МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

ОДЕСЬКИЙ ДЕРЖАВНИЙ ЕКОЛОГІЧНИЙ УНІВЕРСИТЕТ

Факультет Магістерської та аспірантської підготовки

Кафедра Автоматизованих систем моніторингу навколишнього середовища

МАГІСТЕРСЬКА РОБОТА

на тему: Реалізація анімації об’єктів інтерактивної моделі здійснення активних впливів на небезпечні атмосферні процеси

Виконав студент 2 курсу групи МК-62

спеціальності 122 Комп'ютерні науки

Буга Дмитро Ігорович

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

_____БУГА ДМИТРО ІГОРОВИЧ_____
(прізвище, ім'я, по батькові)

1. Тема роботи: Реалізація анімації об'єктів 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 2018 року № 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 анімацію об'єктів 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 анімації об'єктів інтерактивної моделі_____
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 Буга ДмитраІгоровича, натему «Реалізація анімації об’єктівінтерактивної моделі здійснення активних впливів на небезпечні атмосферні процеси».

Вона складається з двох частин: теоретичної та практичної.

Мета даної роботи – призначена для моделювання інтерактивної моделі впливу на небезпечні атмосферні процеси. Вихідною інформацією для даної моделі є модель атмосферної обстановки.

У програмній системі реалізованінаступні 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 режим спостереження за допомогою «міні-карти».

Система представлена в графічному вигляді виконана в 3D моделі.

Модель створена в графічному редакторіBlender 3D.

Об’єкти створенні за допомогою Blender 3D були експортовані в ігровий двіжок Unity3D, де було створено навколишнє середовище для розміщення створених об’єктів, та створена фізика рухів для цих об’єктів.

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

Магістерська робота містить: 79 с., рис. 30, табл. 7 додатки 2, використаних літературних джерел 25.

Ключеві слова: Unity, Blender, об’єкт, 3D графіка, анімація, моделювання.

SUMMARY

Presented master thesis Buga Dmytro, on the theme "Implementation of the Animation of Objects for an Interactive Model of Weather Modification under Hazardous Atmospheric Processes"

It consists of two parts: theoretical and practical.

The purpose of this work – simulate an interactive model of influence on atmospheric processes. Output information for this model is the model of atmospheric environment.

The software system implements the following key features. Select the atmospheric process to be eliminated. Introduction of initial values such us full dispersion, layered clouds, preventing hail. Starting the process of destroying clouds with rockets. There also a mode of monitoring the ability of the minescard.

The system is represented graphically made in 3D models.

The model was 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. Theprogram interface is friendly, easy and requires no special skills to manage theapplication.

Master's work includes: 79 sec., figures30, tables7, 2 applications, usedliterature 25.

Key words: Unity, Blender, object, 3D graphics, animation,modeling.

ЗМІСТ

ПЕРЕЛІК СКОРОЧЕНЬ, УМОВНИХ ПОЗНАЧЕНЬ І ТЕРМІНІВ

AAA - Високобюджетні і дорогі проекти

UI - Користувальницький інтерфейс

OM- Режим об'єкта

LOD-LevelOfDetail- Створення декількох варіантів одного об'єкта з різними ступенями деталізації

FBX- Один із форматів Unity

EM- Режим редагування

3D- Тривимірна графіка

DirectX - це набір API, розроблених для вирішення завдань, пов'язаних з програмуванням під Microsoft Windows

Drag&Drop - форма виконання певних дій у графічних інтерфейсах користувача (GUI), що передбачає використання комп'ютерної миші або сенсорного екрана

Occlusionculling - функція, яка відключає рендеринг тих об'єктів, які в дані момент не бачить камера

Ragdoll - вид процедурної анімації, що прийшов на заміну статичної, пререндерної анімації

ВСТУП

Основною метою магістерської роботи є розробити інтерактивну модель за допомогою движка Unity 3D.Для створення графічних об’єктів доведеться також 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 конструктором Unity 3D;
- ознайомитись з графічним редактором Blender;
- навчитися створювати3Dоб’єкти для нашого проекта у науковому стилі;
- навчитися експортувати об’екти із графічного редактора Blendery Unity 3D;
- навчитися скриптувати та анімувати об’єкти;
- вивчити весь функціонал Unity 3D для правильного функціонування об'єктів;
- додати до проекту ексклюзивний функціонал Unity3D,який сприяє збільшенню FPS;
- скомпелювати проект на OSWindows для запуску.

1 АНАЛІЗ ЗМІСТУ ПРОЦЕДУР АКТИВНИХ ВПЛИВІВ 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 штучної 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 за кілька десятиліть виконати величезну роботу з дослідження атмосферних процесів, яка і стала науковою основою активних впливів. В 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 1921р 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[2].

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 мікрофізики 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 електризованого і неелектризованого

піску, а в лабораторних умовах поставлені дослідження доцільності застосування хладореагентів. 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 з динамічної метеорології і з теорії клімату[3].

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 щодо малих витратах енергії, повинен з'явитися початком загальної зміни стану термодинамічно і колоїдально нестійких хмар і туманів, що в свою чергу має забезпечити практично значущі масштаби одержуваного ефекту в просторі і часі.

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 небезпечних 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 Ельбруський комплексної експедиції АН СРСР, стаціонарно що діяла на схилах Ельбрусу з 1934 по 1941 рік. У 1942р 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 грізними силами природи дуже складне завдання. Досить сказати, що грозоградові хмари можуть досягати потужності в кілька тисяч кубокілометров і мати енергію аналогічної кільком десяткам атомних бомб, скинутих на Хіросіму і Нагасакі разом узятих. Вчені після багаторічних досліджень знайшли всередині цих грозоградових хмарах 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 градонебезбечні 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 кристализуючого реагенту AgJ за допомогою спеціальних протиградових снарядів і ракет (рис. 1.1), при внесенні якого при температурі -10 ºС виділяються порядку 1012 кристалів з одного грама піросостава. 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. Піднімається тепле повітря в міру підйому охолоджується. 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 накопиченої води перерозподіляється між штучно створеною великою

кількістю зростаючих зародків, в результаті чого жоден з них не досягає великих розмірів, а дрібні градини,що утворилися, при падінні встигають розтанути в теплій частини атмосфери. За ефект впливу приймається відсутність граду на землі або локальне випадання дрібного граду в разі розвинення градової хмари. 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 одноразового випадання граду[4,5].

Рисунок 1.1 - Пуск протиградової ракети «Алазань-6»

Основним хладореагентом, використовуваним для впливів на хмари, є тверда вуглекислота (CO_2). Поверхня випаровуються частинок CO_2 має температуру -79°С. 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 льдообразучу 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[6].

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_2). Температурний поріг льдообразуючої активності AgJ - порядку - 6 °С, а PbJ_2 - порядку - 8 °С. Такі льдообразуючі реагенти, як AgJ і PbJ_2, застосовуються шляхом теплової сублімації в різних наземних і літакових димогенераторах і при горінні піротехнічних складів, що містять ці реагенти в ракетах, піропатронах, 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 піддавалася 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 осадкостворення.

При падінні через переохолоджену крапельно-рідку частину хмари крижані кристали укрупнюються як внаслідок перегонки на них води з крапель, так і в результаті намороження 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[7,8].

Для вирішення проблеми запобігання небезпечного градобою 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 градобою і покладений принцип ослаблення лавинного процесу утворення великих градин на порівняно рідкісних природних їх 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 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_2.

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 °, -10 °С, тобто 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 вертикальною потужністю 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-купчастої 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 безопадові дні штучне осадження снігу на поля навіть в малих кількостях може бути використано для

запобігання озимих від вимерзання, а в районах відгінного 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 ефективною для викликання опадів тільки при наявності достатніх водних запасів в хмарах і при сприятливому вертикальному вологообороті. 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 додаток 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 одній містоутворюючій хмарі може бути цих зон і кілька. Вони виявляються за допомогою згаданих вище спеціальних метеолокаторів 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 градоопасних 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 низької 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-рідких хмар або туманів за допомогою кристалізування реагентів - твердої вуглекислоти і деяких йодидів.

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

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 льодообразуючих 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 вологообороту під впливом меліоративних заходів. Тут же проведено деякі методичні 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 Францією й Італією було розпочато експерименти з придушення градом на основі радянської технології («Гроссферзух-4»). У Канаді та Іспанії були розпочаті роботи по збільшенню опадів за допомогою ракет, снаряджених AgJ. До середини 80 -х рр. вітчизняні методи по боротьбі з градом фахівці ВГІ почали застосовувати в Аргентіні, Бразілії, Болгарії, Угорщині, 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 градобою 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 врожай, стають банкрутами, не можуть платити по кредитах і вибувають з оподатковуваної бази. Недофінансування робіт з протиградового захисту знижує її ефективність, призводить до величезних збитків, що наноситься сільгоспугідь і багаторічних насаджень, хоча витрати, вироблені державою на протиградовий захист окупаються до 10 разів, забезпечуючи, як було вище згадано, річний економічний ефект понад 1 млрд грн. в залежності від градонебезпечності року і вартості захистів сільгоспкультур.

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.

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[9,10].

2ОБГРУНТУВАННЯ ВИБОРУ ПРОГРАМНИХ ЗАСОБІВ ДЛЯ МОДЕЛЮВАННЯ АКТИВНИХ ВПЛИВІВ НА АТМОСФЕРНІ ПРОЦЕСИ

2.1 Порівняльна характеристика ігрових движків

Unity- найпопулярніший движок для створення 2D- і 3D-ігор. 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[11].

Основний інтерфейс движка Unity 3Dпоказаний 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 ігри для трьох платформ: Windows, OS X і, що найцікавіше, iOS (і продавати ігри 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 мультиплеєрніігри, які набирають популярність, - наприклад P2P з планшетів.

Основний інтерфейс движка Torque 3Dпоказаний на рис. 2.2.

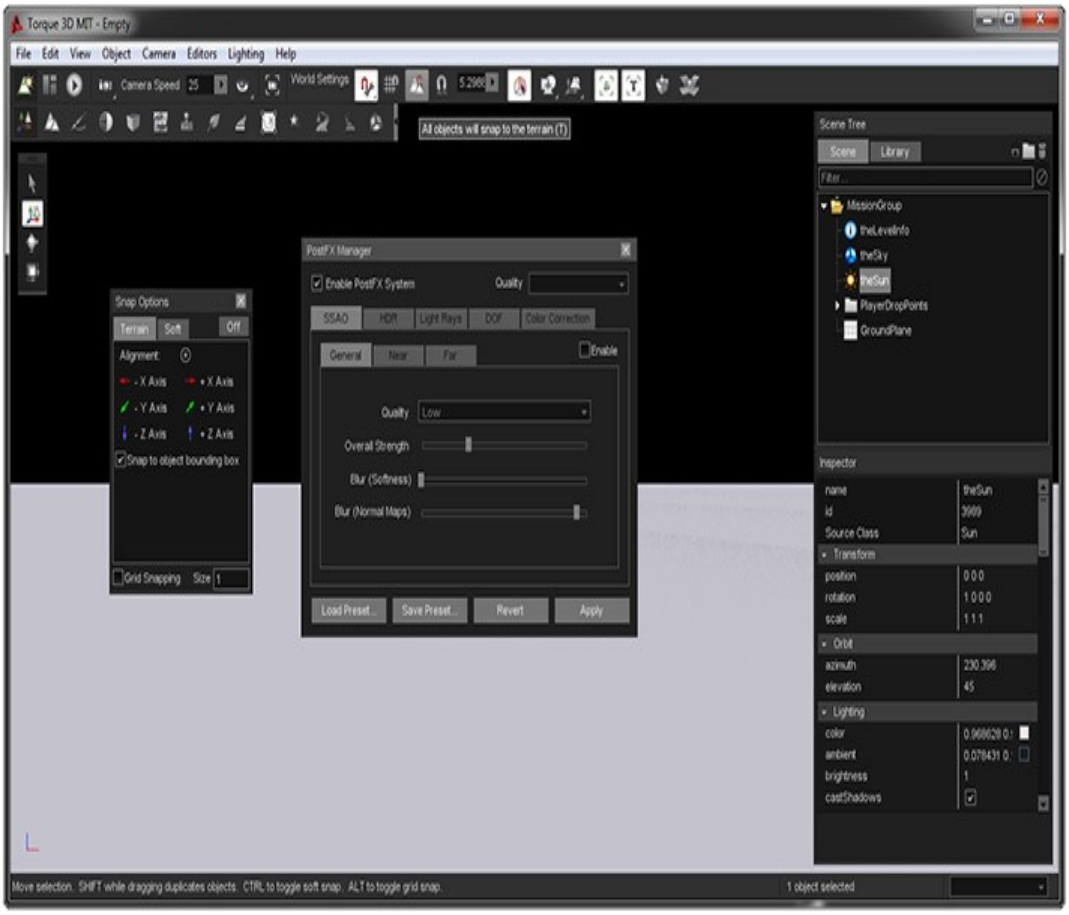

Рисунок 2.2 - Основний інтерфейс Torque 3D

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 ігри, а продавати його як технологію. Отже, всі розробленніCrytek'ом ігрові програми - це з метою зробити додаткову рекламу своєму головному продукту. Хоча для вивчення він доступний безкоштовно, щоб розробляти на ньому комерційні проекти, необхідно заплатити, причому 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 CryEngineпоказаний на рис. 2.3.

