МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
НАЦІОНАЛЬНИЙ АВІАЦІЙНИЙ УНІВЕРСИТЕТ

Кафедра Комп'ютерних інформаційних технологій

ДОПУСТИТИ ДО ЗАХИСТУ
Завідувач випускової кафедри
Аліна САВЧЕНКО.
«_____» _____ 2024р.

КВАЛІФІКАЦІЙНА РОБОТА

(ПОЯСНЮВАЛЬНА ЗАПИСКА)

ЗДОБУВАЧА ОСВІТНЬОГО СТУПЕНЯ **"БАКАЛАВР"**

Тема: "Створення 3D персонажу його анімації з використанням мов програмування"

Виконавець: студент групи УС-411 Йовик Максим Юрійович

Керівник: к.т.н., доцент Колісник Олена Василівна

Нормоконтролер: _____ Олександр ШЕВЧЕНКО
(підпис)

Київ – 2024

НАЦІОНАЛЬНИЙ АВІАЦІЙНИЙ УНІВЕРСИТЕТ

Факультет *Комп'ютерних наук та технологій*

Кафедра *Комп'ютерних інформаційних технологій*

Галузь знань, спеціальність, освітньо-професійна програма: 12 "Інформаційні технології", 122 "Комп'ютерні науки", "Інформаційні управляючі системи та технології"

ЗАТВЕРДЖУЮ
Завідувач випускової кафедри
_____ Аліна САВЧЕНКО
«____»_____2024р.

ЗАВДАННЯ

на виконання кваліфікаційної роботи (проєкту)

Йовика Максима Юрійовича

1. Тема роботи: «Створення 3D персонажу його анімації з використанням мов програмування» затверджена наказом ректора від "05" квітня 2024 р. за № 517/ст.

2. Термін виконання роботи: 06.05.2024 – 07.06.2024

3. Вихідні данні до роботи: проблема необізнаності використання 3D моделей та створенню анімацій до них; існуючі методи розробки моделей та правильності виконання; набори даних з прикладами розробки моделей та анімацій до них; завдання створення 3D персонажу його анімації з використанням мов програмування; необхідність аналізу підходів до правильного створення анімацій на основі моделі.

4. Зміст пояснювальної записки: вступ, огляд методів та алгоритмів розробки моделей та анімацій, аналіз найпопулярніших програм для розробки, вибір технологій для розробки, архітектура проекту, опис вигляду моделі та використання, висновки.

5. Перелік обов'язкового графічного матеріалу: скріншоти створення 3D моделі та анімацій на її основі, демонстрація матеріалів розробки.

6. Календарний план-графік

№ п/п	Завдання	Термін виконання	Підпис керівника
1.	Отримання завдання на дипломну роботу, створення плану кваліфікаційної роботи та побудова плану-графіку виконання робіт.	06.05.2024 – 07.05.2024	
2.	Розроблення та затвердження календарного плану виконання кваліфікаційної роботи.	07.05.2024 – 08.05.2024	
3.	Проведення консультації з науковим керівником.	09.05.2024	
4.	Огляд та аналіз наукової літератури по темі кваліфікаційної роботи та написання Розділу 1.	10.05.2024 – 16.05.2024	
5.	Написання Розділу 2 кваліфікаційної роботи.	17.05.2024 – 22.05.2024	
6.	Написання Розділу 3 кваліфікаційної роботи.	23.05.2024 – 30.05.2024	
7.	Завершення створення пояснювальної записки кваліфікаційної роботи.	31.05.2024 – 02.06.2024	
8.	Оформлення та друк пояснювальної записки. Створення презентації, доповіді та підготовка до захисту кваліфікаційної роботи.	03.06.2024 – 05.06.2024	
9.	Підготовка матеріалів кваліфікаційної роботи для передачі секретарю ДЕК (папка, конверт, диск із файлом диплому, рецензія, відгук).	06.06.2024 – 07.06.2024	

7. Дата видачі завдання: «6» травня 2024 р.

Керівник кваліфікаційної роботи _____ Олена КОЛІСНИК
(підпис)

Завдання прийняв до виконання _____ Владислав ДОРОГАНЬ
(підпис)

РЕФЕРАТ

Пояснювальна записка до кваліфікаційної роботи «Створення 3D персонажу його анімації з використанням мов програмування»: 75 с., 20 рис., 7 табл. 18 літературних джерел.

Об'єкт дослідження: Процес створення 3D персонажу його анімації з використанням мов програмування.

Предмет дослідження: Вивчення різних методів та алгоритмів створення 3D персонажу, їх переваги, недоліки та особливості застосування, та вивчення процесу розробки анімацій з використанням мов програмування.

Мета роботи: Метою даної роботи є створення 3D персонажу, вивчення процесу створення, та напрацювання власних навичок в цій сфері виконанням конкретної роботи по створенню.

Методи дослідження: Аналіз вимог до створення 3D персонажів, порівняння ефективності методів розробки та створення 3D моделей, тренування власних навичок в цій сфері, розробка анімації з використанням мов програмування.

Наукова новизна полягає у розробці 3D моделей, вивченню сфери розробки 3D моделей, їхнє застосування та використання в подальшому.

В результаті виконання роботи створено 3D модель персонажа для подальшого використання в цілях створення, та розробка анімації з використанням мов програмування на основі розробленої 3D моделі персонажа.

Результати кваліфікаційної роботи рекомендується використовувати під час створення 3D моделей персонажів та розробці анімацій для 3D моделей, або інших структур чи моделей для використання в сферах створення загальної візуалізації, наприклад у відеоіграх, кіно, веб-дизайні, архітектурі, медицині.

3D, МОДЕЛІ, ШЕЙДЕРИ, СТРУКТУРИ, СКЕЛЕТИНГ, ТЕКСТУРИ, АНІМАЦІЇ, СТВОРЕННЯ АНІМАЦІЙ ЗАВДЯКИ НАПИСАННЯ КОДУ, ШЕЙДЕРІНГ, ПРОГРАМИ РОЗРОБКИ 3D МОДЕЛЕЙ, КАДРУВАННЯ, ПОЛІГОНИ.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СКОРОЧЕНЬ І ТЕРМІНІВ

ПЗ	Програмне забезпечення.
CG (Computer Graphic)	Комп'ютерна графіка.
2D (2-dimensional)	2-вимірний.
3D (3-dimensional)	Тривимірний простір.
UV	Розгортка. Подання тривимірної моделі в якості двовимірного зображення.
GPU	Графічний процесор.
CPU	Центральний процесор.
FK (Forward Kinematics)	Пряма кінематика.
IK (Inverse Kinematics)	Зворотня/інверсна кінематика.
NURBS (Non-uniform rational B-spline)	Математична форма. Метод моделювання в основі якого B-сплайни.
API (Application Programming Interface)	Програмний інтерфейс додатку.
OS (Operation system)	Операційна система.
MEL (Maya Embedded Language)	Вбудована мова скриптів графічного редактору Autodesk Maya.
PC	Персональний комп’ютер.
RGB (Red, Green, Blue)	Адитивна колірна модель.

ВСТУП

Комп'ютерна анімація швидко розвивається і стає дуже важливою у кіно та відеоіграх. Вона допомагає створювати різні світи і персонажі. Не лише у технологіях, а й у повсякденному житті графіка і анімація дуже важливі. Більшість компаній потребують графічних рішень для рекламних кампаній, а з розвитком Інтернету це стає ще більш актуальним.

Останні 10 років принесли великий розквіт цих галузей, особливо у США та Китаї. Завдяки зниженню цін на комп'ютери та доступним технологіям стало більше інструментів для творчих людей, які працюють з комп'ютерною графікою і анімацією. Це також дало можливість новачкам скористатися безкоштовним програмним забезпеченням для створення анімації.

