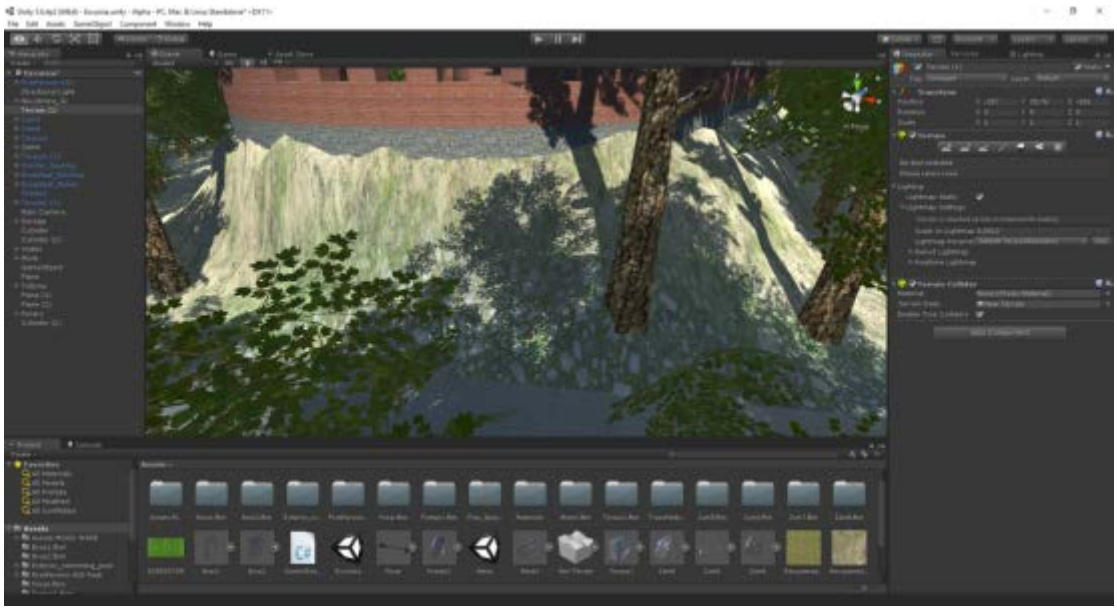


Рисунок 3.23 – Демонстрація підвищення рівня грунту

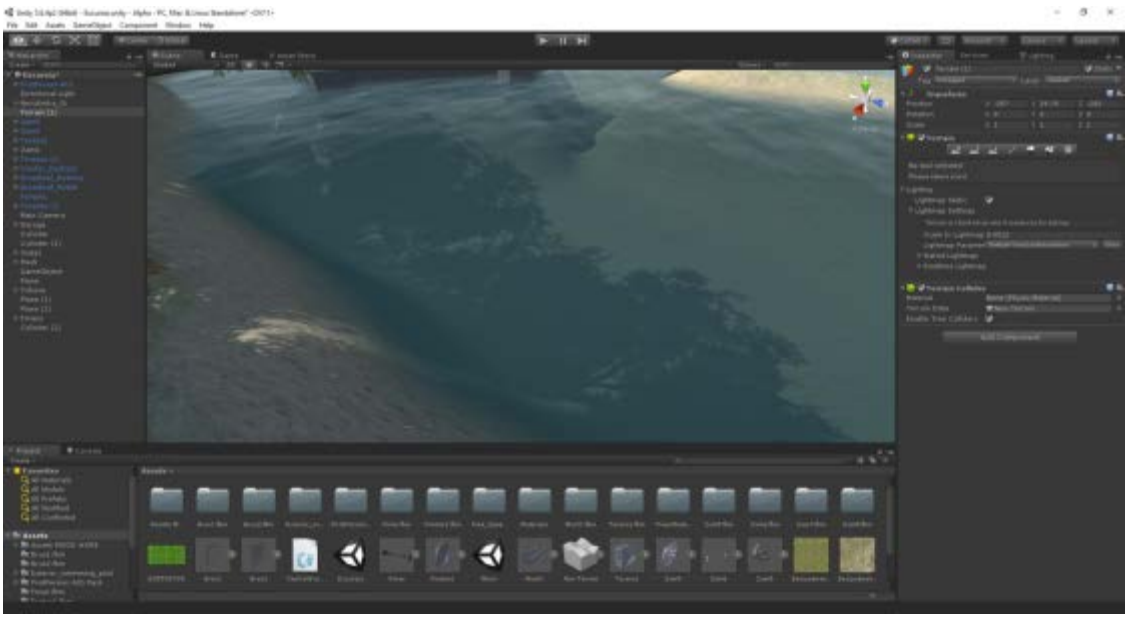


Рисунок 3.24 – Демонстрація плавної зміни висоти ландшафту

Потрібно знизити деталізацію текстур або взагалі замінити їх для плавної роботи програмного додатку.

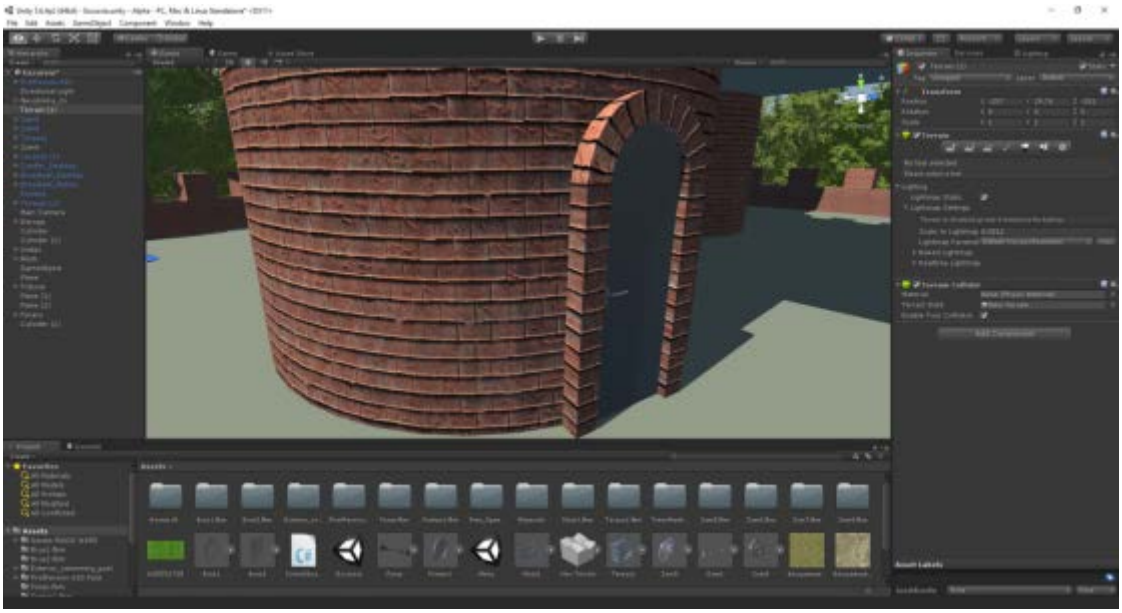


Рисунок 3.25 – Демонстрація зниження якості текстур

Обов’язково потрібно домогтися до плавного переходу між різними типами ландшафтів.

Рисунок 3.26 – Демонстрація переходу між типами ландшафту

Розмістити невидимі стіни для перешкоджання користувачеві потрапити за кінець карти. Периметр закрити густою рослинністю, але

низької якості для запобігання зниження продуктивності програмного продукту.

Рисунок 3.27 – Демонстрація невидимого бар’єра

Освітлення достатньо використати натуральне, додаткове освітлення окремих зон не потребується.

Рисунок 3.28 – Демонстрація джерела світла

Управління передбачено від першого лиця, тому аватар у користувача має лише умовне зображення.

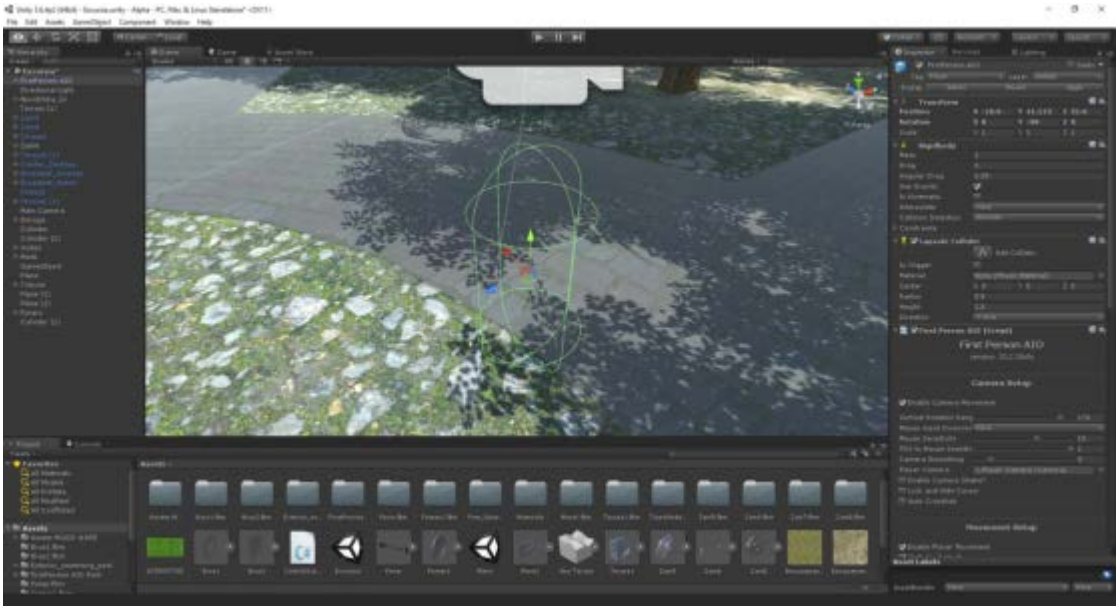


Рисунок 3.29 – Демонстрація аватара користувача

Небо не може бути повністю чистим. Потрібно додати атмосфери шляхом додавання SkyBox.

Рисунок 3.30 – Демонстрація неба

Створення іншої сцени для головного меню.

Рисунок 3.31 – Демонстрація шаблону панелі меню

Кнопки мають анімацію зміни кольору та дуже прості у функціоналі.

Рисунок 3.32 – Демонстрація головного меню

Потрібно створити діалог з користувачем.