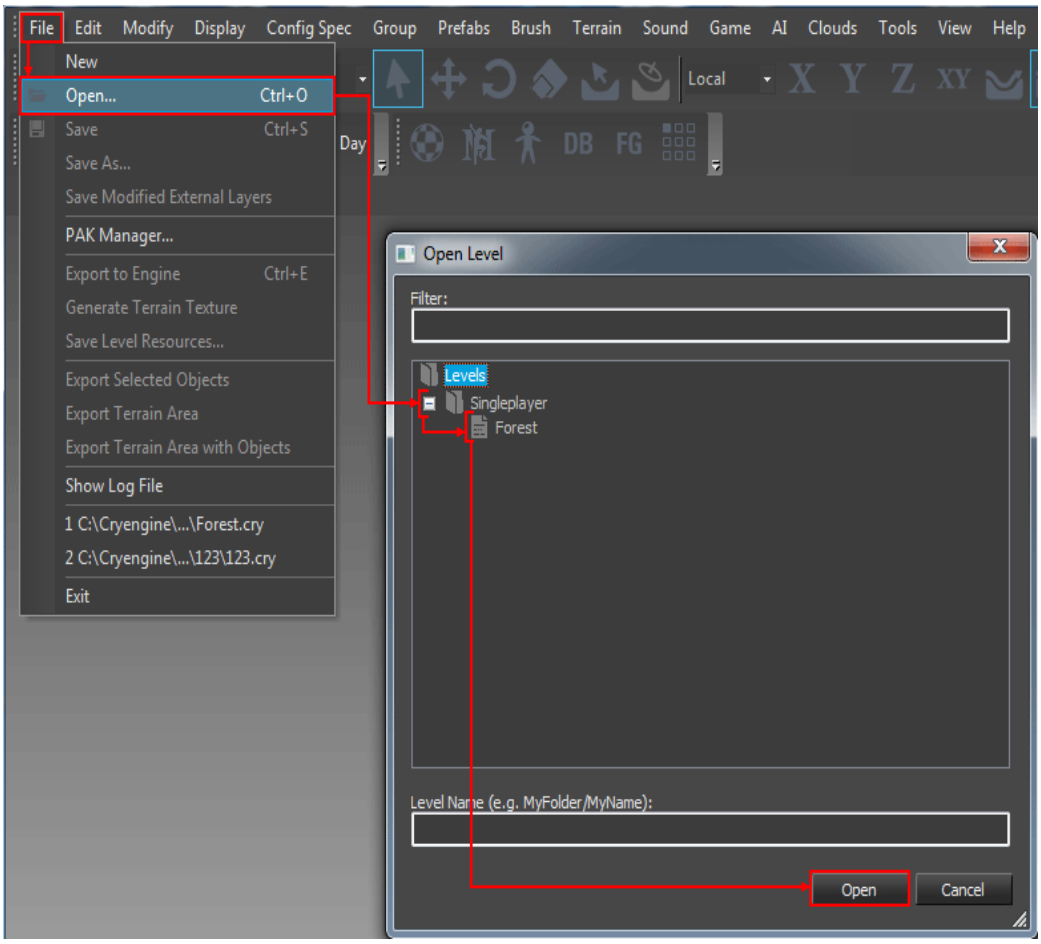

Рисунок 2.3 - Основний інтерфейс CryEngine 3

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 ігор, 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.

UDK-цебезкоштовнаверсіядвижкаUnrealEngine 3, щоволодієвсімуспадкованимінструментаріємостанньогодлястворенняігрових 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.

Основний інтерфейс движка UDKпоказаний на рис. 2.4.

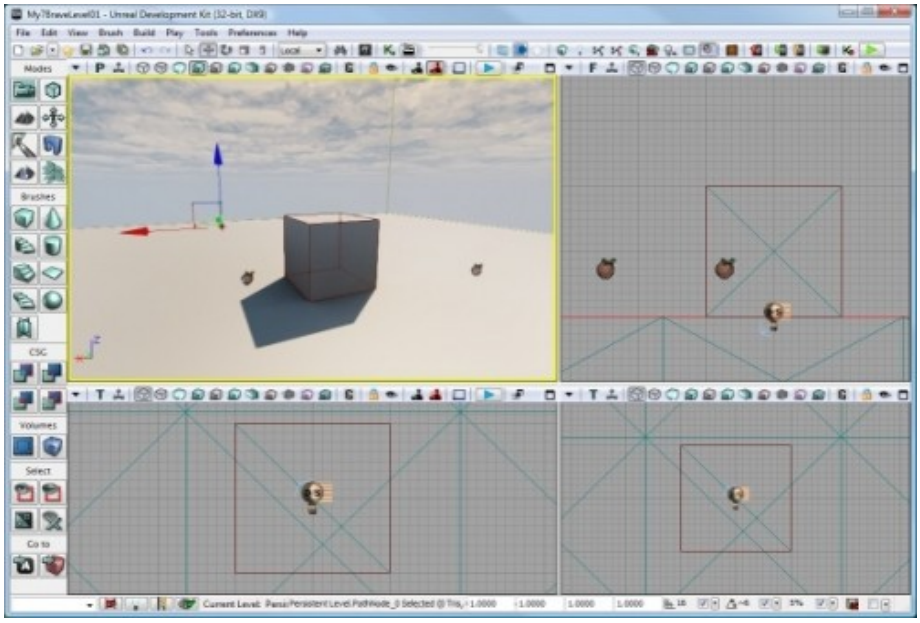

Рисунок 2.4 -ОсновнийінтерфейсUDK

На табл. 2.4 проведений перелік основних недоліків і переваг движкаUDK.

Таблиця 2.4 -ПеревагиінедолікиUDK

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). Движок був розроблений для заміни технічно застарілої технології Refractor Engine, яка використовувалася в попередніх іграх фірми.

Основний інтерфейс движка FrostbiteEngineпоказаний на рис. 2.5.

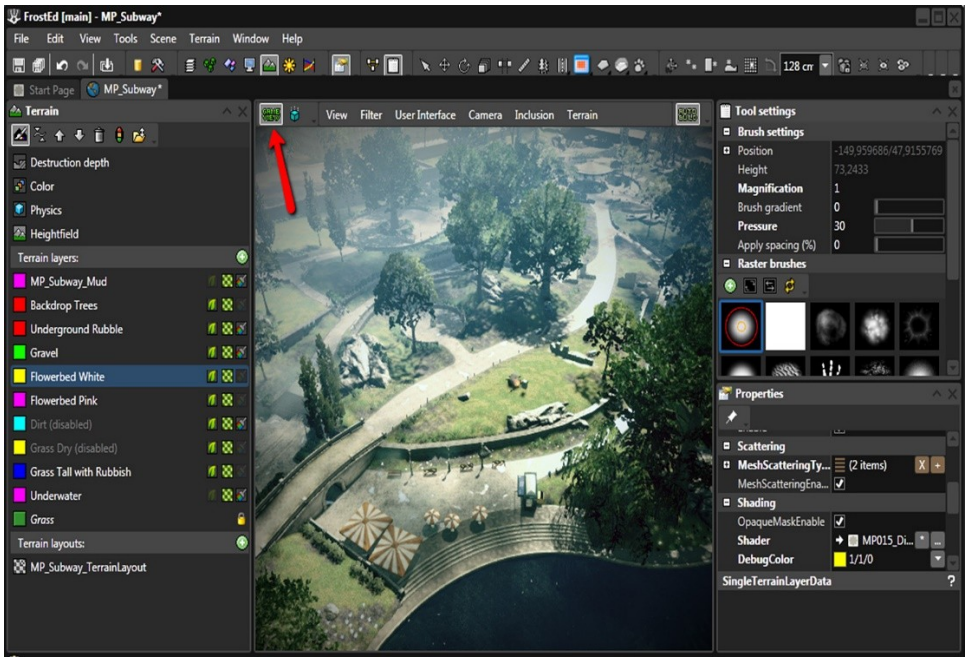

Рисунок 2.5 - Основний інтрефейс FrostbiteEngine

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 MicrosoftWindows ігровий движок підтримує відображення графіки за допомогою Mantleпочинаючи 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.

На табл. 2.5 проведений перелік основних недоліків і переваг движкаFrostbiteEngine.

Таблиця 2.5 -ПеревагиінедолікиFrostbiteEngine

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.

Згадана лише мізерна частина доступних для використання ігрових движків.

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

Таблиця 2.6-Допоміжна порівняльнахарактеристика усіх движків

Характеристика	Вимоги	Unity 3D	Torque 2D/3D	Cry Engine 3	UDK	Frostbite Engine
Режимрендеринга	3D	+	-	+	+	+
Звуковийдвижок		-	-	-	+	+
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 UnrealEngine, 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 OSX. Створені за допомогою Unity програми працюють під операційними системами Windows, OSX, WindowsPhone, Android, AppleiOS, Linux, а також на ігрових приставках Wii, PlayStation 3, PlayStation 4, Xbox 360, XboxOne і

MotionParallax3D дисплеях (пристрої для відтворення віртуальних голограм), наприклад, Nettlebox. Є можливість створювати додатки для запуску в браузерах за допомогою спеціального модуля Unity (UnityWebPlayer) (рис. 2.6), а також за допомогою реалізації технології WebGL (рис. 2.7).

Раніше була експериментальна підтримка реалізації проектів в рамках модуля AdobeFlashPlayer, але пізніше команда розробників Unity прийняла складне рішення щодо відмови від цього[12,13].

Рисунок 2.6 - Браузерна гра запущена за допомогою UnityWebPlayer

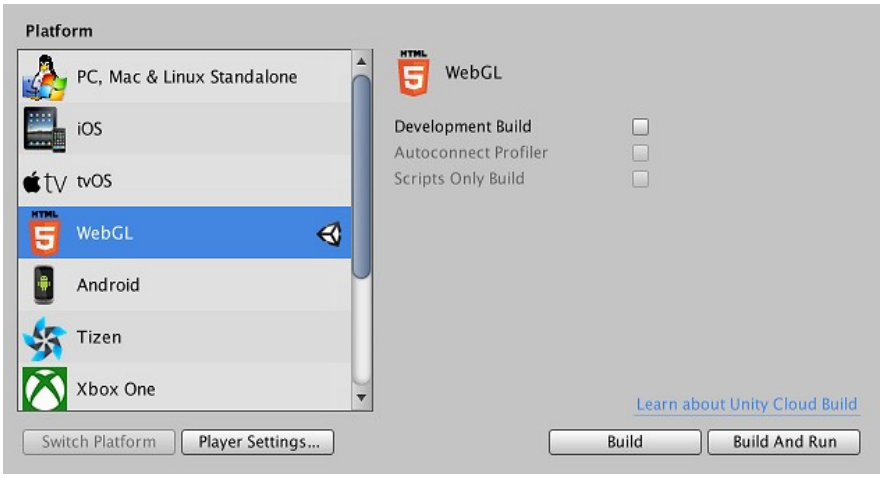

Рисунок 2.7 - Технологія WebGLдля запуску браузерних ігр

Додатки, створенізадопомогою Unity, підтримуютьDirectXiOpenGL. Активнодвижоквикористовуєтьсяяквеликимирозробниками (Blizzard, EA,

QuartSoft, Ubisoft), такірозробникамиIndie-ігор (наприклад, Pathologic, Kerbal, SpaceProgram, Slender: TheEightPages, Slender: TheArrival, SurgeonSimulator 2013, BaeklyseApps: Guesstheactortaінші) всилунаявностібезкоштовноїверсії, зручногоінтерфейсуіпростотироботиздвижком.

РедакторUnityмаєпростийDrag&Dropінтерфейс, якийлегконалаштовувати, щоскладаєтьсязрізнихвікон, завдякичомуможнапроводитиналагодженнягрипрямовредакторі. Движокпідтримуєтрисценарнихмови: C#, JavaScript (модифікація), Boo (діалектPython). Вооприбранийв 5-ійверсії. РозрахункифізикивиробляєфізичнийдвижоквідNVIDIA.

РозробкаAAA-проектіввUnity-найскладнішийпроцес. 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.

Проект в Unity ділиться на сцени (рівні) - окремі файли, свої ігрові світи, що містять, зі своїм набором об'єктів, сценаріїв, і налаштувань. Сцени можуть містити в собі як, власне, об'єкти (моделі), так і порожні ігрові об'єкти - об'єкти, які не мають моделі. Об'єкти, у свою чергу містять набори компонентів, з якими і взаємодіють скрипти.

Також у об'єктів є назва (у Unity допускається наявність двох і більше об'єктів з однаковим назвами), може бути тег (мітка) і шар, на якому він повинен відображуватися. Так, у будь-якого об'єкту на сцені обов'язково є присутнім компонент Transform - він зберігає в собі координати місця розташування, повороту, і розмірів об'єкту по усіх трьох осях. У об'єктів з видимою геометрією також за умовчанням є присутнім компонент Mesh Renderer, що робить модель об'єкту видимої.

До об'єктів можна застосовувати колізії (у Unity т. н. колайдери - collider). Існує декілька типів колайдерів:

- character controller - вид фізичної моделі, створений спеціально під використання його для ігрових персонажів;

- box collider (фізична модель утворює куб, в який потрапляє уся модель об'єкту);
- sphere collider (фізична модель утворює сферу, в яку потрапляє уся модель об'єкту);
- capsule collider (фізична модель утворює капсулу, в яку потрапляє модель об'єкту. На відміну від попереднього типу розміри можна міняти і по одній, і по трьох осях відразу);
- mesh collider (фізична модель повністю повторює реальну геометрію об'єкту);
- wheel collider (фізична модель колеса);
- terrain collider - тип фізичної моделі, створений спеціально для використання на об'єкті типу Terrain, - земля, генерована редактором Unity з можливостями скульптинга і фарбування місцевості.

Unity підтримує фізику твердих тіл і тканини, а також фізикові типу Ragdoll (ганчіркова лялька). У редакторові є система спадкоємства об'єктів; дочірні об'єкти повторюватимуть усі зміни позиції, повороту і масштабу батьківського об'єкту. Скрипти в редакторові прикріпляються до об'єктів у вигляді окремих компонентів[14].