Стандартом у цій справі є Autodesk Maya. Цей редактор є дуже популярним у студіях і вже використовувався в багатьох фільмах та іграх. Він має потужні інструменти для обробки великих сцен і складних анімацій, а також великий вибір інструментів для художників і аніматорів. Крім того, Маya дозволяє розширювати свої можливості за допомогою відкритих моделей, таких як системи частинок, моделювання волосся та жорсткі/м'які тіла. За допомогою 3D анімації у Маya можна створювати дуже реалістичні зображення, які використовуються у архітектурі, дизайні та інших сферах. Основною задачею моєї роботи буде вивчення та розробка 3D моделі та створення анімацій до неї, попередньо буде проводитись ознайомлення з відповідними ресурсами для розробки, вивчення галузі розробки, та систематика спроб та помилок для відточування кінцевого результату для видачі, перед яким буде проводитись тестування функціоналу моделі задля покращення кінцевого результату моєї роботи по створенню.

РОЗДІЛ 1
АНАЛІЗ ОБЛАСТІ СТВОРЕННЯ 3D МОДЕЛІ ТА АНІМАЦІЙ

Під час аналізу предметної області проєкту важливо спочатку розглянути етапи моделювання та анімації, їхні особливості, а також провести дослідження графічних редакторів. Це допоможе спростити вибір технологій та методів для створення цільового продукту.

1.1. Моделювання 3D персонажу, та вивчення сфери

3D моделювання персонажів широко застосовується у різних галузях, включаючи PC і Flash ігри, кінематографію та рекламу.

Наш проект - це система анімації 3D моделі, яка виступає як персонаж у комп'ютерній грі. Щоб провести повний аналіз системи, потрібно детально розглянути процес моделювання, та всі підхідні процеси до цього, включаючи як саме проходить моделювання, етапи моделювання, яким чином зробити так щоб модель в кінцевому результаті мала потрібний нам вигляд, що таке текстуризація моделі, яким чином виконується скелетинг для подальшого анімування рухів, що таке мапа текстури, та який в неї вигляд, та всі етапи до кінцевого створення.

3D моделювання персонажів полягає у створенні віртуальних тривимірних об'єктів за допомогою спеціалізованого програмного забезпечення.

Головною метою 3D моделювання є розробка об'ємного образу цільового персонажу, або якщо простішими словами, візуалізація образу персонажу задопомогою Цей процес включає такі етапи:

- Створення концепції та дизайну.
- Моделювання полігональної сітки та скульптурування.
- Ретопологія.
- Розгортка.
- Текстурування.

Ці кроки будуть розглянуті більш докладно в наступних розділах .для невеликої частки одержувачів, маркетингова кампанія або шахрайська схема все одно будуть успішними[7].

1.2. Розяснення термінів Концепт та Дизайн персонажа

У всіх випадках створення персонажів для відеоігор починається з формування ідеї та концепту. CG-художники відтворюють концептуальне мистецтво, щоб візуально передати особливості та дизайн персонажа. інтегруючи його в основний проект над яким вони працюють.

Персонаж - це вигадана істота або особистість, риси якої стараються передати зовнішнім виглядом, та яка має свій власний характер, настрій та відповідний зовнішній вигляд під середовище в якому цей персонаж повинен перебувати. Для створення образу необхідно користуватися певною кількістю референсів. Референс - це фотографія або зображення, яке в різних аспектах відображає ідею концепту, та підбирається стосовно головної задумки персонажу, референсом може бути буквально що завгодно, головне щоб використане зображення можна було використати для полегшення роботи над проектом.

Деталізація моделі є ще одним ключовим аспектом для концепту, який допомагає створити унікальний образ, стиль і передати характер персонажа. На

рисунку 1.1 показано приклад концептуального мистецтва.

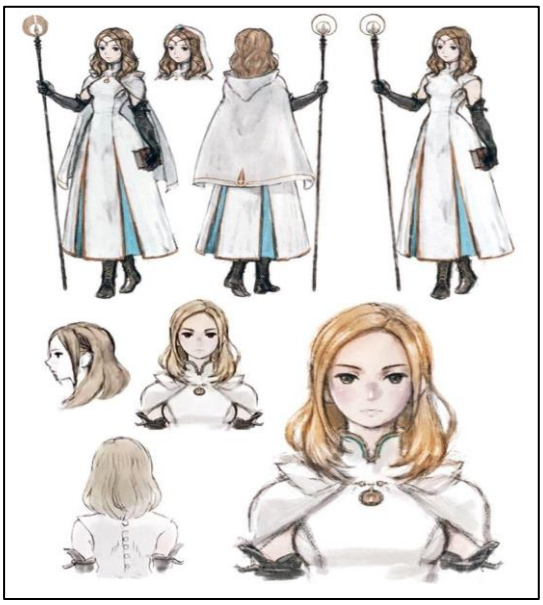

Рис. 1.1. Концепт арт

Після завершення попереднього етапу переходимо до 3D моделювання персонажа - це процес створення віртуальної, тривимірної моделі об'єкта за допомогою спеціалізованого програмного забезпечення. Головною метою цього процесу є створення об'ємного образу цільового персонажа, з яким в подальшому буде проходити вся магія процесу скульптингу , та робота з виглядом, це важливий процес.

1.3. Моделювання, скульптинг, та розяснення ретопології, скульптингу та текстурингу персонажа

Для початку почнемо з основ, та зрозуміємо що таке моделювання. Моделювання - це процес створення моделі або персонажа, під час якого використовуються різні інструменти. Маніпуляції над полігонами допомагають надати правильну форму для досягнення бажаного результату. Полігони, які можуть

бути трикутними або чотирикутними випуклими багатокутниками, використовуються для опису поверхні об'єкта, тобто для самого скелету чи то сітки з якої і складається сама модель. Їх кількість впливає на фінальний вигляд: низькополігональна модель виглядає абстрактно та більш просто стосовно вигляду, тоді як високополігональна модель має більш "згладжений" вигляд, більше деталей та більш чіткіше сприймається на око, хоча одночасно споживає більше ресурсу в порівнянні з низькополігональними моделями, через меншу кількість полігонів моделі.

В свою чергу, полігональна сітка (топологія) представляє собою набір вершин, ребер і граней, які формують об'єкт у 3D просторі. Гарна топологія забезпечує правильну деформацію моделі під час анімації та мінімізує кількість полігонів, необхідних для відтворення бажаної форми.

Скульптинг - це набір маніпуляцій, які виконуються шляхом деформації полігонів 3D-об'єкта, подібно до роботи скульптора з глиною, і являє собою основний процес створення 3D графіки, будь що у графіці такого типу створюється системою скульптингу або використанням скульптингу задля покращення вигляду моделі, приблизний вигляд скульптингу в програмі Zbrush зображений на рисунку 1.2, де видно використанні структури (їх зображають в вигляді кульок) для згладжування чи надання незвичайних форм моделі.

