МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ОДЕСЬКИЙ ДЕРЖАВНИЙ ЕКОЛОГІЧНИЙ УНІВЕРСИТЕТ

Факультет Магістерської та аспірантської підготовки
Кафедра Автоматизованих систем моніторингу навколишнього середовища

М А Г І С Т Е Р С Ь К А Р О Б О Т А

на тему: Реалізація об'єктів навколишнього середовища інтерактивної моделі здійснення активних впливів на небезпечні атмосферні процеси

Виконав студент 2 курсу групи МК-62
спеціальності 122 Комп'ютерні науки
Димитрук Максим Сергійович
Науковий керівник: к. т. н., доц.
Перелигін Борис Вікторович

Консультант: _____
Рецензент: к. т. н., доц.
Гнатовська Ганна Арнольдівна

ОДЕСА 2018

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

ОДЕСЬКИЙ ДЕРЖАВНИЙ ЕКОЛОГІЧНИЙ УНІВЕРСИТЕТ

Факультет Магістерської та аспірантської підготовки
Кафедра Автоматизованих систем моніторингу навколишнього середовища
Рівень вищої освіти магістр
Спеціальність 122 Комп'ютерні науки
(шифр і назва)

ЗАТВЕРДЖУЮ
Завідувач кафедри автоматизованих систем моніторингу навколишнього середовища
Перелигін Б.В.
" 29 " жовтня 2018 року

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

ДИМИТРУК МАКСИМ СЕРГІЙОВИЧ
(прізвище, ім'я, по батькові)

1. Тема роботи: Реалізація об'єктів навколишнього середовища інтерактивної моделі здійснення активних впливів на небезпечні атмосферні процеси

керівник роботи: Перелигін Борис Вікторович, к.т.н., доцент,
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)
затверджений наказом вищого навчального закладу від „05" жовтня 2018року № 271 «С»

2. Строк подання студентом роботи: 10. 12. 2018 р.

3. Вихідні дані до роботи:
на основі використання існуючих програмних засобів складання інтерактивних моделей реалізувати об'єкти навколишнього середовища для здійснення активних впливів на небезпечні атмосферні процеси

4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити):
1. Аналіз змісту процедур активних впливів на атмосферні процеси
2. Обгрунтування вибору програмних засобів для моделювання активних впливів на атмосферні процеси
3. Розробка об'єктів навколишнього середовища інтерактивної моделі у UNITY та BLENDER
4. Управління проектом
5. Перелік графічного матеріалу:

*1.*Презентація роботи

6. Консультанти розділів роботи:

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв

7. Дата видачі завдання: „29" жовтня 2018 року

- **КАЛЕНДАРНИЙ ПЛАН**

№ з/п	Назва етапів магістерської роботи	Термін виконання етапів роботи	Оцінка виконання етапу	
			у %	за 4-х бальною шкалою
1	Одержання завдання на виконання магістерської роботи	29.10.2018	100	відмінно
2	Пошук та підбір літератури та інших джерел інформації	31.10.2018	100	відмінно
3	Проведення аналізу предметної області і написання першого розділу пояснювальної записки до магістерської роботи	05.11.2018	100	відмінно
4	Проведення аналізу предметної області і написання другого розділу пояснювальної записки до магістерської роботи	08.11.2018	100	відмінно
5	Проведення аналізу предметної області і написання третього розділу пояснювальної записки до магістерської роботи	14.11.2018	100	відмінно
6	Рубіжна атестація	19-24.11.2018	100	відмінно
7	Проведення аналізу предметної області і написання четвертого розділу пояснювальної записки до магістерської роботи	30.11.2018	100	відмінно
8	Виготовлення презентації	03.12.2018	100	відмінно
9	Друкування пояснювальної записки	04.12.2018	100	відмінно
10	Одержання висновку керівника магістерської роботи	05.12.2018	100	відмінно
11	Проходження нормативного контролю	06.12.2018	100	відмінно
12	Переплетіння пояснювальної записки	07.12.2018	100	відмінно
13	Здача роботи ка кафедру та одержання висновку кафедри про допуск магістерської роботи до захисту	10.12.2018	100	відмінно
14	Перевірка на оригінальність тексту	13.12.2018	100	відмінно
15	Одержання рецензії	20.12.2018	100	відмінно
16	Здача готової магістерської роботи і документів секретарю АК	20.12.2018	100	відмінно
	Інтегральна оцінка виконання етапів календарного плану (як середня по етапам)		100	відмінно

Студент _____ Буга Д.І.
(підпис) (прізвище та ініціали)

Керівник роботи _____ Перелигін Б.В.
(підпис) (прізвище та ініціали)

АНОТАЦІЯ

Представлена магістерська робота Димитрука Максима Сергійовича, на тему «Реалізація об'єктів навколишнього середовища інтерактивної моделі здійснення активних впливів на небезпечні атмосферні процеси».

Вона складається з двох частин: теоретичної та практичної.

Мета даної роботи – моделювання інтерактивної моделі впливу на атмсофрені процеси. Вихідною інформацією для даної моделі є модель атмосферної обстановки. У програмній системі реалізовані наступні основні можливості.

Вибір атмосферного процесу, який треба усунути. Введення початкових значень, таких як повне розсіювання, шарувата хмарність, запобігання граду. Запуск процесу знищення хмар за допомогою ракет, присутній також режим спостереження за домогою міні карти.

Система представлена в графічному вигляді, виконана в 3D моделі.

Основні об'єкти були створені у графічному редакторі Blender 3D.

Об'єкти створені за допомогою Blender 3D були експортовані в ігровий движок Unity 3D, де було створене навколишнє середовище для розміщення створених об'єктів.

Підключаємі модулі управляються за допомогою Microsoft Visual Studio. Інтерфейс програми зручний і не вимагає особливих навичок для управління додатком.

Магістерська робота містить: 85 с., рис. 51, табл. 7, додатки 2, використаних літературних джерел 25.

Ключові слова: властивості, сцена, текстури, атмосферні процеси, модель, об'єкти.

SUMMARY

Presented master thesis Dymytruk Maksym, on the theme "Realization of objects of the environment interactive model of implementation of active influences on dangerous atmospheric processes".

It consists of two parts: theoretical and practical.

The purpose of this work - simulate an interactive model of influence on atmosfreen processes. Output information for this model is the model of the atmospheric environment. The software system implements the following key features.

Select the atmospheric process to be eliminated. Introduction of initial values such as full dispersion, layered clouds, preventing hail. Starting the process of destroying clouds with rockets, there is also a mode of monitoring the ability of the minescard.

The system is presented in graphical form, executed in the 3D model.

The main objects were created in the graph editor Blender 3D.

Objects created with Blender 3D have been exported to the Unity 3D engine, where the environment was created to place created objects.

Connected modules are managed using Microsoft Visual Studio. The program interface is user friendly and does not require special skills to manage the application.

Master's work includes: 85 pages, pictures 51, tables 7, 2 applications, used literature 25.

Key words: properties, scene, textures, atmospheric processes, model, objects.

ЗМІСТ

ПЕРЕЛІК СКОРОЧЕНЬ, УМОВНИХ ПОЗНАЧЕНЬ І ТЕРМІНІВ

AAA - Високобюджетні і дорогі проекти

UI - Користувальницький інтерфейс

OM - Режим об'єкта

LOD - Level Of Detail - Створення декількох варіантів одного об'єкта з різними ступенями деталізації

FBX - Один із форматів Unity

EM - Режим редагування

3D - Тривимірна графіка

ВСТУП

Ігровий движок - центральний програмний компонент комп'ютерних та відеоігор або інших інтерактивних додатків з графікою, оброблюваної в реальному часі. Він забезпечує основні технології, спрощує розробку і часто дає продукту можливість запускатися на декількох платформах, таких як ігрові консолі та настільні операційні системи, наприклад, GNU / Linux, Mac OS X і Microsoft Windows.

Однак, незважаючи на велику кількість багатофункціональних інструментів, від розробника все одно потрібна дуже велика кількість праці, щоб створити дійсно цікавий продукт. Однозначним плюсом розробки інтерактивної моделі на вже готовому інструменті (движку), є швидкість розробки програмної частини. Якщо раніше програмісту потрібно було писати безліч рядків коду, щоб використовувати просту можливість перевірки зіткнення між двома об'єктами, то тепер всі ці обчислення робляться за все однією командою. Таким чином, такі движки сильно спростили життя програмістам. Однак, програмування це тільки одна частина розробки інтерактивного додатку. Створення якісного продукту буде так само включати в себе створення великого обсягу графічного матеріалу. І до всього іншого, буде потрібно зробити якісне озвучення.

Розробкою інтерактивної моделі може займатися як одна людина, так і фірма. Розробка ААА проекту (вищий клас) коштує від мільйона доларів і більше. Процес розробки звичайної інтерактивної моделі займає близько року, для ААА-проектів може затягнутися до 2-3 років. Цикл розробки звичайних «казуальних» додатків займає близько 4-6 місяців.

Основною метою магістерської роботи є розробити інтерактивну модель за допомогою движка Unity 3D, але так як даний движок є лише конструктором, то доведеться так само використовувати вільний професійний пакет для створення тривимірної комп'ютерної графіки "Blender" він включає в себе засоби моделювання, анімації, рендеринга і постобробки.

Основні задачі, які потрібно виконати у ході виконання магістерської роботи:

- ознайомитись з ігровим движком Unity 3D;
- ознайомитись з графічним редактором Blender;

- навчитися создавати 3D объекти для нашого проекта у науковому стилі;
- навчитися експортувати об’єкти із графічного редактора Blender у Unity 3D;
- вивчити весь функціонал Unity 3D для правильного функціонування об'єктів;
- додати до проекту ексклюзивний функціонал Unity 3D, який сприяє збільшенню FPS;
- скомпелювати проект на OS Windows для запуску.

1 АНАЛІЗ ЗМІСТУ ПРОЦЕДУР АКТИВНОГО ВПЛИВУ НА АТМОСФЕРНІ ПРОЦЕСИ

Статистика небезпечних природних процесів, що відбулися в Україні за останні 10-15 років, показує, що їх наслідки стають все більш небезпечними для об'єктів економіки, населення і навколишнього середовища. Уже в даний час, прямі і непрямі збитки від них становлять 1-2% від валового національного продукту.

Це змушує враховувати можливий економічний збиток від небезпечних природних процесів при розробці державної економічної політики, прогнозів соціально-економічного розвитку держави і макроекономічних програм. Його облік дозволяє розробляти більш реальні стратегічні плани економічного розвитку. Потреба і бажання людей впливати на що відбуваються погодні явища виникли в далекій давнині. Ще Аристотель зміг узагальнити накопичені до III в. до н.е. спостереження за погодою, ніж надав метеорології статус науки. З його праць видно, що ще в ті часи він розмірковував про механізми утворення блискавок, сильних опадів і, зокрема, граду. Але тільки в XX ст. досягнення науки і техніки дали можливість вченим, які досліджують атмосферні процеси, перейти від візуальних спостережень до спостережень за допомогою технічних засобів і подумати про реальність штучного зміни їх ходу, незважаючи на те, що прогноз погоди і її зміна, на думку видатного фізика, Президента АН СРСР , академіка С.І. Вавилова, висловлену ним ще в 40-х рр. минулого століття: «... найактуальніші і важкі наукові та практичні проблеми». Дослідження цих атмосферних процесів, що відповідають за погодні явища, інтенсивно почали розвиватися в першій половині. Складність і багатогранність розглянутих процесів зажадали залучення великих і різнопланових наукових сил. Результати інтенсивних теоретичних і експериментальних досліджень законів природи, що обумовлюють різні погодні явища, і пошуки способів проведення запланованих експериментів по впливу на погоду дали можливість вченим вирішити зазначені складні наукові і практичні проблеми [1].

Найбільший внесок в розуміння механізмів утворення опадів, туманів, грозової діяльності хмар і інших атмосферних процесів внесли вчені нашої країни і США, чому сприяла державна політика цих країн, яка полягає в

тому, що з дослідженнями цього напрямку надавався державний статус, що дозволило виконати величезну роботу з дослідження атмосферних процесів СРСР, а пізніше в Україні та інших країнах СНД, ці дослідження проводилися, головним чином, поруч НДІ Гідрометеослужби, які в свою чергу приваблювали необхідних фахівців інших міністерств і відомств.

Про можливості відтворення деяких атмосферних явищ висловлювалися і такі видатні російські вчені, як Д. І. Менделєєв, А. І. Воєйков, А. В. Клоссовскі. В історії науки відомі і практичні спроби окремих учених і винахідників впливати на атмосферу в цілях регулювання погоди, головним чином - опадів - дощу, снігу, граду і туманів. На жаль, більшість цих спроб в минулому, при слабкій наукової і технічної бази, не давало надійних результатів.

Після Жовтневої революції в нашій країні були розгорнуті дослідження з експериментальної метеорології. Ще в 1921 р при Наркомзему був організований науково-меліоративний інститут, в коло завдань якого входила і розробка проблем штучного дощу. Особливо широкий розвиток ці проблеми отримали після Всесоюзної конференції по боротьбі з посухою, яка відбулася в 1931 році, коли до досліджень з проблеми фізики хмар і опадів і штучного регулювання погоди був притягнутий ряд інститутів, в тому числі знову організований в Москві інститут Експериментальної гідрометеорології.

З його трьох філій - в Ашхабаді, Одесі та Ленінграді - останній незабаром був перетворений в Ленінградський інститут експериментальної метеорології.

Успіхи радянської фізики атмосфери і клімату створили новий напрямок метеорології - вивчення методів активних впливів на погодні процеси і кліматичні умови. В даний час вплив на погоду, або, точніше, на хмари, опади і тумани, досліджуються і теоретично і експериментально. Дослідження ж впливу на кліматичні умови ведуться поки лише в теоретичному плані.

У встановленні основ цього напрямку в метеорології дуже велику роль зіграли дослідження Ленінградського інституту експериментальної метеорології (1931-1941 рр.), Де під керівництвом В. Н. Оболенського були широко розгорнуті роботи з фізики і мікрофізику хмар і туманів з застосуванням літаків-лабораторій та іншої сучасної техніки і розпочато наукова розробка проблеми активного впливу на хмари і тумани. При цьому в польових умовах були випробувані електричні методи впливу і методи з застосуванням гігроскопічних речовин і електризуватися і

неелектрізованного піску, а в лабораторних умовах поставлені дослідження доцільності застосування хладореагентов. Важливу роль у створенні наукових передумов активного впливу на хмари і опади зіграли роботи Аерології Головної геофізичної обсерваторії, де в 1931 р П. А. Молчанов створив перший в світі радіозонд для спостережень за станом вільної атмосфери і заклав теоретичні та експериментальні основи аерології. Суттєве значення для науки про перетворення погодних і кліматичних умов придбали і капітальні роботи Обсерваторії по динамічної метеорології і по теорії клімату.

Все це, а також досягнення суміжних наук - фізичної хімії, кристалографії та інших дисциплін - дозволили більш глибоко дослідити реальні цілі і шляхи впливу на хмари, опади і тумани. З'ясувалося, що принциповою основою таких впливів можуть з'явитися методи штучного регулювання фазових і мікро структурних перетворень хмар і туманів засобами місцевого застосування. При цьому мається на увазі, що результат для протидії таким спробам, при малих витратах енергії, повинен з'явитися початком загальних змін стану термодинамічних і колоїдальних нестійких хмар і туманів, що в свою чергу має забезпечити практично значущі масштаби одержуваного ефекту в просторі і часі [2].

Систематичне дослідження активного впливу на погодні процеси в СРСР розпочато в 1946 р в Головної геофізичної обсерваторії і Центральної аерологічної обсерваторії. На цьому етапі після деяких пошукових розробок почалися випробування в природі методів впливу на переохолоджені крапельно-рідкі хмари і тумани, з метою викликання опадів з хмар, запобігання небезпечного градобою і розсіювання туманів і хмар. Фізичною основою впливу на крапельно-рідкі хмари при негативних температурах є штучне утворення в них крижаних кристалів за допомогою введення хладореагентів або дрібнодисперсних твердих частинок. Поява кристалів льоду серед переохолоджених крапель різко посилює нестійкість стану хмари.

До активних дій, а в подальшому і до вирішення інших проблем, пов'язаних з погодою, активно залучався Високогірний геофізичний інститут (ВГІ). ВГІ був створений в 1961 р. на базі Ельбруської комплексної експедиції АН СРСР, стаціонарно що діяла на схилах Ельбрусу з 1934 по 1941 рік. У 1942 р всі будівлі і лабораторії експедиції були знищені під час запеклих боїв на схилах Ельбрусу, і тільки в 1947 р. вона була відновлена на більш високому науковому рівні з ініціативи та під керівництвом видатного

вченого і організатора науки, Героя Радянського Союзу Є.К. Федорова. ВГІ став головним інститутом Гідрометеослужби СРСР в області розробки методів і технологій активних впливів на небезпечні природні процеси. І особливо видатних результатів інститут досяг в боротьбі з градобою, які тільки плодам нелегкої праці трудівників села на Північному Кавказі щорічно приносили багатомільйонні збитки. Засновник ВГІ, академік Є. К. Федоров, як і його учень і наступник на посту керівника Госкомгідромета СРСР, академік Ю.А. Ізраель, не випадково вважали, що майбутнє Гідрометеослужби буде нерозривно пов'язане з активними діями на природні процеси. Боротьба з цими силами природи дуже складне завдання. Досить сказати, що грозоградові хмари можуть досягати потужності в кілька тисяч кубокілометрів і мати енергію аналогічної кільком десяткам атомних бомб, скинутих на Хіросіму і Нагасакі разом узятих. Вчені після багаторічних досліджень знайшли всередині цих грозоградових хмар зони зародження градових процесів, знайшли відповідні реагенти, головним з яких є йодисте срібло. Ці реагенти поряд з природними ядрами зростання градин стають конкурентами цих природних градин. Таким чином, суть методу активного впливу на градонебезпечні хмари полягає в створенні в окремо взятій хмарі додатково до природних зародків граду великої кількості штучних зародків, здатних вступати в конкуренцію за хмарну воду. На практиці, дана концепція реалізується шляхом внесення в хмару кристалізованого реагенту AgJ за допомогою спеціальних протиградових снарядів і ракет (рис. 1.1), при внесенні якого при температурі 10ºС виділяються порядку 1012 кристалів з одного грама піросоставу. Кристали протягом 4-10 хвилин можуть вирости до декількох міліметрів і стати зародками величезної кількості градин. Залежно від стадії розвитку хмари, реагент вноситься безпосередньо в висхідний потік або у фронтальну периферію висхідного потоку. У розвиненій хмарі, швидкості висхідних потоків невеликі. На певному рівні відбувається конденсація водяної пари і утворення безлічі дрібних крапель. Продовжуючи підйом, дрібні краплі, зливаючись один з одним, збільшуються в розмірі до декількох міліметрів і на рівні, що перевищує максимальну швидкість висхідного потоку, починають зависати, утворюючи зону підвищеної водності. Зародки граду, що потрапили в цю зону, стикаючись з переохолодженими краплями, обмерзають і швидко збільшуються в розмірах. В умовах обмеженої маси води, здатної накопичитися в хмарі при даній швидкості висхідного потоку, маса накопичилася води перерозподіляється між штучно створеним великою

кількістю зростаючих зародків, в результаті чого жоден з них не досягає великих розмірів, а що утворилися дрібні градини при падінні, встигають розтанути в теплій частині атмосфери. За ефект впливу приймається відсутність граду на землі або локальне випадання дрібного граду, в цьому разі розвивається градова хмара. У разі зрілої хмари - припинення випадання граду або значне зменшення його розміру до розмірів сніжної крупи, які не можуть приносити шкоди полям, садам, городам, будівлям, тваринам і т.д., і скорочення площі одноразової випадання граду.

Пуск протиградової ракети «Алазань-6» показаний на рис. 1.1.

Рисунок 1.1 - Пуск протиградової ракети «Алазань-6»

Основним хладореагентом, використовуваним для впливів на хмари, є тверда вуглекислота. Поверхня випаровує частинки CO_2 має температуру 79 градусів. Поблизу таких частинок в хмарі виникає зона значного переохолодження та пересичення водяної пари. У цих умовах за час порядку 10-5 сек утворюються крижані частки розміром близько 10-6 см, які можуть далі рости в навколишньому середовищі рідких переохолоджених крапель за рахунок перегонки на них водяної пари. Свою льдообразуючу активність CO_2 проявляє при температурі -3°С, -4°С і нижче. При випаровуванні одного

грама твердої вуглекислоти в хмарі може виникнути до 1014-1016 крижаних кристалів [3].

Реагенти, найдрібніші частинки яких можуть служити ядрами кристалізації при попаданні в середу переохолоджених крапель, сприяють їх переходу в твердий стан. При цьому вони є або ядрами сублімації (осадження) водяної пари, або ядрами замерзання переохолоджених крапель в процесі контактів з ними, До таких ядер кристалізації відносяться частинки йодистого срібла (AgJ) і свинцю (PbJ). Температурний поріг льдообразуючої активності срібла порядку -6°С, а свинцю порядку -8°С. Такі льдообразуючі реагенти, як AgJ і PbJ, застосовуються шляхом теплової сублімації в різних наземних і літакових димогенераторах і при горінні піротехнічних складів, що містять ці реагенти в ракетах, пиропатронах, димові шашки і т. п. У відповідних умовах при сублімації грама цих речовин отримують до 1011-1013 частинок - ядер кристалізації.

Загалом, процеси в переохолодженій хмарі, що піддавалося штучному впливу, розвиваються в такій послідовності. Спочатку утворюються крижані зародки у холодній поверхні твердої CO^2 безпосередньо з водяної пари або на частинках інших речовин, що грають роль підставок - ядер кристалізації. Далі відбувається дифузне поширення в хмарі крижаних зародків і їх зростання до розмірів, коли гравітаційна складова стає більше складовою дифузного поширення. При цьому хід процесу перегонки водяної пари з крапель на кристали залежить від співвідношення концентрацій. Якщо в деякому обсязі хмари число кристалів мало в порівнянні з числом крапель, то кристали швидко виростають до розмірів, що викликають їх випадання. Навпаки, при зворотному співвідношенні концентрацій рідкі крапельки швидко випаровуються, але розміри кристалів залишаються при цьому малими і в кінці процесу перегонки. Відбудеться кристалізація хмари без необхідного укрупнення частинок для початку процесу осадкоутворення.

При падінні через переохолоджену крапельно-рідку частину хмари, крижані кристали укрупнюються як внаслідок перегонки на них води з крапель, так і в результаті намораживания на них води при зіткненні з краплями. При падінні нижче нульової ізотерми відбувається їх танення. Якщо нульова ізотерма лежить всередині хмари, то краплі, що утворилися внаслідок танення крижинок і мають необхідні розміри для гравітаційної коагуляції, можуть зливатися з краплями, складовими теплу частину хмари, і з нього випадати.

Для вирішення проблеми запобігання небезпечних градобоїв сільськогосподарських культур в останні роки в наукових установах Гідрометеослужби і Академії наук СРСР проведено ряд теоретичних і експериментальних досліджень градонебезпечних хмар. Передбачається, що в потужних конвективних хмарах, на рівні максимальних вертикальних потоків, відбувається накопичення вологи в рідкому переохолодженому стані. Крижані зародки, які утворюються на рівні природній кристалізації, при падінні через цю частину хмари швидко ростуть, перетворюючись в градини.

Радіолокаційні дослідження показали, що зростання градин в хмарі, в такий відносної локалізованої зоні, відбувається протягом 5-10 хв.

В основу сучасних методів боротьби з небезпечним градобоєм і покладений принцип ослаблення лавинного процесу утворення великих градин, на порівняно рідкісних природних зародках. Це робиться в зоні більшої водності хмари і в умовах значного переохолодження крапельно-рідкої води, введенням значної кількості штучних ядер кристалізації. Останні забезпечують розосередження запасу переохолодженої вологи на більшу кількість крижаних зародків, в результаті чого і зменшується можливість утворення великих градин.

Практично задача зводиться до того, щоб, маючи надійний прогноз та поточний діагноз (за допомогою радіолокаторів) стану хмарності, забезпечити точну і оперативну доставку необхідної кількості речовини, що кристалізується в ту частину хмари, яка за своїм станом потенційно сприятлива для лавинного процесу і зростання градин. З цією метою використовуються ракети або реактивні снаряди, на заданій ділянці траєкторії польоту яких при горінні їх піротехнічного складу переганяється і вводиться в хмару реагент. У роботах по запобіганню градобою в даний час в якості реагентів застосовується в більшості випадків AgJ і PbJ.

Дослідження методів штучного викликання опадів з хмар, при всій науковій і технічній складності проблеми, актуальні в практичному відношенні. Такі дії можуть проводитися, наприклад, на літні конвективні хмари. Так, досліди, проведені в конвективної хмарності в різних районах країни, показали, що при тих ситуаціях, коли переохолоджені вершини хмар при своєму розвитку не досягають рівня температур нижче -7°С, -10°С. Коли зазвичай не спостерігається природних опадів, вони можуть бути викликані штучно і при відповідних умовах досягати земної поверхні. Крім того, в областях інтенсивної конвекції в години її максимального розвитку значна

частина потужних купчастих хмар, які можна порівняти за своєю вертикальної потужності з дощовими хмарами, часто не досягає стадії зледеніння і не дає опадів. Є можливість штучного стимулювання опадів і з таких хмар.