Unity 3D підтримує систему Level Of Detail, суть якої полягає в тому, що на далекій відстані від гравця високодеталізовані моделі замінюються на менш деталізовані, і навпаки, а також систему Occlusion culling, суть якої в тому, що у об'єктів, не попадающиих у полі зору камери не візуалізується геометрія і колізія, що знижує навантаження на центральний процесор і дозволяє оптимізувати проект. При компіляції проекту створюється виконуваний файл гри (для Windows), а в окремій теці - ці ігри (включаючи усі ігрові рівні і бібліотеки, що динамічно підключаються).

При імпорті текстури в Unity можна згенерувати alpha -канал, mip -рівні, normal - map, light - map, карту віддзеркалень, проте безпосередньо на модель текстуру прикріпити не можна - буде створений матеріал, якому буде призначений шейдер, і потім матеріал прикріплятиметься до моделі. Редактор Unity підтримує написання і редагування шейдеров.

Редактор Unity має компонент для створення анімації, але також анімацію можна створити заздалегідь в 3D-редакторе і імпортувати разом з моделлю, а потім розбити на файли.

Окрім порожнього ігрового об'єкту і моделей, на сцену можна додавати такі об'єкти типу GameObject :

- система часток;

- камера;
- GUI текст;
- GUI текстура;
- 3D текст;
- точкове світло;
- спрямоване світло;
- освітлення території;
- джерело світла, що імітує сонце;
- стандартні примітиви;
– дерева;
– terrain (земля).

2.3 Основні поняття 3D графіки та Unity3D

Основні поняття 3D:

- Локальний простір vs Світовий простір.

Важливо розуміти різницю між локальним (Local space) і світовим простором (World space). У будь-кому 3D пакеті, технічно, ви працюєте в нескінченному просторі, і може бути скрутно, стежити за розміщенням об'єктів усередині нього. У кожному 3D світі є початкова точка (початок координат), часто звана, - нуль, оскільки її координати (0,0,0). Положення у світі (world positions) усіх об'єктів у 3D світі визначається відносно початку координат або світового нуля (world zero). Проте разом зі світовим ми використовуємо і локальний (об'єктне) простір (Local space або Object space), для визначення позиції об'єктів один відносно одного. Локальний простір припускає, що кожен об'єкт має власну точку відліку (нульову точку - zero point), з якої виходять осі об'єкту. Це звичайно (геометричний) центр об'єкту, і створюючи стосунки між об'єктами, ми можемо порівняти їх положення один відносно одного. Такі стосунки, відомі як батьківсько-дочірні стосунки (parent - child relationships), означають, що, використовуючи локальний простір, ми можемо вичислити відстань відносно батьківського об'єкту до інших об'єктів, в цьому випадку батьківський об'єкт стає новим початком координат для будь-якого з його дочірніх об'єктів.

- Вектори.

Як і 2D векторів, 3D векторами є лінію, тільки в 3D просторі, яка має напрям і довжину. Вектори можуть переміщатися у світовому просторі, але при цьому, самі не змінюються. Вектори грають велику роль в движку,

оскільки дозволяють нам обчислювати відстані, відносні кути між об'єктами, а також напрями об'єктів.

- Камери.

Камери мають велике значення для 3D світу, оскільки вони виступають в ролі viewport -а (вікна перегляду) для екрану. Регульована зона видимості (поле зору - Field of Vision (FOV)) камер має пірамідальну форму.

Камери можуть бути поміщені в будь-яку точку світу, бути анімовані, а так само прикріплені до персонажа або об'єкту, бути частиною сценарію гри. 3D камера - це ваше вікно (viewport) в 3D світ. Ви побачите, що в ігровому движку, різні ефекти, такі як lighting (освітлення), motion blurs (ефект розмитості зображення при відтворенні сцен руху) і інші, застосовуються до камери, щоб зімітувати реальне оточення, оскільки його сприймає людське око, можна також додати і кінематографічні ефекти (ефекти які не випробовує людське око) такі як відблиски (lens flares) від сонця.

Більшість сучасних 3D ігор використовують декілька камер, щоб показати частину ігрового світу, яку камера персонажа в даний момент не бачить. Для Unity це не складно, він (ігровий движок) дозволяє розміщувати декілька камер в одній сцені, які можуть бути налагоджені так, щоб була можливість використовувати будь-яку з них, як основну (головну) камеру (main camera), під час гри. Декількох камер можуть бути використані в грі, щоб управляти візуалізацією (rendering) окремих 2D і 3D елементів, як метод оптимізації. Наприклад, об'єкти можуть бути згруповані по шарах (layers), а камери можуть бути призначені на візуалізацію (render) об'єктів в окремих шарах. Це дає нам більше контролю над індивідуальною візуалізацією (individual renders) певних елементів в грі[15].

- Полігони, грані, вершини і меши.

При побудові 3D фігур, усі об'єкти, кінець кінцем, складаються з пов'язаних 2D фігур - званих полігонами (polygons). При імпорті з 3D застосувань, Unity перетворить усі полігони об'єктів в трикутні полігони (polygon triangles). Трикутні полігони(так само звані faces), у свою чергу, складаються з трьох граней (edges). Місця, в яких ці грані (edges) зустрічаються, називаються точками або вершинами (points або vertices).

Знаючи ці місця, ігровий движок здатний зробити розрахунки точок удару, відомих як collisions (зіткнення), використовуючи складне виявлення зіткнень (complex collision detection) з Mesh Colliders, в шутерах, можливо визначити точне місце, в якому куля уразила інший об'єкт. Завдяки поєднанню пов'язаних полігонів 3D редактори дозволяють створювати

складні фігури, звані мешами (meshes). Дані, що зберігаються в мешах (meshes), можуть використовуватися для ряду інших завдань, відмінних від побудови 3D фігур.

У ігрових проектах, дуже важливо для розробника зрозуміти важливість кількість полігонів (polygon count). Кількість полігонів - це загальне число полігонів, часто по відношенню до моделі, але і з посиланням на увесь рівень гри. Чим більше кількість полігонів, тим більше роботи комп'ютеру необхідно зробити, щоб візуалізувати (render) об'єкт на екрані. Розробникові ігор необхідно знайти баланс між кількістю полігонів (деталізація моделей, кількість об'єктів на рівні і так далі) і продуктивністю візуалізації (render).

- Матеріали, текстури і шейдери.

Матеріали (Materials) загальне поняття для усіх 3D застосувань, оскільки вони забезпечують засоби для створення зовнішнього вигляду 3D моделей. Від базових кольорів до відзеркалювальних поверхонь (reflective image - based), матеріали здатні на усе. Починаючи з простих кольорів і можливість використовувати один або декілька зображень - відомих як текстури (textures) - в одному матеріалі, і закінчуючи, працюй з шейдерами (shader), чий код відповідає за стиль візуалізації (rendering). Наприклад, при використанні шейдеров віддзеркалення (reflective shader), матеріал відбиватиме навколишні предмети, але при цьому зберігають свій колір і зовнішній вигляд зображення, вживаного в якості текстури. У Unity, використовувати матеріали легко. Будь-які матеріали створені в 3D редакторах, будуть імпортовані і автоматично відтворені ігровим движком в якості ресурсу (assets), для подальшого застосування. Ви так само можете створити свій власний матеріал "з нуля", призначивши зображення з файлу текстури і вибравши потрібний шейдер з великої вбудованої бібліотеки. Ви так само можете написати власний скрипт шейдера або використовувати створені членами співтовариства (community) Unity. Важливо відмітити, що при створенні текстури для гри в графічних редакторах, ви маєте бути обізнані про цю угоду. Текстури мають бути квадратними і мати розмір рівний мірі двох[16,17].

2.4Середовище моделювання 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» (NodeCompositing), 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:

- підтримкарізноманітнихгеометричнихпримітивів, включаючиполігональнімоделі, системушвидкогомоделюванняврежимі subdivision surface (SubSurf), кривіБезьє, поверхні NURBS, metaballs (метасфери), скульптурнемоделюваннятавекторнішрифти;
- універсальнівбудованімеханізмирендерингаіінтеграціяззовнішнімре ндерерYafRay, LuxRenderібагатьмаіншими;
- інструментианімації, середякихінверснакінематика, скелетнаанімаціяісітковадеформація, ключовікадри, нелінійнаанімація, редагуванняваговихкоефіцієнтіввершин, обмежувачі, динамікам'якихтіл (включаючивизначенняколізійоб'єктівпривзаємодії), динамікатвердихтілнаосновіфізичногодвижкаBulleticистемаволоссянаосновічастинок;
- 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.
- blenderGameEngine- 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[19].

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:

- Режимиредагування. ДваосновнірежимиОб'єктнийрежим (Objectmode) іРежимредагування (Editmode), якіперемикаютьсяклавішеюTab. 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 VertexPaint та UVFaceselect;
- 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, UVmapping 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 UnrealTournament.

РобочийпростірBlender'авважаєтьсяоднимзнайбільшноваторськихконцепційграфічногоінтерфейсудляграфічнихінструментівінатхненнимдизайномграфічногоінтерфейсупатентованихпрограм, такихякLuxology'sModo[20].

Додаткові особливості:

- упрограмі Blender сутність, щовзаємодієзнавколишнімсвітомііїдані (формаабофункціїоб'єкта) разделяемості. СтавленняОб'єкт-Даніпредставляєтьсявідношенням 1: n (термін, щовідноситьсядотеоріїбазданих, позначаєможливістьдекількохоб'єктіввикористовуватиодніітіждані - одиндобагатьохабосюрьекція);
- 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 bumpmaps, зміна може бути зроблено в інтерактивному режимі, і всі зміни можна реальному часі.

Програма Blender призначена для тривимірного моделювання за допомогою різних вбудованих інструментів. Даний програмний продукт є повністю безкоштовним. Розробники регулярно випускають оновлення, які

включають в себе нові можливості, виправлені недоробки, вдосконалені раніше створені механізми моделювання[21].

Програма не вимагає реєстрації на сайті або створення користувача локально. Після завантаження та встановлення програми Blender, її відразу можна запустити і створювати тривимірні об'єкти за допомогою інструментів самого середовища розробки (рис 2.7):

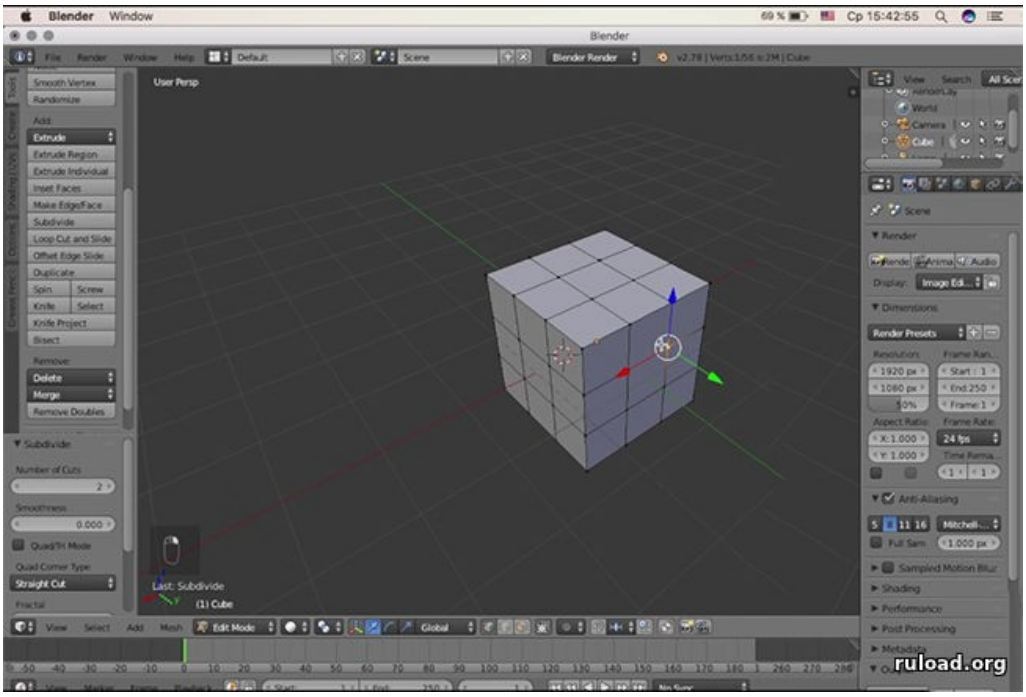

Рисунок 2.7 - Інтерфейс програми Blender

У верхній частині екрану призначеного для користувача інтерфейсу розміщується кілька вкладок для управління файлами, вікнами, настройками і для доступу до довідкової системи.

Нижче розташовується вікно сцени, де відображуються моделюються об'єкти. По краях можна розміщувати різні допоміжні вікна, які надає програма Blender.

Середовище розробки включає можливість створення простих об'єктів: куб, площину, сфера, циліндр, конус тощо. За допомогою даних об'єктів проводиться процес моделювання більш складних об'єктів, за допомогою використання об'єднання, видалення, заміщення, переміщення, зміни розмірів окремих полігонів моделей.

Всі створені об'єкти, які знаходяться на сцені і їх ієрархію можна переглянути у вікні Outliner. Крім цього у вкладці Modifier є велика кількість вбудованих функцій (модифікаторів), які можна застосувати до

модельованого об'єкту. Кожна з них має певні властивості і вимоги до об'єкта, на який накладається відповідна функція. Застосовуючи необхідний модифікатор до об'єкта, полігони автоматично змінюють свої фізичні і геометричні властивості, які залежать від значень відповідного модифікатора. При необхідності, дані значення можна змінювати.