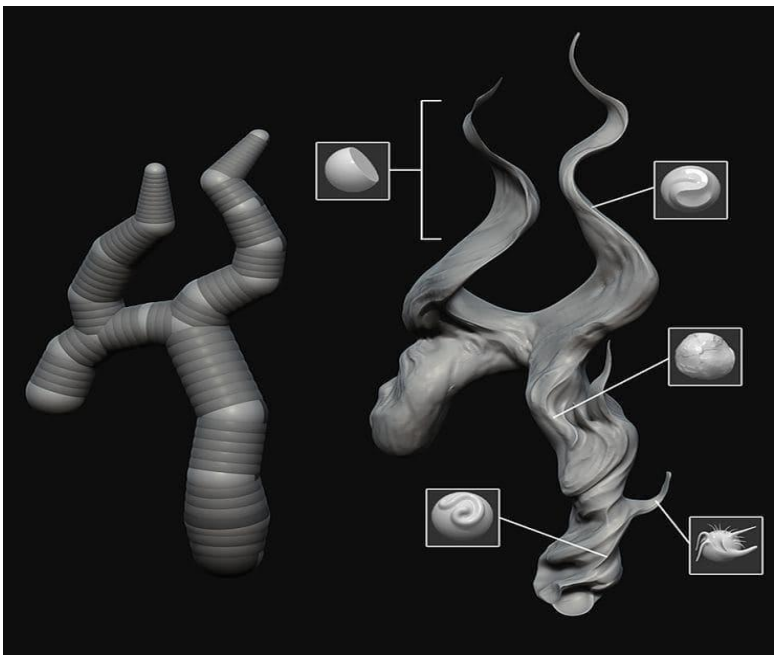

Рис. 1.2. Скульптинг 3D моделі

Загальний вигляд таких моделей залежить від вправності руки, та використанні різних кількостей додаткових засобів в програмі для скульптингу, і враховуючи що для кожної програми є велика частка додаткових ресурсів для завантажування з інтернету (для полегшення роботи), то створити можна абсолютно будь що, головне щоб вистачало розуму та вправності виконання.

1.3.1. Ретопологія

Другий етап топологічної оптимізації відбувається під час підготовки персонажа до анімації. Ця процедура є важливою, оскільки управління моделлю з великою кількістю полігонів у вьюпорті редактора є непрактичним. Суть полягає в перетворенні готової високополігональної форми в більш просту низькополігональну сітку персонажа, це відбувається процесом виділення сітки на моделі що зображено на рисунку 1.3, головна задача зробити так щоб сітка була як умога менше, мала менше звязків, але покривала модель повністю, так як фактично з неї і складається, це треба для спрощення сприйняття моделі, та для того щоб модель не споживала

велику кількість ресурсів при використанні, також це дозволяє легше здійснювати анімацію.

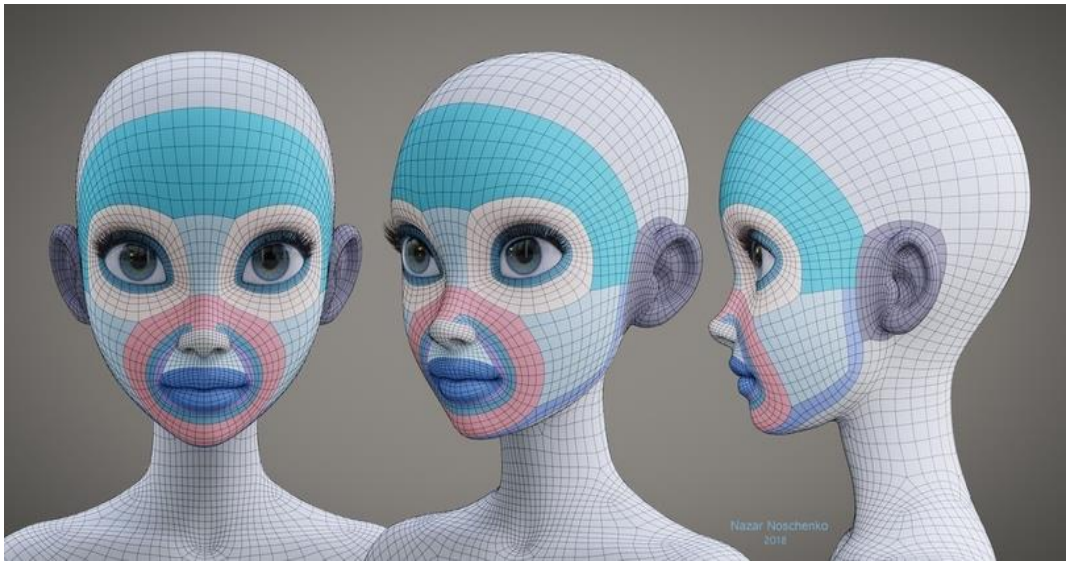

Рис. 1.3. Ретопологія моделі

Виглядає цей процес красиво, але має багато нюансів, так як завжди на цьому етапі потрібно шукати баланс, щоб модель не була надто проста візуально, і надто важка програмно, дотримання цього правила найважче що потрібно зробити при виконанні даного процесу.

1.3.2. Розгортка

Фактично розгортка це процес відображення плоского 2D зображення на 3D об'єкті. Осі координат для розгортки позначаються літерами U та V для зручності, та кожна точка на 3D моделі може мати кілька відповідних точок на розгортці. Кожна з цих точок належить полігону, з яким вона пов'язана, формуючи сітку. Розгортка необхідна для нанесення текстури на модель, що можна зрівняти з нанесенням шкіри на скелет. Кожен полігон моделі має бути представлений на розгортці.

Виглядає це наче знята шкіра розпластована на поверхню абсолютно повністю та без складок, це все потрібно для правильного накладання моделі, демонстрація зображена на рисунку 1.4, там видно персонажа, та з права його модельна розгортка.

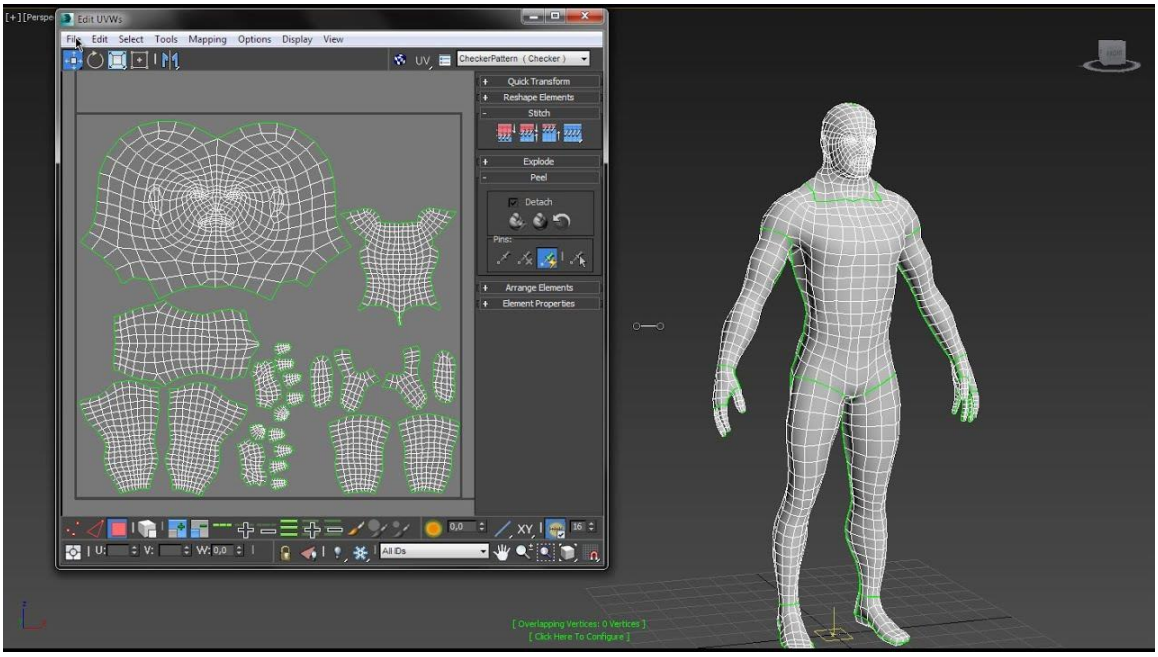

Рис. 1.4. Розгортка моделі

Розгортка є важливим процесом за яким потрібно слідкувати через те що від нього залежить подальша робота з моделлю, і те наскільки добре накладуться текстури, за цим потрібно слідкувати, щоб нічого не пішло не по плану.