Викликання опадів можливо і з зимової шаруватої і шарувато-кучевої хмарності. При цьому дуже важливим є накопичення додаткових опадів взимку для збільшення вологозапасів в грунті до періоду вегетації сільськогосподарських рослин, а також для створення додаткових запасів снігу в горах для збільшення стоку річок, наприклад, в бавовницьких районах. Восени, в сухі холодні хмарні, але безосадочні дні штучне осадження снігу на полях навіть в малих кількостях може бути використано для запобігання озимих від вимерзання, а в районах отгонного тваринництва - для питного споживання худобою разом з травою. Додаткові опади, викликані штучним шляхом в басейнах водозбору річок, на яких є гідроенергетичні споруди, виявляться корисними для забезпечення їх запасами води.

Слід, однак, мати на увазі, що вихідна штучна зміна мікроструктури хмари може бути ефективним для викликання опадів тільки при наявності достатніх водних запасів в хмарах і при сприятливому вертикальному влагообороті. Ці та інші фізико-географічні чинники великою мірою визначають кількісну сторону результатів впливу на хмари. Відповідно по території Радянського Союзу ведеться спрямоване вивчення кліматичних ресурсів, зокрема водних атмосферних ресурсів.

Систематичні польові досліди по засіву хмар твердої CO^2 здійснюються в СРСР з 1957 р Українським науково-дослідним гідрометеорологічним інститутом над територією експериментального метеорологічного полігону. Аналогічні роботи виконуються і в Головної геофізичної обсерваторії з застосуванням інших реагентів. У підсумку можливість штучного збільшення опадів в межах 10-15% за допомогою сучасних методів впливу на переохолоджені хмари на відносно невеликій території, наприклад на площі до кількох тисяч квадратних кілометрів, вважається встановленою. Дослідження згаданого інституту для району полігону показали, що додавання опадів в 10% за вегетаційний період може мати народногосподарське значення в зонах недостатнього зволоження в роки із середньою кількістю опадів, але не дає помітного ефекту в різко посушливі роки.

Співробітниками ВГІ спільно з фахівцями з оборонних підприємств були створені метеорологічні радіолокатори МРЛ-5 і МРЛ-6, спеціальні снаряди з надміцної пластмаси, начинені реагентом. На відміну від звичайних артилерійських штатних снарядів, при розриві яких спостерігаються уламки великих розмірів, здатних вражати при падінні на землю людей і тварин, ці спецснаряди не давали таких осколків. Крім того, як виявилося, через низьку скорострільності зенітні гармати не завжди встигають вносити потрібну кількість реагенту, тому вчені та інженери нашого інституту також в співдружності з військовими інженерами створили ракетні установки залпового вогню, які здатні в лічені секунди внести в зону або кілька зон зародження граду необхідну кількість реагенту, оскільки в одному місці утворення хмари може бути цих зон кілька. Вони виявляються за допомогою згаданих вище спеціальних метеолокаторов МРЛ-5 і МРЛ-6, які отримали широке застосування в системі градозахисту і у всіх аеропортах нашої країни, а також у багатьох зарубіжних країнах для штормооповіщення і градозахисту. У деяких країнах Варшавської співдружності, де спостерігалися градобої, були створені аналогічні протиградові служби з нашими локаторами і ракетами. Ми ж ще в 70-і рр. розгорнули такі протиградові служби у всіх градонебезпечних районах країни в Узбекистані, Таджикистані, Молдові, Україні, Грузії, Азербайджані та Вірменії, а також у всіх республіках Північного Кавказу, Ставропольському і Краснодарському краях. Протиградовий захист тоді охопив площу понад 12 млн га.

Методи штучного розсіювання хмар і туманів над обмеженими територіями актуальні для ряду практичних завдань, особливо для діяльності авіації та морського транспорту. Тут переслідується мета штучної зміни оптичної щільності туману або низького хмари в районі аеродрому, порту і т. п.

До теперішнього часу в Центральній аерологічної обсерваторії розроблені літакові і наземні засоби, що забезпечують розкриття невеликих територій від низьких переохолоджених крапельно-жид-ких хмар або туманів за допомогою кристалізуючих реагентів - твердої вуглекислоти і деяких йодів.

Штучна кристалізація переохолодженого туману або хмари зазвичай супроводжується його розсіюванням внаслідок укрупнення і випадання частинок. Освіта просвіту зі створенням видимості Землі через туман відбувається спочатку у вигляді невеликих зон, з подальшим розширенням їх до повних просвятів шириною, що досягає іноді 4-5 км. Час їх освіти - кілька

десятків хвилин. Час збереження просвятів в хмарах може сягати півтори години. Дослідним шляхом встановлено, що товщина переохолодженого туману або хмари, в якому розсіювання буде ефективним, може мати вертикальну протяжність до 700-800 м. Що стосується застосування методів штучного розсіювання туманів і хмар над великими площами, то такі дії на атмосферні процеси можуть виявитися істотними для цілей зміни радіаційного балансу окремих територій. Експерименти такого роду пророблені Інститутом прикладної геофізики АН СРСР [4].

Застосовувані в даний час методи розсіювання низьких хмар і туманів придатні тільки при температурах нижче 0°С. Тим часом для практики не менш важливо розсіювання їх при позитивних температурах повітря. Розробка ефективних методів активних впливів на «теплі» хмари з метою викликання опадів також актуальна.

Вивчення хмарності над українським експериментальним полігоном і в інших районах показує, що в осінній і зимовий час нерідко хмари мають температуру вище 0°С або негативну, але близьку до нуля, коли застосування льдообразующих реагентів недоцільно. Так, наприклад, в районі цього полігону протягом однієї зими біля верхньої межі хмар температура -4°С і вище зустрічалася в 45 випадках. За розрахунками Українського гідрометеоінстітута, з цих хмар, якщо буде розроблена методика впливу на них при позитивних температурах, може бути додатково викликано 10-11 мм опадів.

У зв'язку з цим перед наукою стоїть невідкладне завдання виконання широкого комплексу досліджень з метою вишукування коштів і методів активного впливу на хмари і тумани при позитивних температурах. Для цієї мети в даний час досліджуються «теплові» методи розсіювання туманів, а також методи з застосуванням гідроскопічних і деяких інших речовин. Обговорюється питання про відновлення дослідження електричних методів впливу.

Дослідження можливостей і шляхів активного впливу на кліматичні умови в даний час ведеться в плані теоретичних розробок. Однак при цьому враховується досвід меліоративних, зрошувальних і інших заходів такого роду, широко розгорнутих в практиці народного господарства. У першому наближенні в Головної геофізичної обсерваторії розроблені питання можливих змін клімату при впливах на підстилаючу поверхню, зокрема на сніговий і льодовий покрив, питання зміни влагооберту під впливом меліоративних заходів. Тут же проведено деяке методичне вивчення стійких і

нестійких атмосферних рухів з метою оцінки можливості впливати на них, а також врахування ролі трансформації повітряних мас при впливі на кліматичні умови.

Завдяки держпідтримці та інноваційної діяльності багатьох поколінь вчених і конструкторів були створені наукоємні автоматизовані технології і технічні засоби, що перевершують світові аналоги і експортуються в багато країн світу. У 1977 р в Швейцарії спільно з Францию й Італією були розпочати експерименти з придушення градом на основі радянської технології («Гроссферзух-4»). У Канаді та Іспанії були розпочаті роботи по збільшенню опадів за допомогою ракет, снаряженних AgJ. До середини 80-х рр. вітчизняні методи у боротьбі з градом фахівці ВГІ почали застосовувати в Аргентині, Бразілії, Болгарії, Венгрії, а по збільшенню опадів фахівці Центральної аерологічної обсерваторії (ЦАО) і Агентства атмосферних технологій (АТТЕХ) - на Кубі, в Монголії, Сирії, Ірані і ін. Є високі наукові досягнення крім ВГІ і ЦАО і у ін. спеціалізованих інститутів Росгідромету: Головної геофізичної обсерваторії (ГГО), Інституту прикладної геофізики (ІПГ), НВО «Тайфун». Сьогодні поставлено питання про оптимізацію діяльності цих інститутів, хоча у них є високий науковий авторитет не тільки в нашій країні. Вони створювали наукову базу, підвищуючи престиж нашої Гідрометеослужби серед служб інших країн.

Сьогодні інститути Гідрометеослужби, як і інститути РАН, через гостру нестачу виділених фінансових ресурсів почали втрачати свої передові позиції у відповідних галузях науки. Сьогодні, наприклад, один Гарвардський університет в США зі своїми інститутами отримує від держави коштів більше, ніж всі наукові установи РАН разом узяті. Звичайно, при такому положенні не може бути мови про порівняння отриманих наукових результатів вченими цих країн, ніж часто користуються і противники академічної науки в нашій країні. В даний час фінансування робіт із захисту сільгоспугідь від градобиття здійснюється шляхом щорічного укладення держконтрактів з міністерствами сільського господарства суб'єктів України і цивільно-правових договорів з агрофірмами. Проблема полягає в тому, що міністерства сільського господарства КБР, КЧР і РСО-Аланія з 2010 року, коли Мінсільгосп України перестав фінансувати протиградові заходи, виділяють кошти на придбання протиградових послуг в 5-6 разів менше від необхідного. В результаті цього на протиградових пунктах впливу постійно не вистачає ракет, в результаті посіви б'є град, а сільгоспвиробники, втративши урожай, стають банкрутами, не можуть платити по кредитах і

вибувають з оподатковуваної бази. Недофінансування робіт по протиградової захисту знижує її ефективність, призводить до величезних збиткам, що наноситься сільгоспугідь і багаторічних насаджень, хоча витрати, вироблені державою на протиградову захист окупаються до 10 разів, забезпечуючи, як було вище згадано, річний економічний ефект понад 1 млрд грн. в залежності від градонебезпеки року і вартості.

Потрібен державний підхід до цієї важливої справи. Тому для вирішення наявних соціально-економічних завдань необхідна федеральна цільова програма, аналогічна тим програмам, які брали з цього питання в СРСР, що дозволяє сконцентрувати і узгодити фінансові, матеріальні та трудові ресурси з метою їх найбільш ефективного використання, що забезпечує узгодженість рішень федеральних і регіональних завдань, і досягти необхідний кінцевий результат у встановлені терміни.

Основною метою цієї програми має бути розвиток технологій активних впливів, спеціалізованих служб прогнозу, оповіщення та запобігання небезпечних природних процесів, захист населення і об'єктів від стихійних лих і забезпечення стійкого економічного розвитку і захисту від стихійних процесів.

У науковому плані, основний акцент програми має припадати на подальше проведення фундаментальних теоретичних і експериментальних досліджень для розвитку фізичних моделей формування небезпечних явищ природного характеру та створення науково обґрунтованих принципів і методів їх прогнозу, моніторингу та запобігання.

З технічної точки зору повинно передбачатися переозброєння протиградових служб і протилавинних підрозділів Росгідромету на новостворені інноваційні технічні засоби, а також створення принципово нових технічних комплексів моніторингу, оповіщення та запобігання небезпечних природних явищ на великих територіях на основі сучасної пілотованої і безпілотної авіації, радіолокаційних, грозопеленгаціонних і супутникових інформаційних систем.

У виробничому плані кошти програми повинні направлятися на фінансування заходів по розширенню площ протиградового захисту, числа об'єктів захисту від снігових лавин, відновлення виробничих робіт по штучному збільшенню опадів, розсіювання туманів і захисту від посух і заморозків [5].

2 ВИБІР ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ ДЛЯ РОЗРОБКИ

2.1 Порівняльна характеристика ігрових движків

Unity - найпопулярніший движок для створення 2D- і 3D-інтерактивних моделей. Безперечно, він став лідером індустрії, і, як тільки з'являється нова ігрова, або графічна технологія, розробники негайно реалізують її в Unity. Движок Unity особливо цінний за низький поріг входження для початківців користувачів, завдяки цьому, а також тому, що інді-версія безкоштовна, навколо движка організувалося величезне співтовариство. Низький поріг входження є результатом грамотного дизайну програми, багато речей можна виконати за допомогою різних редакторів, не написавши при цьому ні строчки коду [6].

Основний інтерфейс движка Unity 3D показаний на рис. 2.1.

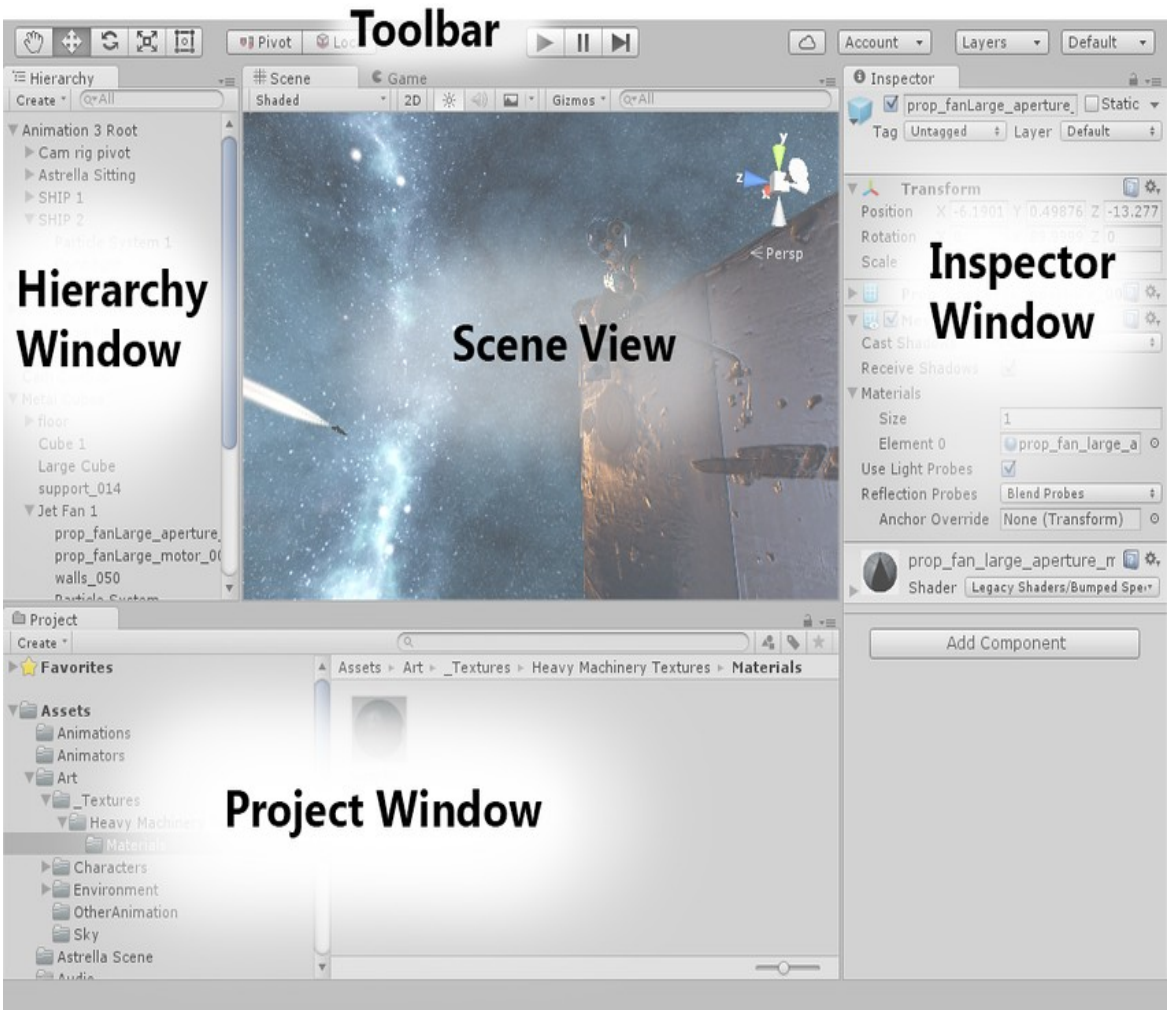

Рисунок 2.1 - Основний інтерфейс Unity 3D

На табл. 2.1 проведений перелік основних недоліків і переваг движка 3D Unity.

Таблиця 2.1 Переваги і недоліки Unity 3D

Недоліки	Переваги
Є недоліки для різних платформ	Безкоштовний для Indie-розробників
Рендер має нарікання	Багатоплатформовий
Немає креслень	Простий в освоєнні
Немає вихідного коду	Багатий магазин активів
	Швидка компіляція
	Багато документації та уроків

Torque 2D / 3D, був свого часу лідером, але під натиском Unity втратив свої позиції. Проте до цих пір на ньому розробляється безліч успішних проектів, оскільки він активно розвивається спільнотою. Відмінності між

двовимірної і тривимірної версіями досить значні, але є і загальні елементи, наприклад розвинена мережева підсистема. Після виходу в світ open source T3D зберіг і навіть збільшив свої можливості, а T2D, навпаки, багато втратив. Наприклад, він втратив абсолютно всі вбудовані редактори, які, очевидно, були вилучені за певних юридичних угод. Проте на ньому можна розробляти проекти для трьох платформ: Windows, OS X і, що найцікавіше, iOS (і продавати в App Store, не відраховуючи ні копійки авторам движка). Даний движок - це одна кодова база на C ++ без додаткових експортерів. Як видно, фундаментальні відмінності 2D і 3D - версій полягають в графічної підсистемі: T2D для візуалізації використовує OpenGL, а T3D - Direct.

В якості скриптової мови в T2D, як і в T3D, використовується Torque Script. Разом з тим в T2D для опису ігрових елементів служить XML-подібна мова TAML. Вона дозволяє визначити властивості об'єктів на стадії ініціалізації рівня гри. Для відтворення звуків T2D використовує бібліотеку OpenAL. Симуляція фізики здійснюється за допомогою движка Box2D, що став стандартом в двовимірних фізичних обчисленнях. Незважаючи на те що в двовимірному TORK ще поки немає конструктора GUI, за допомогою засобів движка (в скриптовому коді) можна створювати призначений для користувача інтерфейс звичними компонентами, а не простими спрайтами. Однак, якщо необхідний компонент відсутній, його можна створити на основі спрайтів. Маючи аналогічну з 3D-версією мережеву систему, на T2D можна розробляти мультиплеєрні ігри, які набирають популярність, - наприклад P2P з планшетів [7].

Основний інтерфейс движка Torque 3D показаний на рис. 2.2.

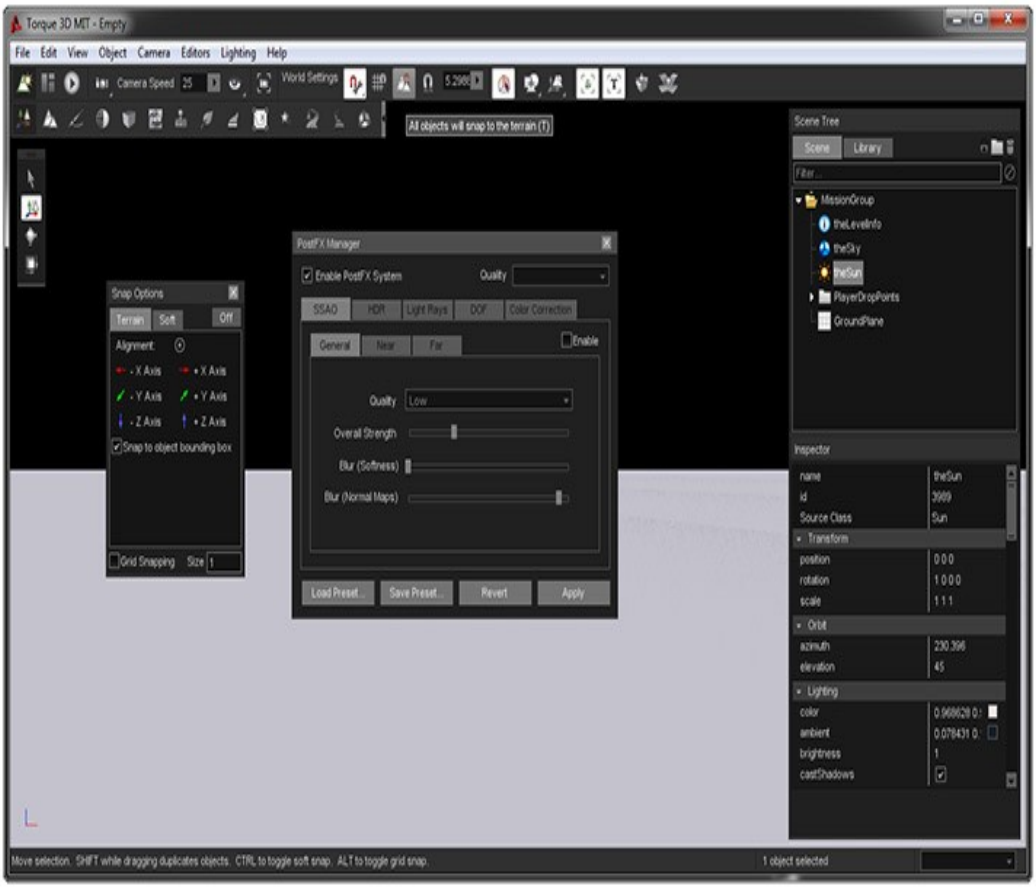

Рисунок 2.2 - Основний інтерфейс Torque 3D

На табл. 2.2 проведений перелік основних недоліків і переваг движка 3D Torque.

Таблиця 2.2 Переваги і недоліки Torque 3D

Недоліки	Переваги
Немає якісного редактора рівнів	Якісний
Повільна компіляція	Багатоплатформовий
Високі вимоги до заліза	Легкий в освоєнні
Немає вихідного коду	Безкоштовний

CryEngine 3 бере початок своєї історії в 2001 році, коли була анонсована перша розроблена на ньому гра Far Cry. З тих пір минуло багато часу, і поточна на даний момент п'ята - остання версія була випущена в жовтні 2016-го. Розробники цього движка з самого початку мали на меті не самим створювати на ньому інтерактивні моделі, а продавати його як технологію. Отже, всі розробленні Crytek'ом програми - це з метою зробити

додаткову рекламу своєму головному продукту. Хоча для вивчення він доступний безкоштовно, щоб розробляти на ньому комерційні проекти, необхідно заплатити, при чому ціна публічно не оголошується. В результаті ліцензіат отримує движок, документацію (навчальні матеріали), вихідний код, а також оперативну підтримку. Крім того, процес ліцензування движка таїть в собі безліч підводних каменів - хоча б те, що ліцензувати його може тільки юридична особа, яка має надати дані про розроблені продукти і в окремих випадках про всіх своїх співробітників.

Основний інтерфейс движка CryEngine показаний на рис. 2.3.

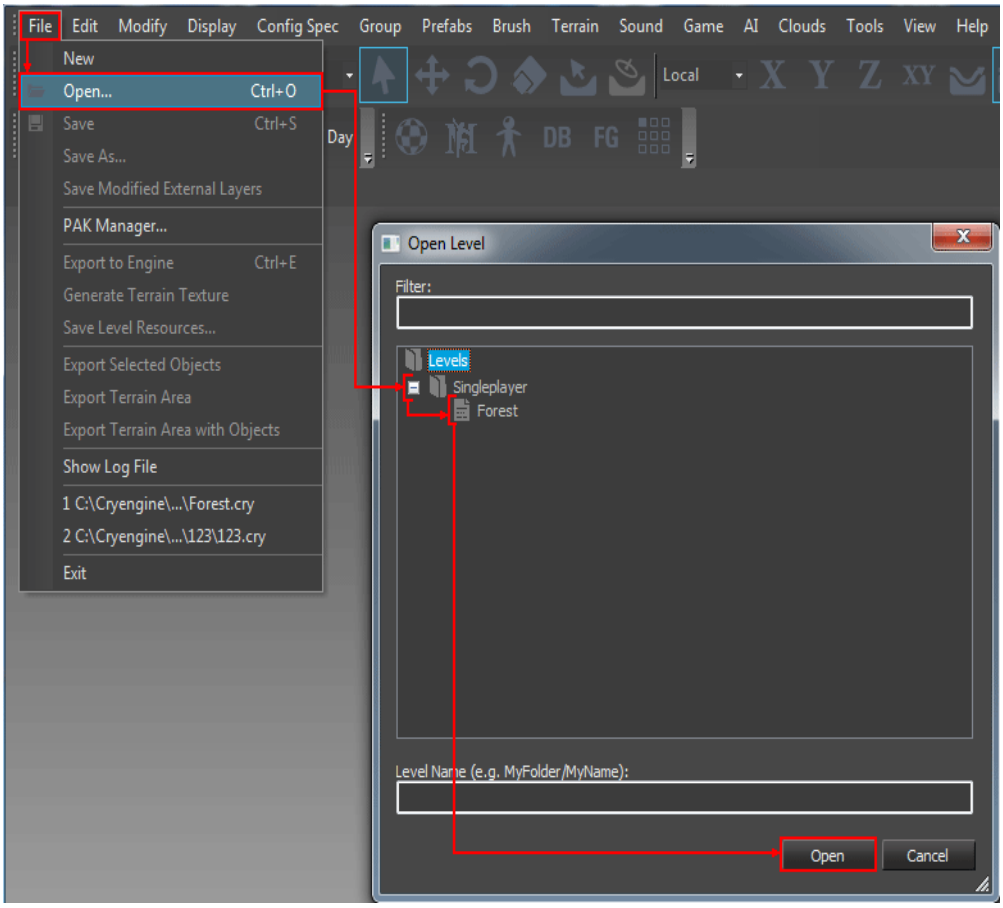

Рисунок 2.3 - Основний інтерфейс CryEngine 3

На табл. 2.3 проведений перелік основних недоліків і переваг движка CryEngine.