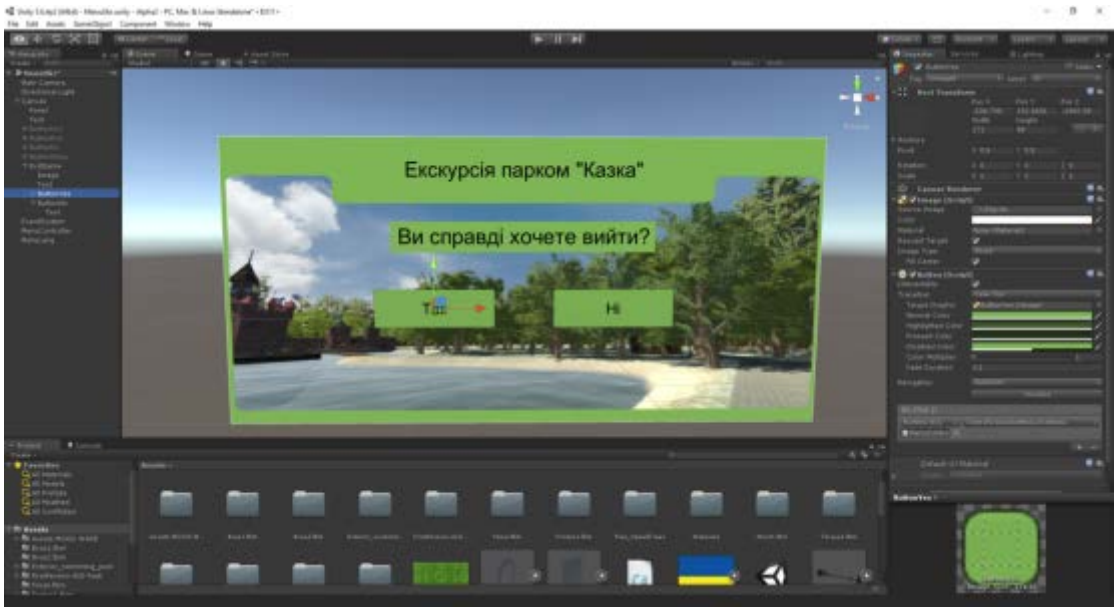


Рисунок 3.33 – Демонстрація підтвердження виходу

Реалізація локалізації.

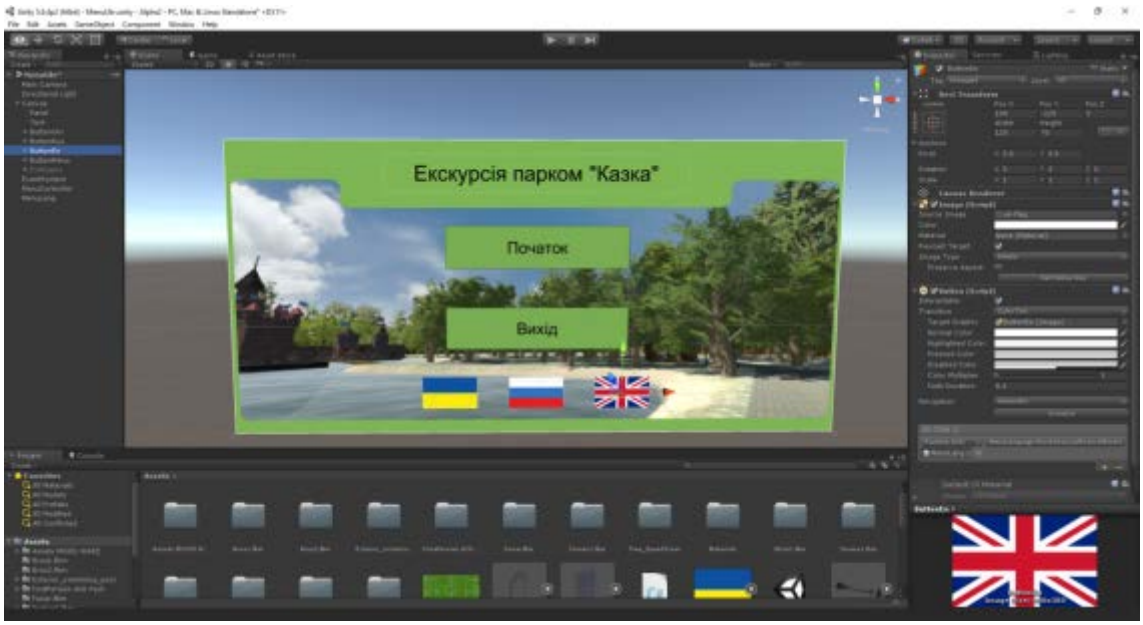


Рисунок 3.34 – Демонстрація локалізації

Рендер сцен та створення файлу програмного продукту.

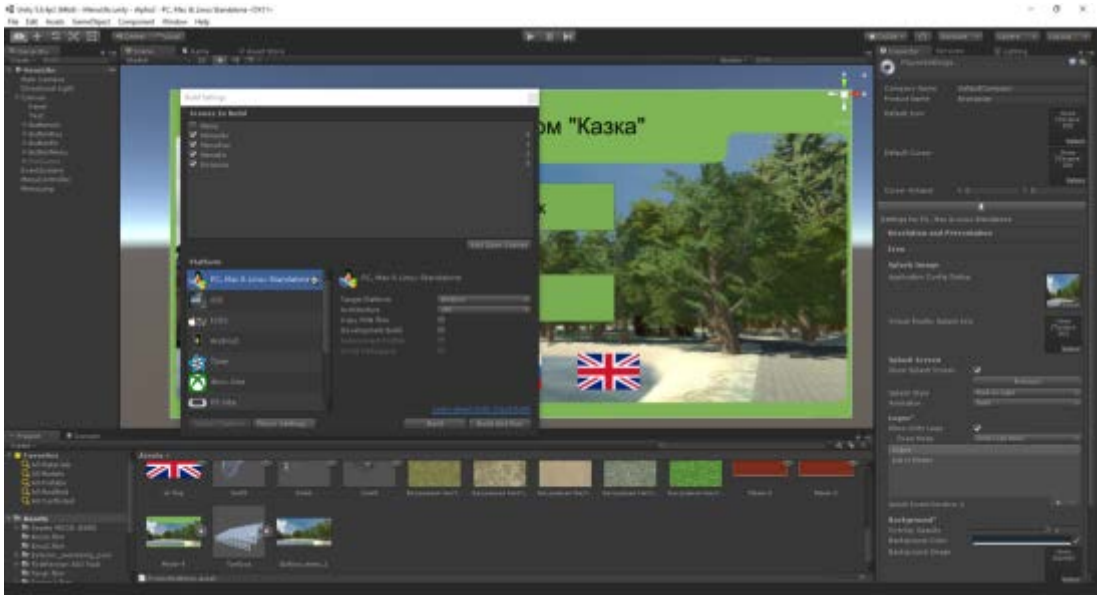


Рисунок 3.35 – Демонстрація меню рендеру сцен

3.3 Використання програмного додатку

Після запуску програмного додатку отримується вікно для налаштування розширення екрану, якості графіки та вибір дисплею. В наступній вкладці присутні налаштування управління.

Налаштування за замовчуванням достатньо змінити лише раз і при кожному наступному запуску, налаштування будуть збережені.

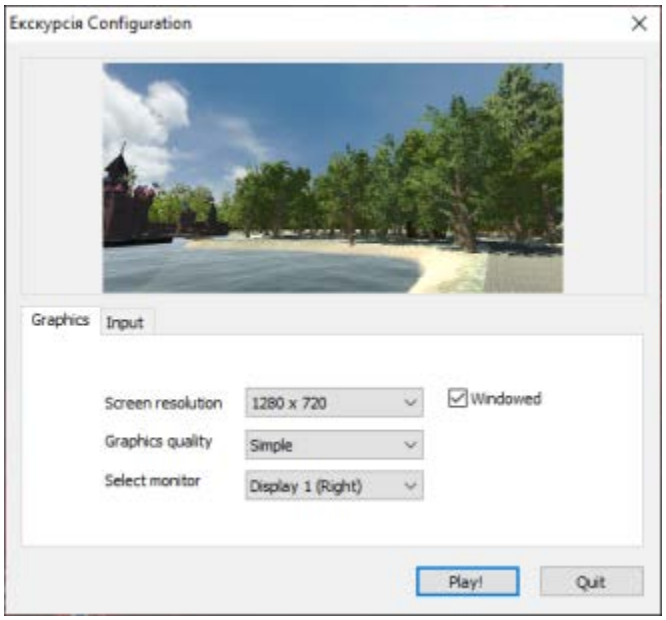


Рисунок 3.36 – Налаштування якості картинки

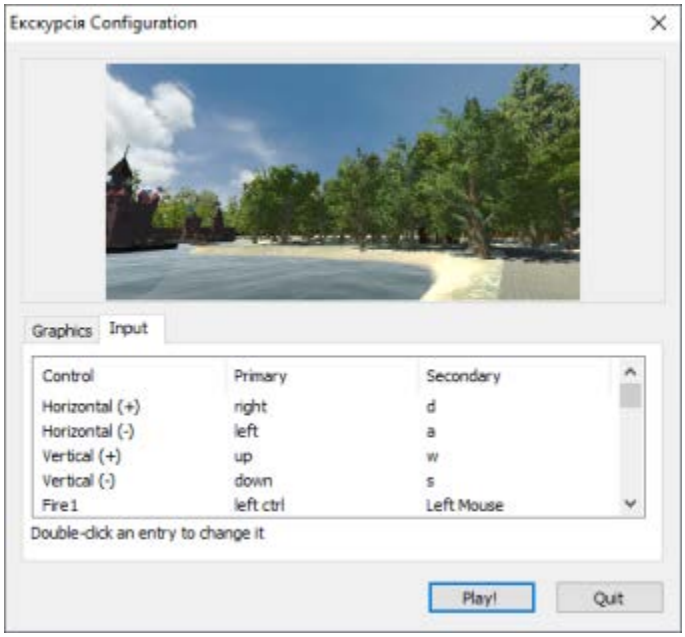


Рисунок 3.37 – Налаштування управління

Після підготовчого налаштування зустрічає дуже простий та дружелюбний інтерфейс з зрозумілим функціоналом.

Рисунок 3.38 – Головне меню

При натисканні на «Вихід» отримується повідомлення, де потрібно підтвердити вихід.

Рисунок 3.39 – Підтвердження виходу

З’являється аватар одразу в центрі локації. Одразу наявні ознаки оптимізації, де тіні об’єктів повільно зникають з відстанню, а самі тіні на об’єктах мають спрощене відображення.

Рисунок 3.40 – Початок роботи додатку

Переміщення імітує швидкість дорослої людини та має легке погойдування, яке не викликає відрази та не набридає гравцю.