1.3.3. Текстурування

Робота над наданням фізичних характеристик, кольору, блиску, текстури та рельєфу моделі називають текстурування. Цей клопіткий етап починається зі зміни кольору моделі на відповідний, враховуючи область об'єкта, яку ми текстуруємо. Далі, для накладання природного блиску матеріалу моделі, створюється спеціальна мапа блиску. Потім визначається шорсткість за допомогою карти нерівностей, яка є основою для створення відбивальних властивостей.

Якість мапінгу та текстурування визначається текселями (скорочення від texture element). Тексель - це конкретне число пікселів, яке відповідає одній одиниці текстури. Для ідеального відображення текстури кількість текселів повинна співпадати з кількістю пікселів на моніторі.

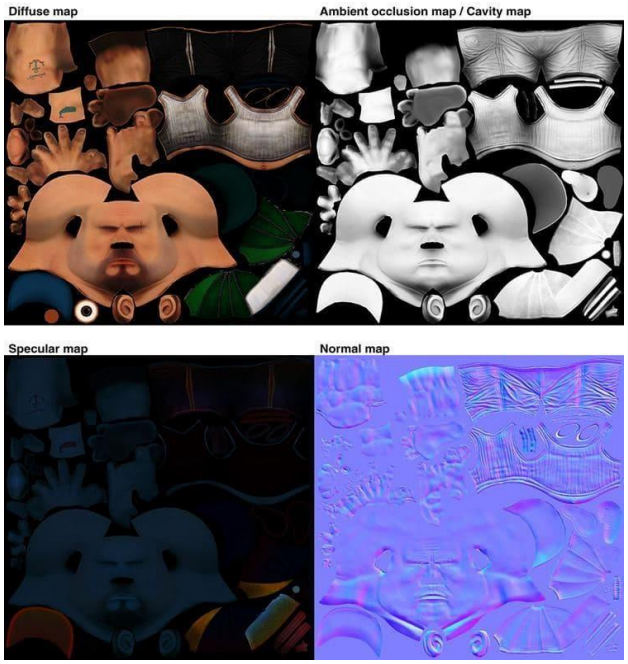

Рис. 1.5. Карти моделі

Процес створення карти моделі (Рис. 1.5) не довгий якщо всі попередні дії були виконані правильно, це автоматично робить програма для скульптингу, головне аби фінальний результат моделі підходив очікуванням.

1.4. Поняття системи анімацій, та що таке анімація в цілому

Система анімації являє собою налаштований віртуальний "скелет" 3D моделі, що містить ключові кадри (Keyframes). Ключові кадри визначають послідовність дій персонажа в різних позах.

Головним принципом цієї системи є правильне розташування персонажа, щоб

його позиції були легко зчитуваними. У 3D анімації найчастіше використовуються три методи:

- Анімація по ключовим кадрам: встановлення ключових кадрів на важливих моментах анімації для визначення позицій персонажа.

- Анімація по кривим руху: використання кривих для визначення плавного переходу між позиціями персонажа.

- Анімація по траєкторіям: створення траєкторій для персонажа, які вказуютьнапрямок його руху або анімаційні шляхи.

Анімація за допомогою ключових кадрів схожа за своїм принципом на роботу традиційних аніматорів. В цьому методі художник визначає ключові позиції персонажа і заповнює проміжні кадри, виконуючи відповідні трансформації фігури. Основна відмінність полягає в тому, що в анімації за допомогою ключових кадрів цю роботу виконують відповідні алгоритми в програмах-редакторах. Ці алгоритми автоматично інтерполюють міжкадрові переходи, враховуючи встановлені ключові кадри, що забезпечує плавність та реалістичність анімації.

Дійсно, для аніматора часто достатньо лише зафіксувати декілька ключових кадрів положення фігури, і процес інтерполяції буде завершено автоматично. Анімація по кривим руху полягає в уявленні переміщення або трансформацій об'єкта у вигляді графіків для кожної з його координат (X, Y, Z). Щодо анімації по траєкторіям, то в цьому методі окремо задається шлях переміщення об'єкта з визначенням його швидкості та можливих змін орієнтації об'єкта в просторі, що регулюється зазвичай тими ж вищезгаданими кривими.

Робота анімації використовує такі процеси, як:

- Риггінг (Rigging): Це процес створення віртуального скелету або системи кісток, які контролюють рухи 3D моделі. Риггінг дає можливість аніматорам керувати рухами персонажів.

Якщо точніше це скелетна анімація - це віртуалізація "скелету" персонажа та

налаштування керуючих елементів для керування рухом та діями моделі. При вправному використанні скелетної анімації можна значно зекономити зусилля, оскільки природно легше рухати декілька "кісток", ніж переміщати групи вершин і полігонів.

Стосовно риггінгу то скелетна анімація дозволяє аніматорам створювати реалістичні рухи персонажів і об'єктів у віртуальному середовищі .

- Та також є процес скінінгу (Skinning): Це процес прикріплення 3D моделі до скелета. Скінінг визначає, які частини моделі будуть рухатися разом з кожною кісткою скелета.

Для кращого розуміння розглянемо скелет як ієрархічну систему, де об'єкти взаємодіють, а положення одних впливає на інших. Вона базується на ідеї "родинних" та "дочірніх" об'єктів і їх взаємозв'язків. Коли родинна кістка перетворюється абсолютно, вона створює локальну систему координат для дочірньої кістки. Остання визначається локальною трансформацією у цій системі і може сама мати свої дочірні кістки, для яких стає родинною. Також родинна кістка може мати свої підлеглі кістки, це все звучить доволі страшно, але на розумінні під час процесу роботи з цим доволі швидко починаєш розбиратись чому така система набагато краще себе показує ніж могла б будь-яка інша.

1.4.1. Рендер

Рендер тема важка, але якщо коротко то анімацію зазвичай виводять у вигляді відеофайлу або послідовності зображень, та наприклад щоб отримати плавну анімацію, зазвичай використовують щонайменше 24 кадри на секунду. Це означає, що одна хвилина анімації може включати до 1440 кадрів для рендерингу, що може забрати чимало часу, в залежності від ресурсу ПК який використовується для рендеру.

Існують два типи візуалізації: візуалізація процесора і візуалізація графічного процесора (у реальному часі). Основна відмінність полягає в їх функціональних

можливостях і характеристиках.

Процесори зазвичай оптимізовані для виконання багатьох менших завдань одночасно, тоді як графічні процесори ефективніше обробляють складні обчислення. Графічний процесор часто працює швидше, що дозволяє досягати частоти відображення близько 60 кадрів в секунду в сучасних іграх. Процесорна візуалізація забезпечує точніші результати освітлення та складніші алгоритми текстурування.

У сучасних системах візуалізації різниця між цими методами стає менш помітною, окрім найскладніших сцен, де може бути помітна різниця в продуктивності та якості відтворення.

Різниця між варіаціями рендеру не велика, так як сам по собі процес достатньо клопіткий, і суттєво існує два варіанти а саме CPU рендер та GPU рендер.

CPU рендер використовує центральний процесор для рендерингу, що іноді називають "попереднім рендерингом", суть полягає в тому, що центральний процесор комп'ютера використовується як основний компонент для обчислень. Цей метод часто використовується кіностудіями та художниками, що працюють у сфері архітектурної візуалізації, оскільки він дозволяє створювати фотореалістичні зображення з високою точністю. У цих галузях час візуалізації не є критично важливим фактором.

Час рендерингу може значно варіюватися і іноді бути дуже довгим. Наприклад, сцена з простим освітленням і матеріалами може бути оброблена всього за декілька секунд. Однак складніші сцени з великою кількістю деталей та складними ефектами можуть вимагати значно більше часу для рендерингу.