Таблиця 2.3 Переваги і недоліки CryEngine 3

Недоліки	Переваги
Не безкоштовний	Якісний
Повільна компіляція	Багато документації та уроків

Високі вимоги до заліза	Легкий в освоєнні
Не багатоплатформовий	
Неможливо продавати свої ассети скріпти	

На відміну від попередніх движків лінійки (які були виключно PC-орієнтованими), CryEngine 3 орієнтований на створення крос-платформних інтерактивних моделей, призначених для PC і консолей. В даний час підтримуються платформи Xbox 360, Xbox One, PlayStation 3-4, WiiU, а також технології візуалізації настільної Windows - DirectX 9-11. Як можна помітити, підтримки мобільних платформ немає. У ньому спочатку присутня підтримка глобальних мультіплеєрних (MMO) ігор. CryEngine 3 володіє приголомшливим списком технологій візуалізації, ось деякі з них: динамічне освітлення і затінення в реальному часі, затуманення, карти нормалей і паралакс-маппінг, підповерхневе розсіювання, світлові промені і хвилі, управління рівнем деталізації ландшафту, а також багато іншого. Фізичний компонент движка CryPhysics також працює незалежно від фізичних API, таких як PhysX. Вбудована система анімації пропонує кілька відмінних підсистем: індивідуалізація персонажів, параметрична скелетна анімація, процедурне деформування руху. Також заслуговує на окрему увагу вбудована система AA, яка дозволяє обробляти поведінку не тільки персонажів, але і транспортних засобів. Вона складається з трьох модулів: розумні об'єкти, алгоритми динамічного виявлення шляху, а також система, керована сценаріями [8].

UDK - це безкоштовна версія движка Unreal Engine 3, що володіє всім успадкованим інструментарієм останнього для створення віртуальних світів. Список підтримуваних платформ не такий широкий, як у Unity, але цього цілком вистачає, щоб окупити розробку: Windows PC, Windows Store, OS X, iOS, Android і консолі передостаннього покоління. Для скриптинга в движку використовується власна мова - UnrealScript. На сайті розробників представлено багато навчальних матеріалів, як текстових, так і відео, як по редактору, так і по скриптингу. UE3 отримав безліч нагород на індустріальних заходах, а також в кінематографі і не раз ставав кращим ігровим / графічним движком року. Можна сказати, що UDK відрізняється від UE3 тільки відсутністю вихідного коду [9].

Основний інтерфейс движка UDK показаний на рис. 2.4.

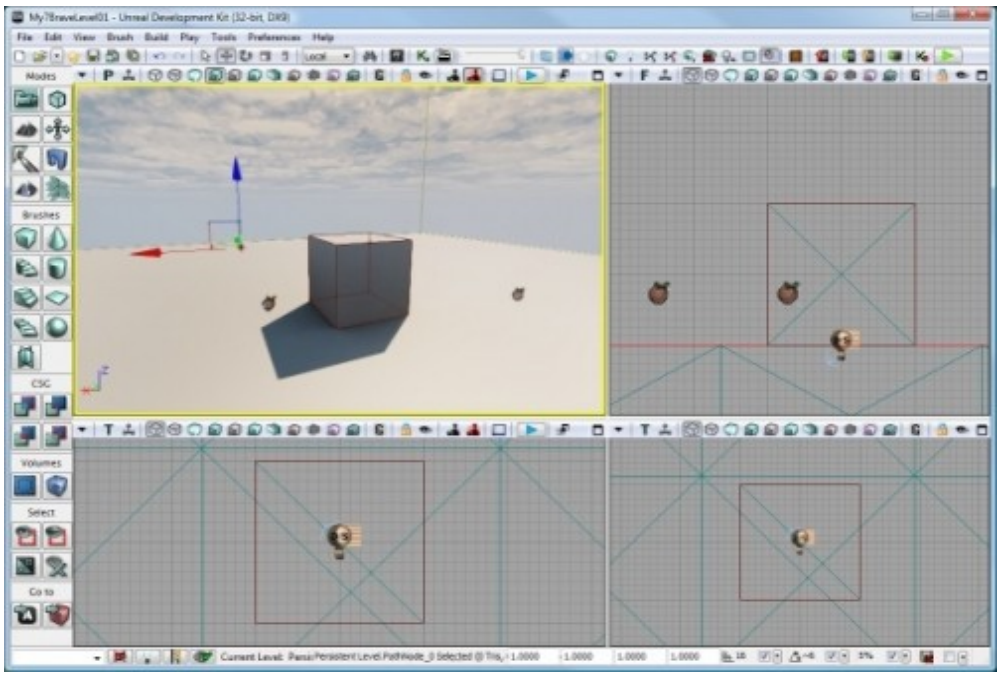

Рисунок 2.4 - Основний інтерфейс UDK

На табл. 2.4 проведений перелік основних недоліків і переваг движка UDK.

Таблиця 2.4 Переваги і недоліки UDK

Недоліки	Переваги
Тільки для професіоналів	Безкоштовне розповсюдження
Повільна компіляція	Простий, зручний інтерфейс;
Не самі передові технології	Великий набір інструментів
Не багатоплатформовий	Багатоплатформовий

Що до функціоналу, гнучка система анімації дозволяє контролювати кожну деталь анімованого об'єкта. Анімаційна модель контролюється системою AnimTree, яка включає наступні механізми: контролер змішання (Blend), контролер керований даними, фізичні, процедурно-скелетні контролери. Для імпортування об'єктів використовується формат FBX, що став стандартом для експорту моделей між редакторами. Для візуалізації UE3 використовує 64-бітний кольоровий HDR графічний конвеєр, який здійснює гамма-корекцію, розмиття рухомих об'єктів, зовнішню окклюзию і інші ефекти постобробки. Движком підтримуються всі сучасні ефекти освітлення і технології візуалізації: нормалізовані карти, параметризоване освітлення по Фонгу, різні анізотропні ефекти та інше. UE3 відомий своєю високо

оптимізованою мережевою архітектурою, що включає підтримку онлайнових баталій для ігор різних жанрів.

Frostbite Engine - ігровий движок, розроблений компанією EA Digital Illusions CE; застосовується як у власних розробках, так і проектах інших філіалів Electronic Arts (EA). Першою грою, створеною з використанням цього движка стала Battlefield: Bad Company 2008 року. Движок був розроблений для заміни технічно застарілої технології Refractor Engine, яка використовувалася в попередніх інтерактивних моделях фірми [10].

Основний інтерфейс движка Frostbite Engine показаний на рис. 2.5.

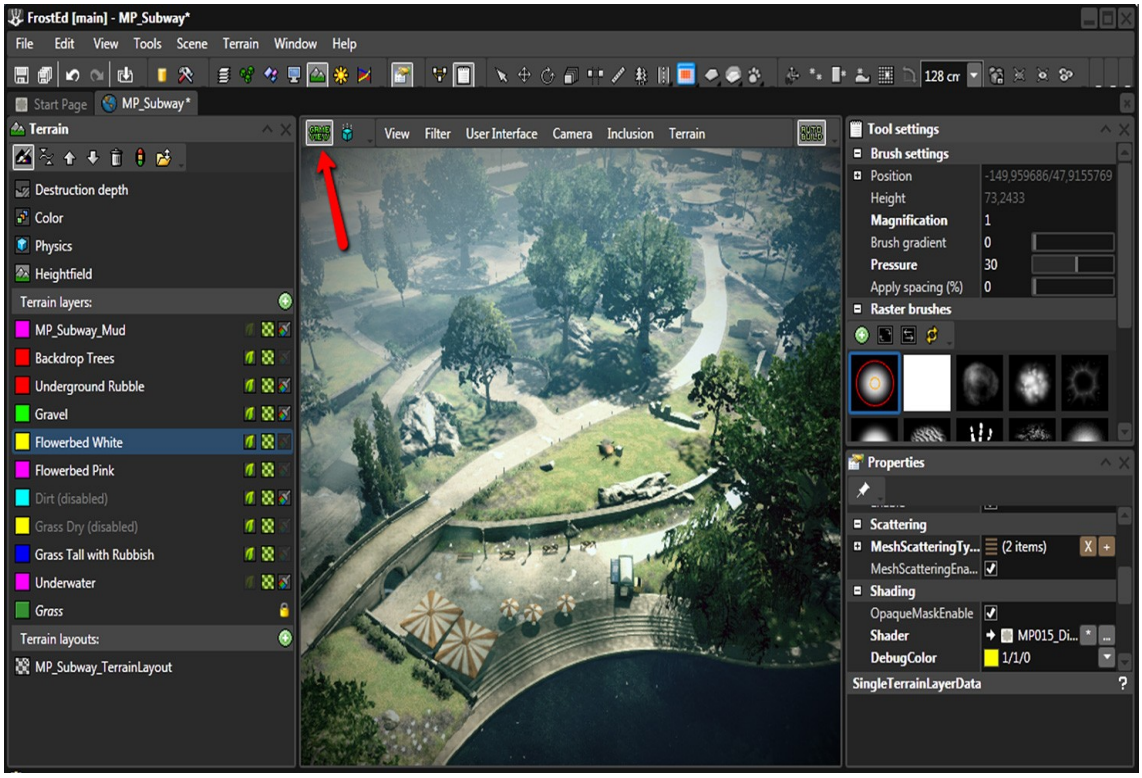

Рисунок 2.5 - Основний інтрефейс Frostbite Engine

Движок відноситься до типу подпрограммного забезпечення (англ. Middleware) і являє собою зв'язку декількох компонентів, таких як графічний движок, звуковий движок і т.д. В операційній системі Microsoft Windows ігровий движок підтримує відображення графіки за допомогою Mantle починаючи з версії 3, DirectX 9, DirectX 10, DirectX 10.1, а починаючи з версії 1.5 - і DirectX 11. Однією із заявлених особливостей є оптимізація для роботи на багатоядерних процесорах [11].

На табл. 2.5 проведений перелік основних недоліків і переваг движка Frostbite Engine.

Таблиця 2.5 Переваги і недоліки Frostbite Engine

Недоліки	Переваги
Не безкоштовний	Ідеальне руйнування ландшафту
Не багатоплатформовий	Динамічне освітлення
Високі вимоги до заліза	Ігровий редактор FrostED
Неможливо продавати свої ассети та	Власний звуковий движок

Технологія здатна обробляти знищення ландшафту і оточення (наприклад, будівель, дерев, автомобілів). Підтримується динамічне освітлення і затінення з функцією HBAO, процедурний шейдинг, різні пост-ефекти (наприклад, HDR і depth of field), система частинок і техніки текстурирования, такі, як бамп-маппінг. Максимальний розмір локації становить обмеження в 32 × 32 кілометри відображаємої площі і 4×4 кілометри ігрового простору. Крім цього, за твердженням творців, максимальна дистанція промальовування дозволяє побачити рівень аж до горизонту. Також вбудований власний звуковий движок, що не вимагає використання спеціалізованих засобів, подібних EAX.

Движок комплектується ігровим редактором FrostED, написаному на мові програмування C #. Програма призначена для створення рівнів, а також роботи з мешамі, шейдерами і об'єктами.

Для спрощення портування движок використовує модульну систему залежних компонентів; підтримує різні системи рендеринга (Direct3D, OpenGL, Pixomatic; в ранніх версіях: Glide, S3, PowerVR), відтворення звуку (EAX, OpenAL, DirectSound3D; раніше: A3D), засоби голосового відтворення тексту, розпізнавання мови, модулі для роботи з мережею та підтримуваних пристроїв введення. Для гри по мережі підтримуються технології Windows Live, Xbox Live, GameSpy і інші, включаючи до 64 гравців (клієнтів) одночасно. Таким чином, движок адаптували і для застосування в іграх жанру MMORPG [12].

Згадана лише мізерна частина доступних для використання ігрових движків.

На табл. 2.6 проведений аналіз усіх основних факторів за якими можно обрати ігровий движок.

Таблиця 2.6 Допоміжна порівняльна характеристика усіх движків

Характеристика	Вимоги	Unity	Torque	Cry	UDK	Frostbite

		3D	2D/3D	Engine 3		Engine
Режим рендеринга	3D	+	-	+	+	+
Звуковий движок		-	-	-	+	+
Фізичний движок		+	+	+	+	+
Ігровий II		+	-	-	+	+
Цільовий жанр	RPG	+	+	+	-	+
Мова розробки	C# или JavaScript	+	-	+	-	+
Інтеграція з IDE	Visual Studio	+	+	+	+	-
ОС для розробки	Windows	+	+	+	+	+
Цільові платформи	Windows, OSX	+	+	+	+	+
Ціна	безкоштовно	+	+	-	+	-
Цільова аудиторія	Для професіоналів	Так	Так	Так	Ні	Так
Якість документації		3	2	2	1	3
Інтерфейс користувача		3	2	3	3	2
Результуючий рейтинг		6	4	5	4	5

2.2 Функціональні можливості Unity 3D

Unity 5 володіє величезною кількістю переваг перед іншими ігровими движками. Ком'юніті Unity 5 на сьогоднішній момент є найбільшим в світі. На офіційному сайті Unity є спеціальний розділ, в якому можна знайти статистику по ігровим движкам. За цими даними Unity 5 використовує понад 50% розробників відеоігор, 20% належать Unreal Engine, а решта ігрових движків - 30%. Для розробки 2D або 3D інді-ігор Unity 5 підходить за всіма параметрами. Unity - це інструмент для розробки двох-і тривимірних додатків та ігор, що працює під операційними системами Windows, Linux і OS X. Створені за допомогою Unity програми працюють під операційними системами Windows, OS X, Windows Phone, Android, Apple iOS, Linux, а також на ігрових приставках Wii, PlayStation 3, PlayStation 4, Xbox 360, Xbox One і MotionParallax3D дисплеях (пристрої для відтворення віртуальних голограм), наприклад, Nettlebox. Є можливість створювати додатки для запуску в браузерах за допомогою спеціального модуля Unity (рис. 2.6), а також за допомогою реалізації технології WebGL (рис. 2.7).

Раніше була експериментальна підтримка реалізації проектів в рамках модуля Adobe Flash Player, але пізніше команда розробників Unity прийняла складне рішення щодо відмови від цього.

Рисунок 2.6 - Браузерна гра запущена за допомогою Unity Web Player

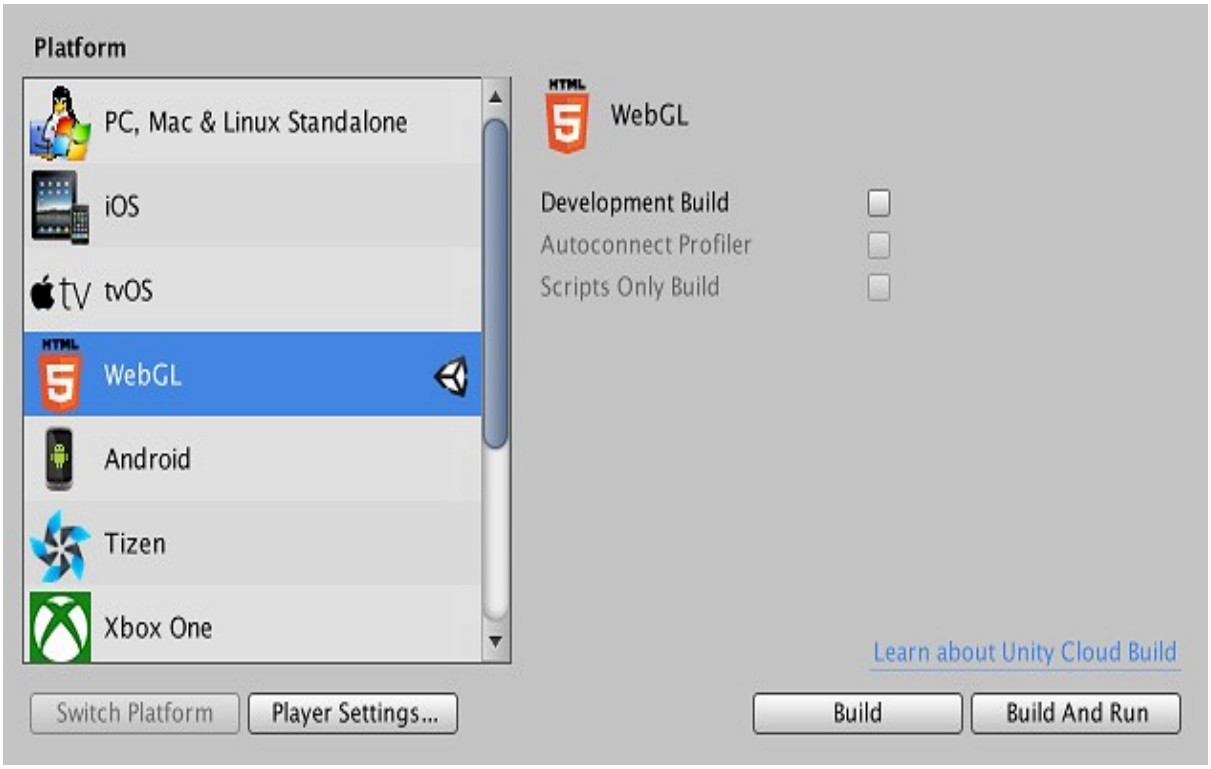

Рисунок 2.7 - Технологія WebGL для запуску браузерних ігр

Додатки, створені за допомогою Unity, підтримують DirectX і OpenGL. Активно движок використовується як великими розробниками (Blizzard, EA, QuartSoft, Ubisoft), так і розробниками Indie-ігор (наприклад, Pathologic, Kerbal, Space Program, Slender: The Eight Pages, Slender: The Arrival, Surgeon Sim-

ulator 2013, Baeklyse Apps: Guess the actor та інші) в силу наявності безкоштовної версії, зручного інтерфейсу і простоти роботи з движком.

Редактор Unity має простий Drag & Drop інтерфейс, який легко налаштовувати, що складається з різних вікон, завдяки чому можна проводити налагодження проекту прямо в редакторі. Движок підтримує три сценарних мови: C#, JavaScript (модифікація), Boo (діалект Python). Boo прибраний в 5-ій версії. Розрахунки фізики виробляє фізичний движок від NVIDIA.

Розробка AAA-проектів в Unity - найскладніший процес. По-перше, будь-який скрипт відразу тягне за собою купу помилок, які в майбутньому необхідно виправити, або переписати скрипт заново. По-друге, Unity 5 все ще має погану оптимізацію. Весь контент який стоїть у вікні Project, але не стоїть в сцені, буде запечений, а значить що гра буде важити в рази більше, ніж передбачалося. Unity в найближчому оновленні виправить цю проблему. У движку є ряд проблем зі скроллингом. При приближенні до об'єкта в певний момент камера наближається повільніше. Якщо потрібно максимально близько наблизиться до землі, то іноді це буває дуже складно зробити. Швидше за все в найближчих оновленнях скролінг так само виправлять [13].

2.3 Графічний 3D редактор Blender

Для 3D-моделювання був використаний редактор Blender - вільний, професійний пакет для створення тривимірної комп'ютерної графіки, що включає в себе засоби моделювання, анімації, рендеринга, постобробки і монтажу відео зі звуком, компонування за допомогою «вузлів» (Node Compositing), а також для створення інтерактивних ігор. В даний час користується найбільшою популярністю серед безкоштовних 3D редакторів в зв'язку з його швидким і стабільним розвитком, якому сприяє професійна команда розробників.

Характерною особливістю пакету Blender є його невеликий розмір в порівнянні з іншими популярними пакетами для 3D-моделювання. У базову поставку не входить розгорнута документація і велика кількість демонстраційних сцен.

Функції пакета:

– підтримка різноманітних геометричних примітивів, включаючи полігональні моделі, систему швидкого моделювання в режимі sub-

division surface (SubSurf), криві Безьє, поверхні NURBS, metaballs (метасфери), скульптурне моделювання та векторні шрифти;

- універсальні вбудовані механізми рендеринга і інтеграція з зовнішнім рендерер YafRay, LuxRender і багатьма іншими;
- інструменти анімації, серед яких інверсна кінематика, скелетна анімація і сіткова деформація, ключові кадри, нелінійна анімація, редагування вагових коефіцієнтів вершин, обмежувачі, динаміка м'яких тіл (включаючи визначення колізій об'єктів при взаємодії), динаміка твердих тіл на основі фізичного движка Bullet і система волосся на основі частинок;
- python використовується як засіб створення інструментів і прототипів, системи логіки в іграх, як засіб імпорту, та експорту файлів (наприклад COLLADA), автоматизації завдань;
- базові функції нелінійного редагування і комбінування відео.
- blender Game Engine - підпроект Blender, що надає інтерактивні функції, такі як визначення колізій, движок динаміки і програмована логіка. Також він дозволяє створювати окремі real-time додатки починаючи від архітектурної візуалізації до відео ігор [14].

Blender мав репутацію програми, складної для вивчення. Практично кожна функція має відповідне їй поєднання клавіш, і враховуючи кількість можливостей, що надаються Blender, кожна клавіша включена в більш ніж одне поєднання (shortcut). С тих пір як Blender став проектом з відкритим вихідним кодом, були додані повні контекстні меню до всіх функцій, а використання інструментів зроблено більш логічним і гнучким. Додамо сюди подальше поліпшення користувальницького інтерфейсу з введенням колірних схем, прозорих плаваючих елементів, новою системою перегляду дерева об'єктів і різними дрібними змінами.

Відмінні риси інтерфейсу користувача:

- Режими редагування. Два основні режими Об'єктний режим (Object mode) і Режим редагування (Edit mode), які перемикаються клавішею Tab. Об'єктний режим в основному використовується для маніпуляцій з індивідуальними об'єктами, в той час як режим редагування - для маніпуляцій з фактичними даними об'єкта. Наприклад, для полігональної моделі в об'єктному режимі ми можемо переміщати, змінювати розмір і обертати модель цілком, а режим редагування використовується для маніпуляції окремих

вершин конкретної моделі. Також є кілька інших режимів, таких як Vertex Paint та UV Face select;

- широке використання гарячих клавіш. Більшість команд виконується за допомогою клавіатури. До появи 2.x і особливо 2.3x версії, це був єдиний шлях виконувати команди, і це було найбільшою причиною створення репутації Blender'у як складної для вивчення програми. Нова версія має більш повне графічне меню;
- управління робочим простором. Графічний інтерфейс Blender'a складається з одного або декількох екранів, кожен з яких може бути розділений на секції і підсекції, які можуть бути будь-якою частиною інтерфейсу Blender'a. Графічні елементи кожної секції можуть контролюватися тими ж інструментами, що і для маніпуляції в 3D просторі, для прикладу можна зменшувати і збільшувати кнопки інструментів тим же шляхом, що і в 3D перегляді. Користувач повністю контролює розташування і організацію графічного інтерфейсу, це робить можливим налаштування інтерфейсу під конкретні завдання, такі як редагування відео, UV mapping і текстурування, і приховування елементів інтерфейсу які не потрібні для даного завдання. Цей стиль графічного інтерфейсу дуже схожий на стиль, використовуваний в редакторі UnrealEd карт для гри Unreal Tournamentx [15].

Робочий простір Blender'a вважається одним з найбільш новаторських концепцій графічного інтерфейсу для графічних інструментів і натхненним дизайном графічного інтерфейсу патентованих програм, таких як Luxology's Modo.

Додаткові особливості:

- у програмі Blender сутність, що взаємодіє з навколишнім світом і її дані (форма або функції об'єкта) разделяемості. Ставлення Об'єкт-Дані представляється відношенням 1: n (термін, що відноситься до теорії баз даних, позначає можливість декількох об'єктів використовувати одні і ті ж дані - один до багатьох або сюрьекція);
- внутрішня файлова система, що дозволяє зберігати кілька сцен в єдиному файлі (званому .blend файл);
- всі «.blend» файли сумісні як із старішими, так і з більш новими версіями Blender. Так само всі вони переносяться з однієї платформи

на іншу і можуть використовуватися як засіб перенесення створених раніше робіт;

- blender робить резервні копії проектів під час всієї роботи програми, що дозволяє зберегти дані при непередбачених обставинах;
- всі сцени, об'єкти, матеріали, текстури, звуки, зображення, post-production ефекти можуть бути збережені в єдиний «.blend» файл;
- налаштування робочого середовища можуть бути збережені в «.blend» файл, завдяки чому при завантаженні файлу ви отримаєте саме те, що зберегли в нього. Ви можете зберегти файл як «власні за замовчуванням», і кожен раз при запуску Blender ви будете отримувати необхідний набір об'єктів та підготовлений до роботи інтерфейс.
- Використовуючи будь-який з безлічі доступних інструментів для проекту сітки, система проста в управлінні текстури простору для даної геометрії. Прогнози можуть бути експортовані у вигляді зображення макетів, розгорнуті області можуть бути адаптовані до існуючих зображень, застосовані численні текстури і спеціальні матеріали, такі як дзеркальне відображення і bump maps, зміна може бути зроблено в інтерактивному режимі, і всі зміни можна реальному часі [16].

2.4 Функціональні можливості редактора Blender 3D

2.4.1 Сцена Blender'a за замовчуванням

Спочатку, Blender запускається з робочим простором за замовчуванням. Дана робоча область розділена на п'ять частин, які містять перераховані нижче редактори:

- Info Editor (інформаційний редактор)у верхній частині робочої області.
- 3D View (3D-вид сцени), займає найбільшу частину робочого простору програми.
- Timeline (тимчасова шкала), розташована внизу.
- Outliner (структура проекту), знаходиться справа вгорі.
- Properties editor (редактор властивостей), розташований справа внизу.