Переміщення по локації може бути швидшим, але обмежену кількість часу. Індикатор знизу повідомляє про закінчення властивості «ривку», «енергія» відновлюється під час звичайної ходьби.

Рисунок 3.41 – Механіка пересування

Не дивлячись на високу ступінь деталізації, більшість об’єктів мають добре деталізовану структуру та легко упізнаються гравцем.

Рисунок 3.42 – Деталізування окремих декорацій

Вода має хорошу деталізацію та плавну і спокійну анімацію. Пересування по воді неможливе, що відповідає реалістичній поведінці людини. Поверхня водойму правильно відбиває світло та відображає тіні.

Рисунок 3.43 – Поверхня води

Правильно підібрані текстури ландшафту мають плавний перехід та візуально досить добре відображені.

Рисунок 3.44 – Перехід типів ландшафту

При віддалянні від об’єктів, змінюється відображення їх. Це зроблено в цілях оптимізації локації та послабити натиск на обчислювані здібності машини.

Рисунок 3.45 – Спрощення візуалізації

Кордон карти має неявне відображення густою рослинністю, яка має погане відображення завжди та невидимі стіни, що не дає гравцю не лише покинути локацію, а й відбиває бажання досліджувати границі карти. Також густа рослинність перешкоджає відображенню гравцеві кінець карти та не навантажує пристрій важкою деталізацією.

Рисунок 3.46 – Кордон локації

Небо має статичні хмарові утворення, що надає атмосферності локації.

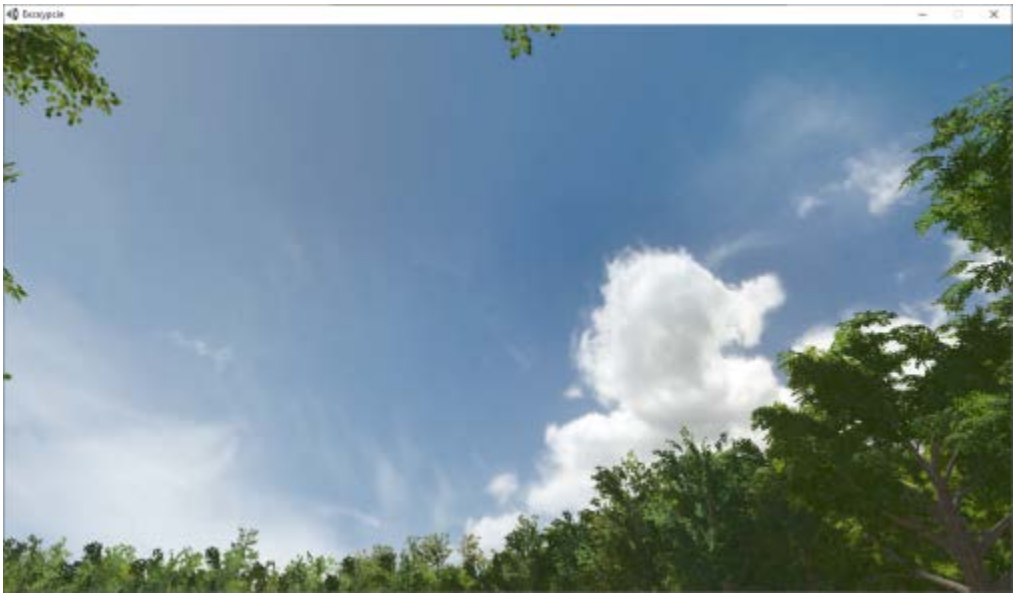


Рисунок 3.47 – Небесні утворення

Освітлення використовується натуральне, та має відтінок сонця. В локації для гравця відображається спрощений вид сонця в цілях оптимізації.

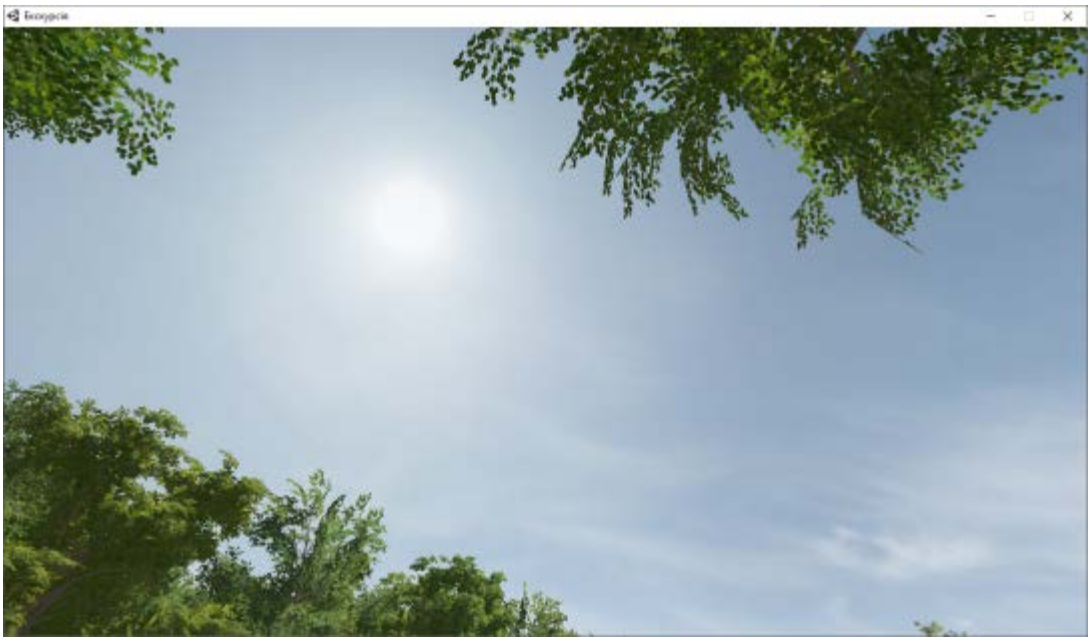


Рисунок 3.48 – Спрощене сонце

ВИСНОВОК

За час виконання бакалаврської роботи була виконана мета створити віртуальну екскурсію, яка надає можливість вільно пересуватися територією парку «Казка» м.Суми.

Проаналізував предметну область та переглянув аналогічні програмні забезпечення, що надали інструменти для створення додатку було обрано ігровий рушій Unity, з необхідною функціональністю та можливостями. Серед аналогів, він єдиний мав модульну систему та підтримку більшості популярних форматів.

На основі аналізу ігрових рушіїв та предметної області було сформовано вимоги до майбутнього додатку - вимоги до наповнення та реалізації віртуальної екскурсії (виконано моделювання контекстної діаграми та діаграми використання, проведено планування робіт, розроблено календарний план, а також проаналізовано можливі ризики).

Розроблено 3d моделі замків, ландшафтів у програмному середовищі 3dMax. В Unity реалізовано зборку та налаштування моделей, освітлення, анімації актора.

Виконано оптимізацію програмного забезпечення. Користувачі мають змогу бачити деталізовану картинку при низьких системних вимогах, об’єкти мають довольно чіткий деталізований контур впізнаються відвідувачами парку.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Hocking J.: Unity in Action / Joseph Hocking – MPC, 2015. – 326 с.
2. Ульянов А.Ю., Рыжкова Н.Г.: Метод трасировки лучей как основная технология фотореалистичного рендеринга / УДК, 2015. – 1128 с.
3. Nvidia [Електронний ресурс] – Режим доступу до ресурсу: https://www.nvidia.com.ua/object/blog-nvidia-ray-tracing-death-ray-ru.html.
4. Макарский Д.Д., Никоноров А.В.: История компьютерной эры / Никоноров А.В. – Эксмо, 2016. – 256 с.
5. Алекс Лайтман, Енди Ларк: Эпоха дополненой реальности / Олимп-Бизнес, 2016. – 528 с.
6. Спрол А.: Думай как програмист. Креативный подход / Спрол Антон – Бомбора, 2018.-272 с.
7. Бычков А.: Дизайн и фриланс. Начало / Бычков А. – М.:АСТ, 2017.-208 с.
8. Васильев А.Н.: Программирование на C++ в примерах и задачах / А.Н. Васильев – Эксмо-пресс, 2017.-368 с.
9. Ashley Godbold, Simon Jackson: Mastering Unity 2D Development / Packt Publishing, 2016. – 506 с.
10. Michelle Menard: Game Development with Unity / Course Techonology, 2011. – 478 с.
11. John P. Doran: Unity 5.x Game Development Blueprints / Packt Publishing Ltd, 2016. – 428 с.
12. Will Goldstone: Unity Game Development Essentials / Packt Publishing Ltd, 2009. – 298 с.
13. Васильев А.Н.: Програмирование на C# для начинающих / Алексей Николаевич Васильев – Бомбора, 2018.-592 с.

14. Шаффлботэм Р.: Photoshop CC для начинающих /Р. Шаффлботэм – Эксмо-пресс, 2017.-272 с.

15. IDEF0 [Електронний ресурс] – Режим доступу до ресурсу: https://uk.wikipedia.org/wiki/IDEF0.

16. Гриффитс И.: Программирование на C# 5.0 / И. Гриффитс – Эксмо, 2014.- 1136 с.

17. Васильев А.Н.: Программирование на C в примерах и задачах / А.Н. Васильев – Эксмо-пресс, 2017.-560 с.

18. МакГрат М.: Программирование на Visual Basic / М. МакГрат – Эксмо-пресс, 2017.-192 с.

19. МакГрат М.: PHP7 для начинающих / М. МакГрат – Эксмо-пресс, 2017.-256 с.

20. Васильев А.Н.: JavaScript в примерах и задачах / А.Н. Васильев – Эксмо-пресс, 2017.-720 с.

21. Грофф Д. SQL: полное руководство / Д. Грофф, П. Вайнберг. – Київ: BHV, 2005. – 608 с.

ДОДАТОК А

ТЕХНІЧНЕ ЗАВДАННЯ

На розробку віртуальної екскурсії

1 Призначення й мета створення додатку

1.1 Призначення додатку

Додаток повинен представляти віртуальну екскурсію парком «Казка» м. Суми.

1.2 Мета створення інформаційної системи –

Ознайомлення с архітектурою парку.

1.3 Цільова аудиторія

Цільова аудиторія додатку може ділитися на такі групи:

1. Діти.
2. Студенти.
3. Туристи.
4. Інші зацікавлені відвідувачі.