А от GPU рендер виконує візуалізацію за допомогою графічного процесора який в свою чергу використовується для рендерингу в режимі реального часу і передбачає використання графічного процесора для основного ресурсу обчислень. Цей тип рендерингу широко застосовується у відеоіграх та інших інтерактивних

додатках, де важливо досягти частоти від 30 до 120 кадрів на секунду для плавного ігрового досвіду.

Для досягнення таких високих частот кадрів, рендеринг в режимі реального часу не може використовувати деякі складні обчислювальні методи, що застосовуються у візуалізації, що не працює в реальному часі. Тому багато обробки виконується за допомогою наближень під час постобробки. Наприклад, ефекти, які допомагають зробити зображення більш плавним, такі як розмиття в русі, використовуються для обману ока.

Завдяки стрімкому прогресу в технологіях і вдосконаленню методів розробки, обмеження візуалізації графічного процесора швидко відходять у минуле. Це пояснює, чому з кожним новим поколінням консолей ігор і подібних медіа їхня графіка значно покращується. З розвитком чіпсетів і вдосконаленням знань розробників покращуються і графічні результати.

1.4.2. Трасування променів та Трасування шляху

Трасування променів це коли кожен піксель на остаточному зображенні розраховується як окрема частинка світла, яка імітується через взаємодію з об'єктами вашої сцени, що в свою чергу дозволяє створювати реалістичні сцени з розширеним відображенням та тінями, але це вимагає значної обчислювальної потужності. Однак завдяки останнім досягненням у технологіях GPU на картах NVIDIA серії 2000, метод трасування променів може наблизитися до загальнодоступних ігор завдяки використанню рендерингу на GPU в найближчому майбутньому.

В відмінність трасування шляху або Відстеження контурів - це метод рендерингу, що реалістично відтворює взаємодію світла з об'єктами у сцені шляхом прослідковування променів. Цей процес визначає кінцевий колір кожного пікселя зображення, враховуючи, як світло взаємодіє з поверхнею об'єктів - відбивається, поглинається або розсіюється, надаючи зображенню високий рівень фотореалізму.

Обидва види трасування активно використовуються в роботі з фінальними стадіями моделей і анімацій, та потрібні для передачі більшого реалізму, без цього нікуди, зараз це застосовують усюди в цифровій графіці.

1.5. Огляд існуючих рішень, порівняння двох систем

Оскільки пряме відтворення анімації у програмному забезпеченні не можливе, огляд існуючих анімацій здійснюється на основі якості процесів моделювання та налаштування віртуального скелету.

Перша система

Концептуальний дизайн (див. рис. 1.7) персонажа відзначається своєю складністю та деталізацією, яка повністю передає характер особистості.

Модель (див. рис. 1.8) також відзначається високою складністю та деталізацією. Великий обсяг роботи був виконаний на етапах скульптування, розгортки та текстурування.

Незважаючи на високу деталізацію моделі, риггінг (див. рис. 1.8) цього персонажа є досить простим і виконується за допомогою простого "скелета", який відображає лише кілька рухів тулуба, в той час як нижні кінцівки залишаються майже нерухомими.

Рис. 1.7. Концепт дизайн

Рис. 1.8. Ретопологія

UV(див. рис.1.9) розгортка така ж сладна, як і модель.

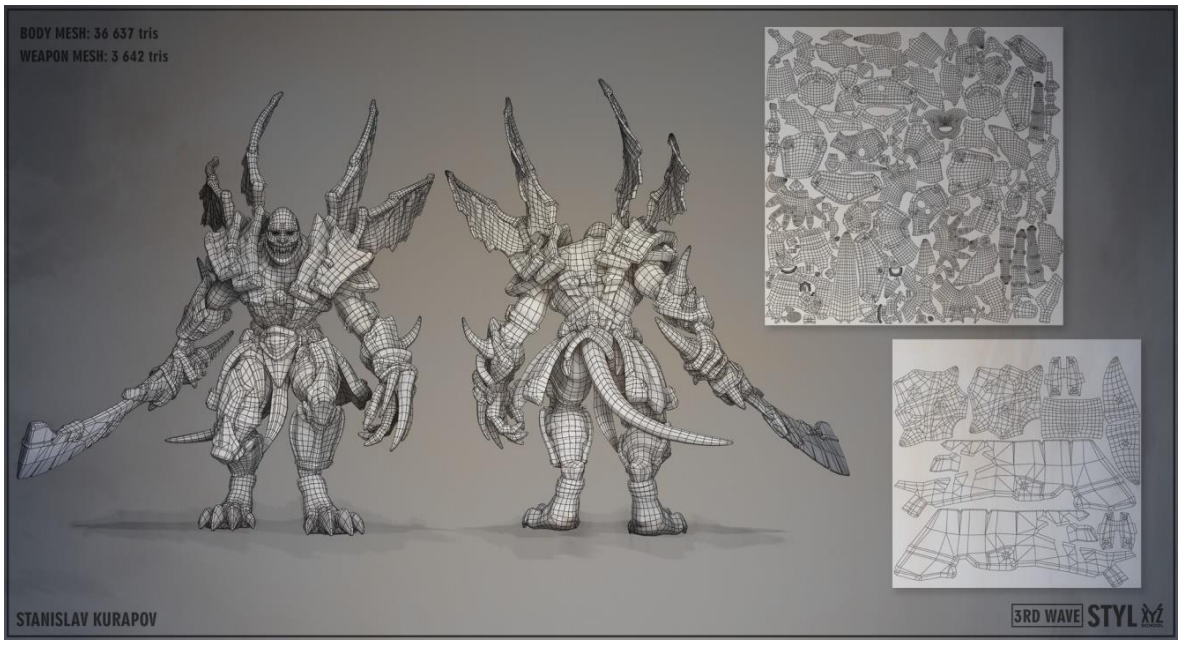

Рис. 1.9. Розгортка

Рис. 1.10 показує кількість використаних текстур для персонажа. Серед них присутні: карта блиску, карта нормалей, карта specular та дифузна карта.

Рис. 1.10. Текстури

Рис. 1.11. Ригінг

Персонаж рухає тулубом, частково рухаються руки та голова. Нижні кінцівки майже нерухомі, скелет персонажу зображений на рисунку 1.11 , де видно всі кістки персонажа якими можна рухати.

Та також друга система:

Порівняно з попередньою системою, ця модель за своїм концептом (див. рис. 1.12) менш деталізована, має простіший скульптинг та менше текстур (див. рис. 1.14). Проте, на відміну від системи №1, вона має складніший ригінг (див. рис. 1.16), оскільки імітує ходу.