Початковий экран Blender показаний на рис. 2.8.

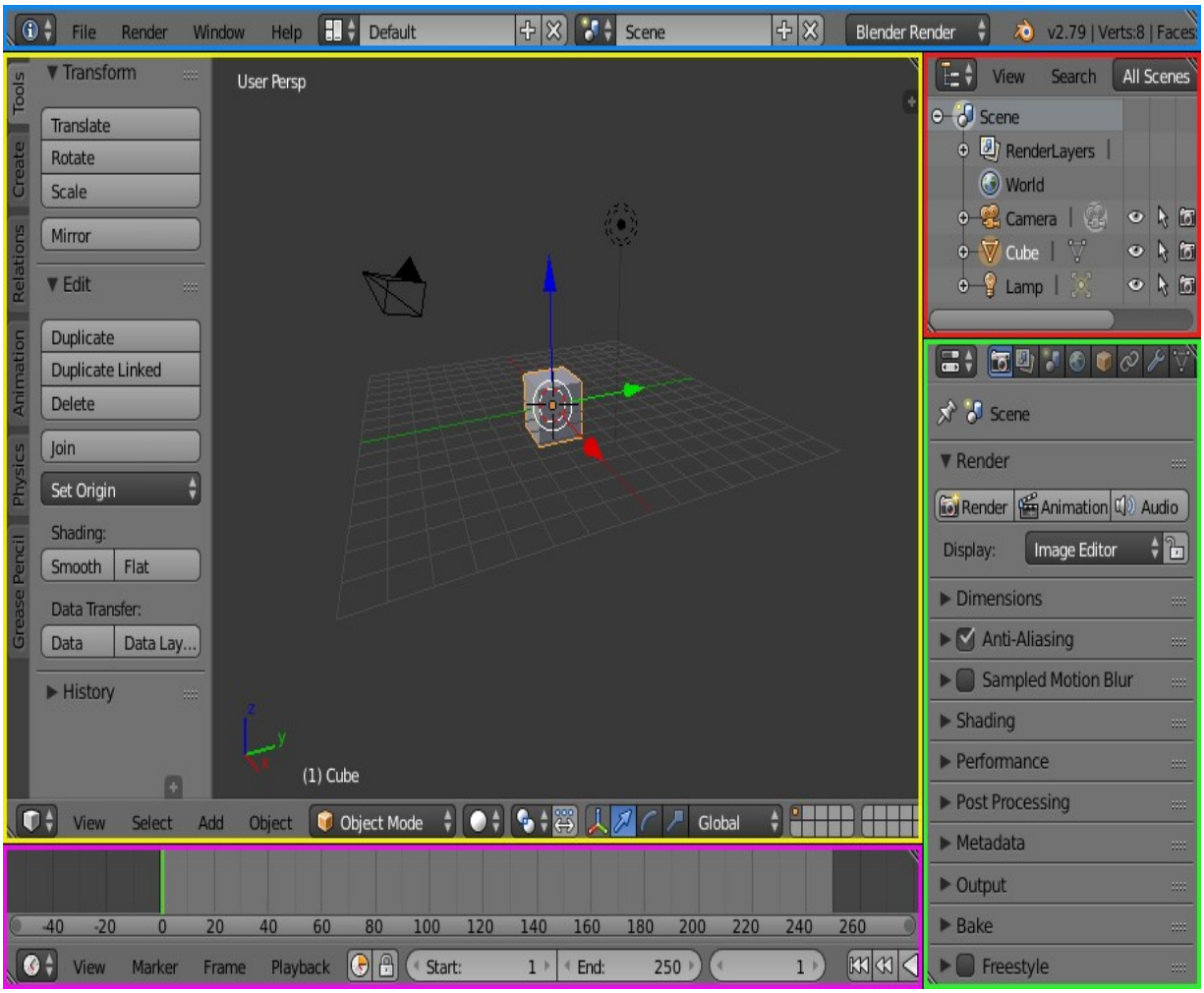

Рисунок 2.8 - Стандартний формат екрана Blender з п’ятьма редакторами

2.4.2 Компоненти редактора

В цілому, кожен з редакторів Blender'a дозволяє використовувати певний інструментарій та переглядати певні параметри об'єктів сцени. Редактори розділені на ділянки редактора. Регіони містять менші структурні елементи, такі як вкладки і панелі з кнопками, різні елементи управління і віджети.

- Transform розташування об'єкта у редакторі,
- Duplicate зброти копію обраного об'єкта,
- Scale змінити розмір об'єкта,
- Shading обрати тіні для об'єкта,

Компоненти редактора показані на рис. 2.9.

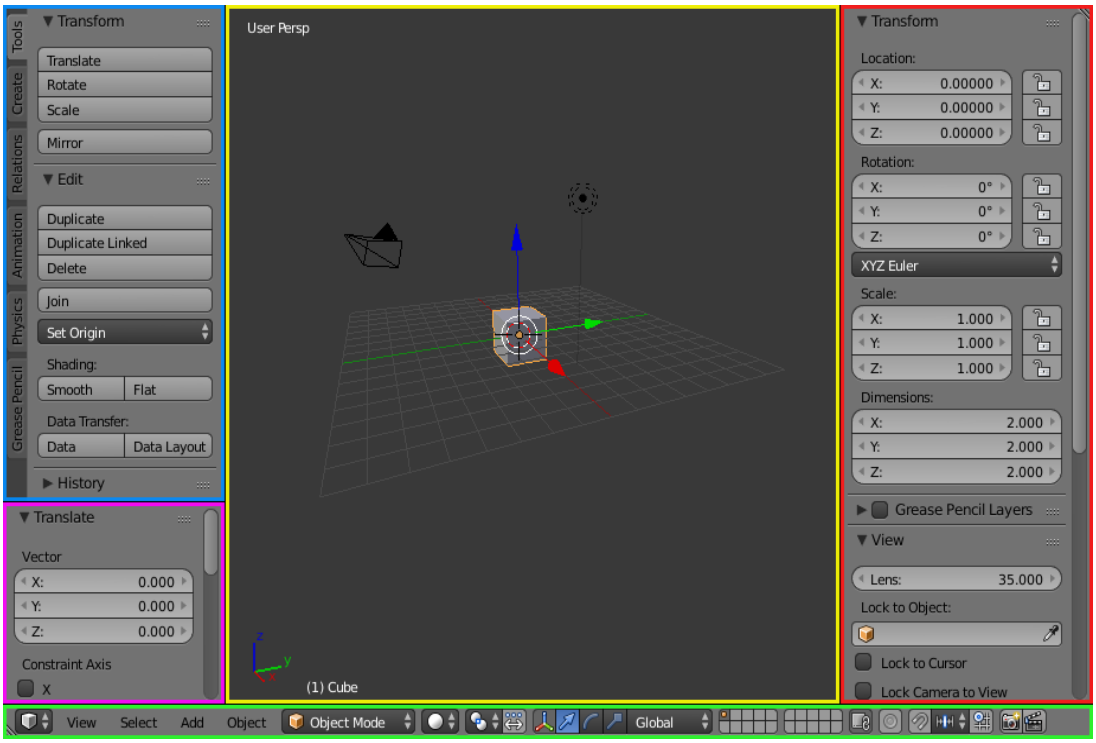

Рисунок 2.9 - Редактор 3D View (3D вид)

2.4.3 Концепції інтерфейсу Blender

Non-Overlapping - інтерфейс користувача Blender розроблений таким чином, щоб дозволити вам відразу побачити всі релевантні інструменти та параметри, без зайвих маніпуляцій з редакторами. Елементи інтерфейсу не перекривають один одного, а розташовані поруч.

Non-Blocking- інструменти і параметри інтерфейсу не перешкоджають використанню інших елементів Blender'a. Blender зазвичай не використовує спливаючі вікна, не блокує роботу інших частин, поки ви не заповните поля і так далі [17].

2.4.4 Об'єкти в Blender

Основні об'єкти, такі як куб, сфера, площину, і багато інших, передбачені в Blender. Ці об'єкти є примітивами, змінюючи які можна отримувати інші, більш складні об'єкти.

Якщо навести курсор миші в 3D вікно, а потім натиснути пробіл, то з'явиться контекстне меню, що містить список різних дій. Перший пункт в меню - Add (додати) - містить список об'єктів і підміню об'єктів, наявних в Blender.

Контекстне меню зображене на рис. 2.10.

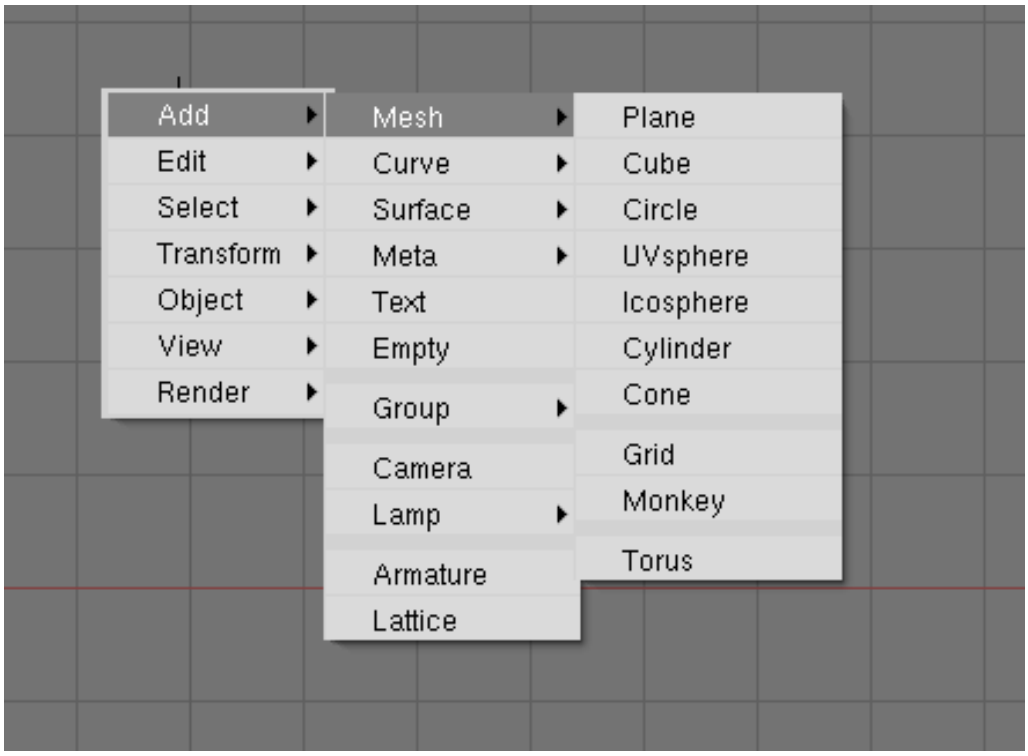

Рисунок 2.10 - Контексте меню що містить список різних дій

Найперший пункт - це так звані меш-об'єкти (Mesh), куди входять площина (Plane), куб (Cub), коло (Circle), сфера (UVsphere), геосфера (Ico-sphere), циліндр (Cylinder), туба (Tube) , конус (Cone), сітка (Grid) і голова мавпи (Monkey) та інші. Дані об'єкти складаються з більш дрібних елементів: вершин, ребер, граней, які формують кінцевий об'єкт. Так геосфера відрізняється від сфери тим, що сформована з трикутників, а не чотирикутників. Дані відмінності мають значення при подальшому редагуванні об'єктів [18].

2.4.5 Додавання об'єктів. Режими об'єктний і редагування

При додаванні нового об'єкта слід мати на увазі, що він розташується там, де знаходиться 3D курсор. Щоб поміняти положення 3D курсора, досить клацнути лівою клавішею миші в обраному місці.

Об'єкти додаються на сцену в режимі редагування. Зокрема, для Mesh-об'єктів це означає, що можна редагувати їх складові частини (вершини, ребра, грані), змінювати їх положення, розмір, кут повороту (природно два останні варіанти для вершин неможливі). Щоб з режиму редагування переключитися в об'єктний режим (коли будь-які зміни застосовуються до всього об'єкта), слід натиснути клавішу Tab. Повторне натискання цієї кнопки знову поверне в режим редагування.

Примітивний об’єкт показаний на рис. 2.11.

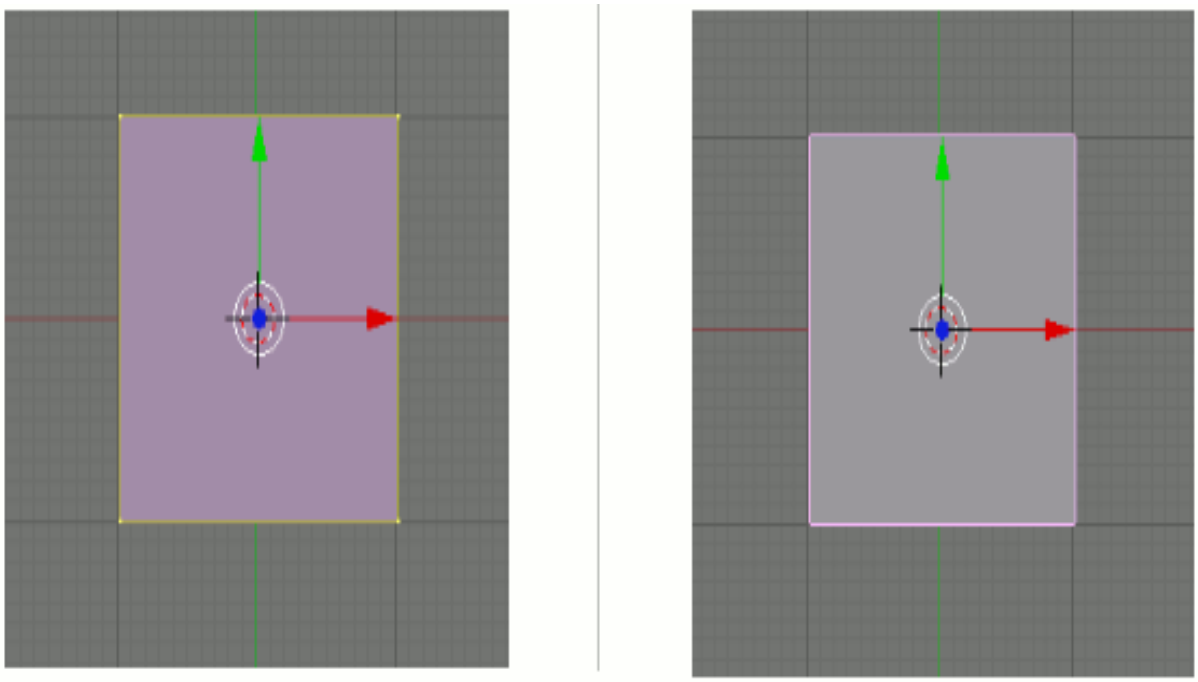

Рисунок 2.11 - Створений в режимі редагування об’єкт

2.4.6 Редагування вершин, ребер і граней

Зміни складових частин об'єкта. Такі зміни можливі лише в режимі редагування. Здійснюються вони за допомогою кнопок меню 3D вікна або за допомогою клавіш G, S, R.

Після створення об'єкта у нього виділені всі частини (в такому стані вони підсвічені жовтим кольором). Якщо зняти виділення (клавіша A) і спробувати виділити будь-якої окремо взятий елемент, то можно виділити або тільки вершини, або ребра, або межі, в залежності від того, який режим включений в даним момент. Кнопки перемикання даних режимів знаходяться в тому ж меню 3D вікна, що і кнопки зміни положення, розміру і повороту.

Після виділення необхідного елемента, його можна пересувати, а в разі ребер і граней ще й змінювати розмір і повертати [19].

Редатування вершин, ребер і граней показане на рис. 2.12.

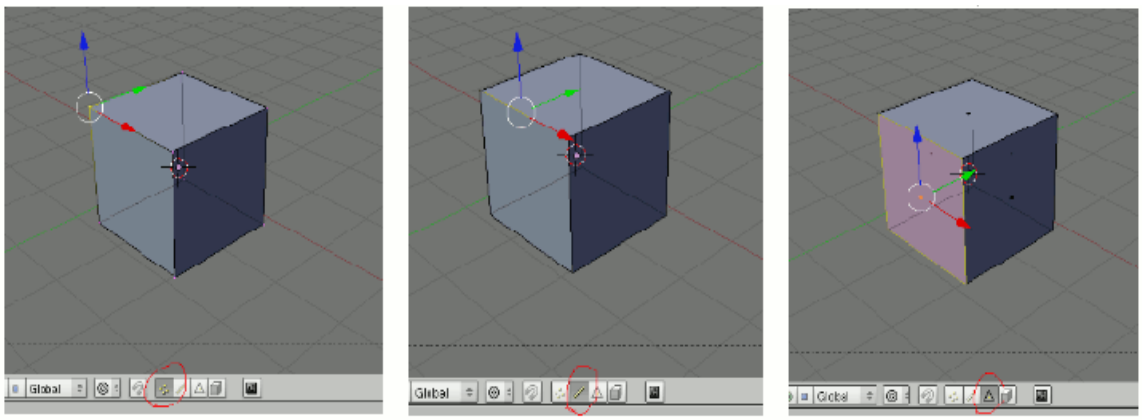

Рисунок 2.12 - Три режими редагування об’єктів

3 РОЗРОБКА ОБ'ЄКТІВ НАВКОЛИШНЬОГО СЕРЕДОВИЩА ІНТЕРАКТИВНОЇ МОДЕЛІ У UNITY ТА BLENDER

3.1 Збірка сцен. Gameobjects і Components

Ігрові об'єкти (GameObjects) - це найважливіші об'єкти в Unity. Дуже важливо розуміти, що таке GameObject і як його використовувати.

Кожен об'єкт в грі - це GameObject. Однак, GameObject'и нічого не роблять самі по собі. Вони вимагають спеціальної настройки, перш ніж стати персонажами, предметами оточенням або спеціальними ефектами.

GameObject'и є контейнерами. Порожня коробка, яка може містити всередині різні елементи, такі як острів з запеченими тінями або фізично коректний автомобіль. Щоб дійсно зрозуміти GameObject'и, потрібно зрозуміти їх складові, який називаються компонентами (Components). Залежно від того, який ми хочемо створити об'єкт, ми будемо додавати різні комбінації компонентів до GameObject'у.

Відкриймо будь-яку сцену Unity, створімо новий GameObject (використовуючи Shift-Control-N в Windows або Shift-Command-N в Mac), виберімо його і поглянимо в інспектор (Inspector) рис. 3.1.

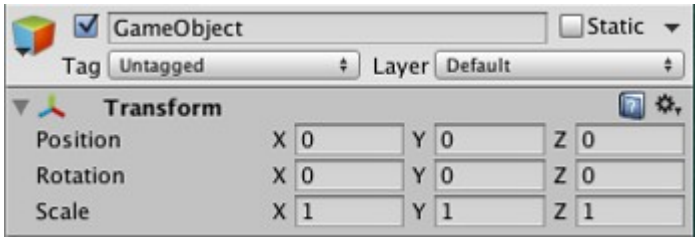

Рисунок 3.1 - Інспектор порожнього GameObject'а

Звернімо увагу, що навіть порожні GameObject'и мають ім'я, тег (Tag), і шар (Layer). Кожен GameObject також містить компонент Transform.

У Unity неможливо створити GameObject без компонента Transform. Компонент Transform - один з найважливіших компонентів, так як всі властивості GameObject'а пов'язані з трансформаціями використовують цей компонент. Він визначає положення, обертання і масштаб GameObject'а в ігровому світі / вікні Scene. Якщо GameObject не матиме компонента Transform, він буде не більше ніж деякою інформацією в пам'яті комп'ютера. Він не зможе ефективно існувати в ігровому світі [20].

Компонент Transform важливий для всіх GameObject'ов, тому він є у кожного GameObject. Але GameObject'и можуть містити й інші компоненти (рис. 3.2).

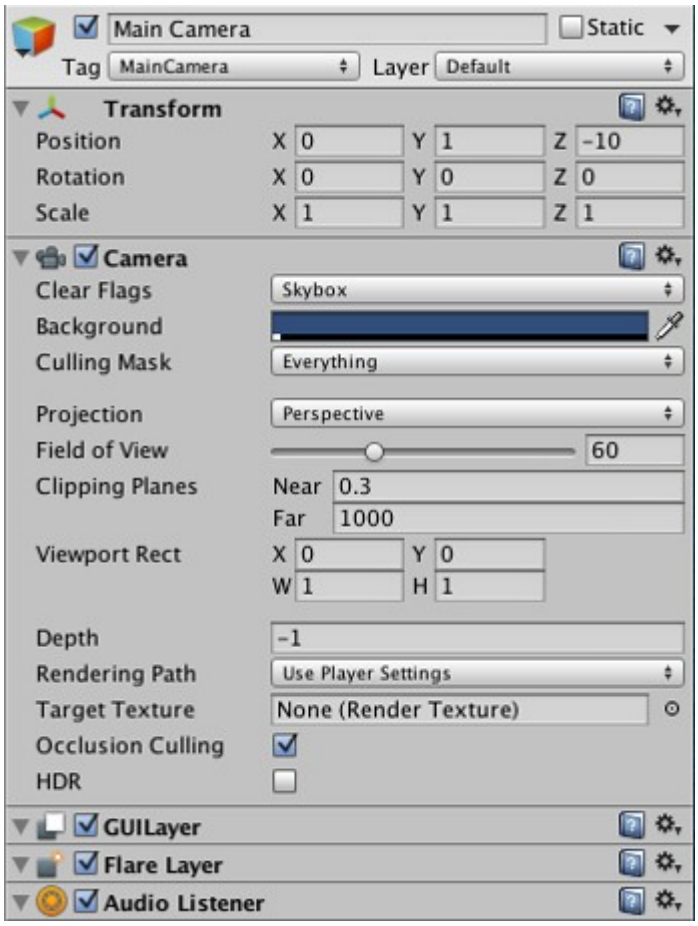

Рисунок 3.2 - GameObject під назвою Main Camera, який додається в кожну нову сцену за замовчуванням

Поглянувши на GameObject Main Camera, ми можемо побачити, що він містить різні колекції компонентів. Особливо, компонент Camera, GUILayer, Flare Layer, і Audio Listener. Всі ці компоненти надають додаткову функціональність GameObject'у. Без них не було б кому займатися рендерингом ігрової графіки для граючої людини! Тверді тіла, колайдери, частинки, аудіо - це все різні компоненти (або комбінації компонентів), які можуть бути додані до будь-якого GameObject'у.

Однією з чудових особливостей компонентів є гнучкість. При підключенні компонента до ігрового об'єкту, існують різні значення або властивості (Properties) в компоненті, які можуть змінюватися в редакторі

при створенні гри або через скрипти в запущеній грі. Є два основних типи властивостей: значення (Values) і посилання (References).

Подивимося на рис. 3.3, це порожній Ігровий Об'єкт з компонентом Audio Source. Всі параметри компонента Audio Source в Інспектора виставлені за замовчуванням [21].

Рисунок 3.3 - порожній Ігровий Об'єкт з компонентом Audio Source

Компонент містить одну властивість-посилання і сім властивостей-значень. Audio Clip - це властивість-посилання. Коли це аудіо джерело починає грати, він буде намагатися програти файл, на який посилається властивість Audio Clip. Якщо такого посилання не виявиться, то виникне помилка, так як ніяке аудіо НЕ буде програно. Ми повинні призначити файл в інспектор. Перетягнемо файл з Project View на властивість-посилання або за допомогою вибору об'єкта (Object Selector).

Всі інші властивості після Audio Clip це властивості-значення. Вони можуть бути відрегульовані прямо в інспектор. Властивості значення після

Audio Clip - це все перемикачі, числові значення і випадаючі меню, але властивості-значення можуть також бути текстовими рядками, кольорами, кривими і іншими типами.

3.1.1 Публікація збірок

У будь-який момент розробки гри можна захотіти подивитися на те, як вона виглядає поза редактора, при складанні в якості самостійного додатка або веб-програвача.

Пункт меню File-> Build Settings ... дозволяє відкрити вікно Build Settings. У ньому виводиться редагований список сцен для включення в збірку гри (рис. 3.4).