2 Вимоги до інформаційної системи

2.1 Вимоги до інформаційної системи в цілому

2.1.1 Вимоги до структури й функціонування інформаційної системи

Додаток повинен бути реалізований у вигляді гри, без обов’язкового доступу до мережі Інтернет. Додаток повинен бути простим у використанні та мати достатню кількість функцій управління.

2.1.2 Вимоги до персоналу

Для підтримки додатку та покращення його роботи потребуються базові знання програмування мовою C# та технологією 3D моделювання в Autodesk 3ds Max. Обов’язково мати загальні навички роботи з персональним комп'ютером.

2.2 Вимоги до збереження інформації

У системі додатку не передбачено зберігання даних користувача.

2.2.1 Вимоги до розмежування доступу

Користувачів додатку можна розділити на 2 групи відповідно до можливостей:

1.Клієнт

2.Адміністратор

Клієнти мають загальні функції управління.

Доступ до адміністративної частини – закритий.

Адміністратор може редагувати матеріали та розміщення об’єктів. Має доступ до редагування інтерфейсу та програмного коду. Доступ до адміністративної частини для клієнта не передбачено.

2.3 Вимоги до функцій, виконуваних додатком

2.3.1 Основні вимоги

Структура додатку

Додаток повинен складатися з наступних розділів:

– Головне меню – надає вибір для налаштування додатку, налаштування та вихід з додатка.

– Початок екскурсії – головна частина контенту, дає можливість пересуватися парком.

– Зміна мови – дає можливість змінити мову програми.

Навігація

Інтерфейс користувача повинен бути простим та мати логічне розміщення, мати швидкий перехід між пунктами меню. Управління має бути інтуїтивно зрозумілим користувачеві. Кнопки інтерфейсу повинні бути підписані та мати чіткий та розбірливий шрифт.

Управління пересуванням парком повинне бути зрозумілим та легким у використанні. Інтерфейс користувача під час екскурсії не повинен навантажувати клієнта надлишковою інформацією та відволікати від екскурсії.

Система навігації (карта додатку)

Взаємозв'язок між розділами й підрозділами додатку (карта додатку) представлено на рисунку А.1.

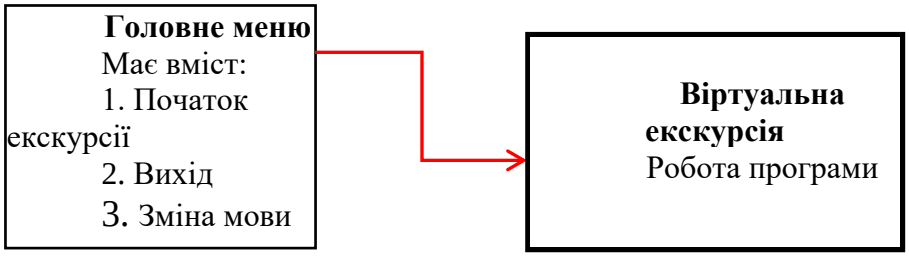


Рисунок А.1– *Карта сайту*

Вимоги до функціональних можливостей

Система управління вмістом проекту (адміністративна частина управління) повинна давати можливість додавати, редагувати та видаляти об’єкти з подальшим редагуванням текстур та анімацією для них. Повинна бути доступна можливість редагування програмного коду як для всього проекту, так і для окремих об’єктів.

Функціональні можливості розділів

На головній сторінці будуть представлені наступні елементи:

- Початок екскурсії;
- Можливість налаштування додатку;
- Можливість завершити роботу додатку;

Загальні вимоги

Стиль додатку можна описати класичним та стандартним. У якості фону рекомендується використовувати робочі скріншоти додатку.

Оформлення не повинне мати приємну кольорову тематику. Інтерфейс користувача повинен бути ненав’язливий, але при цьому має бути зручним з логічним розміщенням елементів.

Розташування елементів в головному меню додатку схематично показано на рис. А.2.

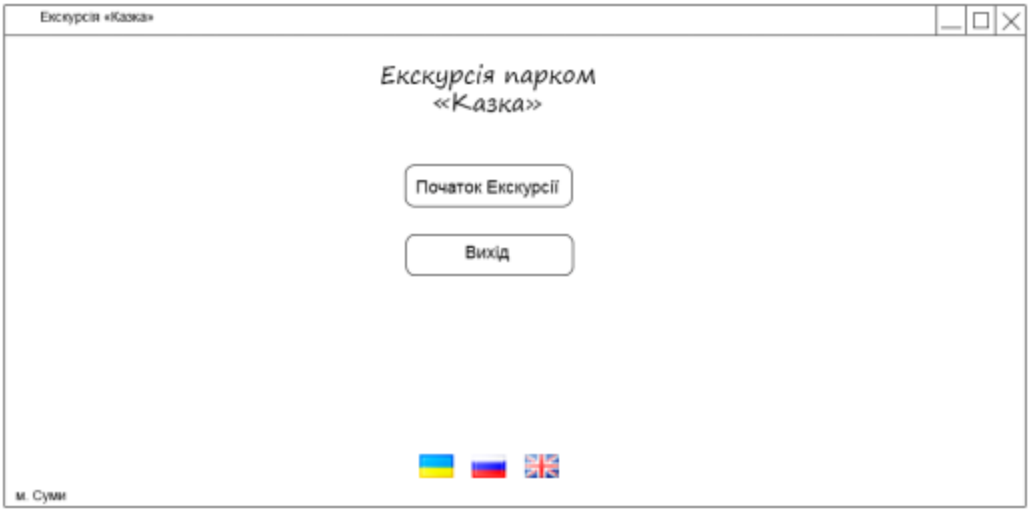


Рисунок А.2 – *Типова сторінка*

Типові навігаційні й інформаційні елементи

– Шапка додатку

– Головне меню

– Меню зміни мови

Шапка додатку

Шапка додатку повинна містити назву додатку.

Головне меню

Головне меню повинне розташовуватися у центральній частині вікна (під шапкою) і містити кнопки на всі розділи, які використовуються для управління додатком. Стильове оформлення повинні мати всі кнопки для всього додатку.

Зміна мови

1 Зміна мови повинне знаходитися під головним меню та мати зрозуміле і логічне оформлення.

2.4 Вимоги до видів забезпечення

Вимоги до інформаційного забезпечення

Реалізація додатку відбувається з використанням:

- 3ds Max 2018
- Adobe Photoshop CC 2019
- Unity 5.6.4p2 (64-bit)
- V-RAY for 3ds Max 2018
- Visual Studio 2018

Вимоги до лінгвістичного забезпечення

Додаток має бути реалізований українською та англійською мовами.

Вимоги до програмного забезпечення

Програмне забезпечення клієнтської частини має наступні мінімальні вимоги:

– Операційна система: Windows 7 SP1 і вище;

– Графічний API: DX10, DX11, DX12;

Вимоги до апаратного забезпечення

Апаратне забезпечення повинне задовольняти наступні вимоги:

– CPU: архітектура x86, x64 з підтримкою набору команд SSE2;

– Не менше 2 ГБ вільного місця на диску;

– Мінімум 500 МБ оперативної пам’яті

3 Склад і зміст робіт зі створення додатку

Докладний опис етапів роботи зі створення додатку наведено в табл.1.

Таблиця 1 – Етапи створення додатку

№	Склад і зміст робіт	Строк розробки (у робочих днях)
1	Розробка ескізу: Проектування інтерфейсу та головного меню	5 день
2	Моделювання об’єктів: Створення моделей графічному редакторі	20 днів
3	Накладання текстур: Створення та накладання текстур та матеріалів на об’єкти	18 днів
4	Ландшафтний дизайн: Створення ландшафту та накладання текстури на нього	7 днів
5	Моделювання сцени: Розміщення об’єктів в сцені	5 дні
6	Головне меню: Створення головного меню та локалізації. Створення елементів інтерфейсу	1 дні
7	Написання скриптів: Створення скриптів (програмного коду) та налаштування управління	15 дні
8	Завершення роботи: Тестування всього функціоналу, виправлення косметичного характеру	5 день

Продовження таблиці 1 – Етапи створення сайту

	Загальна тривалість робіт (з урахуванням резервного строку на налагодження й виправлення помилок) і строк закінчення проекту	76

4 Вимоги до складу й змісту робіт із введення сайту в експлуатацію

Для правильного функціонування в умовах сьогодення та відповідати всім вимогам ТЗ, а також задовольнити потреби користувача, потрібно зробити цілий комплекс робіт.1

Для правильної роботи додатку у користувача, потрібно щоб РС користувача відповідав мінімальним вимогам зазначених у ТЗ. Додаток переноситься на РС користувача за допомогою файлового обміну.

ДОДАТОК Б

Б.1 Деталізація мети проекту методом SMART

Сутність деталізації мети проекту за допомогою SMART-методу випливає з розшифровки термінів, які формують його назву: конкретна (Specific), вимірювана (Measurable), досяжна (Achievable), реалістична (Relevant), обмежена у часі (Time-framed).

S – конкретність, специфічність. Вимагає щоб сформульована мета давала чітке якісне уявлення про специфічні унікальні та інноваційні властивості майбутнього продукту проекту порівняно з іншими його альтернативами.

M – вимірюваність. Передбачає показників вартості які вимірюються. При відсутності фізичних способів та інструментарію виміру використовуються експерти – як інструмент для виміру.

A – узгодженість. Встановлює, що мета повинна випливати з реальних проблем, місії, стратегічних планів, планів розвитку, а також узгоджуватись з інтересами зацікавлених сторін проекту.

R – реалістичність, релевантність. Показує, що мета є такою, яку можливо досягти з урахуванням реально доступних ресурсних можливостей та обмежень (людських, фінансових тощо).

Поставлена мета є досяжною, адже вона формулювалася на основі реально доступних ресурсних можливостей та проведеного аналізу вже наявних досліджень експертів у даній сфері.

T – обмеженість в часі. Зумовлює необхідність «прив’язки» мети до певних обґрунтованих термінів її досягнення (або початку та тривалості дій по її досягненню).

Таблиця Б.1 – Деталізація мети методом SMART

Specific (конкретна)	Створити віртуальну екскурсію.
Measurable (вимірювана)	Використовуючи мінімум ресурсів розробити якісний програмний продукт.
Achievable (досяжна)	Поставлена мета випливає у результаті актуальних проблем.
Relevant (реалістична)	У наявності є всі необхідні технічні та програмні засоби. Розробники достатньо кваліфіковані для виконання поставлених задач.
Time-framed (обмежена у часі)	Ціль має часове обмеження. Терміни досягнення мети проекту визначаються за домовленість замовником та виконавцем.

Після проведення аналізу методом SMART можна визначити кінцеву мету: вчасно створити якісний програмний продукт з використанням мінімальних витрат.