Рис. 1.12. Концепт арт

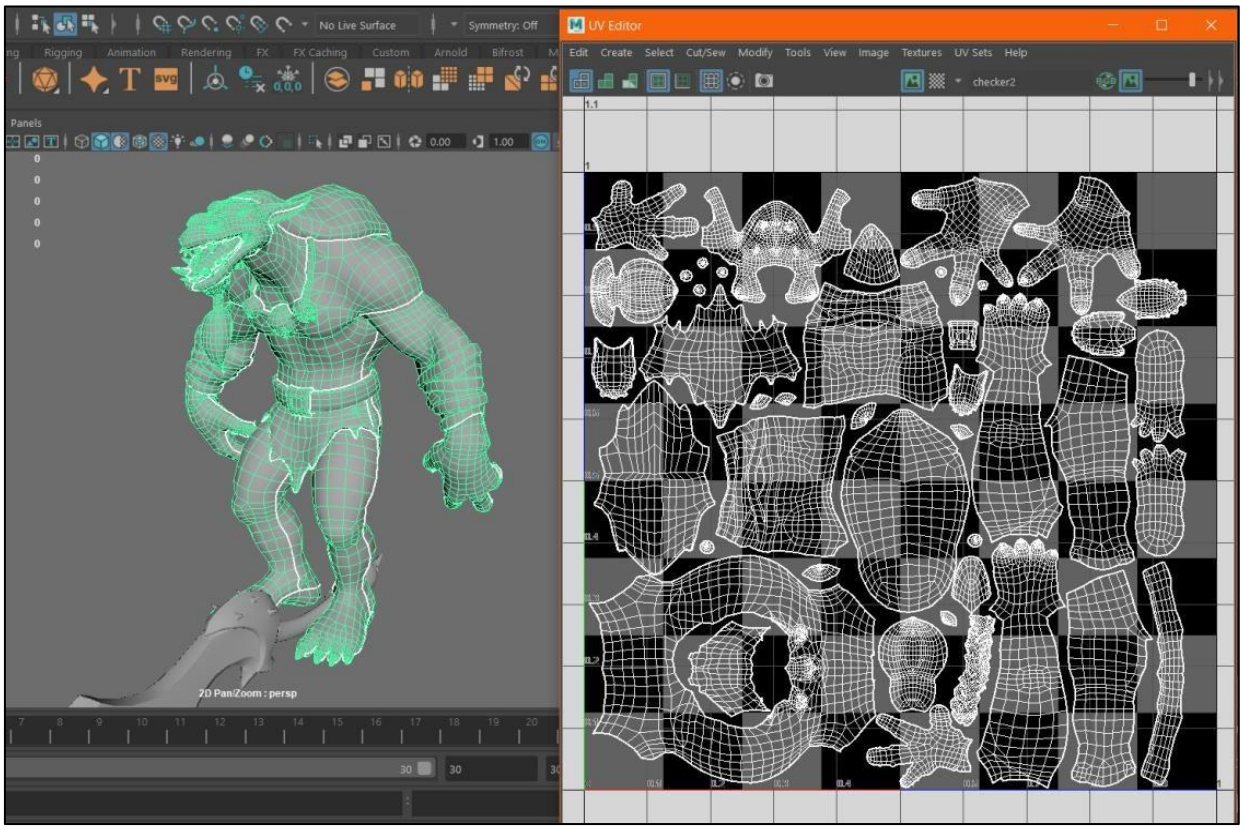

Рис. 1.13. Розгортка

У порівнянні з розгорткою моделі з першої системи, розгортка цієї моделі має менше деталей, що зумовлено меншою деталізацією персонажа, і тим самим більшою продуктивністю або легшим сприйняттям моделі пристроєм, хоча і гіршим виглядом.

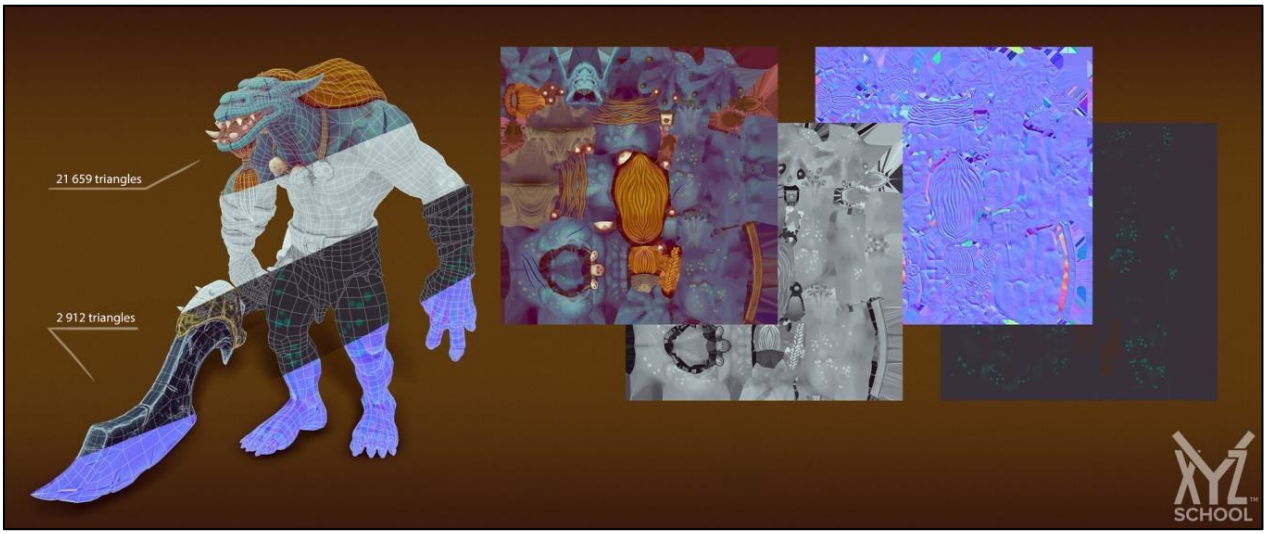

Рис. 1.14. Текстура

Рис. 1.15. Ригінг моделі

Структура кісткової системи або ригінг моделі, показані на рисунку 1.15.

Побудовані з урахуванням силуету персонажа, який ґрунтується на концепції.

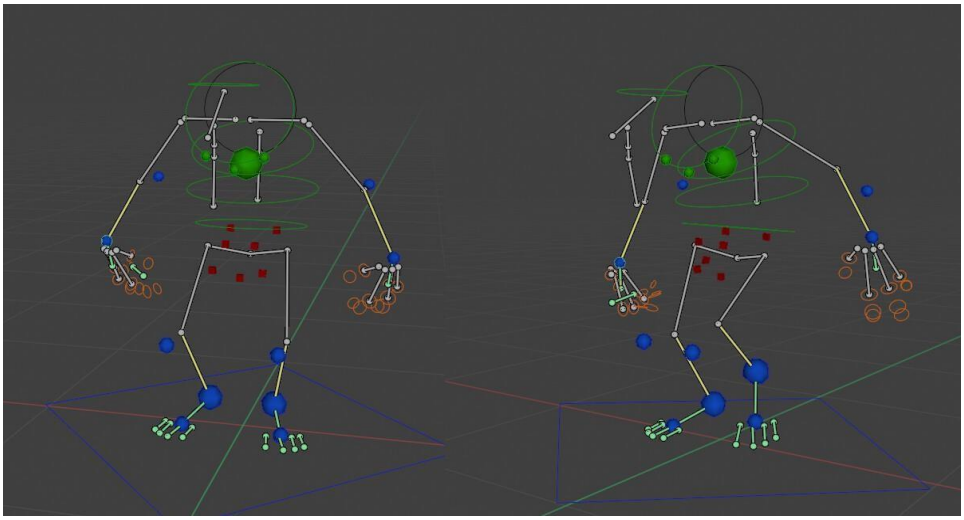

Рис. 1.16. Система кісток ригінгу

На рисунку 1.16 синіми маркерами показані суглоби моделі, а зелені кільця виступають контролерами анімаційних рухів тулуба для цієї моделі. Червоні кільця відповідають контролерам анімаційних рухів верхніх кінцівок персонажа. Червоні квадрати відповідають за рух одягу. Зелені кулі є ключовими в цій системі, оскільки центр мас тіла персонажа знаходиться у верхній частині.

1.6 Висновки першого розділу

Висновки стосовно розділу після попереднього аналізу:

1. Враховуючи можливість реалізації системи анімації у багатьох редакторах і складнення її за допомогою різних технологій моделювання та анімації, вирішено використовувати стандартний технічний процес для моделювання та анімації. Це забезпечує узгодженість та надійність кінцевого результату, мінімізуючи технічні проблеми та оптимізуючи робочий процес.

2. Детально проаналізовано різні технології для створення 3D моделей та анімацій, включаючи вивчення новітніх програмних засобів та методів, що підвищують якість та реалістичність кінцевого продукту. Зокрема, розглядалися інструменти для високополігонального моделювання, текстурування, освітлення та процедурної анімації.