У вкладці Physics модельованого об'єкту можна привласнити деякі фізичні значення: масу, структуру (рідина, твердий матеріал, дим), опір і багато іншого. Кожне з властивостей гнучко настроюється користувачем.

На основі заданих фізичних властивостей у вкладці Particle є можливість створювати об'єкти, які складаються з безлічі частинок малих розмірів. Такими об'єктами можуть бути: краплі дощу, град, зірки, оскільки вибуху або зіткнення, пил тощо. Додаючи систему частинок певного об'єкту можна досить гнучко її налаштувати під певний тип розробляється моделі.

Програма Blender дозволяє накладати текстуру на об'єкт і на окремі полігони. Дані маніпуляції можна зробити у вкладці Material, де можна створити матеріал для певного об'єкта і налаштувати його в разі потреби. Після чого об'єкт набуває відповідний зовнішній вигляд.

Після створення необхідної моделі з усіма необхідними ефектами і елементами користувач може отримати двовимірну картинку за допомогою рендеру.Дана дія відбувається у вкладці Render. Вказавши необхідні формат картинки, її дозвіл і якість комп'ютер почне розрахунок і створить готове зображення. Дане зображення можна використовувати як звичайну картинку, а можна імпортувати в гру як елемент управління призначеним для користувача інтерфейсом або як завантажувальний заставку.

У деяких випадках моделювання створення анімації є необхідністю. Анімації можуть бути простими, наприклад, пересування. А можуть бути і складними, де окрема частина об'єкта пересувається незалежно від інших частин. Для цього в програмі Blender є можливість створювати кістки, які в подальшому об'єднуються в скелет і зв'язуються безпосередньо з модельований об'єктом. Після чого рух окремих кісток призводять до руху відповідних частин об'єкта.

Дана процедура в основному необхідна для об'єктів, що імпортуються в ігри. Перед імпортуванням в середу розробки ігор програма Блендер експортує модель в обраний користувачем формат. Блендер підтримує експорт кінцевого файлу о восьмій різних форматів (.dae, .3ds, .fbx, .bvh, .ply, .obj, .x3d, .stl). Після визначення необхідного формату, користувач має можливість відповідним чином налаштувати

експортований об'єкт.

Дана програма підходить для розробки комп'ютерних ігор в зв'язку з тим, що вона безкоштовна, постійно доопрацьовується і включає в себе безліч функцій, які необхідні для моделювання ігрових об'єктів[22].

2.5Функціональні можливості редактора Blender 3D

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

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

Початковий екран Blender показаний на рис. 2.8.

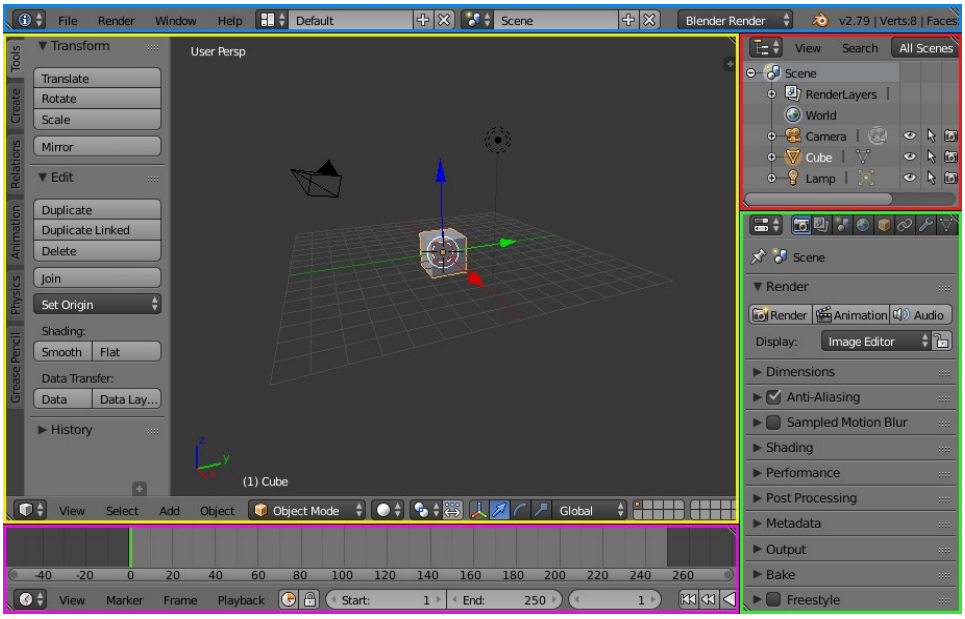

Рисунок 2.8 - Стандартний формат екрана Blender з п’ятьма редакторами

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 віджетих[23].

Компоненти редактора показані на рис. 2.9.

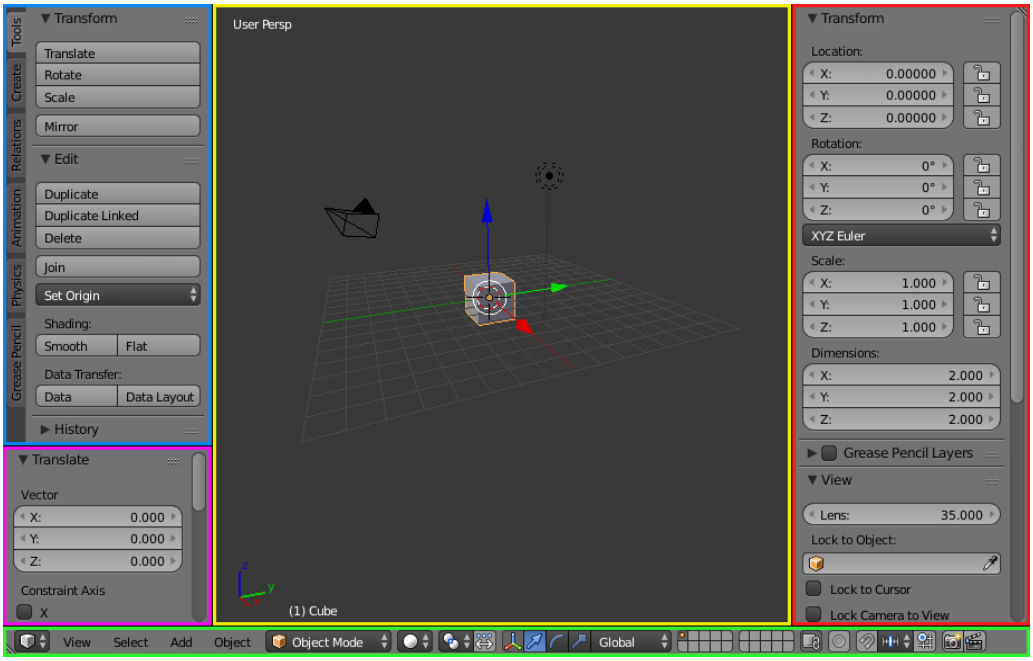

Рисунок 2.9 - Редактор 3DView (3D вид)

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.

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.

Контексте меню зображене на рис. 2.10.

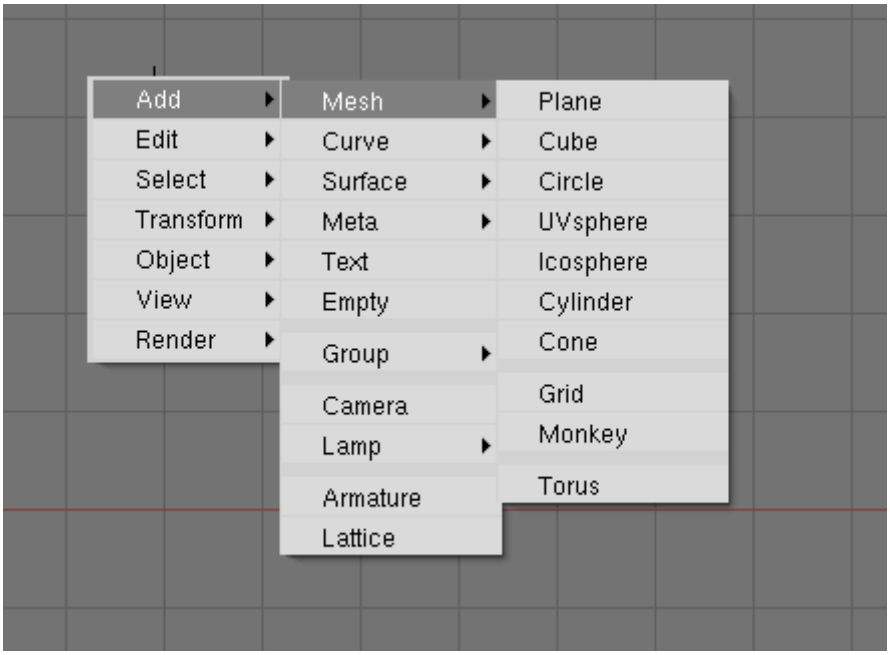

Рисунок 2.10 - Контексте меню що містить список різних дій

Найперший пункт - це так звані меш-об'єкти (Mesh), куди входять площина (Plane), куб (Cub), коло (Circle), сфера (UVsphere), геосфера (Icosphere), 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.

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.

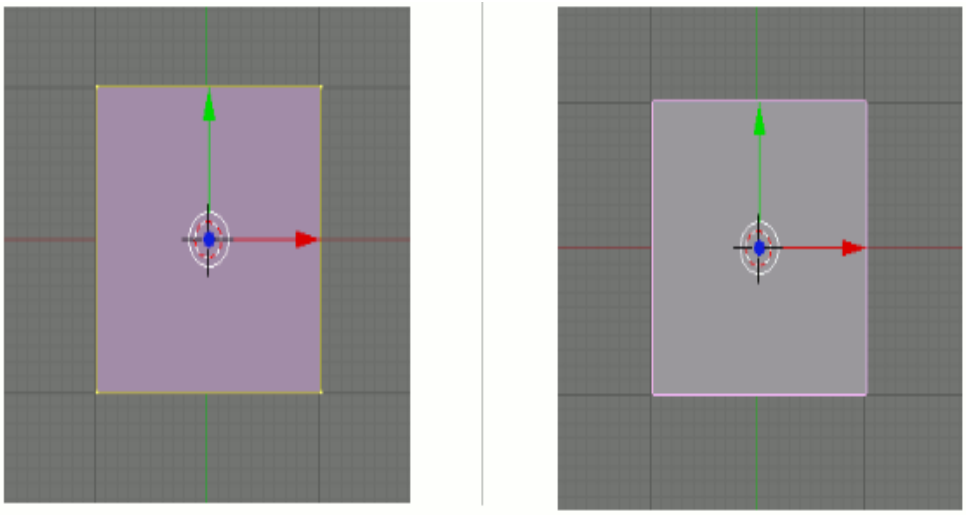

Рисунок 2.11 - Створений в режимі редагування об’єкт

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.

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

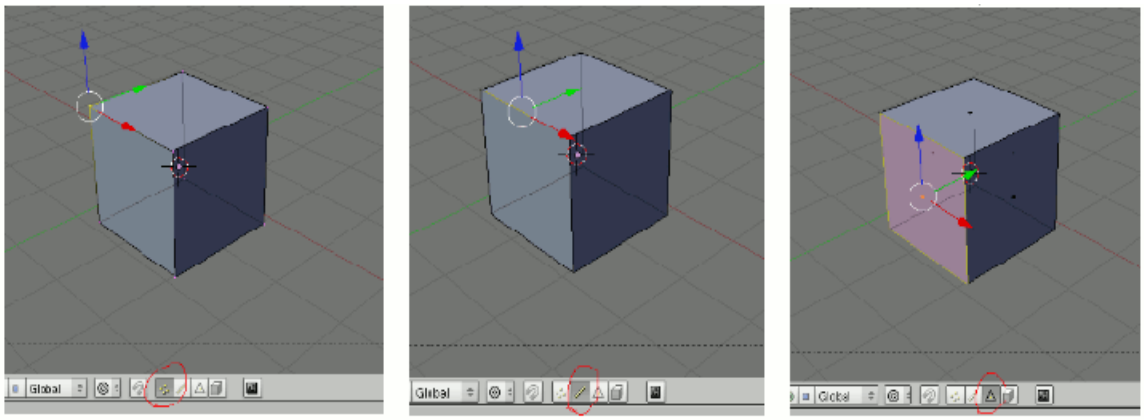

Рисунок 2.12 - Три режими редагування об’єктів

3 РОЗРОБКА АНІМАЦІЇ ОБ'ЄКТІВ ІНТЕРАКТИВНОЇ МОДЕЛІ

3.1 Комп’ютерне анімування

Комп'ютерна анімація - вид мультиплікації, створюваний за допомогою комп'ютера. На відміну від більш загального поняття «графіка CGI», що відноситься як до нерухомих, так і до рухомих зображень, комп'ютерна анімація має на увазі тільки рухомі. На сьогодні набула широкого застосування як в області розваг, так і у виробничій, науковій та діловій сферах. Будучи похідною від комп'ютерної графіки, анімація успадковує ті ж способи створення зображень:

1) векторна графіка;

2) растрова графіка;

3) фрактальна графіка;

4) тривимірна графіка (3d).

Існує безліч видів анімації об'єктів:

1) Анімація по ключовим кадрам. Розстановка ключових кадрів проводиться аніматором. Проміжні ж кадри генерує спеціальна програма. Цей спосіб найбільш близький до традиційної мальованої мультиплікації, тільки роль фазовщіка бере на себе комп'ютер, а не людина.

2) Запис руху. Дані анімації записуються спеціальним обладнанням з реально рухаються об'єктів і переносяться на їх імітацію в комп'ютері.