Б.2 Планування змісту структури робіт ІТ-проекту (WBS)

WBS – це графічне подання згрупованих елементів проекту у вигляді пакета робіт, які ієрархічно пов’язані з продуктом проекту. На верхньому першому рівні WBS фіксується продукт проекту. Він повинен відповідати продукту проекту. Наступний ІІ рівень відповідає діям або основним заходам для досягнення продукту проекту. Потім триває розбивка цих дій доти, поки не відбувається виконання дій елементарних робіт.

Ієрархічна структура робіт являє собою, по суті, перелік завдань проекту. Вона може бути представлена в графічному вигляді або у вигляді опису, що відображає вкладення робіт. Ієрархічна структура робіт організовує і визначає весь зміст проекту. Роботи, не включені is WBS, не є роботами проекту.

Виконаємо побудову WBS структури, у якій зазначимо всі виконувані роботи в залежності від головних етапів:

1. Формування технічного завдання - розробка технічного завдання, що встановлює основне призначення, показники якості, техніко - економічні та спеціальні вимоги до розроблюваного інструментального засобу. Формування технічного завдання включає в себе підпункти:

- визначення предметної області;
- визначення мови написання;
- визначення цільової аудиторії;
- визначення вимог дизайну програмного продукту;
- визначення вимог засобів перегляду та вимог до системи управління контентом.

Планування проекту включає в себе розробку OBS структури, матриці відповідальності, календарного плану, а саме діаграми Ганта, управління ресурсами та ризиками.

Реалізація матиме 4 етапи:

1. Підготовка – збір потрібної інформації, формування можливостей та цілей проекту.
2. Проектування – розробка технічної моделі, створення ескізів основних об’єктів та ландшафту.
3. Розробка – створення програмного коду та графічної складової програми, тестування та оптимізація.
4. Реліз – випробування ПП, аналіз проблем та виправлення помилок.

І останній етап створення проекту завершення має на увазі здачу проекту в експлуатацію і закриття проекту.

Діаграму WBS наведено рис. Б.1.

Б.3 Організаційна структура проекту (OBS)

OBS-структура проекту – організаційна структура виконавців (організацій) проекту. Визначається за переліком пакетів робіт нижнього рівня кожної гілки WBS-структури. Представляється відповідальними (відповідальні – це не обов'язково керівники організацій (відділів), а ті люди які безпосередньо організують виконання робіт) за виконання пакетів робіт.

Організаційна структура представляє собою графічне відображення учасників проекту та їх відповідальних осіб, які задіяні в реалізації проекту. На верхньому рівні OBS розташована команда проекту. На наступному рівні фіксуються виконавці: організації, відділи тощо. Потім, рівнем нижче, для кожного виконавця вказують прізвища конкретних осіб, які будуть відповідати за виконання елементарних робіт WBS. Потрібно пам'ятати, що відповідальні – це не обов'язково керівники, а ті співробітники, які безпосередньо організовують і відповідають у виконавця за виконання елементарної роботи, зазначеної у WBS. Для них ця елементарна робота також є проектом (у порівнянні з загальним проектом). Для себе вони також можуть побудувати WBS- структуру й застосовувати інші інструменти планування. Саме на цьому рівні закладається певна якість майбутнього продукту проекту. Діаграма OBS представлена на рис. Б.2.

Б.4 Побудова календарного графіка виконання ІТ-проекту

Для того щоб мати реальне уявлення про тривалість виконання робіт з урахуванням обмеженості у використанні ресурсів, на підставі часткової мережевої моделі будують календарний графік робіт.

Діаграма Ганта – горизонтальна лінійна діаграма, на якій задачі проекту представляються протяжними в часі відрізками, що характеризуються датами початку та закінчення, затримками і, можливо, іншими тимчасовими параметрами.

Кожен відрізок відповідає окремому завданню або підзадачі. Завдання і підзадачі, складові плану, розміщуються по вертикалі. Початок, кінець і довжина відрізка на шкалі часу відповідають початку, кінцю і тривалості завдання. На деяких діаграмах Ганта також показується залежність між завданнями.

На наступному рисунку представлено діаграму Ганта розроблюваного проекту. На рис. Б.3 представлено побудовану діаграму Ганта.

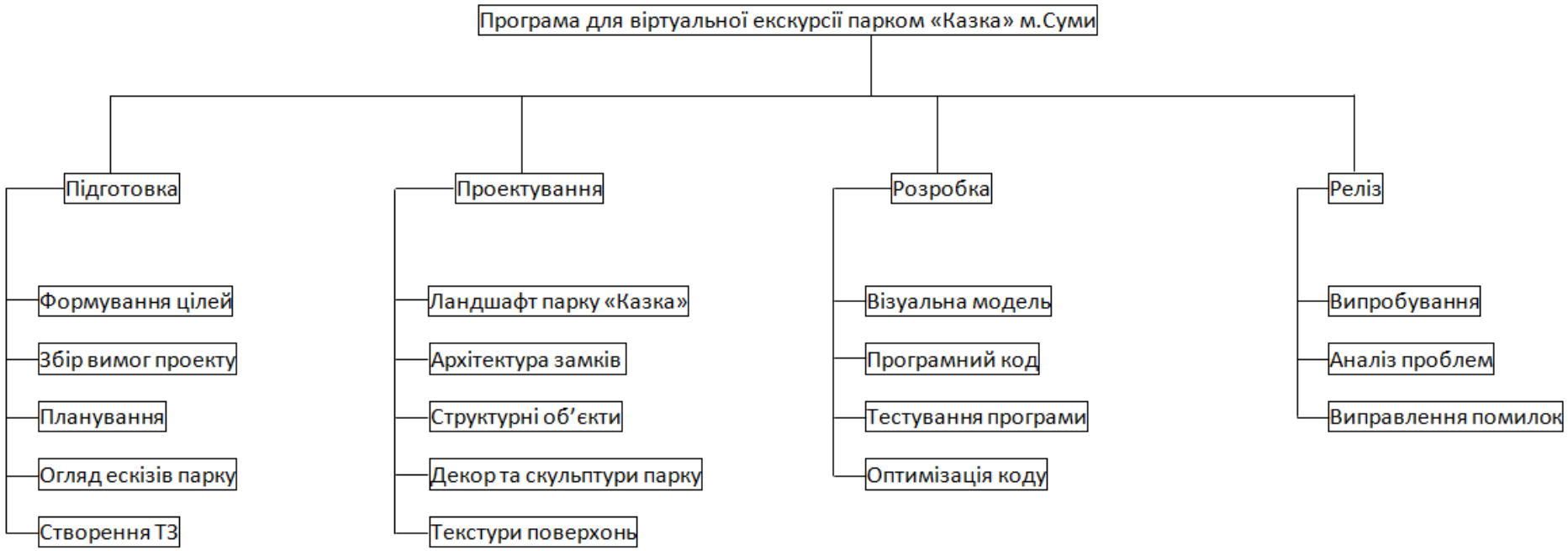


Рисунок Б.1 – Структуруа WBS

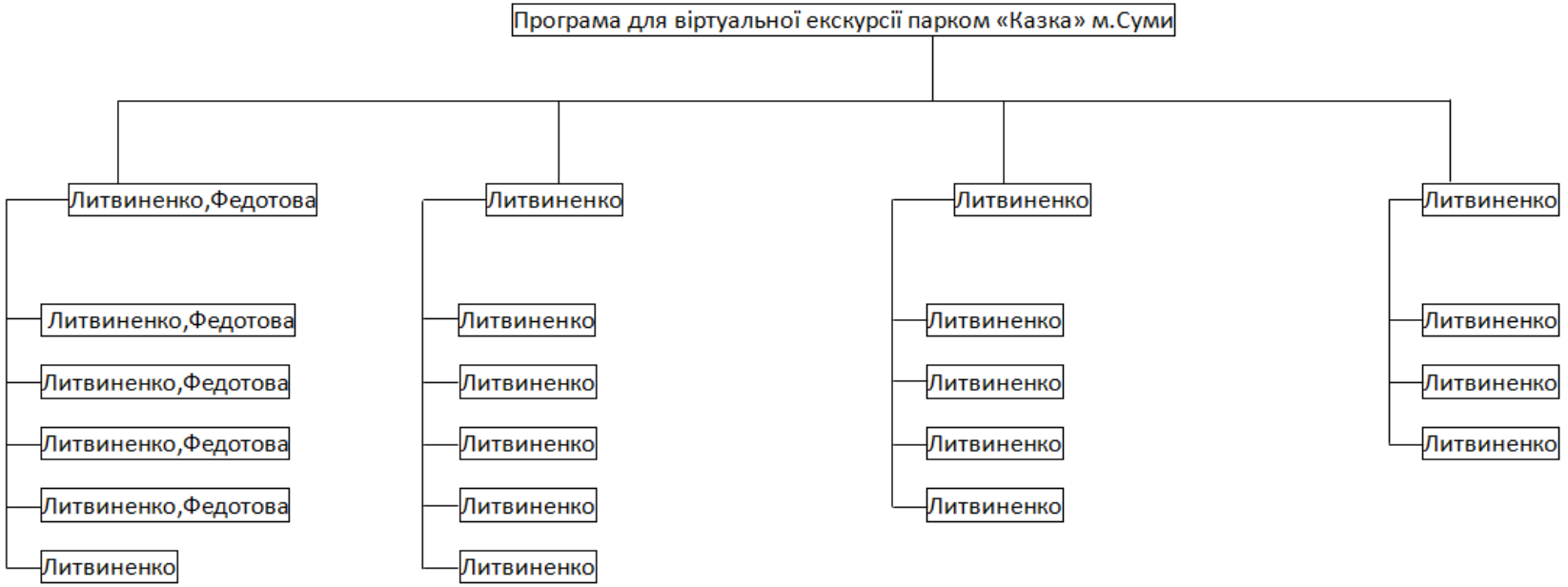


Рисунок Б.2 – Структура OBS

Task	Start Date	Days to Complete
Збір інформації	12 сен	30
Проектування ландшафту	4 окт	15
Проектування архітектури	18 окт	20
Проектування декору	1 ноя	20
Розробка текстур	15 ноя	18
Розробка віртуальної моделі	29 ноя	8
Розробка коду	6 дек	25
Тестування	27 дек	5
Оптимізація	30 дек	5
Випробування	3 янв	3