3. Проведено всебічний огляд існуючих рішень, створених іншими розробниками. Оцінено якість та складність реалізації моделей на всіх етапах їх створення — від концептуального дизайну до фінальної анімації. Особливу увагу приділено методам створення анімації, таким як ключова анімація, процедурна анімація та технології motion capture. Аналізовані системи анімації різнилися за складністю, рівнем деталізації та процесом створення ригінгу, що впливає на загальну ефективність і гнучкість системи.

РОЗДІЛ 2
ВИБІР І ОБГРУНТУВАННЯ ЗАСОБІВ РЕАЛІЗАЦІЇ

2.1. Аналіз програмних середовищ для розробки

Під час виконання дипломного проєкту був проведений аналіз графічних редакторів для роботи з 3D графікою.

Цей аналіз дозволяє оцінити процес реалізації бажаної системи та її відповідність початковим умовам. На сьогодні існує багато 3D редакторів для моделювання та анімації. Вони мають схожий набір інструментів і функціональних можливостей, проте в роботі з деталізованими, високополігональними моделями та складною анімацією у кожного з них є свої переваги та недоліки. Нижче наведені таблиці (таблиця 2.1), що описують функціональні можливості 3D редакторів.

	3Ds Max	Maya	Blender
Ос	Microsoft Windows	Mac OS, Microsoft Windows, Linux	Mac OS, Microsoft Windows, Linux
Об’єм пам’яті	1 ГБ	2 ГБ	52,5 МБ / 90,8 МБ
Скриптовані мови	MaxScrip	Mel, Python	Python

Таблиця 2.1– Порівняння 3D редакторів

Згідно з даними таблиці, можна зробити висновок, що не всі редактори сумісні з основними операційними системами. Наприклад, 3Ds Max і Maya, які мають розширені можливості для роботи з анімацією та моделюванням, вимагають більшого об'єму пам'яті. Особливості редакторів надані в таблиці 2.2, переваги та недоліки в таблиці 2.3 на наступній сторінці.

Таблиця 2.2

Особливості 3D редакторів

	3Ds Max	Maya	Blender
Особливість редактору	Ядро з графічним прискоренням Nitrous(Підтримка необмеженої кількості джерел світла, Тонування зображення і прозорість об’єктів більш високої якості).	Модуль PlainEffects, за допомогою якого можна створити тривимірні об’єкти, візерунки, квіти, траву.	Наявність симулятора флюїдів, який відкриває перед користувачем великі можливості у створенні ефектів, таких як дим та рідини.
Особливості створення тривимірної моделі	Доступність всіх необхідних інструментів, в тому числі редаговані полігон, каркас та лоскут.	Серед засобів моделювання присутні засоби реагування NURBS кривих і полігонів, а також різні складні механізми накладання текстур і налаштування їхніх координат.	Підтримка різноманітних геометричних примітивів, включаючи полігональні моделі, систему швидкого моделювання в режимі Subdivision surface face, метасфери, скульптинг та векторні шрифти
Анімація	Зручний механізм анімації персонажів, котрий називається інверсною кінематикою, який дозволяє побудувати суглобів і складових частин механізму.	Пряма та інверсна кінематика, спеціальні деформатори, які дозволять створення або руху волосся в повітрі.	Інверсна кінематика, скелетна анімація і анімація по ключовим кадрам, нелінійна анімація, динаміка м’яких тіл, динаміка частинок.

Продовження таблиці 2.2

	3Ds Max	Maya	Blender
Переваги	• V-ray — найпотужніший рендер. • Потужна система частиць. • Найкраще в індустрії полігональне моделювання •Фотореалітисність.	• Створення та редагування полігональньної графіки. • Скульптинг на рівні ZBrush. • UV текстури, нормалі та колірне кодування.	•Безкоштовний • Простий та зручний інтерфейс. • Відкритий вихідний код на Python. • Анімація об'єктів та їхніх частин. •Текстури, шейдери, мапінг.
Недоліки	•Перевантажений інтерфейс. • Потреба в підключені модулів, для інтеграції в інші редактори. • Кросплатформність. Доступний лише для Windows, а для Mac OS доведеться встановити додаткове ПЗ. • Ціна. • Високі системні вимоги.	• Обмежений продукт для звичайного користувача. • Високі системні вимоги. • Ціна. • Не підходить для створення дизайну екстер'єрів/інтер'єрів та архітектури, оскільки відсутні спеціальні	•Мала кількість бібліотек моделей. • Рідко використовується студіями в пайплайні. • Середній фунціонал. • Не сумісний з іншими редакторами.

2.2. Вибір методу моделювання персонажу

Для моделювання персонажа необхідно з'ясувати, які існують способи моделювання, і визначити, який з них найкраще підходить для створення 3D моделі. Варіацій фактично є три: полігональне, NURBS моделювання, та моделювання САПР

Полігональне моделювання є найпопулярнішим способом розробки 3D моделей є полігональне моделювання. Суть цього методу полягає у створенні та редагуванні сітки з полігонів, що складаються з вершин і ребер.

Створити полігональну модель можна різними способами:

- Шляхом з’єднання примітивів: за основу береться просте геометричне тіло, наприклад, куля, куб або тор. При необхідності можна змінити кількість граней, що дозволяє задати будь-які розміри примітивам.

- Ручне моделювання: коли інші методи не підходять, об’єкти створюються шляхом малювання вручну.

- Екструзія: об’єкти створюються шляхом витягування нових граней з вихідного полігону.

На рисунку 2.1 представлений приклад полігонів. Натискаючи кнопку миші, ми створюємо нову вершину, яка з'єднується ребром. Такий процес моделювання можна уявити як форму, покриту прямокутниками з різним ступенем перспективного спотворення.

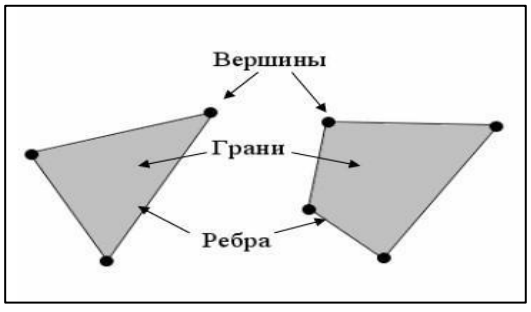

Рис. 2.1. Атрибути полігонів

Оцінка переваг та недоліків полігонального методу моделювання наведена в таблиці 2.1.

Таблиця 2.1.

Переваги та недоліки полігонального моделювання

Переваги	Недоліки
Оскільки сучасні комп'ютери оптимізовані для обробки полігонів, полігональні моделі найпростіші для рендеру та візуалізації.	Побудова моделі з багатокутників є неточною.
Полігональне моделювання дозволяє дизайнерам створювати більш унікальні / органічно виглядають конструкції (люди, тварини тощо).	Полігональні моделі не витримують усіх резолюцій.
Оскільки полігональні моделі створюються з менших компонентів (трикутні або чотирикутні полігони), вони можуть деформуватися та анімуватися більш природно.	Полігональне моделювання дуже трудомістке, особливо для складних конструкцій

NURBS моделювання - або неоднорідні раціональні B-сплайни, являють собою спосіб математичного відображення тривимірної геометрії, який дозволяє точно описувати будь-які форми - від найскладніших тривимірних органічних поверхонь або твердих об'єктів до простих 2D ліній, кіл, дуг або кривих. Основна відмінність цього методу від полігонального моделювання полягає в його плавності. NURBS моделі складаються не з полігонів, а зі сплайнів (кривих).

Цей метод застосовується для створення плавних органічних форм і моделей, особливо коли потрібно реалізувати складну тривимірну поверхню природного походження. Хоча таку поверхню можна описати вершинами та розбити на примітиви, це займе багато часу, а змінювати криву поверхню в полігонах буде незручно. У таких випадках використовуються можливості моделювання NURBS.