Поширений приклад такої техніки - Motion capture (захоплення рухів). Актори в спеціальних костюмах з датчиками здійснюють руху, які записуються камерами і аналізується спеціальним програмним забезпеченням. Підсумкові дані про переміщення суглобів і кінцівок акторів застосовують до тривимірним скелетам віртуальних персонажів, ніж домагаються високого рівня достовірності їх руху. Такий же метод використовують для перенесення міміки живого актора на його тривимірний аналог в комп'ютері.

3) Процедурна анімація. Це вид комп'ютерної анімації, який автоматично генерує анімацію в режимі реального часу відповідно до встановлених правил, законів і обмежень. На відміну від визначеної анімації, коли аніматор вручну визначає кожен кадр і всі параметри створюваної анімації, при процедурної анімації результат може бути в деякій мірі непередбачуваний і при кожному запуску може генерувати різноманітну анімацію. Процедурна анімація використовується для створення та моделювання системи частинок (дим, вогонь, вода), тканини і одягу, динаміки твердих тіл, динаміки волосся та хутра, а також для анімації гуманоїдних і негуманоїдних персонажів. У комп'ютерних іграх процедурна анімація часто використовується для таких простих речей, як повертання голови персонажа, коли гравець озирається на всі боки.

Фізика Ragdoll може вважатися видом процедурної анімації. Фізика Ragdoll використовує фізичний движок для створення анімації смерті персонажа і його реалістичного падіння на поверхню. При використанні фізики Ragdoll персонаж складається з послідовності пов'язаних твердих тіл (руки, ноги, торс, голова і т. д.), Які запрограмовані з використанням ньютонової фізики, що діє на них. Завдяки фізики Ragdoll можуть бути створені дуже реалістичні анімації, які дуже складно створити, використовуючи традиційну зумовлену анімацію. Наприклад, з використанням фізики Ragdoll тіла персонажів можуть котитися або ковзати по похилих поверхнях, перевертатися, перекидатися і т.д., причому кожен раз анімація буде інша.

4) Шейпова анімація. Назва походить від англійського слова Shape. Переклад означає фігуру або анімацію геометричних фігур. Добре виконані анімації не залишають глядача байдужим, захоплюючи його гіпнотично в те, що відбувається. Прості фігури, випадково з'являючись з нізвідки, також зникаючи - трансформуються один в одного. При цьому сенс відео послання ясний кожному.

5) Програмована анімація. Широке застосування в мережі отримали дві мови, за допомогою яких програмуються руху анімованих об'єктів:

- JavaScript - браузерні мову;
- ActionScript - мова роботи з додатками Flash.

Перевага програмованої анімації - в зменшенні розміру вихідного файлу, недолік - навантаження на процесор клієнта.

6) Створення анімації за допомогою цифрового фотоапарата. Сьогодні програмне забезпечення, що дозволяє задіяти цифровий фотоапарат для зйомки анімації, застосовується також часто, як і звичні 3D- або 2D-пакети. Будь-яка програма такого типу забезпечує управління цифровим фотоапаратом через комп'ютер і роботу з отриманими кадрами[24].

3.2 Основні поняття анімування та створення фізики в Unity

Система анімації в Unity дозволяє створювати чудово анімованих персонажів. Вона підтримує блендінг, мікшування, додавання анімацій, синхронізацію циклу ходьби, анімаційні шари, контроль всіх аспектів програвання (час, швидкість, ваги блендінга), скіннінг мешів з 1, 2 або 4 кістками на вершину, а також засновані на фізиці rag-dolls і процедурну анімацію.

Створення анімованого персонажа включає в себе дві речі - переміщення в просторі сцени і відповідна анімація. Компонент Character Controller (рис.3.1та рис. 3.2) в основному використовується для управління від третьої або першої особи, де не потрібно фізика Rigidbody.

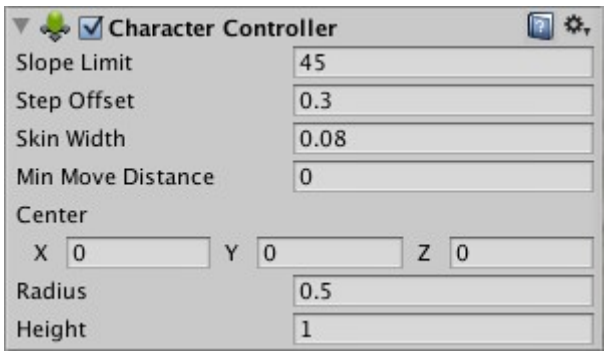

Рисунок 3.1 -Компонент Character Controller

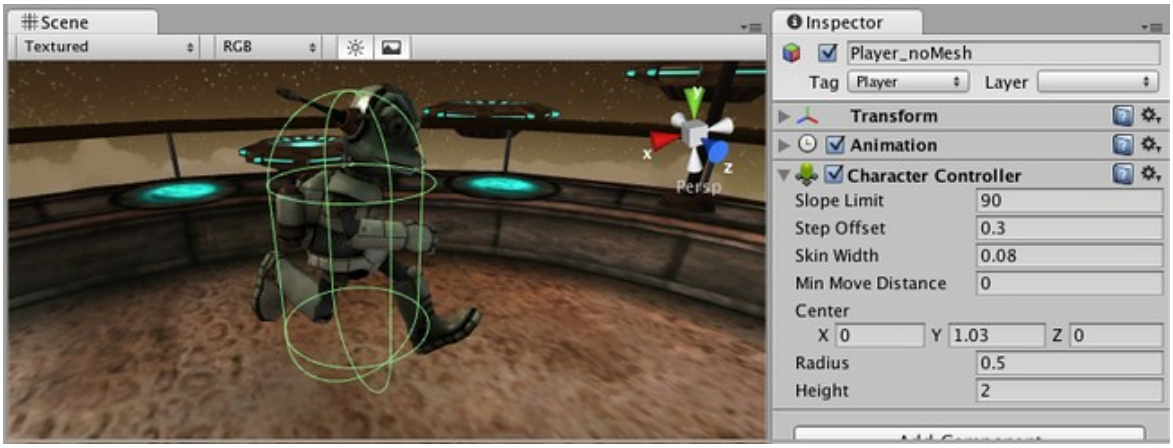

Рисунок 3.2 - Приклад компоненту Character Controller

Традиційне управління від першої особи в стилі Doom не реалістичні, з точки зору фізики. Персонаж біжить зі швидкістю 90 миль на годину, зупиняється миттєво і повертається за лічені миті. Оскільки це нереалістично, використання Rigidbody і фізики щоб створити подібне управління непрактично і ефект буде не той. Рішенням є спеціалізований Character Controller. Він просто створює коллайдер (Collider) у формі капсули, який можна рухати в деякому напрямок за допомогою скрипта. Controller подбає про рух, але зіткнення його стримують. Він буде ковзати уздовж стін, підніматися по сходах (якщо сходинки нижче ніж Step Offset) і ходити по нахилам враховуючи Slope Limit. Controller не реагує на сили сам по собі і не відштовхує Rigidbody об'єкти автоматично.

Якщо необхідно штовхати Rigidbody-тіла або об'єкти, що використовують Character Controller, можна застосувати сили до будь-якого об'єкту, в якого він врізається, через скриптинг за допомогою функції OnControllerColliderHit ().

З іншого боку, якщо ви хочете, щоб персонаж гравця піддавався впливу фізики, то було б краще використовувати Rigidbody замість Character Controller.

Rigidbody - це основний компонент, що включає фізичне поводження для об'єкта (рис. 3.3). З прикріпленим Rigidbody, об'єкт негайно почне реагувати на гравітацію. Якщо доданий один або кілька компонентів Collider, то при колізіях (зіткненнях) об'єкт буде пересуватися.

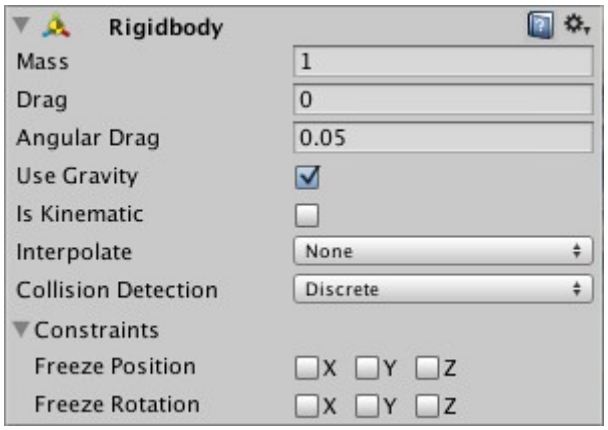

Рисунок 3.3 - Компонент Rigidbody

Так як компонент Rigidbody управляє переміщенням об'єкта, до якого він прикріплений, вам не слід намагатися впливати на об'єкт з коду за допомогою зміни таких властивостей Transform, як position і rotation. Замість цього вам слід застосовувати сили для того, щоб штовхати об'єкт і дозволити фізичній движку розрахувати результати.

Існують випадки, при яких ви можете захотіти, щоб у об'єкти був Rigidbody, але фізичний движок не керував його рухом. Наприклад, ви можете побажати управляти своїм персонажем безпосередньо з коду, але при цьому хочете, щоб він взаємодіяв з тригерами. Цей тип не фізичного руху, вироблений за допомогою коду, називається кінематичним рухом. У компонента Rigidbody є властивість Is Kinematic, яке може виключити об'єкт з-під контролю фізичного движка, і дозволити переміщати його кінематично з скрипта. Значення Is Kinematic можна міняти з коду, щоб ввімкнути або вимкнути фізику для об'єкта, але ця можливість вимагає додаткових ресурсів і повинна використовуватися як можна рідше.

Тверді тіла дозволяють вашим ігровим об'єктам взаємодіяти за допомогою фізики. Для реалістичного переміщення твердих тіл, на останні впливають сила обертання і інші сили. Будь-ігровий об'єкт повинен містити в собі тверде тіло, щоб бути схильним до гравітації, діяти відповідно до призначеним шляхом скриптинга силам, або взаємодіяти з іншими об'єктами через фізичний движок NVIDIA PhysX.

Колайдери це інший тип компонентів, які повинні бути додані разом з твердими тілами, щоб задіяти зіткнення. Якщо два твердих тіла врізаються один в одного, фізичний движок не буде прораховувати зіткнення, поки до обох об'єктів не буде призначений коллайдер (рис. 3.4). Тверді тіла не мають

коллайдеров будуть просто проходити крізь один одного при прорахунку зіткнень.

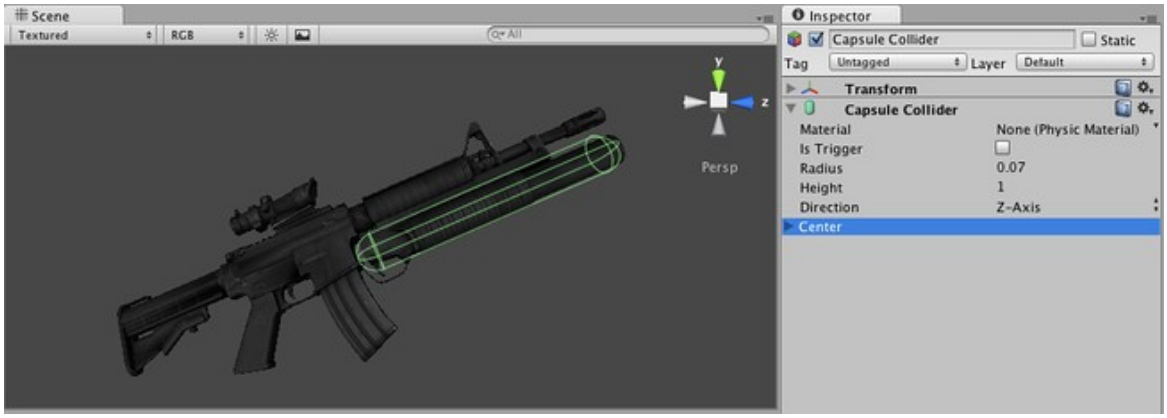

Рисунок 3.4 - Приклад колайдеру

Складові колайдери - це комбінації примітивних коллайдеров, які спільно діють як один коллайдер (рис. 3.5). Вони прігождаются, коли у вас є модель, яка буде занадто складною або дорогої з точки зору продуктивності, щоб точно імітувати, і хочете імітувати зіткнення фігури оптимальним способом, використовуючи прості наближення. Щоб створити об'єднаний коллайдер, створіть дочірні об'єкти вашого стикається об'єкта, а потім додайте компонент коллайдера до кожного дочірньому об'єкту. Це дозволяє легко і незалежно один від одного позиціонувати, повертати і масштабувати кожен коллайдер. Ви можете побудувати складний коллайдер з ряду примітивних коллайдеров і / або опуклих колайдерів.

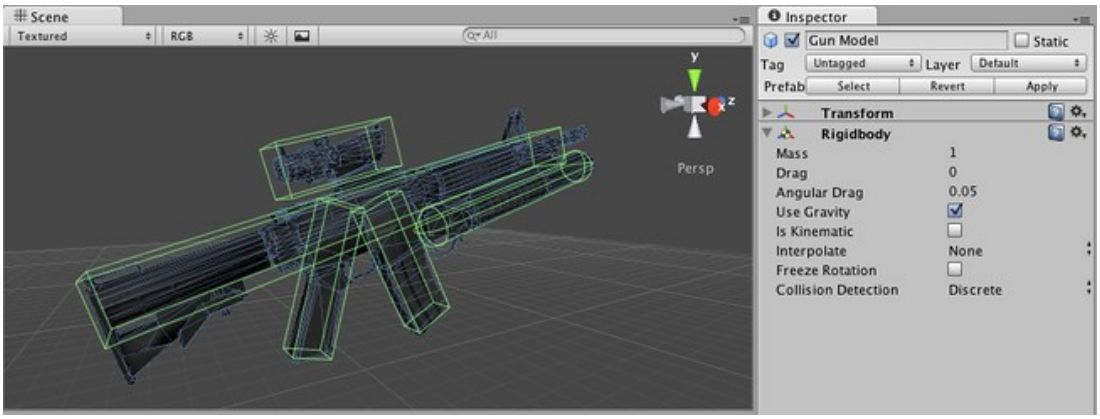

Рисунок 3.5 - Приклад складового колайдеру

На рис.3.5 ігровий об'єкт Gun Model з призначеним твердим тілом і

декількома примітивними колайдерами як нащадків ігрового об'єкта. При русі Rigidbody батька під впливом сил, нащадки колайдери будуть рухатися разом з ним в тому ж напрямку. Примітивні колайдери будуть стикатися з меш колайдер навколишнього середовища, і батьківський Rigidbody змінить спосіб свого переміщення, ґрунтуючись на силах, застосованих до нього і тому, як його нащадки колайдери взаємодіють з іншими колайдері в сцені.