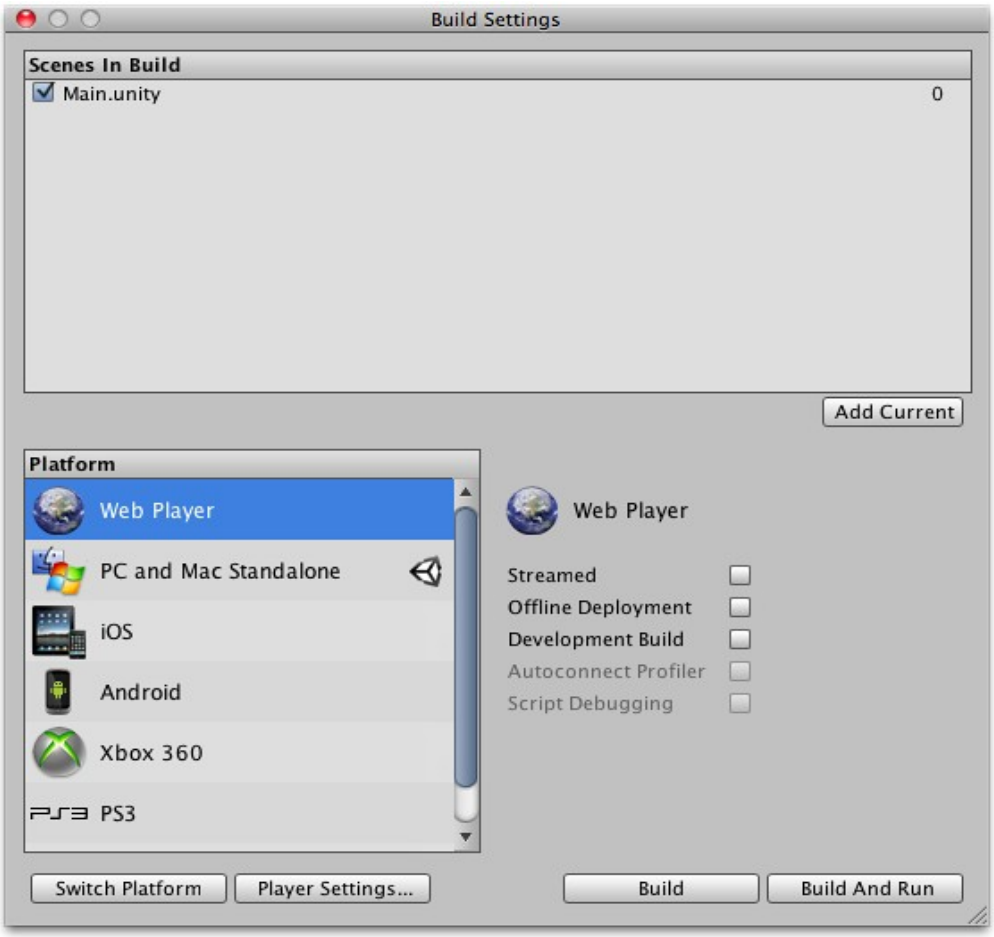

Рисунок 3.4 -Build Setting у якому можно скомпілювати наш проект

Список буде порожній при першому відкритті цього вікна в проекті. В такому випадку, при складанні, в гру буде включена лише поточна відкрита сцена.

Коли будемо готові до публікації своєї збірки, вибераємо потрібну платформу в списку Platform, навпроти неї знаходиться логотип Unity; якщо це не так, тоді натиснимо кнопку Switch Platform, щоб повідомити Unity про те, під яку платформу ми бажаємо здійснювати збірку. Після цього, натиснимо кнопку Build. Відкриється стандартне діалогове вікно збереження файлу, в якому ми можемо вибрати ім'я і розташування для гри. Після натискання кнопки "Зберегти" Unity збере ваш додаток [22].

3.1.2 Редагування властивостей

Properties (властивості) - параметри і опції компонентів, які можна редагувати в інспекторі (рис. 3.5).

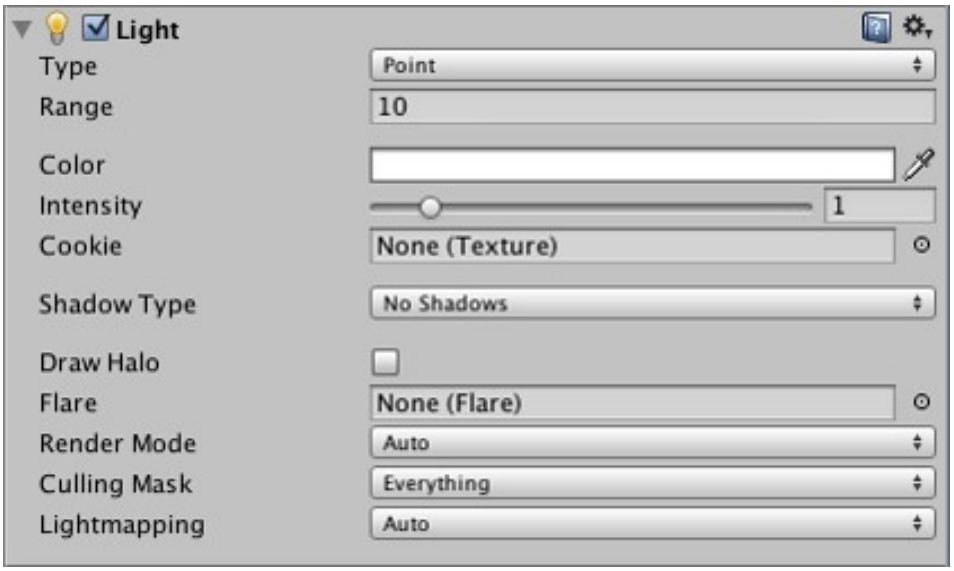

Рисунок 3.5 - Компонент Light відображає різні значення і властивості-посилання

Властивості можна широко розділити на категорії "посилання" (зв'язки з іншими об'єктами і Ассетами) або величини (числа, прапорці, кольори і т.д.).

Посилання можуть бути призначені шляхом перетягування об'єкта або Ассета відповідного типу на відповідний пункт в інспекторі. Наприклад, компоненту Mesh Filter треба посилатися на Ассет Mesh десь в цьому проекті. Але коли компонент тільки створено, посилання ще не призначене (рис. 3.6):

Рисунок 3.6 - Посилання не призначене

але ми можемо призначити йому Mesh шляхом перетягування Ассет-міша на нього (рис. 3.7):

Рисунок 3.7 - Посилання призначене

Ми також можемо використовувати Object Picker, щоб вибрати об'єкт і зв'язати його з властивістю. Якщо ми клацнімо на значок невеликого кола в правій частині властивості в інспекторі, ми побачимо подібне вікно (рис. 3.8):

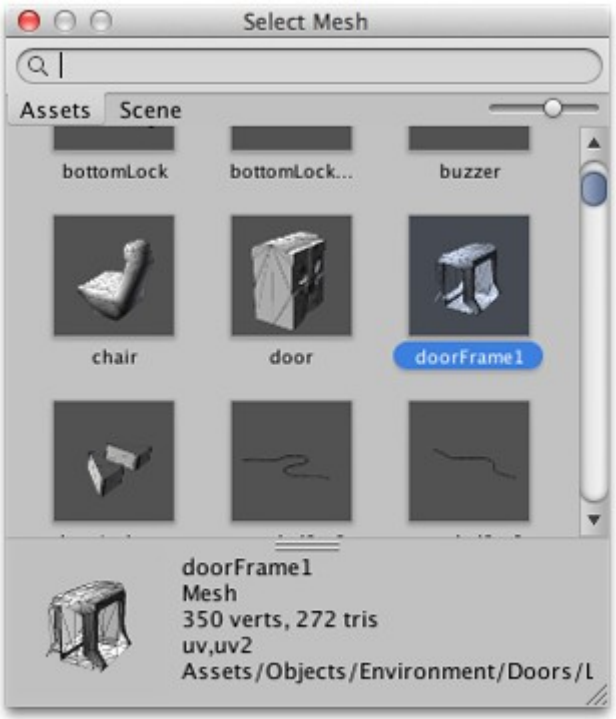

Рисунок 3.8 - Object Picker для пошуку об'єкта в сцені або ассетах

Object picker (вибір об'єкта) дозволяє шукати и вібіраті об'єкти в сцені або в Ассет проекту (інформаційні панелі в Нижній части вікна можна підніматі и опускаті за Бажанов). Для Вибори об'єкта на властівість-ПОСИЛАННЯ, просто необходимо двічі натіснуті на него в Object Picker.

Коли властівість-ПОСИЛАННЯ відноситься до компоненту типу (например Transform), Ми можемо помістити в Цю властівість будь-який об'єкт; Unity Визначіть перший компонент цього типу зустрічаючійся в цьом об'єкті и призначила на ПОСИЛАННЯ. Если в об'єкта НЕ виявило компонентів відповідного типу, призначення буде перервати.

При кліку по колірної Властивості, відкріється Палітра кольорів (Color Picker) рис. 3.9.

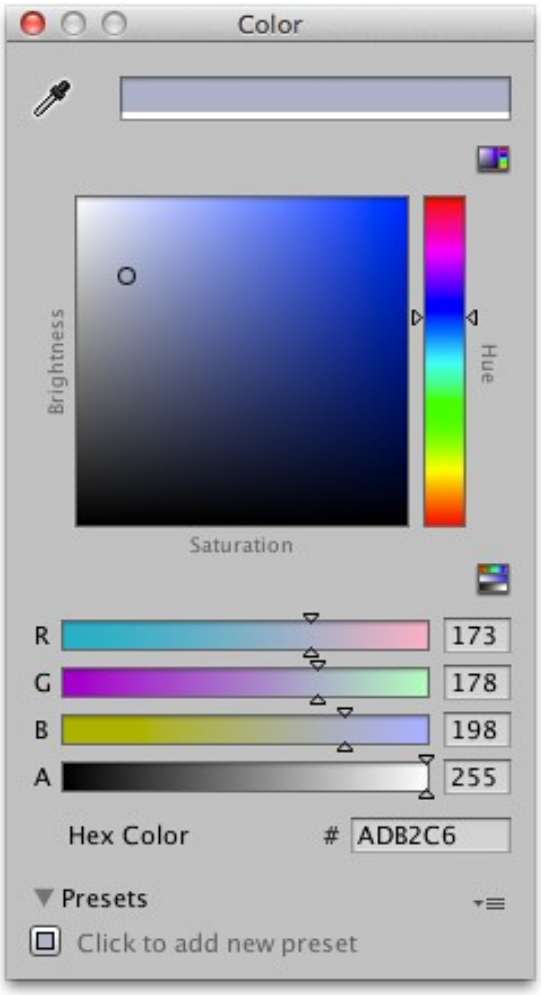

Рисунок 3.9 - Вікно палітри кольорів

Unity використовує власну систему кольорової палітри, але на Mac OS X ми можемо вибрати системну систему з меню Preferences (меню: Unity> Preferences і потім Use OS X Color Picker з панелі General).

3.2 Використання вікна Scene View

Вікно Scene View - це інтерактивна пісочниця. Будемо використовувати Scene View для вибору і розташування оточень, гравця, камери, ворогів, і всіх інших ігрових об'єктів. Управління та маніпулювання об'єктами з використанням вікна Scene View є однією з найбільш важливих функцій Unity, тому важливо вміти робити це швидко.

3.2.1 Навігація у вікні Scene

Вікно Scene має набір навігаційних елементів управління, які допомогають орієнтуватися в ньому швидко і ефективно (рис. 3.10).

Для переміщення по сцені можна використовувати клавіші зі стрілками. Клавіші вгору і вниз переміщують камеру вперед і назад в тому напрямку, в яке вона дивиться. Клавіші вліво або вправо повертають камеру в сторони. При використанні стрілок, затиснимо клавішу Shift для прискореного переміщення.

Якщо вибрати геймоб'єкт в ієрархії, потім помістити миш над Scene View, і натиснути Shift + F, вид концентрується на вибраному об'єкті. Ця можливість називається кадрувати виділене (frame selection).

Рисунок 3.10 - Елементи фокусування

Переміщення, обертання по орбіті і масштабування - ключові операції навігації у вікні Scene View, тому Unity надає кілька альтернативних шляхів їх виконання для максимальної зручності [23].

3.2.2 Позиціонування ігрових об'єктів

Під час створення гри, я повинен розмістити багато різних об'єктів у ігровому світі. Для цього будуть використані інструменти трансформацій в панелі інструментів для переміщення, обертання і масштабування окремих об'єктів. Кожен інструмент має відповідне Гизмо, яке з'являється навколо виділеного ігрового об'єкта у вікні Scene. Можна використовувати миш і маніпулювати будь якою віссю Гизмо для зміни компонента Transform

ігрового об'єкта, або можна ввести значення безпосередньо в числові поля компонента в інспекторі.

Кожен з трьох режимів може бути обраний гарячими клавішами - W для переміщення, E для обертання і R для масштабування (рис. 3.11).

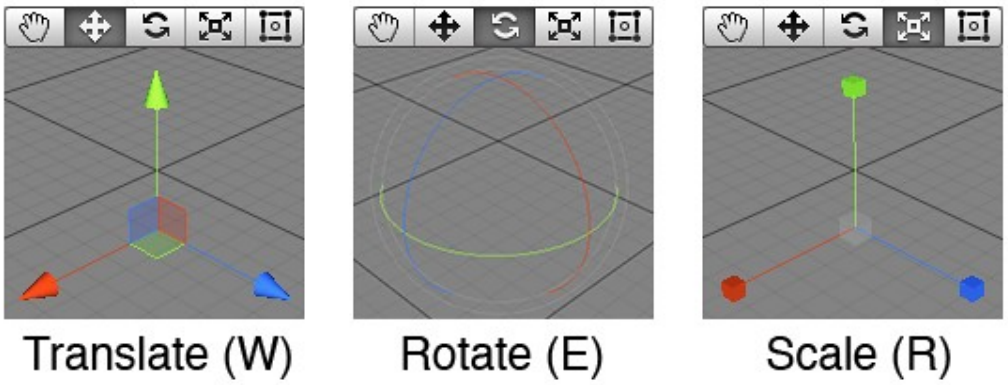

Рисунок 3.11 - Переміщення, обертання і масштабування

3.2.3 Пошук в Scene

При роботі з великими і складними сценами буде корисний пошук певних об'єктів. Використовуючи можливість Search в Unity, можна відфільтрувати об'єкт або групу об'єктів. Можна шукати Ассет по імені, по типу компонента, і, в деяких випадках, по мітках Ассет. Можна вибрати режим пошуку з випадаючого меню Search.

У вікнах Scene і Hierarchy є поле пошуку, що дозволяє фільтрувати об'єкти по імені. Так як ці вікна, по суті, просто два варіанти відображення одних і тих же об'єктів, будь-якій пошуковій запит буде дублюватися в обох полях пошуку і застосовуватися однаково до обох вікон.

Звернемо увагу, що коли пошук активний, обидва вікна трохи відрізняються: вікно Scene буде показувати об'єкти, які не пройшли по фільтру сірим, а у вікні Hierarchy будуть відображатися тільки ті об'єкти, імена яких відповідають пошуковому запиту (рис. 3.12):

Невеликий хрестик праворуч від поля пошуку видаляє пошукової запит і повертає вікно до нормального стану. В меню зліва від поля можна вибрати фільтр пошуку - по імені об'єкта, за його типу, або обом цим параметрам.

Текстове поле дозволить фільтрувати існуючі мітки або додавати нові. Коли друкуєте, ви можете натиснути клавіші пробіл або ентер для додавання

Ассет новий тег. Мітки, призначені Ассет відображаються галочкою в лівому меню

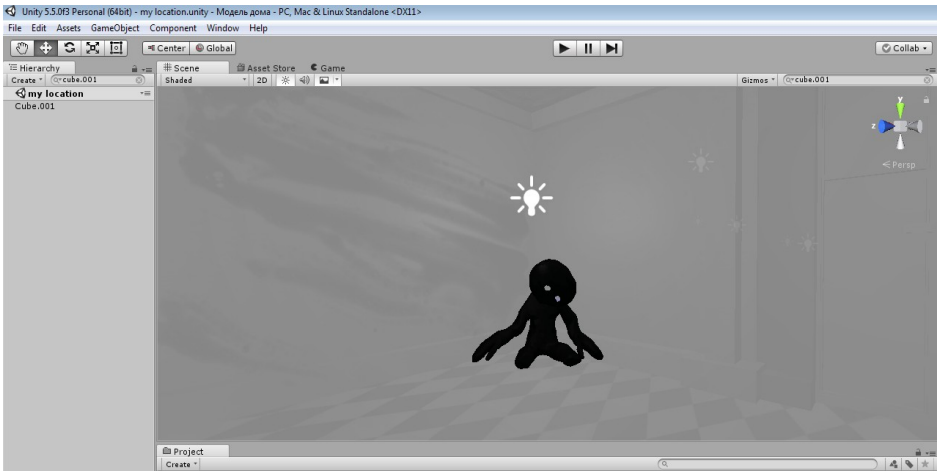

Рисунок 3.12 - Пошук певного об'єкта у проекті

3.3 Префаб (Prefabs)

Досить зручно працювати з GameObject в сцені, додаючи компоненти і змінюючи їх значення на потрібні у інспекторі. Однак, це може створити ряд проблем в таких випадках, коли працюємо над створенням NPC, об'єктом або предметом, який часто зустрічається в сцені. Звичайно, можна просто скопіювати ці об'єкти для створення дублікатів, але всі вони будуть редагуватися незалежно один від одного.

В Unity можна створювати префаб. Це особливий тип Ассета, що дозволяє зберігати весь GameObject з усіма компонентами і значеннями властивостей. Префаб виступає в ролі шаблону для створення екземплярів зберігаючого об'єкта в сцені. Будь-які зміни в префабі негайно відображаються і на всіх його примірниках, при цьому можна перевизначати компоненти і настройки для кожного екземпляра окремо. Коли перетягуємо файловий Ассет (наприклад, меш) в сцену, буде створений новий екземпляр такого об'єкта і всі такі екземпляри зміняться при зміні оригінального Ассета. Однак, хоч його поведінка і схожа, але Ассет - це не префаб, так що не вийде додати до нього компоненти або використовувати будь-які інші описані нижче властивості префабов.

Можна створити префаб, вибравши Asset> Create Prefab і перетягнувши об'єкт зі сцени в "порожній" префаб, що з'явився в проекті. Після чого можна створювати екземпляри префаба просто перетягуючи його з вікна Project на

сцену. Імена об'єктів реалізованих префабом, будуть підсвічуватися синім у вікні Hierarchy (імена звичайних об'єктів мають чорний колір).

Як уже згадувалося вище, зміни в префаб автоматично застосуються до всіх її екземплярів, однак можна змінювати і окремі екземпляри. Це корисно наприклад в разі, коли бажаємо створити кілька схожих NPC, але з зовнішніми відмінностями, щоб додати реалістичності. Щоб було чітко видно, що властивість в екземплярі префаб змінено, воно показується в інспекторі жирним шрифтом, якщо до примірника префаба доданий абсолютно новий компонент, то всі його властивості будуть написані жирним шрифтом (рис. 3.13).

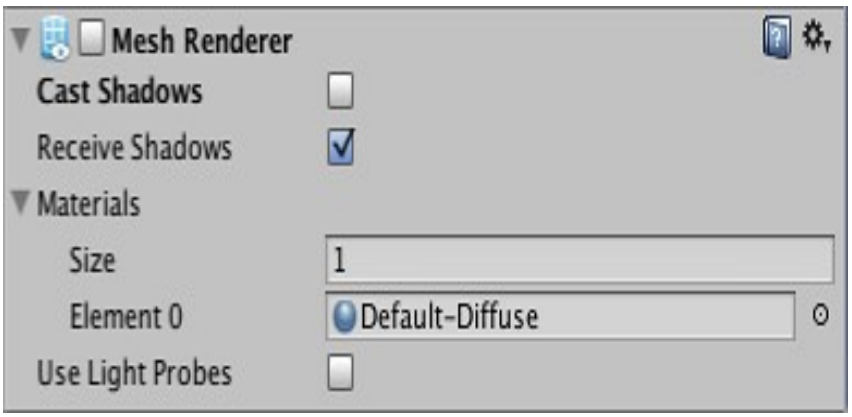

Рисунок 3.13 - Mesh Renderer на екземплярі префаба з перевизначеною властивістю "Cast Shadows"

3.4 Джерела світла

Джерела світла є невід'ємною частиною кожної сцени. У той час як меши і текстури визначають форму і зовнішній вигляд сцени, джерела світла визначають колір і атмосферу вашого 3D оточення. Організація їх одночасної роботи вимагає невеликої практики, але результат може бути приголомшливим (рис. 3.14).

Unity підтримує різні способи рендеринга (Rendering Paths). Ці способи зачіпають в основному джерела світла і тіні, тому вибір правильного способу рендеринга, в залежності від вимог гри, може поліпшити продуктивність проекту.

Якщо графічний адаптер не підтримує обраний спосіб рендеринга, Unity автоматично переключиться на менш вимогливий. Так що якщо GPU не підтримує відкладене освітлення (Deferred Lighting), буде використаний

попереджуючий рендеринг (Forward Rendering). Якщо і попереджуючий рендеринг не підтримується, тоді буде використано вершинний освітлення (Vertex Lit).

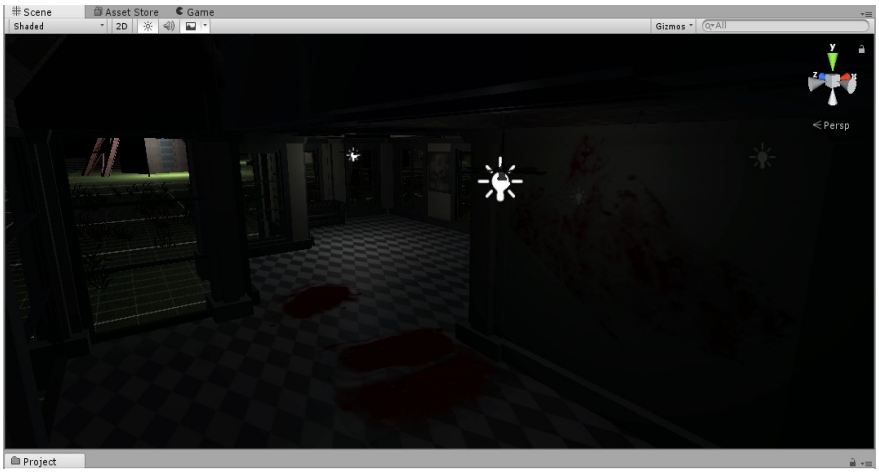

Рисунок 3.14 - Проста компоновка двох джерел світла

Джерела світла можуть бути додані в сцену з використанням меню GameObject-> Create Other. Після того, як джерело світла буде додано до сцени, можна керувати ним як будь-яким іншим GameObject'ом. Також, можна додати компонент Light до будь-якого виділеного об'єкту використовуючи Component-> Rendering-> Light (рис. 3.15).

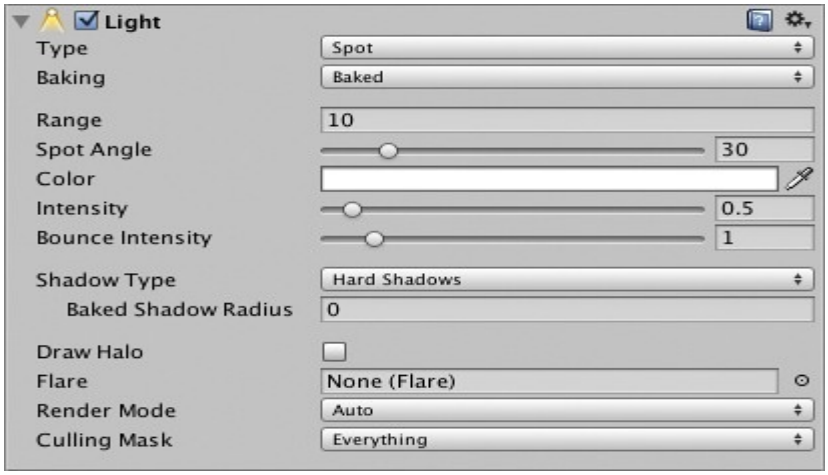

Рисунок 3.15 - Властивості компонента Light в інспектора

Просто змінюючи властивість Color (колір) джерела світла, можна додати сцені зовсім інший настрій (рис. 3.16).

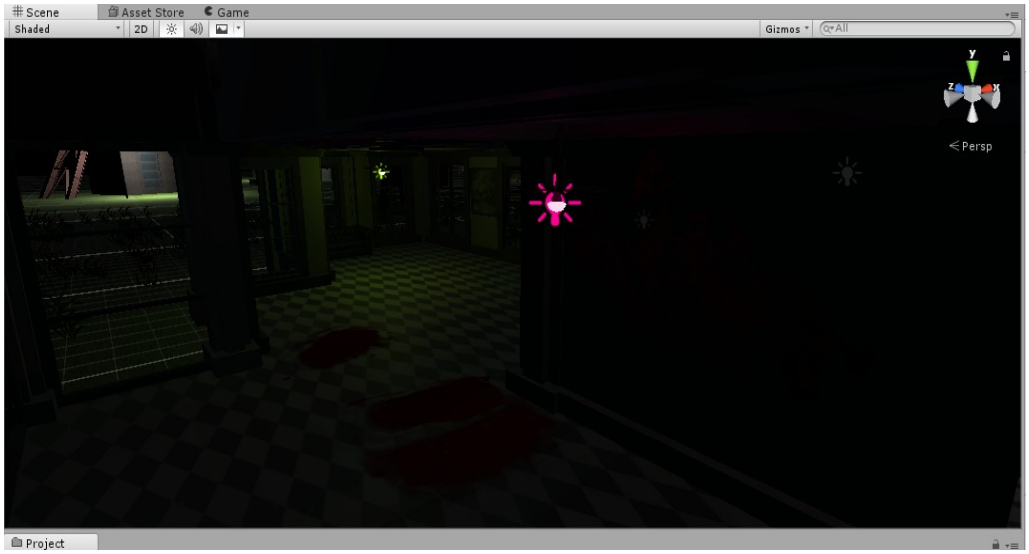

Рисунок 3.16 - Похмурі джерела світла

3.5 Створення та редагування terrain'ов

Можна додати об'єкт terrain'a в сцену, вибравши GameObject> Create Other> Terrain з меню, це також додасть відповідний Accet terrain'a в вікно Project (рис. 3.17). При цьому, ландшафт спочатку буде нічим іншим, як просто великий, плоскою рівниною. Однак, якщо подивитися на інспектор при виділеному об'єкті terrain'a, побачимо , що там представлений ряд інструментів, які можна використовувати для створення будь-яких потрібних ландшафтів.