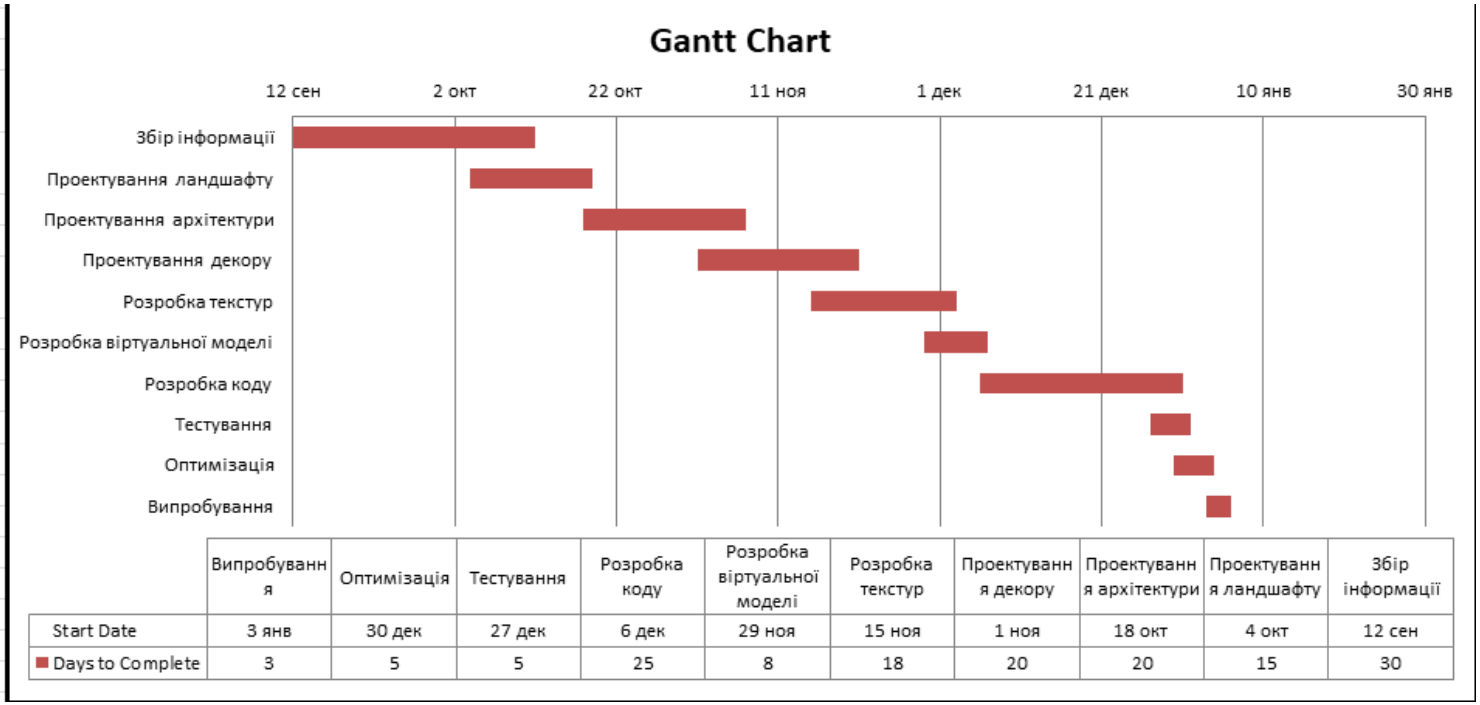


Рисунок Б.3 – Діаграма Ганта

Б.5 Управління ризиками

Ризик – це ймовірнісна подія, яка у випадку своєї появи негативно або позитивно впливає на проект.

Управління ризиком – це процес реагування на події та зміни ризиків у процесі виконання проекту. При цьому важливим є проведення моніторингу ризиків.

Процес управління ризиками включає в себе такі пункти:

1) Ідентифікація ризиків (виявлення ризиків)
2) Оцінювання ризиків (оцінка ймовірності та впливу)
3) Заходи реагування на ризики
4) Моніторинг ризиків

Ідентифікація ризиків – це виявлення ризиків, здатних вплинути на проект, і документальне оформлення їх характеристик. Це ітеративний процес, який періодично повторюється на всьому протязі проекту, оскільки в рамках його життєвого циклу можуть виявлятися нові ризики. Найбільш розповсюдженою характеристикою ризику є загроза або небезпека виникнення невдач у тій чи іншій діяльності, небезпека виникнення несприятливих наслідків, змін зовнішнього середовища, які можуть викликати втрати ресурсів, збитки, а також небезпеку, від якої слід застрахуватися.

Планування реагування на ризики – це процес розробки шляхів і визначення дій із збільшення можливостей і зниження погроз для цілей проекту. Даний процес зачинається після проведення якісного і кількісного аналізу ризиків.В процесі аналізу для визначення числових значень ймовірності виникнення ступеня впливу, зазвичай застосовується метод експертних оцінок. На їх основі визначається ранг ризику, як потенційний вплив ризику на проект, який оцінюється як добуток ймовірності виникнення та ступеню впливу.

Матриця ризиків

Ймовірність виникнення:

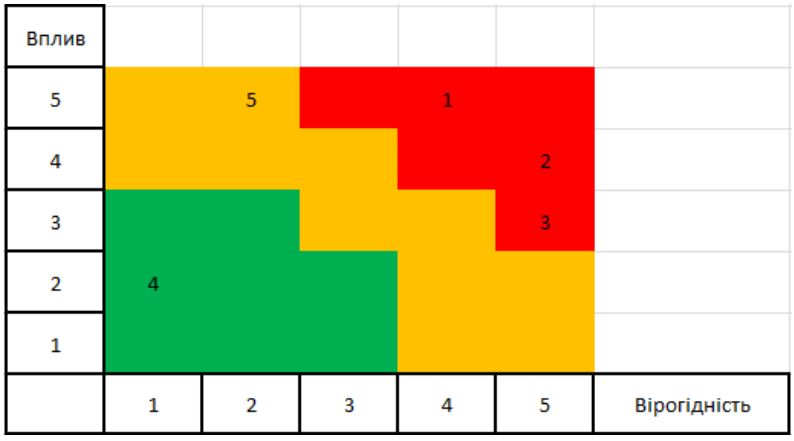
1 Слабкоймовірно
2 Малоймовірно
3 Ймовірно
4 Вельми ймовірно
5 Майже можливо

Величина втрат:
1 Мінімальна
2 Низька
3 Середня
4 Висока
5 Максимальна

Таблиця Б.4 – Ймовірність втрат

Ідентифікатор ризику	Ризики	Вірогідність	Вплив
1	Нестабільність програмного забезпечення	4	5
2	Помилка в розрахунках	5	4
3	Помилка у відображенні	5	3
4	Затримка фінансування	1	2
5	Додаткові вимоги	2	5

Таблиця Б.5 – Ймовірність втрат



Класифікація за ступенем впливу:

- ігноровані (1≤R≤4);
- незначні (5≤R≤8);
- помірні (9≤R≤11);
- вагомі (12≤R≤19);
- критичні (20≤R≤25).

Класифікація за рівнем ризику:

- прийнятні ризики;
- виправданні ризики;
- недопустимі ризики;

Таблиця Б.6 – Класифікація за ступенем впливу та за рівнем ризику

Ризик		**Ступінь впливу**	**Рівень ризику**
Нестабільність програмного забезпечення	1	20	Критичні ризики
Помилка в розрахунках	2	20	Критичні ризики
Помилка у відображенні	3	15	Вагомі ризики
Затримка фінансування	4	2	Ігноровані ризики
Додаткові вимоги	5	10	Помірні ризики

План по усуненню ризиків:

– Вибір потужного обладнання для виконання проекту
– Зіставлення структурованого плану роботи
– Періодичні поставки тестових версій ПП замовнику
– Безперервна взаємодія з замовником
– Враховувати досвід проектів-аналогів.
– Резервувати час на випадок помилок планування та виникнення непередбачених обставин.
– Ретельний вибір інструментів виконання проекту.

ДОДАТОК В

Програмний код

```
//ControlScenseAndMenu

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

public class ControlScenseAndMenu : MonoBehaviour {

   public GameObject buttonsMenu;
   public GameObject buttonsExit;
        // Use this for initialization
        void Start () {

        }

        // Update is called once per frame
        void Update () {

        }

   public void ShowExitButtons() {
      buttonsMenu.SetActive(false);
      buttonsExit.SetActive(true);
   }

   public void BackInMenu()
   {
      buttonsMenu.SetActive(true);
      buttonsExit.SetActive(false);
   }

   public void ExitGame()
   {
      Application.Quit();
   }

   public void NewGameLoadScenceExcursia()
   {
      Application.LoadLevel("Excursia");
   }
}

//MenuLanguage

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

```
public class MenuLanguage : MonoBehaviour {

    public GameObject buttonsUkr;
    public GameObject buttonsRus;
    public GameObject buttonsEn;

        // Use this for initialization
        void Start () {

        }

        // Update is called once per frame
    void Update()
    {

    }
    public void NewGameLoadScenceMenuUkr()
    {
        Application.LoadLevel("MenuUkr");
    }

    public void NewGameLoadScenceMenuRus()
    {
        Application.LoadLevel("MenuRus");
    }

    public void NewGameLoadScenceMenuEn()
    {
        Application.LoadLevel("MenuEn");
    }
        }

/// FirstPerson

using UnityEngine;
using UnityEngine.UI;
using System.Linq;
using System.Collections;
using System.Collections.Generic;
#if UNITY_EDITOR
    using UnityEditor;
#endif

[RequireComponent(typeof(CapsuleCollider)),RequireComponent(typeof(Rigidbody)),AddCom
ponentMenu("First Person AIO")]

public class FirstPersonAIO : MonoBehaviour {


    #region Variables

    #region Input Settings
```

```
#endregion

#region Look Settings
public bool enableCameraMovement = true;
public enum InvertMouseInput{None,X,Y,Both}
public InvertMouseInput mouseInputInversion = InvertMouseInput.None;
public float verticalRotationRange = 170;
public float mouseSensitivity = 10;
public float mouseSensitivityInternal;
public  float   fOVToMouseSensitivity = 1;
public float cameraSmoothing = 5f;
public bool lockAndHideCursor = false;
public Camera playerCamera;
public bool enableCameraShake=false;
internal Vector3 cameraStartingPosition;
float baseCamFOV;


public bool autoCrosshair = false;
public bool drawStaminaMeter = true;
float smoothRef;
Image StaminaMeter;
Image StaminaMeterBG;
public Sprite Crosshair;
public Vector3 targetAngles;
private Vector3 followAngles;
private Vector3 followVelocity;
private Vector3 originalRotation;
#endregion