Меш колайдери не можуть нормально стикатися один з одним. Якщо меш коллайдер позначений як Convex, тоді він може стикатися з іншим меш колайдером. Звичайним виходом із ситуації буде використання примітивних коллайдеров для будь-яких рухомих об'єктів, і меш коллайдеров для статичних другорядних об'єктів [25].

3.3 Скриптування та аналіз об’єктів моделі

Завданням дипломного проекту було розробка інтерактивної моделі на рушії Unity3D, тобто скриптування та взаємодія с різними об’єктами гри на сцені, таких як головний персонаж, локатори, атмосферні явища.

Сцена – це локація в грі, у якій перебуває головний персонаж. В різних іграх сцени можуть виконувати різні функції та мають певні характеристики. За ними їх можна умовно класифікувати:

- прості або лінійні (де мета — просто дійти з пункту А до пункту Б) та ускладнені (проходячи, здійснити певні дії: вбити всіх ворогів, віднайти якийсь предмет тощо);
- обмежені та необмежені в часі; якщо гравець не встигне пройти обмежений у часі рівень, це зараховується як поразка;
- сюжетні (обов'язкові для проходження гри) та бонусні або додаткові (звичайно це секретні рівні, які не є обов'язковими для проходження, містять додаткові бонуси).

Кожна сцена завантажується з використанням наступного коду: Application.LoadLevel(“Назва сцени, або його порядковий номер”);

Головна сцена (рис. 3.6, рис 3.7)зробленавідкритою місцевістюдля більшої схожості з реальністю.

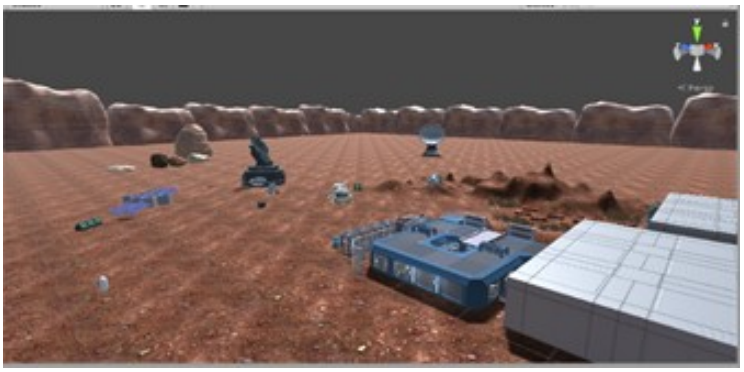

Рисунок 3.6– Головна сцена

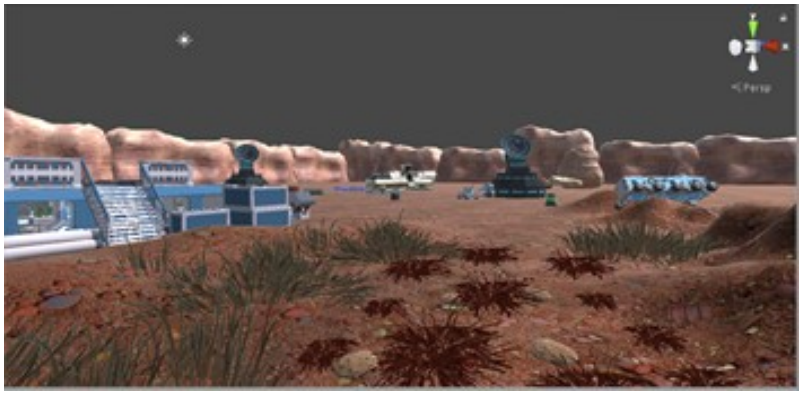

Рисунок 3.7 - Місцевість у локації

У Unity3D для створення рівнів(сцен) використовуються різні геометричні фігури такі як: сфера, куб, конус, капсула, площина та модифікація площини яка може приймати будь яку форму та має назву називають полігонна сітка (Polygon mesh).

Полігональна сітка (англ. Polygon mesh) — це набір вершин, ребер, та граней, що описують форму багатогранного об'єкта в тривимірній графіці та твердотільному моделюванні. Грані зазвичай складаються з трикутників, чотирикутників, чи інших опуклих многокутників, що спрощує їх рендеринг, хоча можуть використовуватись і загальніші, неопуклі багатогранники, чи багатогранники з дірками.

В першому рівні я використав полігонні сітки створені за допомогою 3D редактора Blender.Приклад фігур можна побачити на рис. 3.8 та рис. 3.9.

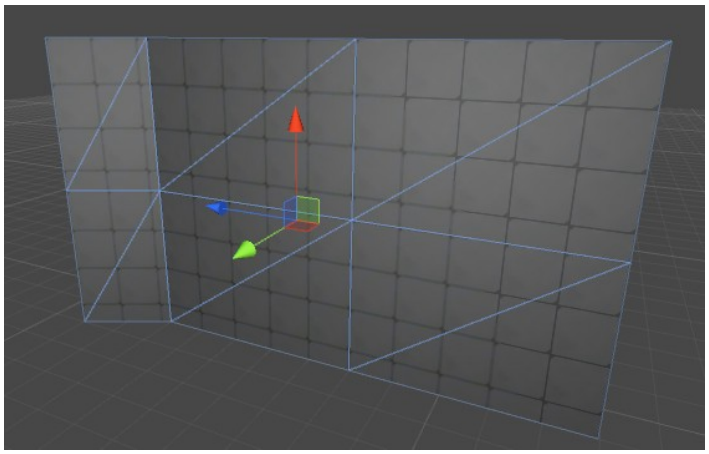

Рисунок 3.8- Приклад фігури із полігонної сітки

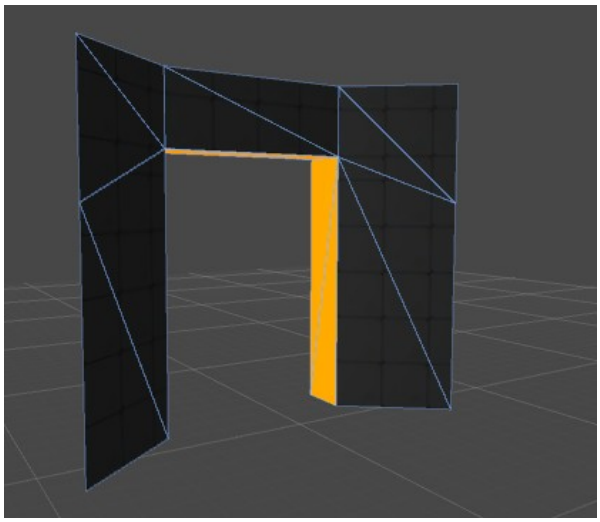

Рисунок 3.9- Приклад фігури із полігонної сітки

На рис. 3.10 зображений приклад локатору , після повної реалізації якого він буде повертатися після ввімкнення.

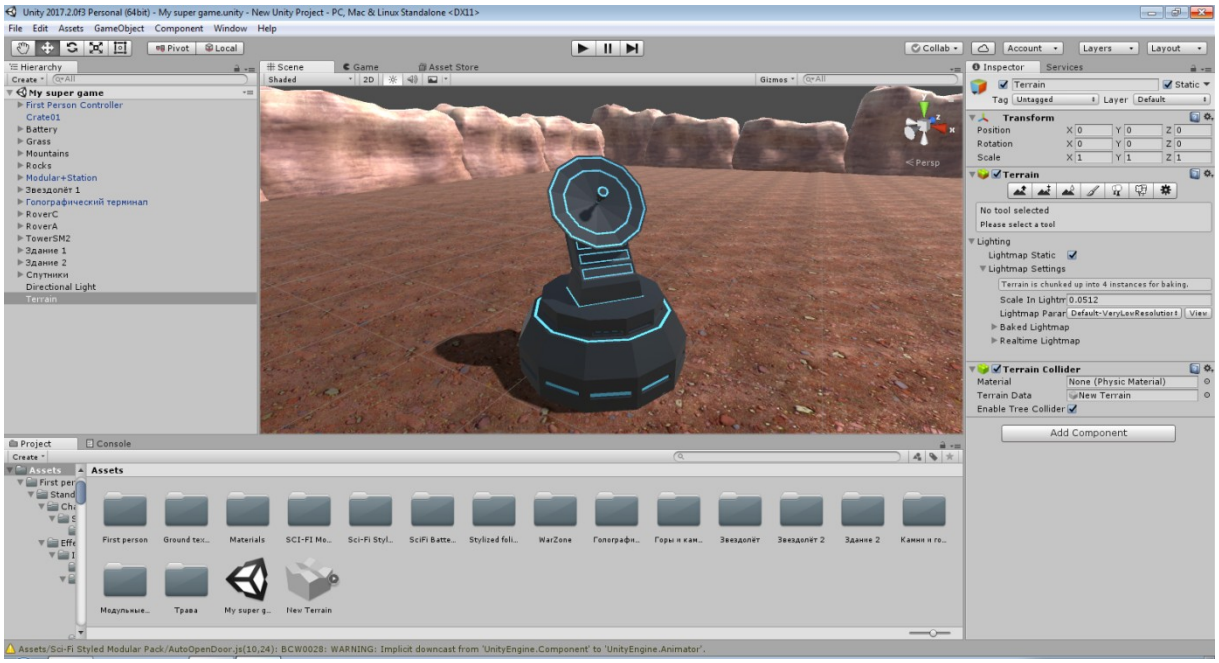

Рисунок 3.10 - Приклад локатору

Код занесений у клас MonoBehaviour, де і виконує свою роботу за допомогою методу Update та компонентів (класів), таких як:

- Transform - кожен об'єкт в сцені має свій компонент Transform. Він використовується для зберігання і обробки позиціювання, повороту і масштабу об'єкта. Кожен компонент Transform може мати батька, який дозволяє застосовувати положення, повороту і масштабування ієрархічно.
- Vector3 - цей клас використовується для створення тимчасових об’єктів, або може служити аргументом для методів класа Transform у тривимірному просторі. Він також містить функції для виконання спільних векторних операцій.

Щоб була взаємодія між скриптом та рушієм гри, на початку коду підключаємо наступні простори назв:

- UnityEngine – дає нам знати, що ми працюємо з Unity
- System.Collections – містить базові методи та об’єкти мови програмування C#.

Важливе значення у грі займає такий компонент як Rigibody – це об’єкт, який контролює положення об’єкта через симуляцію фізики. Rigibody прораховує, як предмети будуть падати у низ під дією сили тяжіння, і обчислює як вони будуть реагувати на зіткнення. При маніпулюванні з параметрами Rigibody, я працює у середині функції FixedUpdate.

Моделювання фізики здійснюється в дискретних тимчасових кроків. Функція FixedUpdate викликається безпосередньо перед кожним кроком.

Відтворення звуку здійснюється за допомогою двох компонентів:

- AudioSource – компонент, який відтворює звуки. Може бути здійснений як 2D, або 3D, тобто звуки відтворенні у 2D чуються завжди, у 3D – залежно від того, де знаходиться.
- AudioListener – компонент, який обробляє звуки, відправлені компонентом AudioSource.

При написані структури гри, для відтворення графічного інтерфейсу я використовую компоненти GUI Text, GUI Texture, 3D Text, які беруть за основу можливості батьківського класу – GUI. Тобто графічний інтерфейс користувача — тип інтерфейсу користувача, який дозволяє користувачам взаємодіяти з електронними пристроями через графічні зображення та візуальні вказівки, на відміну від інтерфейсів заснованих на використанні тексту, текстовому наборі команд та текстовій навігації.

Переваги GUI:

- графічний інтерфейс є «дружнім» для користувачів, котрі розпочали знайомство з комп'ютером з графічного інтерфейсу;
- в програмах обробки графіки він найчастіше є єдино можливим.

Недоліки GUI:

- більше споживання пам'яті в порівнянні з текстовим інтерфейсом;
- складніше організувати дистанційну роботу;
- неможливість автоматизації, якщо вона не була закладена автором програми.

У своєму проекту я використовував GUI для наступних дій:

- швидкість бігу;
- камера головного персонажу;
- допоміжні функції.

У сучасних іграх, змішування анімації є необхідною функцією для забезпечення персонажа плавними анімаціями. Аніматори створюють окремі анімації, наприклад, цикл ходьби, цикл бігу, анімація в стані спокою або стрільби. У будь-який момент під час гри, повинна бути можливість переходу з анімації спокою в цикл ходьби і назад. Природно, ви хочете щоб цей перехід був плавним і без раптових ривків в русі.

В цьому випадку змішування анімації стає корисним. У Unity можна мати будь-яку кількість анімацій, які програються на одному і тому ж персонажа. Всі анімації змішуються або складаються разом для створення остаточної

анімації.