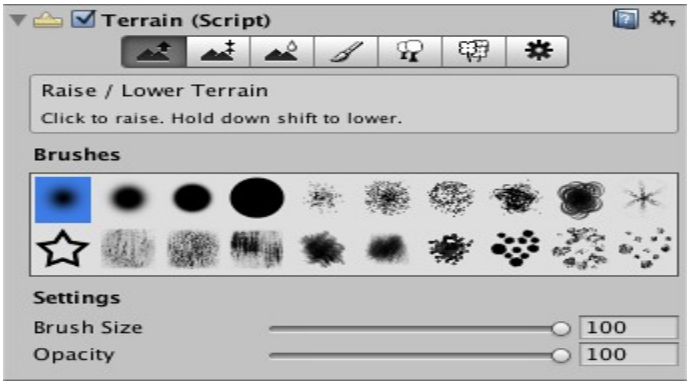

Рисунок 3.17 - Інструменти редагування terrain'a

За винятком інструменту розміщення дерев і панелі налаштувань, всі інші інструменти на панелі інструментів забезпечують набір "кистей" і налаштувань для розміру і прозорості кисті. Те що, ці кисті схожі на кисті з графічних редакторів - не випадково, тому що саме так створюються деталі у

terrain'a, за допомогою їх "малювання" на ландшафті. Якщо виберати перший зліва інструмент на панелі інструментів (Raise / Lower Terrain) і пересунути миш над terrain'ом у вікні Scene, ми побачимо, що курсор нагадує прожектор на поверхні. Клікнувши кнопку миші, можна намалювати поступові зміни в висоті ландшафту в точці розташування мишки. Можна змінювати форму кисті в панелі Brushes. Опції Brush Size і Opacity впливають на область кисті і силу "натиску" відповідно.

Перші три інструменти на панелі інструментів інспектора terrain'a використовуються для малювання змін висоти на terrain (рис. 3.18).

Рисунок 3.18 - Інструменти для змін висоти

Починаючи зліва, перша кнопка активує інструмент Raise / Lower Height (підвищити / знизити висоту). Коли малюємо використовуючи цей інструмент, висота буде збільшуватися поки керуємо мишкою з затиснутою лівою кнопкою по terrain'у. Висота буде підсумовуватися, якщо будемо утримувати курсор в одному місці, аналогічно ефекту аерографії в графічних редакторах. Якщо затиснути кнопку shift, висота буде знижуватися. Можна використовувати різні кисті для створення різних ефектів. Наприклад, можна створити горбисту місцевість за допомогою збільшення висоти пензлем з м'якими краями і потім вирізаючи круті скелі і рівнини за допомогою зниження висоти пензлем з гострими краями (рис. 3.19).

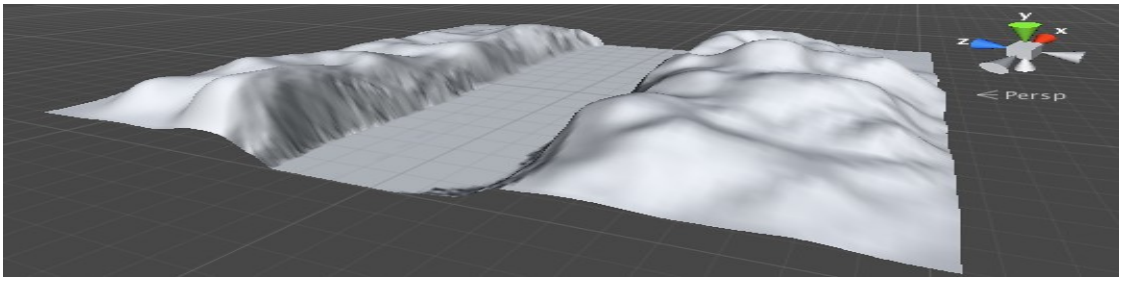

Рисунок 3.19 - Горбиста місцевість з різко виділеною рівнинною

Другий зліва інструмент, Paint Height, аналогічний інструменту Raise / Lower, тільки цього разу є додаткова властивість для установки цільової висоти. Коли малюємо на об'єкті, terrain занижуватиметься в областях вище цієї висоти і завищуватиметься в областях нижче цільової висоти. Можна використовувати слайдер властивості Height для установки висоти вручну,

або клікати по terrain'у з затиснутою клавішею shift для взяття зразка висоти в поточній позиції мишки (аналогічно "піпетке" в графічних редакторах). Поряд з потрібними Height є кнопка Flatten, яка просто встановлює висоту у всього terrain'а в задане значення. Це зручно для підняття рівня землі, наприклад, якщо бажаємо щоб ландшафт включав як пагорби вище цього рівня, так і рівнини нижче нього. Paint Height зручний для створення плато в сцені і також для додавання штучних елементів, таких як дороги, платформи або ступені (рис. 3.20).

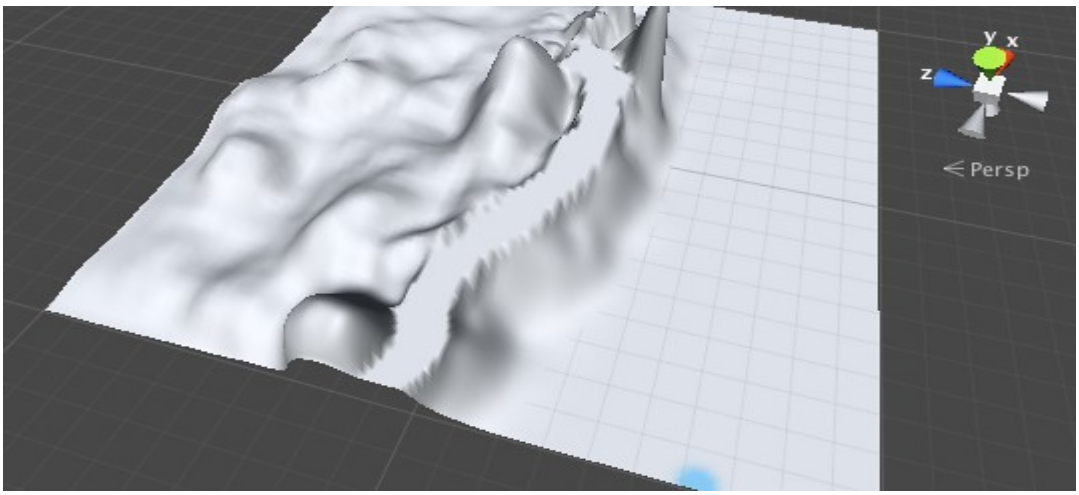

Рисунок 3.20 - Схил пагорба з плоскою дорогою

Третій зліва інструмент, Smooth Height, не піднімає або опускає значно висоту terrain'а, а скоріше, усереднює сусідні області. Це пом'якшує ландшафт і зменшує поява різких змін, щось на зразок інструменту розмиття в графічному редакторі.

3.6 Текстури

Можна додавати текстури на поверхню terrain'а для створення забарвлення і дрібних деталей. Так як terrain'и досить великі об'єкти, до них зазвичай застосовують текстури, які можна бесшовно стикувати, щоб замостити ними поверхню (повтор зазвичай не помітний з точи зору персонажа, який знаходиться близько до землі). Одна текстура буде служити як "фонова" картинка по всьому ландшафту, але також можна малювати області, використовуючи інші текстури, щоб симулювати різні покриття, такі як трава, пустеля і сніг (рис. 3.21). Намальовані текстури можуть бути

застосовані з різною прозорістю, так, щоб вийшов плавний перехід між трав'янистої місцевістю і піщаним пляжем, наприклад.

Спочатку у terrain'a НЕ буде текстур, призначених для малювання. Якщо клацнути кнопку Edit Textures і з меню вибрати Add Texture (рис. 3.22), ми побачимо вікно, в якому можна вказати текстуру і її властивості[24].

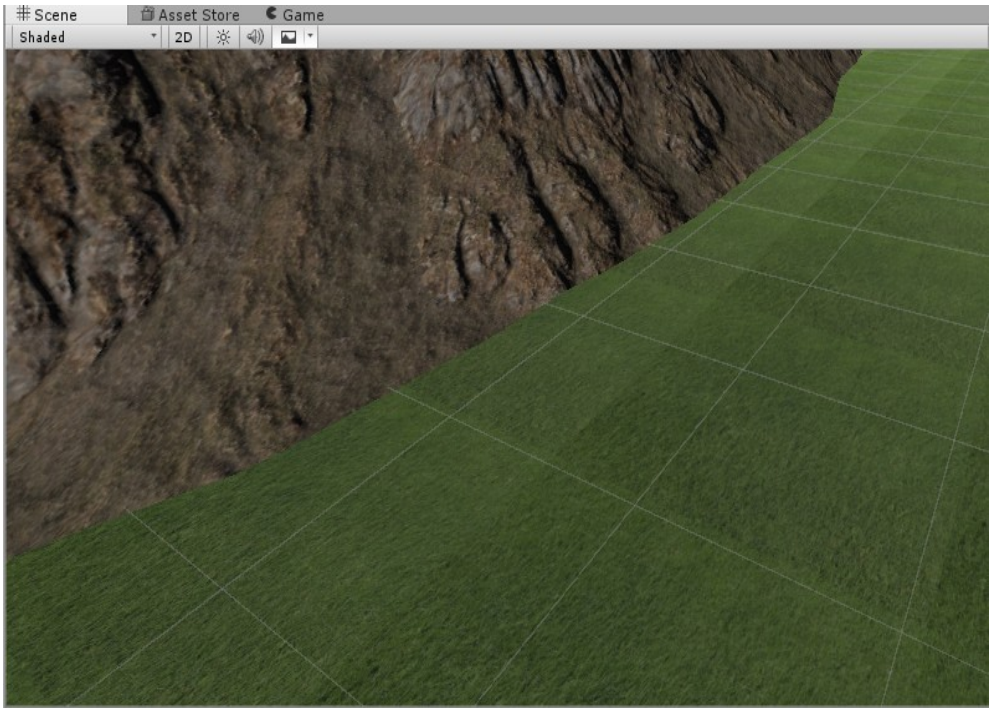

Рисунок 3.21 - Terrain с зеленою текстурою на фоні гір

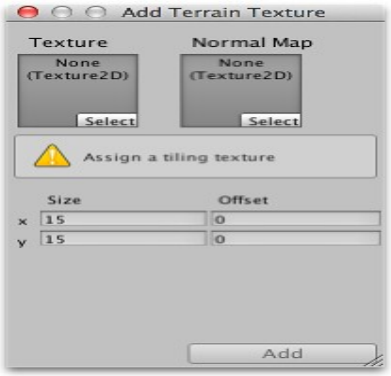

Рисунок 3.22 - Вікно Add Texture

3.7 Створення дерев

Terrain'и в Unity можуть бути доповнені деревами. Острівці з деревами можна малювати прямо на terrain'е, аналогічно малювання карт висот і текстур, але дерева - повноцінні 3D об'єкти, які ростуть з поверхні. Unity

використовує оптимізації (наприклад, перетворює вилучені дерева в спрайт) для збереження гарної продуктивності рендеринга, так що можна мати щільно засаджені ліси з тисячами дерев, і зберігати при цьому прийнятну частоту кадрів.

У Unity є свій інструмент для створення дерев (Tree creator), який можна використовувати для створення нових Ассет дерев (рис. 3.23), але також можна використовувати і стандартні програми для 3D моделювання для цього завдання. Меш дерева повинен мати щонайменше 2000 трикутників (для підвищення продуктивності) і центр обертання повинен бути розташований прямо в підставі дерева, там, де воно стикується з землею. Меш завжди повинен мати рівно два матеріали, один для стовбура і гілок, інший для листя.

Дерева повинні використовувати шейдери Nature / Soft Occlusion Leaves і Nature / Soft Occlusion Bark. Щоб використовувати ці шейдери, потрібно помістити дерево в спеціальну папку, яка містить в імені "Ambient-Occlusion". Коли помістімо модель в цю папку і повторно імпортуємо її, Unity прорахує м'яке розсіяне затінення спеціально створеним для дерев способом. Шейдери "Nature / Soft Occlusion" спираються на угоду про іменування папок, і дерево не отмалюється коректно, якщо не будемо його дотримуватися [25].

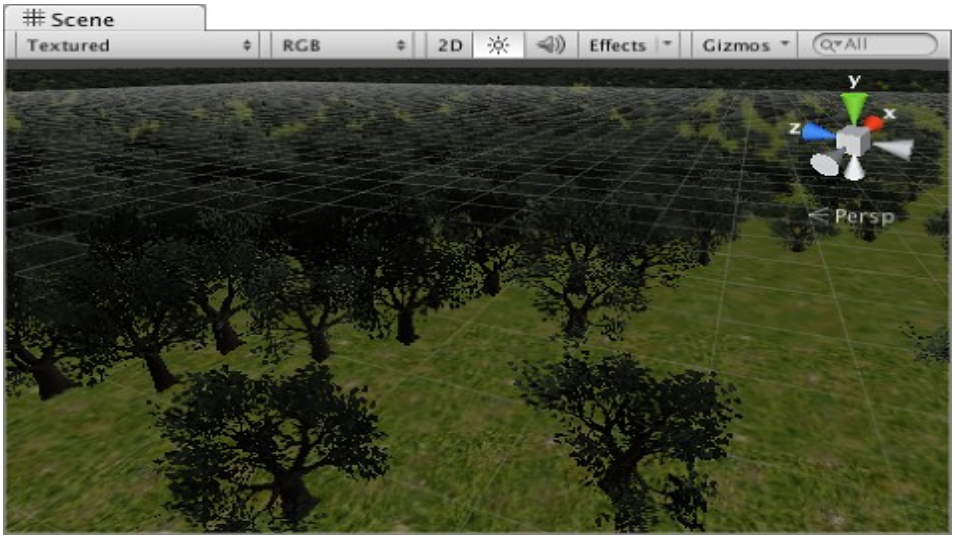

Рисунок 3.23 - Terrain з деревами

3.8 Створення моделі ракети за допомогою Blender

1. Створюю UV-сферу (я використовував для неї налаштування за замовчуванням).

2. Видаляю її нижню половину, і масштабую залишившийся верх по осі Z, щоб сформувати ніс. Також можна використовувати Пропорційне редагування для настройки зовнішнього вигляду (рис 3.24).

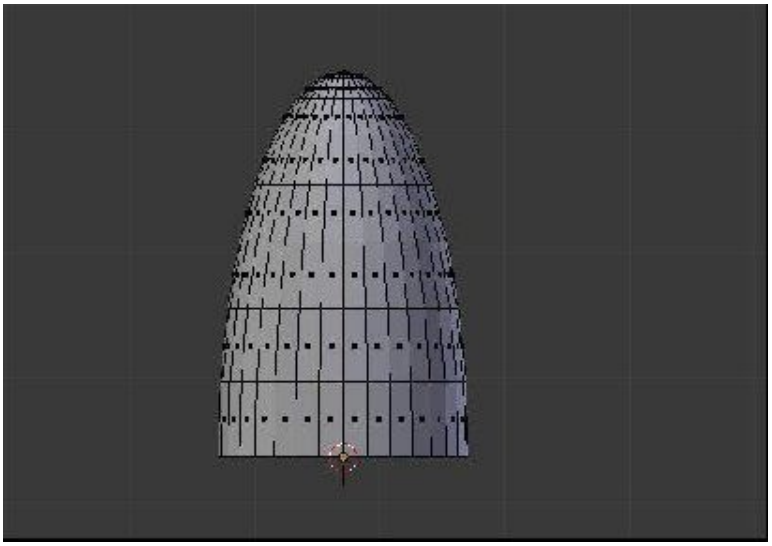

Рисунок 3.24 - Формування носу

3. Видавлюю ребра, щоб сформувати бажану довжину (рис. 3.25).

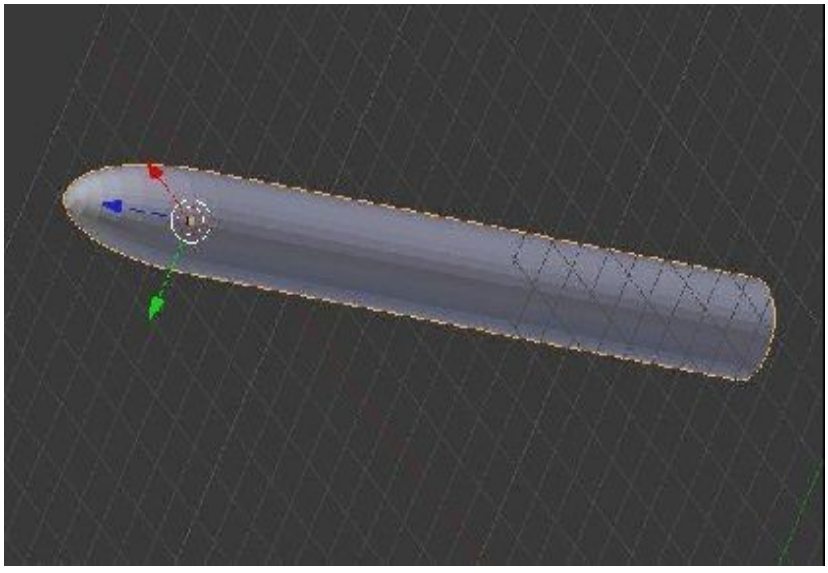

Рисунок 3.25 - Видавлення ребер

4. Наступна частина дуже суб'єктивна. Додаю реберні цикли і видавлюю деякі грані, щоб отримати форму, яка потрібна (рис. 3.26).

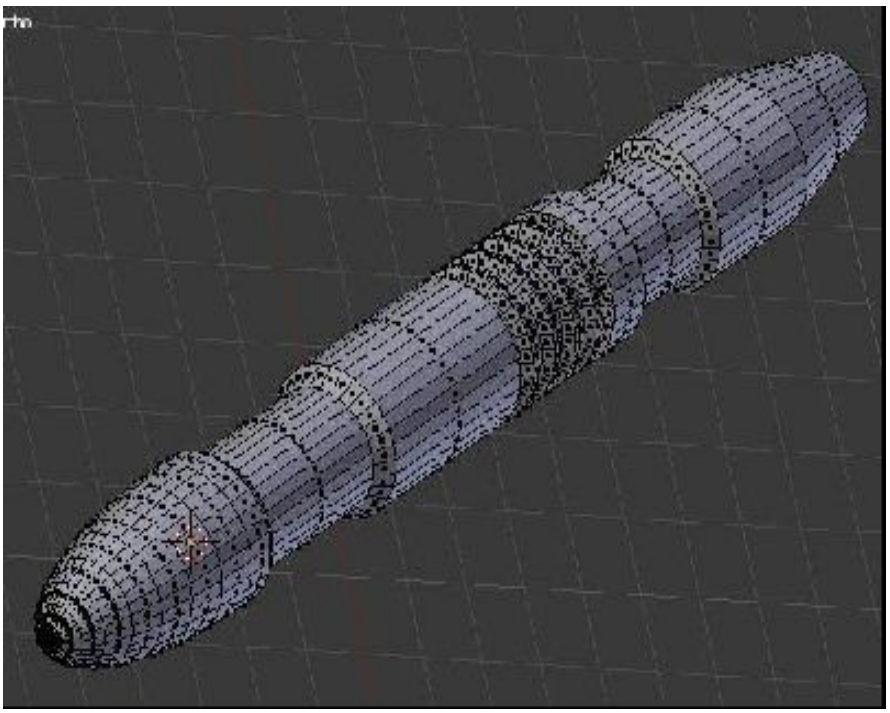

Рисунок 3.26 - Додавання реберних циклів

5. Вибираю дві бічні грані в одній із секцій циліндра (рис. 3.27).

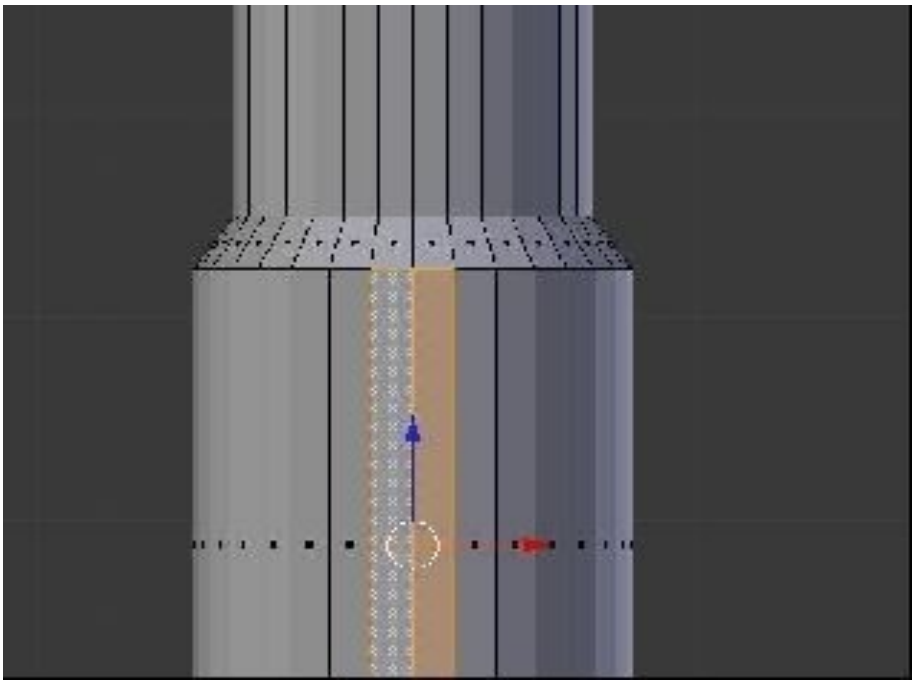

Рисунок 3.27 - Виділення граней

6. Дублікую їх і переміщаю назовні з корпусу (рис. 3.28).

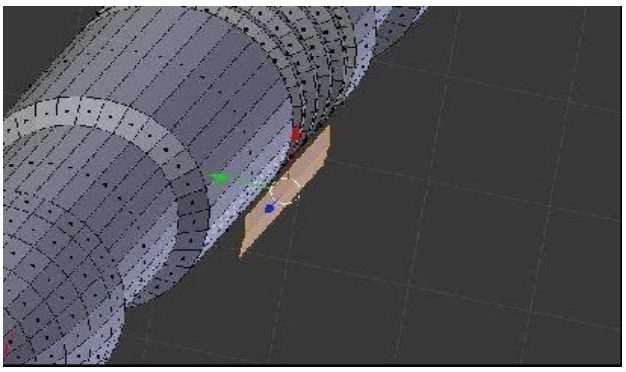

Рисунок 3.28 - Дублікат і переміщення

7. Видаляю середнє ребро ("X", "Петля ребер") (рис. 3.29).

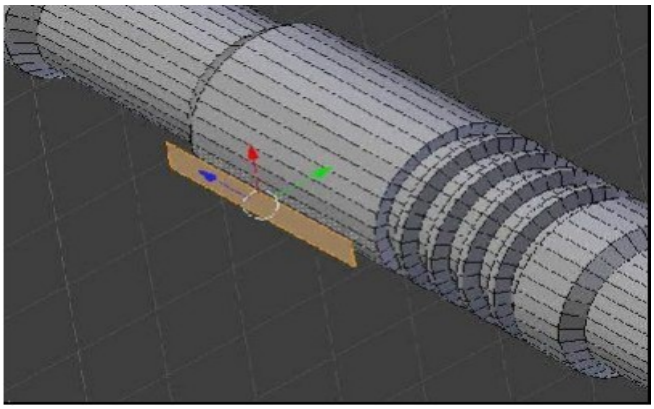

Рисунок 3.29 - Видалення ребра

8. Масштабую його, як потрібно, і переміщаю до кінчика стабілізатора. Потім видавлюю всередину (рис. 3.30).

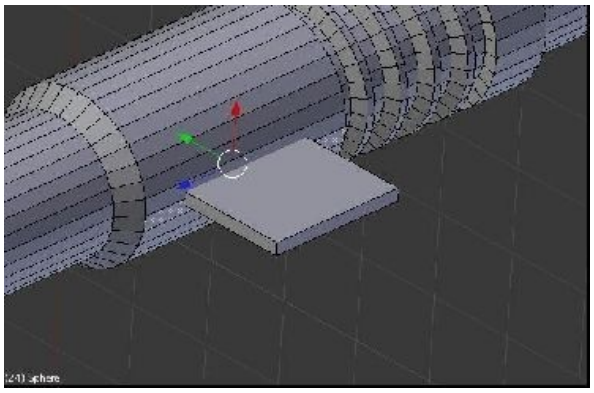

Рисунок 3.30 - Масштабування, переміщення та видавлення

9. Регулюю розміри і розташування граней, поки не з'явиться бажана форма стабілізатора.

10. Додаю декілька деталей (рис. 3.31).

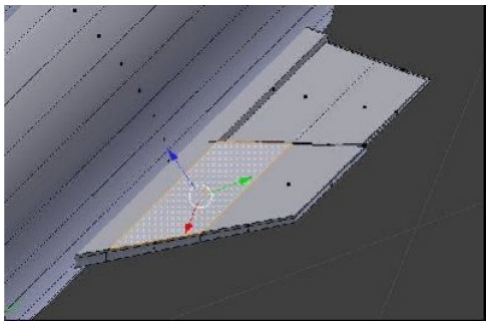

Рисунок 3.31 - Додавання дрібних деталей

11.Вигляд зверху ракети і обираю одне з кілець на носі (рис. 3.32).