#region Movement Settings

public bool playerCanMove = true;
public bool walkByDefault = true;
public float walkSpeed = 4f;
public KeyCode sprintKey = KeyCode.LeftShift;
public float sprintSpeed = 8f;
public float jumpPower = 5f;
public bool canJump = true;
public bool canHoldJump;
bool didJump;
public bool useStamina = true;
public float staminaDepletionSpeed = 5f;
public float staminaLevel = 50;
public float speed;
public float staminaInternal;
internal float walkSpeedInternal;
internal float sprintSpeedInternal;
internal float jumpPowerInternal;

[System.Serializable]
```

```
public class CrouchModifiers {
    public bool useCrouch = true;
    public bool toggleCrouch = false;
    public KeyCode crouchKey = KeyCode.LeftControl;
    public float crouchWalkSpeedMultiplier = 0.5f;
    public float crouchJumpPowerMultiplier = 0f;
    public bool crouchOverride;
    internal float colliderHeight;

}
public CrouchModifiers _crouchModifiers = new CrouchModifiers();
[System.Serializable]
public class FOV_Kick
{
    public bool useFOVKick = false;
    public float FOVKickAmount = 4;
    public float changeTime = 0.1f;
    public AnimationCurve KickCurve = new AnimationCurve();
    public float fovStart;
}
public FOV_Kick fOVKick = new FOV_Kick();
[System.Serializable]
public class AdvancedSettings {
    public float gravityMultiplier = 1.0f;
    public PhysicMaterial zeroFrictionMaterial;
    public PhysicMaterial highFrictionMaterial;
    public float _maxSlopeAngle = 70;
    public float maxStepHeight = 0.2f;
    internal bool stairMiniHop = false;
    public RaycastHit surfaceAngleCheck;
    public float lastKnownSlopeAngle;
}
public AdvancedSettings advanced = new AdvancedSettings();
private CapsuleCollider capsule;
private const float jumpRayLength = 0.7f;
public bool IsGrounded { get; private set; }
Vector2 inputXY;
public bool isCrouching;

bool isSprinting = false;

public Rigidbody fps_Rigidbody;

#endregion

#region Headbobbing Settings
public bool useHeadbob = true;
public Transform head = null;
public bool snapHeadjointToCapsul = true;
public float headbobFrequency = 1.5f;
public float headbobSwayAngle = 5f;
public float headbobHeight = 3f;
```

```
public float headbobSideMovement =5f;
public float jumpLandIntensity =3f;
private Vector3 originalLocalPosition;
private float nextStepTime = 0.5f;
private float headbobCycle = 0.0f;
private float headbobFade = 0.0f;
private float springPosition = 0.0f;
private float springVelocity = 0.0f;
private float springElastic = 1.1f;
private float springDampen = 0.8f;
private float springVelocityThreshold = 0.05f;
private float springPositionThreshold = 0.05f;
Vector3 previousPosition;
Vector3 previousVelocity = Vector3.zero;
Vector3 miscRefVel;
bool previousGrounded;
AudioSource audioSource;

#endregion

#region Audio Settings

public float Volume = 5f;
public AudioClip jumpSound = null;
public AudioClip landSound = null;
public List<AudioClip> footStepSounds = null;
public enum FSMode{Static, Dynamic}
public FSMode fsmode;

[System.Serializable]
public class DynamicFootStep{
    public enum matMode{physicMaterial,Material};
    public matMode materialMode;
    public List<PhysicMaterial> woodPhysMat;
    public List<PhysicMaterial> metalAndGlassPhysMat;
    public List<PhysicMaterial> grassPhysMat;
    public List<PhysicMaterial> dirtAndGravelPhysMat;
    public List<PhysicMaterial> rockAndConcretePhysMat;
    public List<PhysicMaterial> mudPhysMat;
    public List<PhysicMaterial> customPhysMat;

    public List<Material> woodMat;
    public List<Material> metalAndGlassMat;
    public List<Material> grassMat;
    public List<Material> dirtAndGravelMat;
    public List<Material> rockAndConcreteMat;
    public List<Material> mudMat;
    public List<Material> customMat;
    public List<AudioClip> currentClipSet;

    public List<AudioClip> woodClipSet;
    public List<AudioClip> metalAndGlassClipSet;
```

```
        public List<AudioClip> grassClipSet;
        public List<AudioClip> dirtAndGravelClipSet;
        public List<AudioClip> rockAndConcreteClipSet;
        public List<AudioClip> mudClipSet;
        public List<AudioClip> customClipSet;
    }
    public DynamicFootStep dynamicFootstep = new DynamicFootStep();

    #endregion

    #region BETA Settings
    /*
     [System.Serializable]
public class BETA_SETTINGS{

}

          [Space(15)]
    [Tooltip("Settings in this feild are currently in beta testing and can prove to be unstable.")]
    [Space(5)]
    public BETA_SETTINGS betaSettings = new BETA_SETTINGS();
     */

    #endregion

    #endregion

    private void Awake()
    {
        #region Look Settings - Awake
        originalRotation = transform.localRotation.eulerAngles;

        #endregion

        #region Movement Settings - Awake
        walkSpeedInternal = walkSpeed;
        sprintSpeedInternal = sprintSpeed;
        jumpPowerInternal = jumpPower;
        capsule = GetComponent<CapsuleCollider>();
        IsGrounded = true;
        isCrouching = false;
        fps_Rigidbody = GetComponent<Rigidbody>();
        _crouchModifiers.colliderHeight = capsule.height;
        #endregion

        #region Headbobbing Settings - Awake

        #endregion

        #region BETA_SETTINGS - Awake

#endregion
```

```
    }

    private void Start()
    {
        #region Look Settings - Start

        if(autoCrosshair || drawStaminaMeter){
            Canvas canvas = new GameObject("AutoCrosshair").AddComponent<Canvas>();
            canvas.gameObject.AddComponent<CanvasScaler>().uiScaleMode =
CanvasScaler.ScaleMode.ScaleWithScreenSize;
            canvas.renderMode = RenderMode.ScreenSpaceOverlay;
            canvas.pixelPerfect = true;
            canvas.transform.SetParent(playerCamera.transform);
            canvas.transform.position = Vector3.zero;

            if(autoCrosshair){
                Image crossHair = new GameObject("Crosshair").AddComponent<Image>();
                crossHair.sprite = Crosshair;
                crossHair.rectTransform.sizeDelta = new Vector2(25,25);
                crossHair.transform.SetParent(canvas.transform);
                crossHair.transform.position = Vector3.zero;
            }

            if(drawStaminaMeter){
                StaminaMeterBG = new GameObject("StaminaMeter").AddComponent<Image>();
                StaminaMeter = new GameObject("Meter").AddComponent<Image>();
                StaminaMeter.transform.SetParent(StaminaMeterBG.transform);
                StaminaMeterBG.transform.SetParent(canvas.transform);
                StaminaMeterBG.transform.position = Vector3.zero;
                StaminaMeterBG.rectTransform.anchorMax = new Vector2(0.5f,0);
                StaminaMeterBG.rectTransform.anchorMin = new Vector2(0.5f,0);
                StaminaMeterBG.rectTransform.anchoredPosition = new Vector2(0,15);
                StaminaMeterBG.rectTransform.sizeDelta = new Vector2(250,6);
                StaminaMeterBG.color = new Color(0,0,0,0);
                StaminaMeter.rectTransform.sizeDelta = new Vector2(250,6);
                StaminaMeter.color = new Color(0,0,0,0);
            }
        }
        mouseSensitivityInternal = mouseSensitivity;
        cameraStartingPosition = playerCamera.transform.localPosition;
        if(lockAndHideCursor) { Cursor.lockState = CursorLockMode.Locked; Cursor.visible =
false; }
        baseCamFOV = playerCamera.fieldOfView;
        #endregion

        #region Movement Settings - Start
        staminaInternal = staminaLevel;
        advanced.zeroFrictionMaterial = new PhysicMaterial("Zero_Friction");
        advanced.zeroFrictionMaterial.dynamicFriction =0;
        advanced.zeroFrictionMaterial.staticFriction =0;
        advanced.zeroFrictionMaterial.frictionCombine = PhysicMaterialCombine.Minimum;
```

```
        advanced.zeroFrictionMaterial.bounceCombine = PhysicMaterialCombine.Minimum;
        advanced.highFrictionMaterial = new PhysicMaterial("Max_Friction");
        advanced.highFrictionMaterial.dynamicFriction =1;
        advanced.highFrictionMaterial.staticFriction =1;
        advanced.highFrictionMaterial.frictionCombine = PhysicMaterialCombine.Maximum;
        advanced.highFrictionMaterial.bounceCombine = PhysicMaterialCombine.Average;
        #endregion

        #region Headbobbing Settings - Start

        originalLocalPosition = snapHeadjointToCapsul ? new Vector3(head.localPosition.x,
(capsule.height/2)*head.localScale.y ,head.localPosition.z) : head.localPosition;
        if(GetComponent<AudioSource>() == null) {
gameObject.AddComponent<AudioSource>(); }

        previousPosition = fps_Rigidbody.position;
        audioSource = GetComponent<AudioSource>();
        #endregion

        #region BETA_SETTINGS - Start
        fOVKick.fovStart = playerCamera.fieldOfView;
        #endregion
    }

    private void Update()
    {
        #region Look Settings - Update

            if(enableCameraMovement){
            float mouseYInput;
            float mouseXInput;
            float camFOV = playerCamera.fieldOfView;
            mouseYInput = mouseInputInversion == InvertMouseInput.None || mouseInputInversion
== InvertMouseInput.X ? Input.GetAxis("Mouse Y") : -Input.GetAxis("Mouse Y");
            mouseXInput = mouseInputInversion == InvertMouseInput.None || mouseInputInversion
== InvertMouseInput.Y ? Input.GetAxis("Mouse X") : -Input.GetAxis("Mouse X");
            if(targetAngles.y > 180) { targetAngles.y -= 360; followAngles.y -= 360; } else
if(targetAngles.y < -180) { targetAngles.y += 360; followAngles.y += 360; }
            if(targetAngles.x > 180) { targetAngles.x -= 360; followAngles.x -= 360; } else
if(targetAngles.x < -180) { targetAngles.x += 360; followAngles.x += 360; }
            targetAngles.y += mouseXInput * (mouseSensitivityInternal - ((baseCamFOV-
camFOV)*fOVToMouseSensitivity)/6f);
            targetAngles.x += mouseYInput * (mouseSensitivityInternal - ((baseCamFOV-
camFOV)*fOVToMouseSensitivity)/6f);
            targetAngles.y = Mathf.Clamp(targetAngles.y, -0.5f * Mathf.Infinity, 0.5f *
Mathf.Infinity);
            targetAngles.x = Mathf.Clamp(targetAngles.x, -0.5f * verticalRotationRange, 0.5f *
verticalRotationRange);
            followAngles = Vector3.SmoothDamp(followAngles, targetAngles, ref followVelocity,
(cameraSmoothing)/100);
            playerCamera.transform.localRotation = Quaternion.Euler(-followAngles.x +
originalRotation.x,0,0);
```