Спочатку зробимо плавний перехід між анімаціями спокою і ходьби. Щоб полегшити роботу написання скрипта, спершу потрібно змінити Wrap Mode анімації на Loop. Потім потрібно відключити Play Automatically для впевненості, що ніхто, крім нашого скрипта, не програє анімацію.

Наш перший скрипт для анімації персонажа досить простий; нам потрібен спосіб для визначення швидкості руху персонажа, а після цього робити перехід між анімаціями ходьби і спокою. Для цього простого тесту ми будемо використовувати стандартні осі введення:

```
function Update () {
  if (Input.GetAxis("Vertical") > 0.2)
     animation.CrossFade ("walk");
  else
    animation.CrossFade ("idle");
}
```

Щоб використовувати цей скрипт у нашому проекті:

1) створіть Javascript файл використовуючи Assets-> Create Other-> Javascript;

2) скопіюйте та вставте код в нього;

3) перетягніть скрипт на персонажа (він повинен бути прив'язаний до GameObject, який має анімацію).

Після натискання на кнопку Play, персонаж почне крокувати на місці, поки ви будете утримувати кнопку зі стрілкою вгору, і повернеться в позу очікування, якщо ви відпустіть її.

Змішування анімацій дозволяє урізати кількість анімацій, який потрібно створити для вашої гри. Це досягається тим, що деякі анімації впливають тільки на частину тіла. Це означає, що такі анімації можуть бути використані спільно з іншими анімаціями в різноманітних комбінаціях.Додавання трансформації змішування анімацій проводиться викликом методу AddMixingTransform () наявного AnimationState.

Як приклад змішування можна привести щось на зразок анімації махання рукою. Ви можете захотіти зробити махання рукою або коли персонаж знаходиться в спокої, або коли він йде. Без змішування анімацій, довелося б зробити окремі анімації махання рукою для стану спокою і ходьби. Але якщо ви додасте трансформацію плечей як трансформацію змішування в анімацію махання рукою, то вона буде мати повний контроль

тільки від плечового суглоба до руки. Так як решта тіла не буде схильна до впливу анімації махання рукою, воно продовжить грати анімацію спокою або ходьби. Як наслідок, знадобиться тільки одна анімація, щоб зробити махання рукою, тоді як решта тіла буде використовувати анімацію спокою або ходьби.

```
/// Adds a mixing transform using a Transform variable
var shoulder : Transform;
animation["wave_hand"].AddMixingTransform(shoulder);
```

Адитивні анімації і технологія змішування анімацій дозволяють зменшити загальну кількість анімацій, які вам треба створити для вашої гри, і ці техніки також важливі для створення лицьової анімації.

Припустимо, вам захотілося створити персонажа, який нахиляється в сторони під час поворотів, коли він ходить або бігає. Це призводить до 4 комбінаціям (йти-нахилятися-вліво, йти-нахилятися-вправо, бігти-нахилятися-вліво, бігти-нахилятися-вправо), для кожної з яких потрібна анімація. Створення окремої анімації на кожну комбінацію, очевидно, веде до безлічі додаткової роботи, навіть в такому простому випадку. Але кількість комбінацій збільшується з кожною додатковою дією. На щастя, адитивні анімації і змішування дозволяє уникнути необхідності створення окремих анімацій для комбінацій простих рухів.

Адитивні анімації дозволяють накладати ефекти однієї анімації поверх будь-яких інших запущених анімацій. Коли генеруються адитивні анімації, Unity вирахує різницю між першим і поточним кадрами анімаційного кліпу. Потім різниця застосується поверх всіх інших програються анімацій.

На основі попереднього прикладу, створимо анімації для нахилів вправо і вліво, і Unity зміг би накласти їх на цикл ходьби, спокою або бігу. Це може бути досягнуто за допомогою коду наступним чином.

```
privatevarleanLeft : AnimationState;
private var leanRight : AnimationState;

function Start () {
  leanLeft = animation["leanLeft"];
  leanRight = animation["leanRight"];

  // Put the leaning animation in a separate layer
  // So that other calls to CrossFade won't affect it.
leanLeft.layer = 10;
  leanRight.layer = 10;
```

```
  // Set the lean animation to be additive
  leanLeft.blendMode = AnimationBlendMode.Additive;
  leanRight.blendMode = AnimationBlendMode.Additive;

  // Set the lean animation ClampForever
  // With ClampForever animations will not stop
  // automatically when reaching the end of the clip
  leanLeft.wrapMode = WrapMode.ClampForever;
  leanRight.wrapMode = WrapMode.ClampForever;

  // Enable the animation and fade it in completely
  // We don't use animation.Play here because we manually adjust the time
  // in the Update function.
  // Instead we just enable the animation and set it to full weight
  leanRight.enabled = true;
  leanLeft.enabled = true;
  leanRight.weight = 1.0;
  leanLeft.weight = 1.0;

  // For testing just play "walk" animation and loop it
  animation["walk"].wrapMode = WrapMode.Loop;
  animation.Play("walk");
}

// Every frame just set the normalized time
// based on how much lean we want to apply
function Update () {
  var lean = Input.GetAxis("Horizontal");
  // normalizedTime is 0 at the first frame and 1 at the last frame in the clip
  leanLeft.normalizedTime = -lean;
  leanRight.normalizedTime = lean;
}
```

Іноді може знадобиться анімувати кістки персонажа процедурно. Наприклад, щоб голова персонажа дивилася в певну точку в 3D просторі. Ця дія найкраще реалізувати за допомогою скрипта, який відстежує цільову точку. На щастя, Unity робить це дуже простим завданням, тому що кістки всього лише є трансформаціями, які керують шкірою (skinned mesh). Можна керувати кістками персонажа з скрипта так само, як трансформаціями GameObject.

Одна з важливих речей, яку необхідно знати, це те, що анімаційна система оновлює трансформації після функції Update (), і до функції

LateUpdate (). Тому, якщо потрібно викликати функцію LookAt (), слід робити це в LateUpdate (), щоб упевнитися, що насправді переопределяется анімація.

Ганчіркові ляльки (Ragdolls) створюються таким же способом. Просто потрібно додати компоненти Rigidbody, Character Joint і Capsule Collider до різних кісток. Це дозволить створити анімацію персонажа засновану на фізиці.

Безперервне виявлення зіткнень це особливість, що запобігає проникнення швидко рухомих об'єктів один в одного. Таке може статися при використанні нормального (Discrete) виявлення зіткнень, коли в одному кадрі об'єкт знаходиться там же де і коллайдер, а в іншому він уже пройшов через цей коллайдер. Щоб вирішити цю проблему, можна включити безперервне виявлення зіткнень на твердому тілі що швидко рухається. Встановіть режим виявлення зіткнень в Continuous, щоб запобігти проникнення твердих тіл через будь-які статичні (тобто не є rigidbody) меш колайдери. Встановіть їх також в Continuous Dynamic, щоб запобігти проникненню твердих тіл в будь-який інший підтримуваний тип твердих тіл з встановленим режимом виявлення зіткнень Continuous або Continuous Dynamic. Безперервне виявлення зіткнень підтримується для Box-, Sphere- і Capsule коллайдеров. Треба враховувати, що безперервне виявлення зіткнень служить в якості системи підтримки для вилову зіткнень в тих випадках, коли об'єкти могли б пройти крізь один одного, але вона не надає фізично акуратних результатів зіткнень, тому ви можете все ще порахувати потрібним зменшити значення фіксованого тимчасового кроку (fixed Time step) в інспектора менеджера часу (TimeManager), щоб симуляція пройшла більш точно якщо б відчувались проблеми з швидко рухомими об'єктами.

Розмір ігрового об'єкта грає велику роль в порівнянні з масою твердого тіла. Якщо ви помітили що ваше тверде тіло поводиться не так, як ви спочатку задумували - повільно рухається, коливається, або некоректно стикається - спробуйте налаштувати масштаб свого меш Ассет. Стандартна одиниця вимірювання в Unity 1 юніт = 1 метр, Наприклад, руйнується на частини хмарочос виглядає набагато інакше ніж вежа, зібрана з іграшкових блоків, тому об'єкти різного розміру повинні бути змодельовані в масштабі.

Якщо ви моделюєте людини, переконайтеся, що він займає близько 2 метрів в єдності. Щоб перевірити, чи має ваш об'єкт правильний розмір, порівняйте його з кубом за замовчуванням. Ви можете створити куб, використовуючи GameObject> 3D Object> Cube. Висота куба буде

дорівнювати 1 метр, тому ваш чоловік повинен бути в два рази вище.

Якщо ви не можете налаштувати саму сітку, ви можете змінити єдину шкалу для певного об'єкта сітки, вибравши її у вікні «Проект» і вибравши «Активи -> Параметри імпорту ...» в меню. Тут ви можете змінити масштаб і повторно імпортувати свою сітку.

Якщо у вашій грі необхідно зробити так, щоб створювалися екземпляри об'єктів типу GameObject різного розміру, буде цілком нормальним змінювати значення масштабу осей трансформацій вашого об'єкта. Поганий стороною цього процесу є те, що фізична симуляція повинна брати на себе основну частину роботи під час створення екземплярів об'єкта, а це в свою чергу може позначитися на зниженні продуктивності у вашій грі. Але це не жахлива втрата, і це не так уже й ефективно як при використанні для масштабування ваших об'єктів двох інших опцій. Також не забувайте про те, що нерівномірне масштабування може привести до непередбачуваного поведінки при використанні успадкування. Тому, в таких випадках краще в своєму пакеті тривимірного моделювання з створюваних об'єктів в правильному масштабі.

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 розробціінтерактивної 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
Вивченнядокументції для створення об'єктіву Blender3D	30 днів	25.08.18	24.09.18
Вивчення основ та характеристик Unityта Blender	5 днів	24.09.18	29.09.18
Створення об'єктів в Blender	20 днів	29.09.18	19.10.18
Створennядекількохоб'єктіввUnity 3D	19 днів	19.10.18	08.11.18
Експортоб'єктівуUnity	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

Прирозробціданогонауковогопроектунездійснювалосярйознеуправлінняризиками, протевраховуваласяможливістьвиникненняпевнихфорс-

мажорів, такихякхворобивиконавців, забутіроботиіт.п. Дляцьоговкоженетапбуладоданафіктивназадачазмінімальнимпріоритетом, підназвою«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.

4.3 Оцінка трудомісткості і швидкості 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 3 грудня 2018 року, як і було заплановано. Загальна тривалість розробки і реалізації проекту склала 222 днів, або 2664 годин. Під час розробки були задіяні всі наявні ресурси.

ВИСНОВКИ

В результаті виконання магістерської роботи була створена інтерактивна модель, для реалізації якої був виконаний аналіз існуючих движків та графічних 3D редакторів.

Створення моделей здійснювалось за допомогою графічного 3D редактора Blender.

Основна сцена моделі створювалась за допомогоюUnity 3D, та складається з більш ніж 100 об'єктів створених у Blender, таінших графічних редакторахі взаємодією між ними.

Анімування та створення фізики створювались у Unity.

Скриптування об’єктів здійснювалось за допомогою мови програмування C#.

Було вивчено декілька десятків компонентів, які взаємодіяли із сценаріями. Такі як, RigiBody, Colliders, Meshes та інші. Головний клас рушія MonoBehaviour, де описувалися усі можливості Unity.

Розробка моделівелась для операційної системи Windows. Булавибрана ця операційна система, тому що у даний час вона є найбільш популярна та швидко розвивається.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАНЬ

1. Фінкельштейн М. І. Основи радіолокації. М.: Радіо і зв'язок, 1983. 536 с.
2. Павлов Н.Ф. Аерологія, радіометеорологія і техніка безпеки - моделювання Л .:Гидрометеоиздат, 1980. 432 с.
3. Квитків М.В., Кузьменко В.П., Павлов Н.Ф., Цівенко Н.В. Радіометеорологія -моделювання М.: Військове видавництво, 1984. 210 с.
4. Радіолокаційні станції бічного огляду. А. П. Реутов, Б. А. Михайлов, Г. С. Кондратенков, Б. В. Бойко; Под ред. А. П. Реутова.М.: Сов.радних.Наступні, 1970. 358 с.
5. Виноградов Б. В, Космічні методи вивчення природного середовища. -моделювання М .: Думка, 1976. 288 с.
6. Сулаквелідзе Г. К. Зливові опади і град. Л.: Гидрометеоиздат, 1967.412 с.
7. Міервін В. Є., Шупяцкій А. Б. Радіолокаційний метод визначення фазового стану хмар і опадів. Праці ЦАО, 1963, вип. 47.84 с.
8. Віглеб Г. Датчики. М .: Світ, 1989. 4. Федотов Я.А. Основи фізики напівпровідникових приладів. М .: Сов.радіо, 1969. 860 с.
9. Фогельсон І.Б. Транзисторні термодатчики. М.: Сов.радіо, 1972. 400с.
10. Гордов А.Н., Жагулло О.М., Іванова А.Г. Основи температурних вимірювань. М .: Вища школа, 1992. 690 с.
10. Шефтель І.Т. Терморезистор. М .: Наука, 1973. 300 с.
11. Большаков Д. І. 3D моделювання. / Д. І. Большаков-М.:Техатека, 2011. 34 с.
12. Бочков М. Д. Основи 3D-моделювання. / М. Д. Бочков - К.: КНЭУ, 2003. 106 с.
13. Донован Т. Ігрова індустрія. Створення ігор. / Т. Донован- М.: Видавничий дім «Вільямс», 2007. 412 с.
14. Дацко М. А. Моделювання складних об'єктів. / М. А. Дацко - К.: Максімас, 2015р. 111 с.
15. Хокінг Д. Unity в дії. Мультиплатформенна розробка. / Д. Хокінг - М.:Видавничий центр «Академія», 2016. 336с.