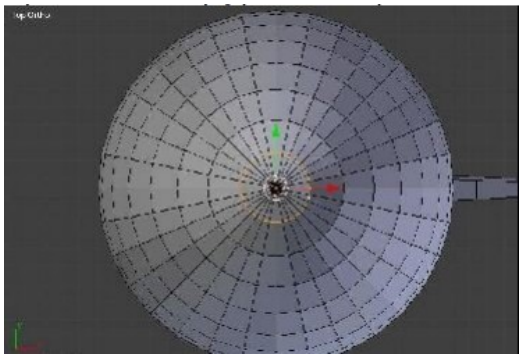

Рисунок 3.32 - Вигляд зверху

12. Акцентую на ньому 3D-курсор (SHIFT + S, "Курсор до виділення").
13. Скасовую вибір ("A"), вибираю всі грані стабілізатора (рис. 3.33).

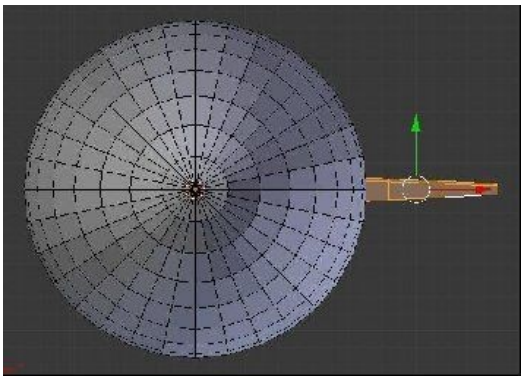

Рисунок 3.33 - Видаляю усі грані

14. Обираю 3D-Курсор внизу вікна 3D-виду.

15. Дублікую стабілізатор (SHIFT + D) і повертаю його на 180 градусів ("R","180").
16. Обираю обидва стабілізатора і повторюю поворот, тепер на 90 градусів (рис. 3.34).

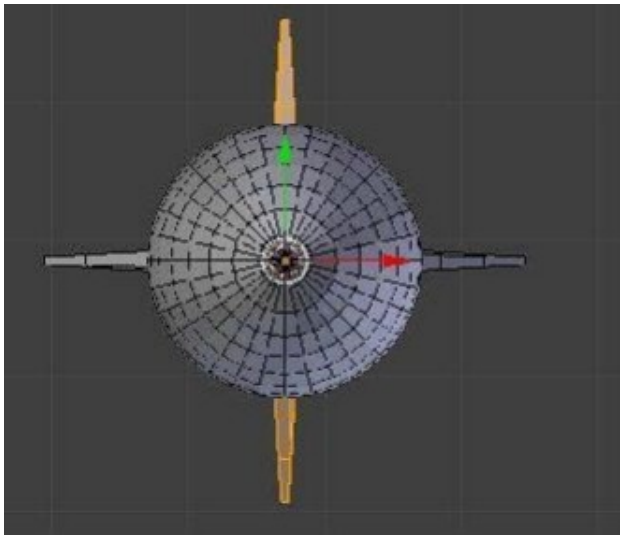

Рисунок 3.34 - Повертаю стабілізатори на 90 градусів

17. У цей момент я вважав за краще повернути всі чотири стабілізатора на 45 градусів:
18. Потім, дублюю ці стабілізатори і перетаскую їх назад (рис. 3.35).

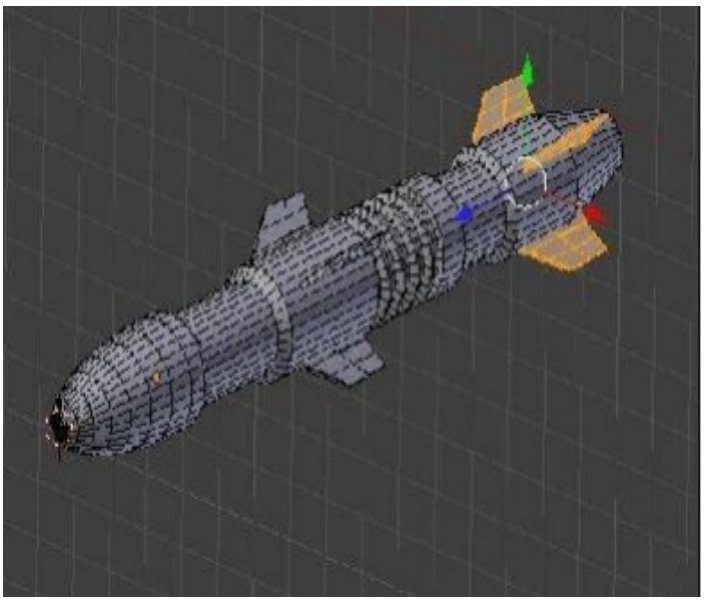

Рисунок 3.35 - Стабілізатори встановлені на 45 градусів

19. Тепер я збираюся вибрати ці межі на носі (рис. 3.36).

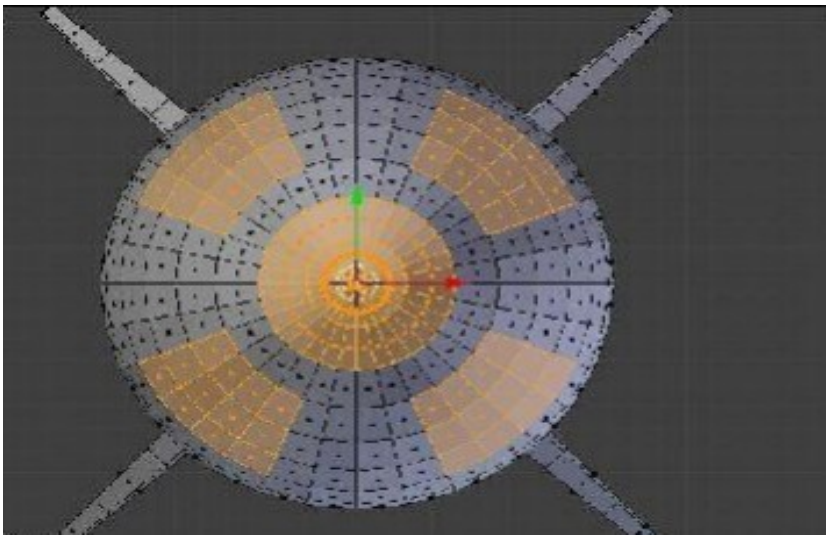

Рисунок 3.36 - Виділяю межі на носовій частині

20. І видавити їх всередину
21. Обираю ребра на задній стороні і сформую грань ("F") (рис. 3.37).

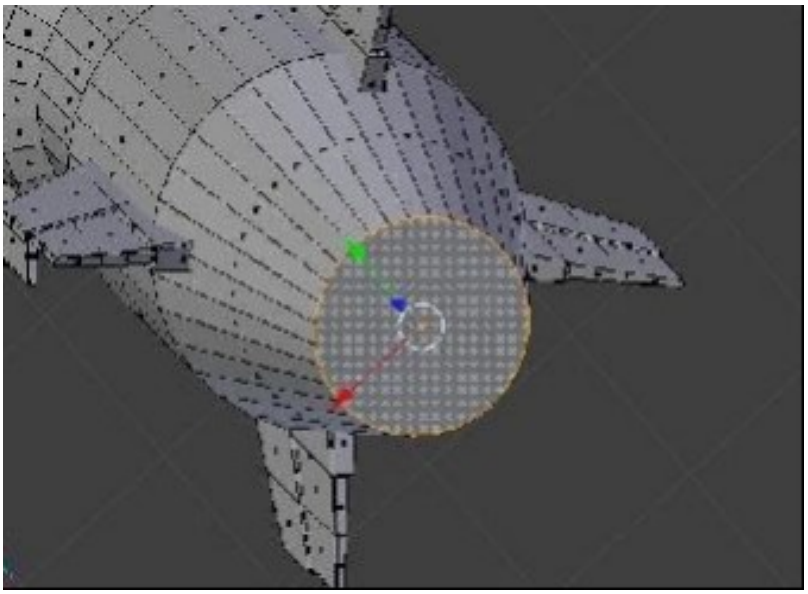

Рисунок 3.37 - Формую грань на задній стороні ракети

22. Видавлюю грань до центру кола і трохи всередину:
23. Налаштовую об'єкт на гладке затінення.
24. Видаляю дублікати.
25. Вибираю все і вивертаю нормалі назовні (CTRL + "N").
26. Дороблена ракета у кінцевому результаті в Unity 3D (рис. 3.38).

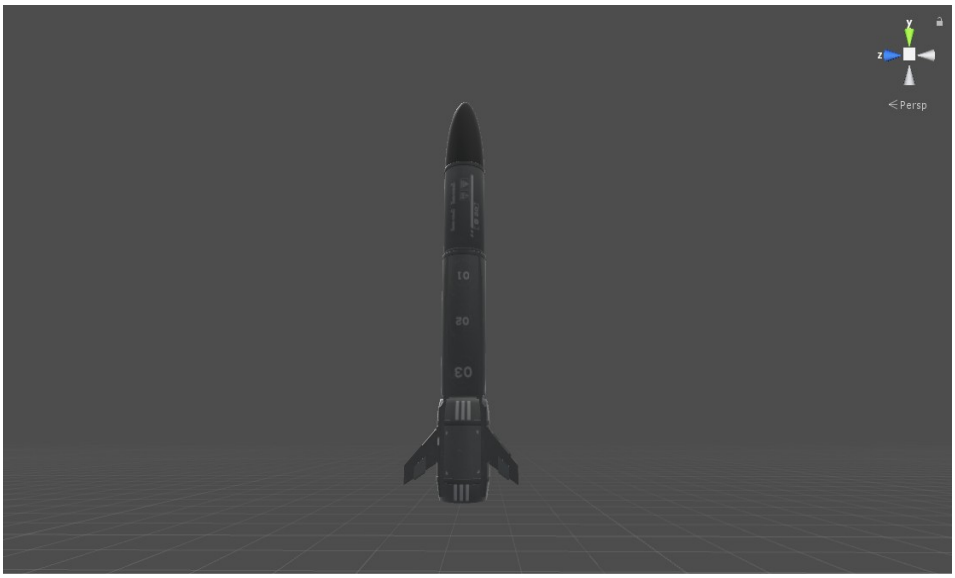

Рисунок 3.38 - Кінцевий результат ракети

4 УПРАВЛІННЯ ПРОЕКТОМ

Проект – унікальний набір процесів, що складаються зі скоординованих і керованих завдань з початковою і кінцевою датами, вжитих для досягнення мети. Досягнення мети проекту вимагає отримання результатів, що відповідають певним заздалегідь поставленим вимогам, у тому числі обмеження на отримання результатів, таких як час, гроші і ресурси.

Управління проектом – область діяльності, в ході якої визначаються та досягаються чіткі цілі проекту при балансуванні між обсягом робіт, ресурсами (такими як гроші, праця, матеріали, енергія, простір та ін.), часом, якістю та ризиками. Ключовим фактором успіху проектного управління є наявність чіткого заздалегідь визначеного плану, мінімізації ризиків і відхилень від плану, ефективного управління змінами.

4.1 Планування проекту

Для поставленої задачі планування використовувалася програма MS Project 2010. Microsoft Project (або MSP) – програма керування проектами, розроблена компанією Microsoft.

Microsoft Project створений, щоб допомогти менеджеру проекту в розробці планів, розподілі ресурсів за завданнями, відстеженні прогресу і аналізі обсягів робіт. Microsoft Project створює розклад критичного шляху. Розклади можуть бути складені з урахуванням використовуваних ресурсів. Ланцюжок візуалізується на діаграмі Ганта.

Проект дозволяє досягти певного результату в певні терміни і за певні гроші. План проекту складається для того, щоб визначити, за допомогою яких робіт буде досягатися результат проекту, які люди й устаткування потрібні для виконання цих робіт, в який час ці люди й устаткування будуть зайняті роботою по проекту. Тому проектний план містить три основні елементи: завдання (task), ресурси (resource) і призначення (assignment).

У табл. 4.1 наведений план проекту, відповідно до якого велася його розробка і реалізація.

Дана таблиця містить всі завдання, які виконувалися при розробці інтерактивної моделі, тривалість кожної з задач, терміни її виконання, а також які ресурси для цього використовувалися.

Таблиця 4.1 – План проекту

Назва задачі	Термін	Початок	Кінець
1	2	3	4
Початок реалізації проекту	0 днів	25.04.18	25.04.18
Проектування системи	15 днів	25.04.18	10.05.18
Ознайомлення з предметною областю	10 днів	10.05.18	20.05.18
Постановка задачі	1 днів	20.05.18	21.05.18
Аналіз предметної області	5 днів	21.05.18	26.05.18
Побудова функціональної моделі	10 днів	26.05.18	05.06.18
Інші роботи	4 днів	05.06.18	09.06.18
Проектування закінчено	0 днів	09.06.18	09.06.18
Реалізація системи	50 днів	09.06.18	29.07.18
Аналіз доступних середовищ розробки	10 днів	29.07.18	08.08.18
Вибір середовищ розробки	1 днів	08.08.18	09.08.18

Продовження таблиці 4.1

1	2	3	4

Створення проекту в Unity	1 днів	09.08.18	10.08.18
Вивчення основ та характеристик Unity 3D	9 днів	10.08.18	19.08.18
Ознайомлення з Blender 3D	6 днів	19.08.18	25.08.18
Вивчення документції для створення об'єктів у Blender 3D	30 днів	25.08.18	24.09.18
Вивчення основ та характеристик Unity та Blender	5 днів	24.09.18	29.09.18
Створення об'єктів в Blender	20 днів	29.09.18	19.10.18
Створення декількох об'єктів в Unity 3D	19 днів	19.10.18	08.11.18
Экспорт об'єктів у Unity	1 днів	08.11.18	09.11.18
Написання коду програмного модулю	11 днів	09.11.18	20.11.18
Інші роботи	10 днів	20.11.18	30.11.18
Реалізація закінчена	0 днів	30.11.18	30.11.18
Відладка розробленої інтерактивної моделі	2 днів	30.11.18	02.12.18
Кінець проекту	1 днів	02.12.18	03.12.18

4.2 Управління ризиками проекту

Причиною виникнення ризиків є невизначеності, існуючі в кожному проекті. Ризики можуть бути відомі – ті, які визначені, оцінені, для яких

можливе планування. Ризики невідомі – ті, які не ідентифіковані і не можуть бути передбачені. Хоча специфічні ризики й умови їх виникнення не визначені, менеджери проекту знають, виходячи з минулого досвіду, що більшу частину ризиків можна передбачити.

Реалізуючи проекти, що мають високий ступінь невизначеності в таких елементах, як цілі і технології їх досягнення багато компаній приділяють увагу розробці та застосуванню корпоративних методів управління ризиками. Дані методи враховують як специфіку проектів, так і корпоративних методів управління.

Управління ризиками – це процеси, пов'язані з ідентифікацією, аналізом ризиків та прийняттям рішень, які включають максимізацію позитивних і мінімізацію негативних наслідків настання ризикових подій. Процес управління ризиками проекту зазвичай включає виконання наступних процедур:

1) планування управлінням ризиками – вибір підходів та планування діяльності по керуванню ризиками проекту;

2) ідентифікація ризиків – визначення ризиків, здатних впливати на проект і документування їх характеристик;

3) якісна оцінка ризиків – якісний аналіз ризиків і умов їх виникнення з метою визначення їх впливу на успіх проекту;

4) кількісна оцінка – кількісний аналіз ймовірності виникнення і впливу наслідків ризиків на проект;

5) планування реагування на ризики – визначення процедур і методів з послаблення негативних наслідків ризикових подій і використання можливих переваг;

6) моніторинг і контроль ризиків – моніторинг ризиків, визначення ризиків, що залишаються, виконання плану керування ризиками проекту і оцінка ефективності дій з мінімізації ризиків.

Всі ці процедури взаємодіють одна з одною, а також з іншими процедурами. Кожна процедура виконується, принаймні, один раз в кожному проекті. Незважаючи на те, що процедури розглядаються як дискретні елементи з чітко визначеними характеристиками, на практиці вони можуть частково збігатися і взаємодіяти

При розробці даного наукового проекту не здійснювалося серйозне управління ризиками, проте враховувалася можливість виникнення певних форс-мажорів, таких як хвороби виконавців, забуті роботи і т.п. Для цього в кожен етап була додана фіктивна задача з мінімальним пріоритетом, під

назвою «інші роботи» для кожного ресурсу, що можна побачити в табл. 4.1. Після вирівнювання ресурсів ці завдання виявляються в кінці етапу. Тривалість цих завдань залежить від імовірності виникнення і ступеня впливу ризиків, вона залежить від способу визначення оцінок тривалості задач, здоров'я членів команди та інших факторів.

Тривалість «інших робіт» для кожного етапу розробки виставлялася приблизно до чверті довжини етапу. Такий метод обліку ризиків дозволяє, по-перше назвати терміни виконання проекту та його етапів, причому аргументовано і з високим ступенем достовірності, і по-друге оцінити зразкові трудовитрати по проекту.

Трудомісткость розробки програмного забезпечення. На даному етапі необхідно визначити, скільки часу буде займати реалізація проекту і в які терміни необхідно вирішити ту чи іншу задачу. Це, в свою чергу, визначить, коли саме потрібно залучити ті чи інші ресурси і як буде контролюватися розподіл коштів проекту.

Варто відзначити, що в управлінні термінами пріоритетним є визначення взаємозв'язків задач проекту і способів їх реалізації; саме це і впливає в першу чергу на те, як вибудовуються завдання в календарному плані робіт. Даний підхід має ряд переваг.

По-перше, робота повинна визначати графік, а не навпаки, тому що досить легко розробити календарний план, який на ділі буде розходитися з намірами.

По-друге, даний підхід повністю співпадає з інструментами управління проектами, такими як Microsoft Project.

По-третє, використовуючи даний підхід, можна одержати цілком реалістичний графік результатів, менш схильних до змін, на відміну від графіків робіт, які є більш гнучкими і визначаються іншими способами.

Таким чином, процеси управління термінами проекту включають в себе наступні фази:

1) визначення складу операцій – виявлення всіх планових операцій, які необхідно здійснити. Це досягається за допомогою таких методів, як: декомпозиція, шаблони, планування методом хвилі що набігає, експертна оцінка і т.д.;

2) визначення взаємозв’язків операцій включає ідентифікацію і документування логічних взаємозв’язків між плановими операціями. Завдання послідовності може бути виконане за допомогою програмного забезпечення для керування проектами або вручну;

3) оцінка ресурсів операцій повинна визначати, які ресурси (людські, матеріальні і т.д.) будуть використовуватися і в якій кількості, та коли кожен з цих ресурсів буде доступний для виконання проектних рішень;

4) оцінка тривалості операцій використовує інформацію про склад робіт планової операції, типах необхідних ресурсів, розрахунковій кількості ресурсів і календарях ресурсів з указівкою їх доступності;

5) розробка розкладу здійснюється за допомогою визначення планових дат початку і кінця кожної з операцій на проекті. Розробка розкладу здійснюється безперервно по всьому проекті по мірі виконання робіт, зміни плану керування проектом і виникненні або припиненні очікуваних ризиків або виявленні нових;

6) керування розкладом інтегрує в собі усі вищенаведені фази.

Проект був повністю завершений у строки 11 грудня 2018 року, як і було заплановано. Загальна тривалість розробки і реалізації проекту склала 223 днів, або 2676 годин. Під час розробки були задіяні всі наявні ресурси.

ВИСНОВКИ

Була створена інтерактивна модель за допомогою якої можна усувати атмосферні процеси. Для реалізації данної моделі був виконаний аналіз існуючих ігрових движків та графічних 3D редакторів, та у результаті був обраний движок Unity, та графічний редактор Blender 3D.

За допомогою графічного редактору Blender 3D було створено більш ніж 20 ігрових об' єктів власноруч, для використання їх у проекті.

Основною розробкою моделі була сцена в Unity 3D, яка скалається з більш ніж 150 об'єктів створених у Blender і взаємодією між ними.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАНЬ

1. Большаков Д. И. 3D моделирование. / Д. И. Большаков - М.: Техатека, 2011. 34 с.
2. Бочков М. Д. Основи 3D-моделювання. / М. Д. Бочков - К.: КНЭУ, 2003. 106 с.
3. Донован Т. Ігрова індустрія. Створення ігор. / Т. Донован - К.: Видавничий дім «Вільямс», 2007. 412 с.
4. Дацко М. А. Моделювання складних об'єктів. / М. А. Дацко - К.: Максімас, 2015. 111 с.
5. Хокінг Д. Unity в дії. Мультиплатформенна розробка. / Д. Хокінг - М.: Видавничий центр «Академія», 2016. 336 с.
6. Кролиестер Д. Основы Blender начальный учебник 4-е издание. / Д. Кролиестер - СПб.: Питер, 2012. 416 с.
7. Максимов А. П. Створення найпростіших моделей, побудова сцени. / А. П. Максимов - К.: Мітра, 2011. 38 с.
8. Мейкісон Л. Моделювання - це легко. / Л. Мейкісон - К.: Інтуїт, 2013. 313 с.
9. Петренко. С. М. Изучаем Blender 3D. / С. М. Петренко - М.: Росийский новый университет, 2009. 542 с.
10. Коротаєв А.В., Малков С.Ю., Гринин Л.Є. Історія і Математика: Аналіз і моделювання соціально-історичних процесів. / Вид.3 - К .: URSS, 2016. 360 с.
11. В. Є. Михайленко, В. В. Ванін, С. М. Ковальов. Інженерна такомп'ютерна графіка: підруч. для студ. ВНЗ. / 5-те вид. - К .:Каравела, 2010. 360 с.
12. В. П. Іванов, А. С. Батраков. Комп'ютерна графіка. Під ред. Г. М. Поліщука. / К .: Радіо та зв’язок, 1995. 224 с.
13. Kajiya, James T., "The rendering equation". / N.Y .: Siggraph 1986. 143 p.
14. Тозік В.Т. Л.М. Корпан. Комп'ютерна графіка та дизайн: Підручник для поч. проф. Освіти. / К .: ВЦ Академія, 2013. 208 с.
15. Заставне Л.А. Комп'ютерна графіка. Елективний курс: Практикум. / М .: БИНОМ. ЛЗ, 2011. 245 с.
16. Дегтярьов В.М. Інженерна і комп'ютерна графіка: Підручник для установ вищої професійної освіти. / В .: ВЦ Академія, 2011. 240 с.

17. Кейлер В.А., Економика предприятельства. / М.: Новосибирск: НГАЕИУ, 1999. 131 с.
18. Кондратева М.Н. Економика предприятельства: нач. учебник / М.Н. Кондратева, Е.В. Тен. 2-е изд. Пред. - Ульяновський: УлГТУ, 2006. 171 с.
19. Pasetsky V. Meteorological Center. / L.: Doubleday 1978. 430 p.
20. Jonhson E. The first woman geophysics and meteorology. / University of California Press, 1989. 430p.
21. Mickelson S. Meteorology and climatology for the geographical departments. / L.: Hydrometeorological Publishing House, 1964. 500 p.
22. Gural'nik I., Larin V. Meteorology, methods and technical means of observation. / Hydrometeorological Publishing House, 1991. 336 p.
23. Perafinho M. Meteorology and Climatology. Textbook. / Interurban Press, 2001. 527 p.
24. Ненашев А.П., Конструювання радіоелектронних засобів: навч. для радіотехнічних спец. Вузів / К .: Вища школа, 1990. 432 с.
25. Сулаквелидзе Г. К. Ливневые осадки и град / Л .: Гидрометеоиздат, 1967. 412 с.