```
            transform.localRotation =  Quaternion.Euler(0, followAngles.y+originalRotation.y, 0);
        }

        #endregion

        #region  Input Settings - Update
        didJump = canHoldJump?Input.GetButton("Jump"): Input.GetButtonDown("Jump");

        if(_crouchModifiers.useCrouch){
            if(!_crouchModifiers.toggleCrouch){ isCrouching = _crouchModifiers.crouchOverride ||
Input.GetKey(_crouchModifiers.crouchKey);}
            else{if(Input.GetKeyDown(_crouchModifiers.crouchKey)){isCrouching = !isCrouching
|| _crouchModifiers.crouchOverride;}}
            }
        #endregion

        #region Movement Settings - Update

        #endregion

        #region Headbobbing Settings - Update

        #endregion

        #region BETA_SETTINGS - Update

        #endregion
    }

    private void FixedUpdate()
    {
        #region Look Settings - FixedUpdate

        #endregion

        #region Movement Settings - FixedUpdate

        bool wasWalking = !isSprinting;
        if(useStamina){
            isSprinting = Input.GetKey(sprintKey) && !isCrouching && staminaInternal > 0 &&
(Mathf.Abs(fps_Rigidbody.velocity.x) > 0.01f || Mathf.Abs(fps_Rigidbody.velocity.x) > 0.01f);
            if(isSprinting){
                staminaInternal -= (staminaDepletionSpeed*2)*Time.deltaTime;
                if(drawStaminaMeter){
                    StaminaMeterBG.color = Vector4.MoveTowards(StaminaMeterBG.color, new
Vector4(0,0,0,0.5f),0.15f);
                    StaminaMeter.color = Vector4.MoveTowards(StaminaMeter.color, new
Vector4(1,1,1,1),0.15f);
                }
            }else if((!Input.GetKey(sprintKey)||Mathf.Abs(fps_Rigidbody.velocity.x)< 0.01f ||
Mathf.Abs(fps_Rigidbody.velocity.x)< 0.01f || isCrouching)&&staminaInternal<staminaLevel){
                staminaInternal += staminaDepletionSpeed*Time.deltaTime;
```

```
        }
            if(drawStaminaMeter&&staminaInternal==staminaLevel){
                StaminaMeterBG.color = Vector4.MoveTowards(StaminaMeterBG.color, new
Vector4(0,0,0,0),0.15f);
                StaminaMeter.color = Vector4.MoveTowards(StaminaMeter.color, new
Vector4(1,1,1,0),0.15f);
            }
            staminaInternal = Mathf.Clamp(staminaInternal,0,staminaLevel);
            float x =
Mathf.Clamp(Mathf.SmoothDamp(StaminaMeter.transform.localScale.x,(staminaInternal/stamin
aLevel)*StaminaMeterBG.transform.localScale.x,ref smoothRef,(1)*Time.deltaTime,1),0.001f,
StaminaMeterBG.transform.localScale.x);
            StaminaMeter.transform.localScale = new Vector3(x,1,1);
        } else{isSprinting = Input.GetKey(sprintKey);}

        Vector3 dMove = Vector3.zero;
        speed = walkByDefault ? isCrouching ? walkSpeedInternal : (isSprinting ?
sprintSpeedInternal : walkSpeedInternal) : (isSprinting ? walkSpeedInternal :
sprintSpeedInternal);
        if(IsGrounded || fps_Rigidbody.velocity.y < 0.1) {
            RaycastHit[] hits = Physics.SphereCastAll(transform.position - new
Vector3(0,((capsule.height/2)*transform.localScale.y)-0.01f,0),
capsule.radius,Vector3.down,0,Physics.AllLayers,QueryTriggerInteraction.Ignore);
            float nearest = float.PositiveInfinity;
            IsGrounded = false;
            for(int i = 0; i < hits.Length; i++) {
                if(hits[i].distance < nearest && hits[i].collider != capsule) {
                    IsGrounded = true;
                    advanced.stairMiniHop = false;
                    nearest = hits[i].distance;
                }
            }
        }




        if(advanced._maxSlopeAngle>0 && Physics.Raycast(transform.position - new
Vector3(0,((capsule.height/2)*transform.localScale.y)-capsule.radius,0),new Vector3(dMove.x,-
1.5f,dMove.z),out advanced.surfaceAngleCheck,1.5f)){
            dMove = (transform.forward * inputXY.y * speed + transform.right * inputXY.x *
walkSpeedInternal) * SlopeCheck();
            if(SlopeCheck()<=0){didJump = false;}
        }
        else{
        dMove = transform.forward * inputXY.y * speed + transform.right * inputXY.x *
walkSpeedInternal;
        }

        RaycastHit WT;
```

```
        if(IsGrounded && advanced.maxStepHeight > 0 && Physics.Raycast(transform.position -
new Vector3(0,((capsule.height/2)*transform.localScale.y)-0.01f,0),dMove,out
WT,capsule.radius+0.15f) && Vector3.Angle(WT.normal, Vector3.up)>88){
            RaycastHit ST;
            if(!Physics.Raycast(transform.position - new
Vector3(0,((capsule.height/2)*transform.localScale.y)-(advanced.maxStepHeight),0),dMove,out
ST,capsule.radius+0.25f)){
                advanced.stairMiniHop = true;
                transform.position += new Vector3(0,advanced.maxStepHeight*1.2f,0);
            }
        }
        float horizontalInput = Input.GetAxis("Horizontal");
        float verticalInput = Input.GetAxis("Vertical");
        inputXY = new Vector2(horizontalInput, verticalInput);
        if(inputXY.magnitude > 1) { inputXY.Normalize(); }

        float yv = fps_Rigidbody.velocity.y;

        if (!canJump) didJump = false;

        if(IsGrounded && didJump && jumpPowerInternal > 0)
        {
            yv += jumpPowerInternal;
            IsGrounded = false;
            didJump=false;
        }

        if(playerCanMove)
        {
            fps_Rigidbody.velocity = dMove + (Vector3.up * yv);
        } else{fps_Rigidbody.velocity = Vector3.zero;}

        if(dMove.magnitude > 0 || !IsGrounded) {
            capsule.sharedMaterial = advanced.zeroFrictionMaterial;
        } else { capsule.sharedMaterial = advanced.highFrictionMaterial; }

        fps_Rigidbody.AddForce(Physics.gravity * (advanced.gravityMultiplier - 1));
        /* if(fOVKick.useFOVKick && wasWalking == isSprinting &&
fps_Rigidbody.velocity.magnitude > 0.1f && !isCrouching){
            StopAllCoroutines();
            StartCoroutine(wasWalking ? FOVKickOut() : FOVKickIn());
        } */

        if(_crouchModifiers.useCrouch) {

            if(isCrouching) {
                capsule.height = Mathf.MoveTowards(capsule.height,
_crouchModifiers.colliderHeight/1.5f, 5*Time.deltaTime);
                    walkSpeedInternal = walkSpeed*_crouchModifiers.crouchWalkSpeedMultiplier;
                    jumpPowerInternal = jumpPower*
_crouchModifiers.crouchJumpPowerMultiplier;
```

```
            } else {
            capsule.height = Mathf.MoveTowards(capsule.height,
_crouchModifiers.colliderHeight, 5*Time.deltaTime);
            walkSpeedInternal = walkSpeed;
            sprintSpeedInternal = sprintSpeed;
            jumpPowerInternal = jumpPower;
        }
    }

    #endregion

    #region BETA_SETTINGS - FixedUpdate

    #endregion

    #region Headbobbing Settings - FixedUpdate
    float yPos = 0;
    float xPos = 0;
    float zTilt = 0;
    float xTilt = 0;
    float bobSwayFactor = 0;
    float bobFactor = 0;
    float strideLangthen = 0;
    float flatVel = 0;

    //calculate headbob freq
    if(useHeadbob == true || fsmode == FSMode.Dynamic){
        Vector3 vel = (fps_Rigidbody.position - previousPosition) / Time.deltaTime;
        Vector3 velChange = vel - previousVelocity;
        previousPosition = fps_Rigidbody.position;
        previousVelocity = vel;
        springVelocity -= velChange.y;
        springVelocity -= springPosition * springElastic;
        springVelocity *= springDampen;
        springPosition += springVelocity * Time.deltaTime;
        springPosition = Mathf.Clamp(springPosition, -0.3f, 0.3f);

        if(Mathf.Abs(springVelocity) < springVelocityThreshold && Mathf.Abs(springPosition)
< springPositionThreshold) { springPosition = 0; springVelocity = 0; }
        flatVel = new Vector3(vel.x, 0.0f, vel.z).magnitude;
        strideLangthen = 1 + (flatVel * ((headbobFrequency*2)/10));
        headbobCycle += (flatVel / strideLangthen) * (Time.deltaTime / headbobFrequency);
        bobFactor = Mathf.Sin(headbobCycle * Mathf.PI * 2);
        bobSwayFactor = Mathf.Sin(Mathf.PI * (2 * headbobCycle + 0.5f));
        bobFactor = 1 - (bobFactor * 0.5f + 1);
        bobFactor *= bobFactor;

        yPos = 0;
        xPos = 0;
        zTilt = 0;
        if(jumpLandIntensity>0 && !advanced.stairMiniHop){xTilt = -springPosition *
(jumpLandIntensity*5.5f);}
```

```
        else if(!advanced.stairMiniHop){xTilt = -springPosition;}

        if(IsGrounded){
            if(new Vector3(vel.x, 0.0f, vel.z).magnitude < 0.1f) { headbobFade =
Mathf.MoveTowards(headbobFade, 0.0f,0.5f); } else { headbobFade =
Mathf.MoveTowards(headbobFade, 1.0f, Time.deltaTime); }
            float speedHeightFactor = 1 + (flatVel * 0.3f);
            xPos = -(headbobSideMovement/10) * headbobFade *bobSwayFactor;
            yPos = springPosition * (jumpLandIntensity/10) + bobFactor * (headbobHeight/10) *
headbobFade * speedHeightFactor;
            zTilt = bobSwayFactor * (headbobSwayAngle/10) * headbobFade;
        }
    }
```