16. Кроніестер Д.Основи Blender навчальний посібник 4-е видання./ Д. Кроніестер-СПб.: Пітер, 2012. 416с.
17. Максимов А. П. Створення найпростіших моделей, побудова сцени. / А. П. Максимов -М.: Мітра, 2011. 38 с.
18. Мейкісон Л.Моделювання - це легко. / Л. Мейкісон - М.: Інтуїт, 2013. 313 с.
19. Петренко. С. М. Вивчаємо Blender 3D. / С. М. Петренко - М.: Російський новий університет, 2009. 542 с.
20. Богатырев Р. Природа и эволюция сценарных языков./Р. Богатырев. - Мир ПК, 2001. 160 с.
21. GerzE. diJustoP. AtmosphericMonitoringwithArduino-O'ReillyMe-dia, Inc.2013 89p.
22. Chen V.C., Tahmoush D., Miceli W.J.Radar Micro-Doppler Signatures. (Eds.), 2014. 408 p.
23. Mahafza B.R.Radar Systems Analysis and Design Using Unity., 2013. 772 p.
24. MacGormanD.R.,StrakaJ.M.,ZieglerC.L.ALightningParameterizationfor NumericalCloudModels.–J.Appl.Meteor.,2001,vol.40,459–478pp.
25. DiFranco J.V., Rubin W.L.Radar Detection., 2004. 654 p.

Д О Д А Т К И

ДодатокА
Графічна частина магістерської роботи

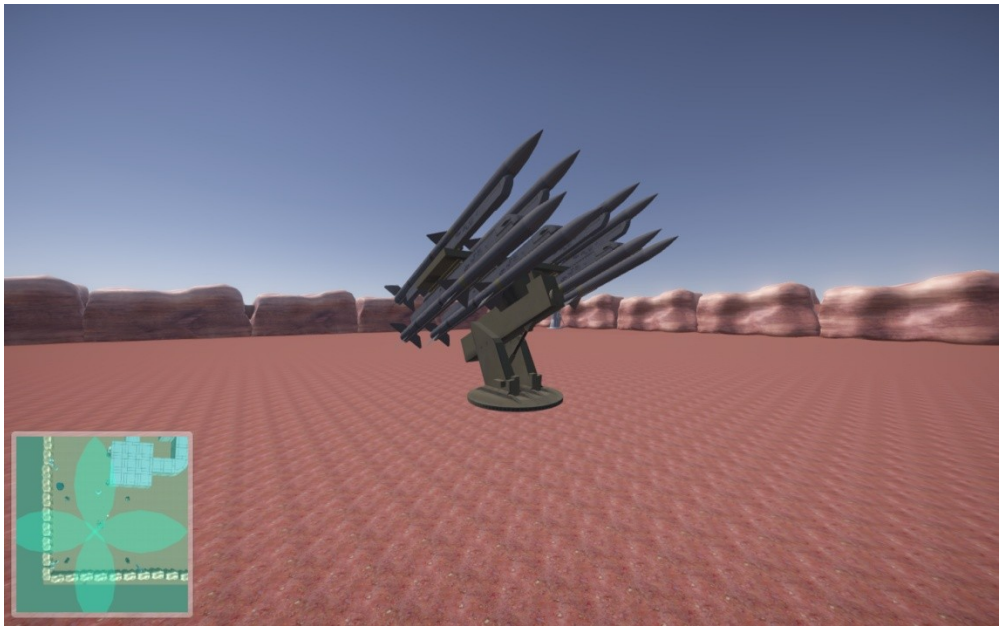

Рисунок А.1 – Протиградова ракетна установка

Рисунок А.2 – Запуск ракетв купчасті хмари

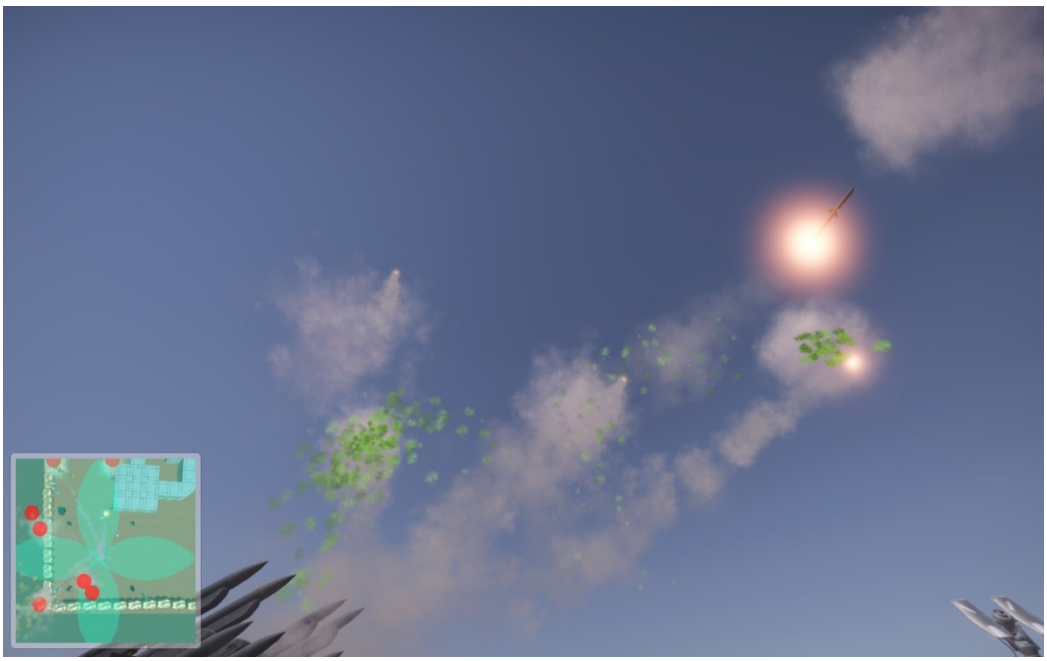

Рисунок А.3 – Розсіювання реагенту

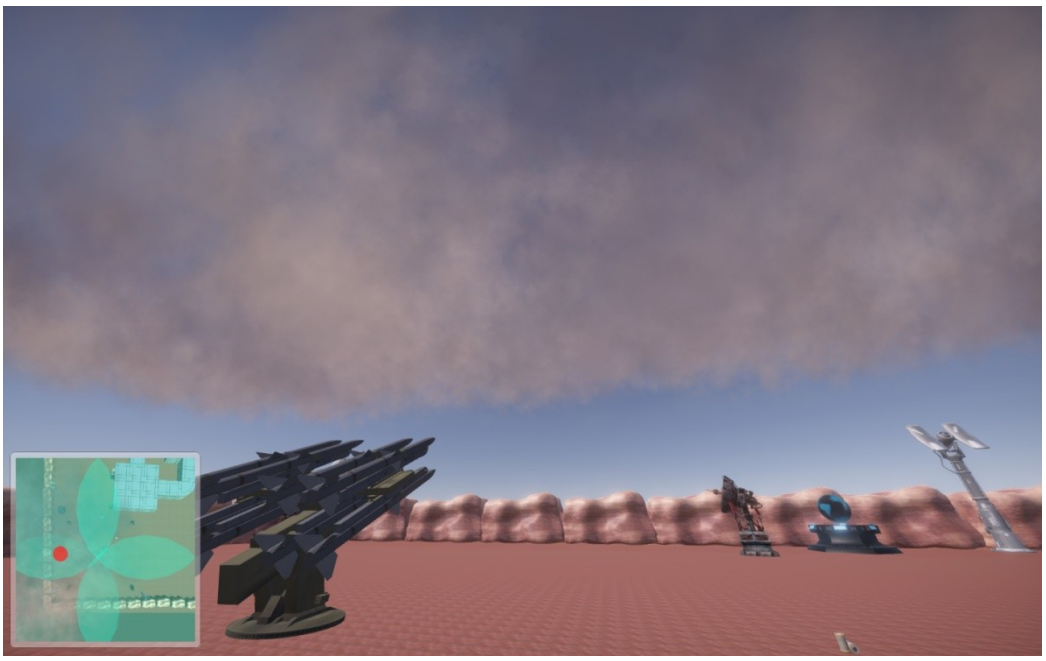

Рисунок А.4 –Появашаруватої хмарності

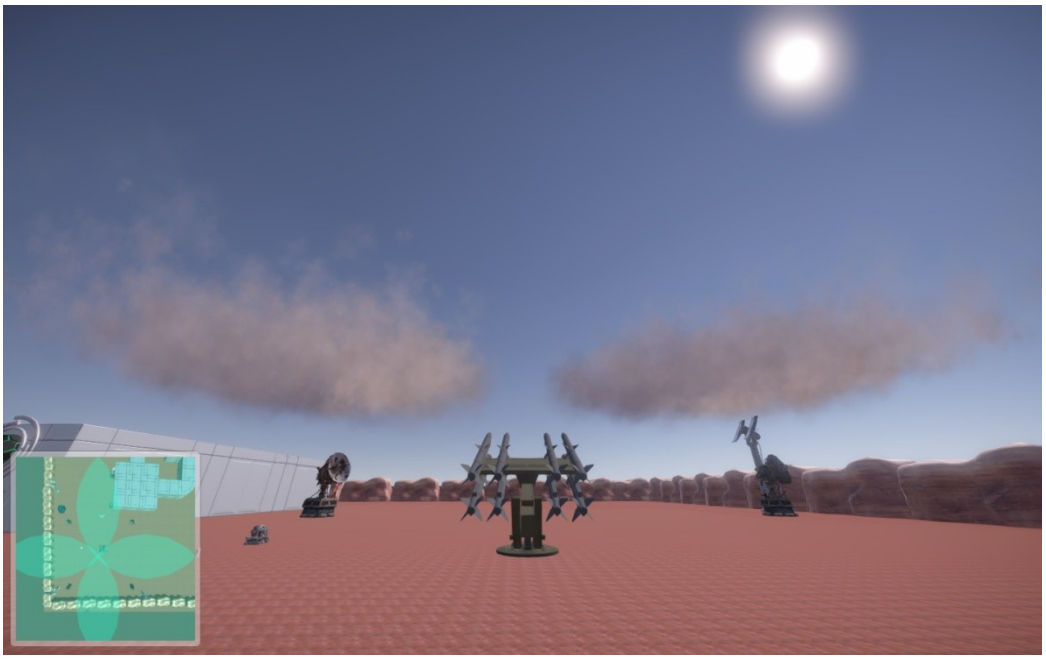

Рисунок А.5 – Шарувата хмарність після запуску ракети з реагентом

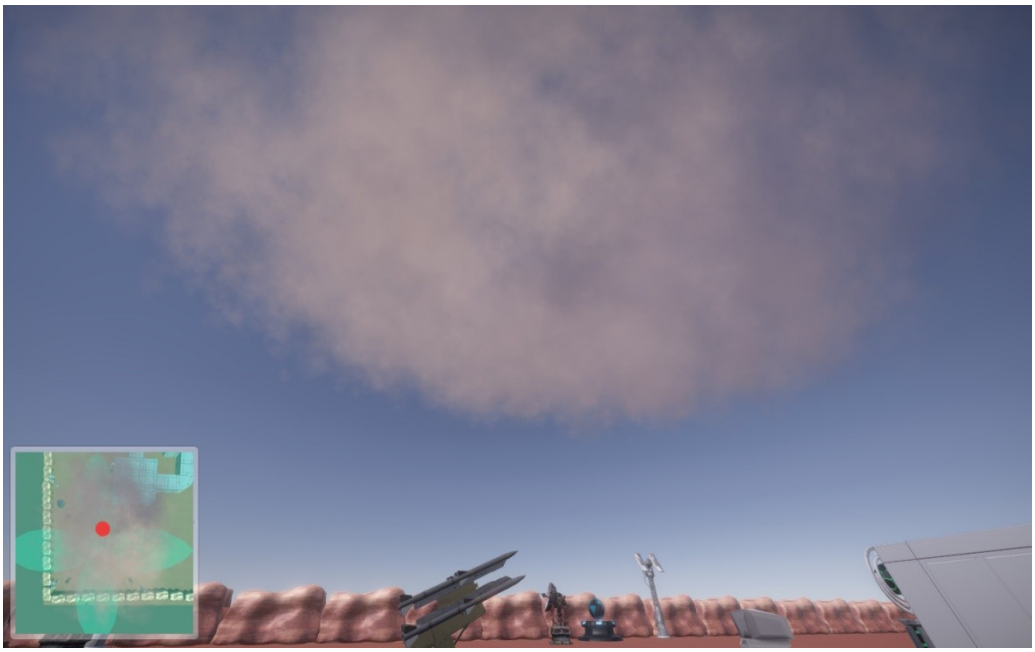

Рисунок А.6 – Поява грозоградової хмари

Рисунок А.7 – Розсіювання грозоградової хмари та поява дощу

ДодатокБ
Програмнийкод

```
publicclassHDK_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
after 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,
ExaminedReveal.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_obj.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_InteractObject>()){
                        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_InteractObject> ().ExaminableObject == null) {
                                                        ExaminingObject = false;
                                                        this.GetComponent<AudioSource>        ().clip       =
CantPickup;
                                                        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_DigitalCamera> ().broken) {

        Player.GetComponent<HDK_DigitalCamera> ().brokenGUI.GetComponent<CanvasGroup> ().alpha = 0;

        Player.GetComponent<HDK_DigitalCamera> ().camera_effect.enabled = false;
                                                                        }
                                                                }
                if (!RaycastedExamineObj.GetComponent<HDK_InteractObject> ().Examined) {
                                                                        StartCoroutine (RevealExamined ());
                                                                }
                                                                if    (GetComponentInParent<HDK_SimpleInventory>
() != null) {

        GetComponentInParent<HDK_SimpleInventory> ().close ();
                                                                }
```