Д О Д А Т К И

Додаток А
Графічна частина магістерської роботи

Рисунок А.1 - Ракетна установка

Рисунок А.2 - Керуюча панель для запуску ракет

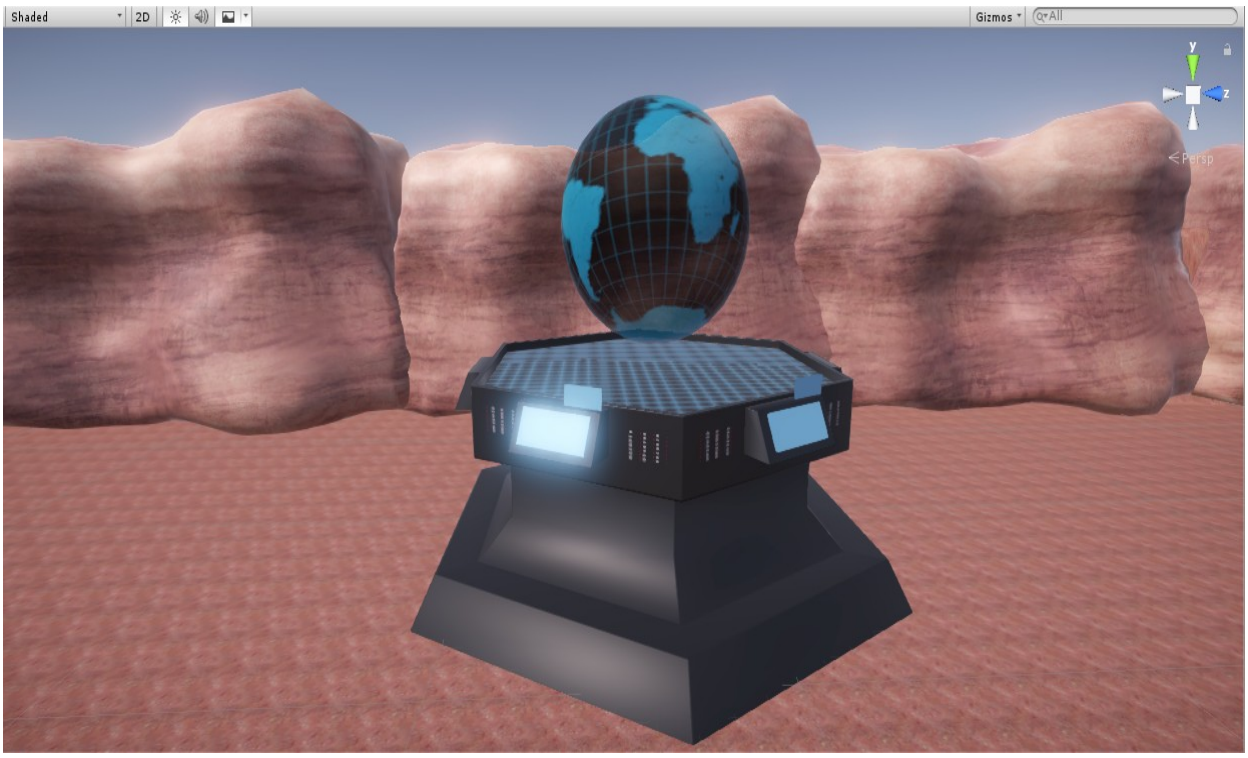

Рисунок А.3 – Голографічна модель землі

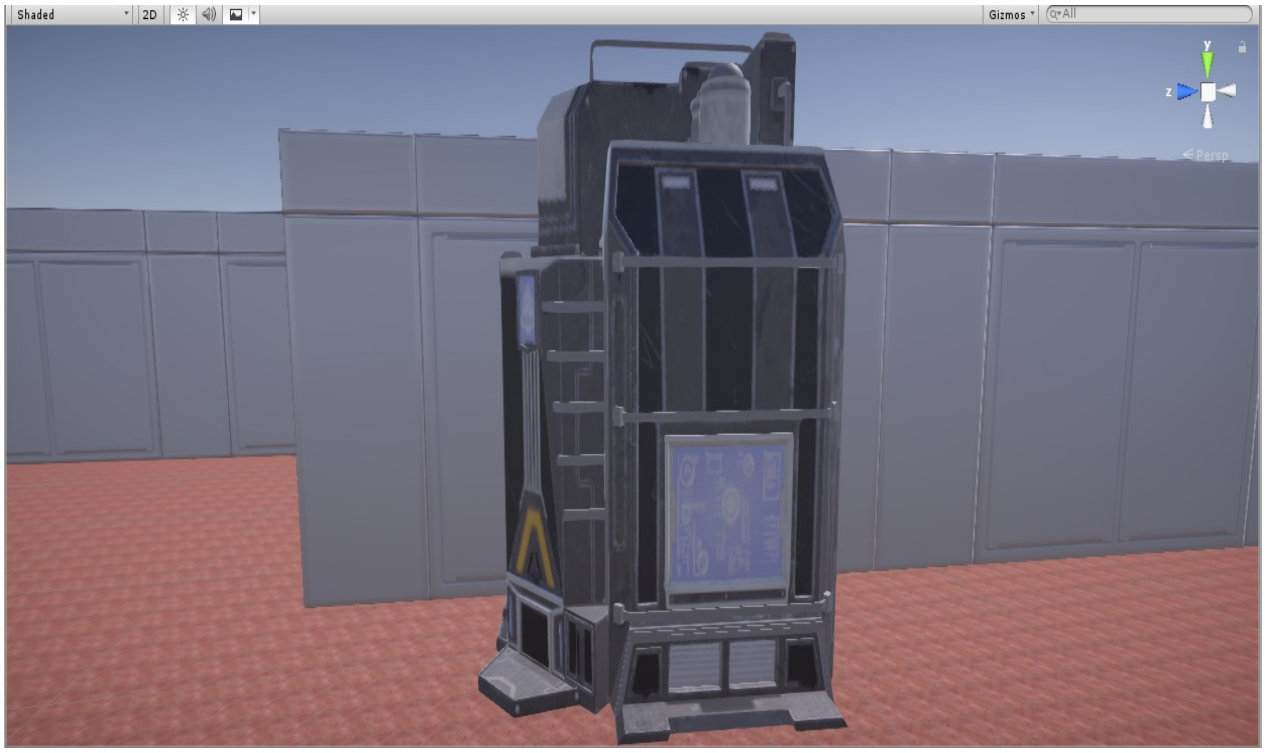

Рисунок А.4 - Генератор

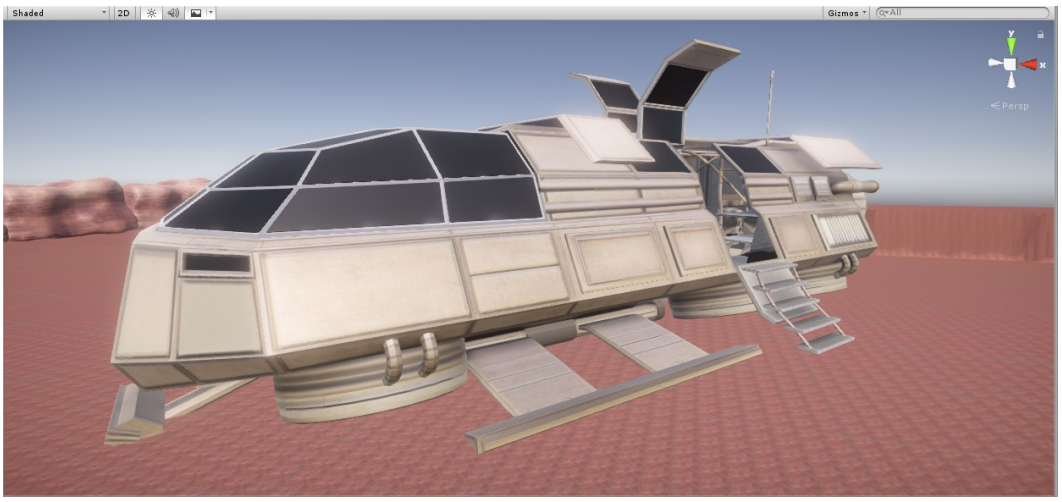

Рисунок А.5 – Транспортний шатл

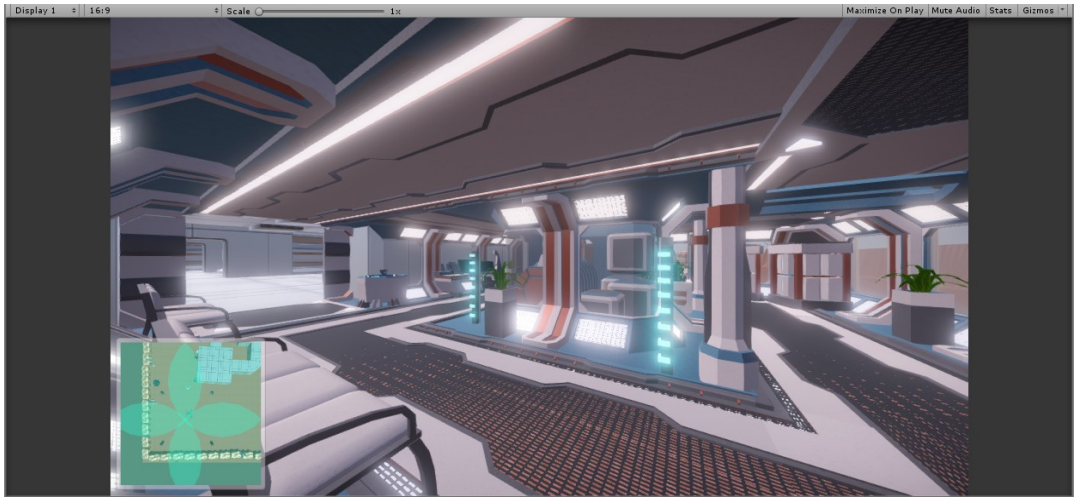

Рисунок А.6 – Лабораторія

Рисунок А.7 - Повне розсіювання

Рисунок А.8 – Шарувата хмарність

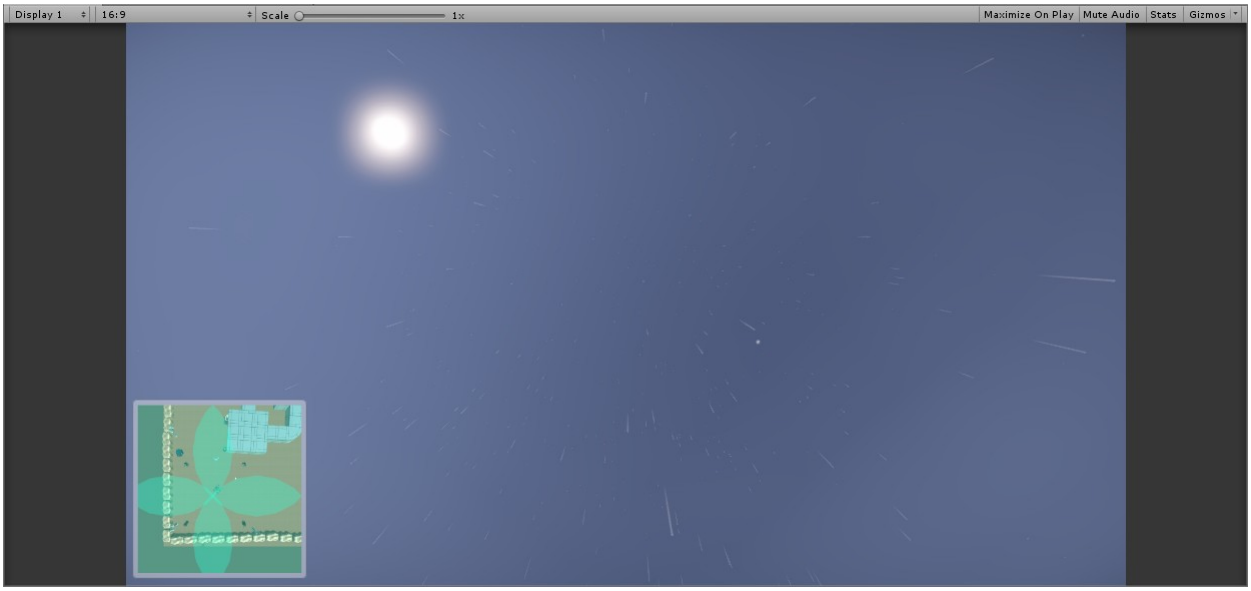

Рисунок А.9 – Ефект дощу після запобігання граду

Додаток Б
Код магістерської роботи

```
public class HDK_RaycastManager : MonoBehaviour {

        [Header ("Crosshair")]
        public GameObject normal_Crosshair;
        public GameObject interact_Crosshair;

        [Header ("Raycast")]
        public float distance = 2.0f;
        public LayerMask layerMaskInteract;
    public Color RayGizmoColor;
    RaycastHit hit;

    [Header ("Tags")]
        string KeyTag = "Key";
        string FlashlightTag = "Flashlight";
        string FlashlightBatteryTag = "FlashlightBattery";
        string DoorTag = "Door";
        string PaperTag = "Paper";
        string TelecameraTag = "Telecamera";
        string LampTag = "Lamp";
        string ExamineTag = "Examine";
    string PlayAudioTag = "PlayAudio";
    string SecurityCamTag = "SecurityCamera";
    string FoodTag = "Food";

    [Header ("SFX")]
        public AudioClip[] KeyPickup;
        public AudioClip[] GeneralPickup;
        public AudioClip[] PaperPickup;
        public AudioClip[] ExaminedReveal;
        public AudioClip CantPickup;
        public float pickupVolume;
        public float revealVolume;

        [Header ("Tags Bools")]
        bool OnTagKey;
        bool OnTagFlashlight;
        bool OnTagFlashlightBattery;
        bool OnTagDoor;
        bool OnTagPaper;
        bool OnTagTelecamera;
        bool OnTagLamp;
        bool OnTagExamine;
    bool OnTagPlayAudio;
    bool OnTagSecurityCam;
    bool OnTagFood;

    [Header ("Other")]
        GameObject Player;                                          //Player GameObject
```

```
        GameObject targetDoor;                                  //The target door/drawer...
        bool hasFlashlight;                                             //Do we have the flashlight?
        GameObject doorRaycasted;                               //The Door/Drawer... raycasted
        GameObject targetPaperNote;                             //The target paper note
        public GameObject raycasted_obj;          //The raycasted item
   GameObject RaycastedLamp;                //The raycasted functional lamp
        GameObject RaycastedExamineObj;                         //The raycasted examinable item
        public static bool ExaminingObject;               //Are we examining an item?
        public static bool ReadingPaper;          //Are we reading a paper?
   public static bool UsingSecurityCam;       //Are we using a security camera?
   GameObject ExamineObjectInfoGUI;           //The Examine Object Info GUI
        bool ShowExaminingInfoGui;                               //Do we need to show the Examine Object Info
GUI?
        public float RevealWait;                          //The time we need to wait for the item to be revealed af-
ter the examination
        GameObject ItemNameText;                                //The text that shows us the name of the item
        bool FadeInteractInfoGUI;                               //Do we need to fade the Interact Info GUI?
        GameObject examineEyeIcon;
   bool hasPeak;
   bool hasHBob;

   void Start()
        {
                Player = GameObject.Find("Player");
                ExaminingObject = false;
                ExamineObjectInfoGUI = GameObject.Find ("examineControls");
                ItemNameText = GameObject.Find ("itemName");
                examineEyeIcon = GameObject.Find ("icon_examining");
        }

        IEnumerator RevealExamined()
        {
                yield return new WaitForSeconds (RevealWait);
                if (ExaminingObject)
    {
                        ItemNameText.GetComponent<Text>                         ().text                 =
RaycastedExamineObj.GetComponent<HDK_InteractObject> ().ItemName;
                        RaycastedExamineObj.GetComponent<HDK_InteractObject> ().Examined = true;
                        this.GetComponent<AudioSource> ().clip = ExaminedReveal [Random.Range (0, Exam-
inedReveal.Length)];
                        this.GetComponent<AudioSource> ().volume = revealVolume;
                        this.GetComponent<AudioSource> ().Play ();
                        FadeInteractInfoGUI = true;
                }
        }

        void Update() {

                if (FadeInteractInfoGUI) {
                        ItemNameText.GetComponent<HDK_UIFade> ().TextOut = false;
                        ItemNameText.GetComponent<HDK_UIFade> ().TextIn = true;
                } else {
                        if (!ExaminingObject) {
                                ItemNameText.GetComponent<HDK_UIFade> ().TextOut = true;
                                ItemNameText.GetComponent<HDK_UIFade> ().TextIn = false;
```

```
                }
            }
            if (ShowExaminingInfoGui) {
                ExamineObjectInfoGUI.GetComponent<HDK_UIFade> ().TextIn = true;
                ExamineObjectInfoGUI.GetComponent<HDK_UIFade> ().TextOut = false;
            } else
            {
                ExamineObjectInfoGUI.GetComponent<HDK_UIFade> ().TextIn = false;
                ExamineObjectInfoGUI.GetComponent<HDK_UIFade> ().TextOut = true;
            }
            bool canusecamera = HDK_DigitalCamera.canUse;
    Vector3 position = transform.parent.position;
            Vector3 direction = transform.TransformDirection(Vector3.forward);
    Debug.DrawRay(transform.position, direction * distance, RayGizmoColor);

    if (Physics.Raycast (position, direction, out hit, distance, layerMaskInteract.value)) {
                if (hit.transform.gameObject.GetComponent<HDK_InteractObject> ()) {
                    raycasted_obj = hit.transform.gameObject;
                    if (raycasted_obj.GetComponent<HDK_WithEmission> ())
                    {
                        raycasted_obj.GetComponent<HDK_WithEmission>  ().rayed
= true;
                    }else  if(!raycasted_obj.GetComponent<HDK_WithEmission>()  &&
raycasted_obj.GetComponentInChildren<Light>() && !raycasted_obj.GetComponent<HDK_WithNoEmission>())
                    {
                        raycasted_obj.GetComponentInChildren<Light>  ().enabled  =
true;
                    }
                    normal_Crosshair.SetActive (false);
                    interact_Crosshair.SetActive (true);
                    FadeInteractInfoGUI = true;
                    if (raycasted_obj.GetComponent<HDK_InteractObject> ().Examined) {
                        ItemNameText.GetComponent<Text>  ().text  =  raycasted_ob-
j.GetComponent<HDK_InteractObject> ().ItemName;
                    } else
                    {
                        ItemNameText.GetComponent<Text> ().text = null;
                    }
                }
            }
            else
            {
                if (raycasted_obj != null) {
                    if (raycasted_obj.GetComponent<HDK_WithEmission> ()) {
                        raycasted_obj.GetComponent<HDK_WithEmission>  ().rayed
= false;
                    } else if (!raycasted_obj.GetComponent<HDK_WithEmission> () &&
raycasted_obj.GetComponentInChildren<Light> () && !raycasted_obj.GetComponent<HDK_WithNoEmission>())
{
                        raycasted_obj.GetComponentInChildren<Light>  ().enabled  =
false;
                    }
                }
                normal_Crosshair.SetActive (true);
                interact_Crosshair.SetActive (false);
```

```
                FadeInteractInfoGUI = false;
        }
        if (Physics.Raycast (position, direction, out hit, distance, layerMaskInteract.value)) {
                if (hit.transform.CompareTag (KeyTag) && hit.transform.GetComponent<HDK_Key>
()) {
                        targetDoor = hit.transform.GetComponent<HDK_Key> ().targetDoor;
                        OnTagKey = true;
                }
                else
                {
                        OnTagKey = false;
                }
        }
        else
        {
                OnTagKey = false;
        }

        if (Physics.Raycast (position, direction, out hit, distance, layerMaskInteract.value)) {
                if (hit.transform.CompareTag(FlashlightTag)){
                        OnTagFlashlight = true;
                }
                else
                {
                        OnTagFlashlight = false;
                }
        }
        else
        {
                OnTagFlashlight = false;
        }
        if (Physics.Raycast (position, direction, out hit, distance, layerMaskInteract.value)) {
                if (hit.transform.CompareTag(FlashlightBatteryTag)){
                        OnTagFlashlightBattery = true;
                }
                else
                {
                        OnTagFlashlightBattery = false;
                }
        }
        else
        {
                OnTagFlashlightBattery = false;
        }
        if (Physics.Raycast (position, direction, out hit, distance, layerMaskInteract.value)) {
                if (hit.transform.CompareTag(DoorTag)){
                        OnTagDoor = true;
                        doorRaycasted = hit.transform.gameObject;
                }
                else
                {
                        OnTagDoor = false;
                }
        }
        else
```

```
		{
			OnTagDoor = false;
		}
		if (Physics.Raycast (position, direction, out hit, distance, layerMaskInteract.value)) {
				if (hit.transform.CompareTag(PaperTag) &&
hit.transform.GetComponent<HDK_Note>()){
					targetPaperNote = hit.transform.GetComponent<HDK_Note>
().UI_Note;
				OnTagPaper = true;
			}
			else
			{
					OnTagPaper = false;
			}
		}
		else
		{
				OnTagPaper = false;
		}
		if (Physics.Raycast (position, direction, out hit, distance, layerMaskInteract.value)) {
			if (hit.transform.CompareTag(TelecameraTag)){
				OnTagTelecamera = true;
			}
			else
			{
				OnTagTelecamera = false;
			}
		}
		else
		{
			OnTagTelecamera = false;
		}
		if (Physics.Raycast (position, direction, out hit, distance, layerMaskInteract.value)) {
			if (hit.transform.CompareTag(LampTag) &&
hit.transform.GetComponent<HDK_SwitchableLamp>()){
				OnTagLamp = true;
				RaycastedLamp = hit.transform.gameObject;
			}
			else
			{
					OnTagLamp = false;
			}
		}
		else
		{
				OnTagLamp = false;
		}
		if (Physics.Raycast (position, direction, out hit, distance, layerMaskInteract.value)) {
			if (hit.transform.CompareTag(ExamineTag) || hit.transform.GetComponent<HDK_Inter-
actObject>()){
				OnTagExamine = true;
				RaycastedExamineObj = hit.transform.gameObject;
			}
			else
			{
```

```
                                OnTagExamine = false;
                        }
                }
                else
                {
                                OnTagExamine = false;
                }
if (Physics.Raycast(position, direction, out hit, distance, layerMaskInteract.value))
{
   if (hit.transform.CompareTag(PlayAudioTag))
   {
      OnTagPlayAudio = true;
   }
   else
   {
      OnTagPlayAudio = false;
   }
}
else
{
   OnTagPlayAudio = false;
}

if (Physics.Raycast(position, direction, out hit, distance, layerMaskInteract.value))
{
   if (hit.transform.CompareTag(SecurityCamTag))
   {
      OnTagSecurityCam = true;
   }
   else
   {
      OnTagSecurityCam = false;
   }
}
else
{
      OnTagSecurityCam = false;
}

if (Physics.Raycast(position, direction, out hit, distance, layerMaskInteract.value))
{
   if (hit.transform.CompareTag(FoodTag))
   {
      OnTagFood = true;
   }
   else
   {
      OnTagFood = false;
   }
}
else
{
   OnTagFood = false;
}
```

```
    if (ExaminingObject)
                {
                                if (Input.GetKeyDown (KeyCode.Mouse1))
                                {
                                                if (Player.GetComponent<HDK_DigitalCamera>().UsingCamera)
                                                {
                                                                Player.GetComponent<HDK_DigitalCamera>          ().CameraUI.GetComponent<CanvasGroup> ().alpha = 1;
                                                                if (Player.GetComponent<HDK_DigitalCamera> ().broken)
                                                                {
                                                                                Player.GetComponent<HDK_DigitalCamera> ().camera_effect.enabled = true;
                                                                                Player.GetComponent<HDK_DigitalCamera>   ().brokenGUI.GetComponent<CanvasGroup> ().alpha = 1;
                                                                }
                                                }
                                if (hasHBob)
                                {
                                  GetComponentInParent<HeadBobController>().enabled = true;
                                }
                                if (hasPeak)
                                {
                                  GetComponentInParent<HDK_Peak>().enabled = true;
                                }
                                if (Player.GetComponent<HDK_Stamina>() != null)
                                {
                                  Player.GetComponent<HDK_Stamina>().Busy(false);
                                }
                                examineEyeIcon.GetComponent<HDK_UIFade> ().TextIn = false;
                                                examineEyeIcon.GetComponent<HDK_UIFade> ().TextOut = true;
                                                ExaminingObject = false;
                                                ShowExaminingInfoGui = false;
                                                Player.GetComponent<FirstPersonController> ().enabled = true;
                                                Player.GetComponentInChildren<HDK_ExamineRotation>    ().enabled = false;
                                                Player.GetComponentInChildren<HDK_ExamineRotation>  ().target = null;
                                                RaycastedExamineObj.GetComponent<HDK_InteractObject> ().ExaminableObject.SetActive (false);
                                                RaycastedExamineObj.SetActive (true);
                                                if (Player.GetComponent<HDK_DigitalCamera>().UsingCamera && ! Player.GetComponent<HDK_DigitalCamera>().broken)
                                                {
                                                                Player.GetComponentInChildren<HDK_DigitalCameraZoom> ().canZoom = true;
                                                }
                                                canusecamera = true;
                                                Player.GetComponentInChildren<BlurOptimized> ().enabled = false;
                                                if (!Player.GetComponent<HDK_DigitalCamera>().UsingCamera)
                                                {
                                                                Player.GetComponent<HDK_MouseZoom>    ().canZoom    = true;
                                                }
                                }
                }
```

```
            if (OnTagExamine) {
                        if (Input.GetKeyDown (KeyCode.Mouse1)) {
                                if (!ExaminingObject) {
                                        if    (RaycastedExamineObj.GetComponent<HDK_InteractOb-
ject> ().ExaminableObject == null) {
                                                ExaminingObject = false;
                                                this.GetComponent<AudioSource>   ().clip   =   Cant-
Pickup;
                                                this.GetComponent<AudioSource> ().volume = 1f;
                                                this.GetComponent<AudioSource> ().Play ();
                                        } else {
                                                if
(Player.GetComponent<HDK_DigitalCamera>().UsingCamera) {

        Player.GetComponent<HDK_DigitalCamera> ().CameraUI.GetComponent<CanvasGroup> ().alpha = 0;
                                                        if (Player.GetComponent<HDK_DigitalCam-
era> ().broken) {
                                                                Player.GetComponent<HDK_Digi-
talCamera> ().brokenGUI.GetComponent<CanvasGroup> ().alpha = 0;
                                                                Player.GetComponent<HDK_Digi-
talCamera> ().camera_effect.enabled = false;
                                                        }
                                                }
                        if (!RaycastedExamineObj.GetComponent<HDK_InteractObject> ().Examined) {
                                                        StartCoroutine (RevealExamined ());
                                                }
                                                if    (GetComponentInParent<HDK_SimpleInventory>
() != null) {
                                                        GetComponentInParent<HDK_SimpleInven-
tory> ().close ();
                                                }
```