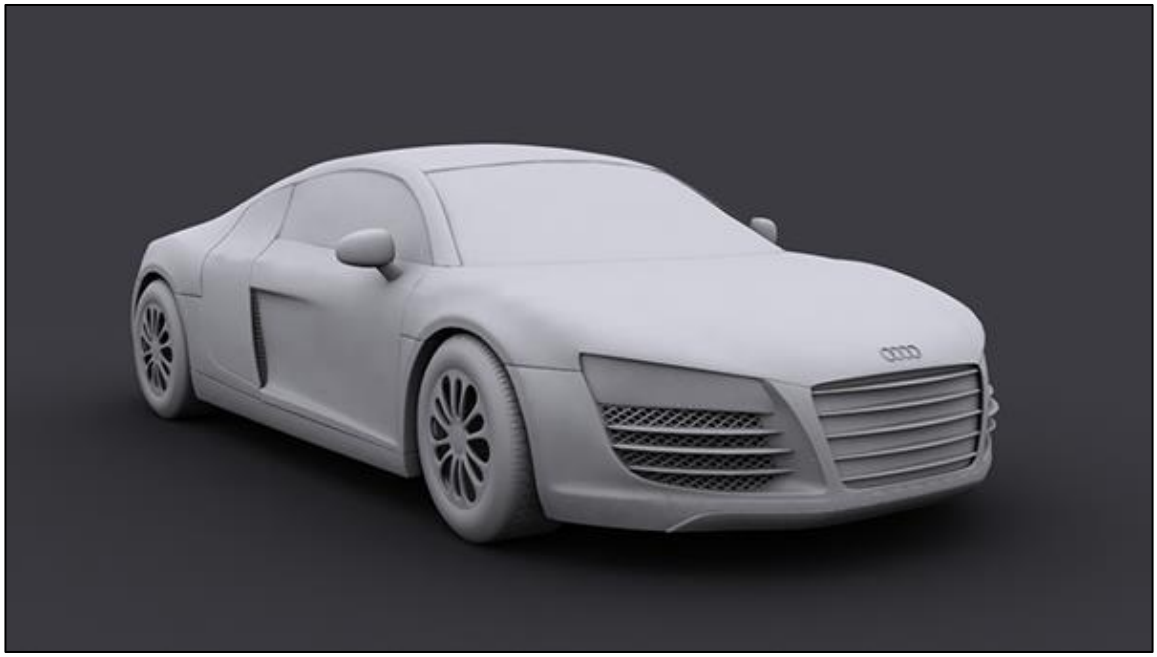

Рис. 2.2. Модель створена методом NURBS моделювання

Та також моделювання в САПР - Цей метод моделювання використовує математичні формули для створення поверхневих моделей, які залишаються абсолютно гладкими при будь-якому збільшенні, з точністю до мілліметра. Він

застосовується у випадках, коли висока точність є ключовою, а не естетичний вигляд. Теоретично, цей метод можна використовувати для створення персонажів, але це вимагає значних часових і обчислювальних витрат, порівняно з полігональним моделюванням, приклад моделі на рисунку 2.3

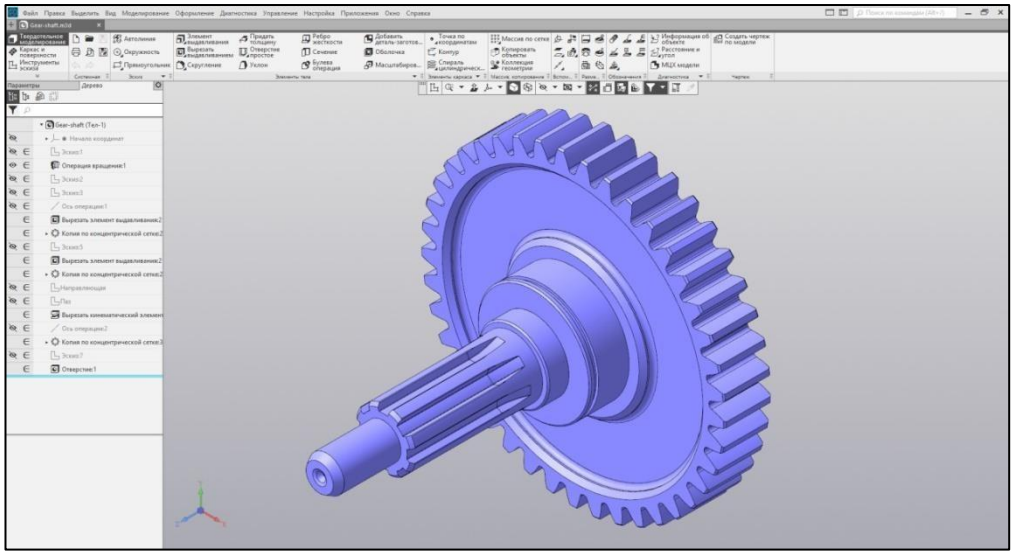

Рис. 2.3. Модель створена методом САПР моделювання

2.3. **Вибір методу реалізації анімації персонажу**

Обертанням моменту вибору методики для анімації персонажа в ігровому світі залежить успішність у створенні бажаної системи. Під час вибору методу необхідно керуватися наступними критеріями:

- Забезпечення реалістичності рухів персонажа.

- Ефективне використання ресурсів комп’ютера для реалізації системи.

Наприклад анімація по ключовим кадрам використовується у процесі створення 3D-анімації, де використовуються різні параметри, такі як положення, орієнтація та освітлення. Ключові кадри мають велике значення, оскільки вони визначають моменти на часовій шкалі, коли починається або закінчується перехід. Послідовність

таких ключових кадрів створює ілюзію руху для глядача. Наприклад, якщо анімація працює із швидкістю 20 кадрів на секунду, то за одну секунду на екрані відображається двадцять зображень. Між цими ключовими кадрами відбуваються інтервали. У комп'ютерній анімації числові параметри автоматично налаштовуються для плавного переходу до наступного кадру, що забезпечує гармонійний рух. Процеси "твінінгу" та "проміжку часу" - це методи інтерполяції, які допомагають зробити переходи між кадрами більш плавними.

А от для анімації по траекторії, стркутурі об'єкта вздовж траєкторії, потрібно визначити не лише сам об'єкт, який буде анімований, але й його шлях руху. Цей метод часто використовується для анімації рухомих об'єктів, камери та інших технічних елементів. Основна концепція полягає у такому:

1. Визначенні початкової точки (початку шляху об'єкта).

2. Побудові траєкторії (шляху, яким буде рухатися об'єкт).

3. Встановленні кінцевої точки (місця, де об'єкт має зупинитися).

Вибравши об'єкт, який буде рухатися вздовж траєкторії, призначаємо йому цей шлях для анімації. Потім об'єкт розміщується на лінії траєкторії та з'єднується з нею. Під час цього процесу програма автоматично створює два ключових кадри: один, що показує початкове положення об'єкта, та інший, що визначає його місцезнаходження в кінці траєкторії. Решта кадрів програма інтерполює. В результаті об'єкт рухається вздовж визначеної траєкторії під час програвання анімації.

Є також створення анімацій при динамічних симуляціях, в такому випадку у тривимірних середовищах властивості об'єктів не визначаються наперед, і для того, щоб кожен об'єкт ведів себе подібно до реального світу, використовуються динамічні симуляції. Ці симуляції здійснюються за допомогою програм тривимірної графіки, а потім генеруються анімаційні ключі, що містять дані про поведінку кожного елемента. Динамічні симуляції широко використовуються для моделювання різних матеріалів, таких як рідини, тканини, а також тверді і м'які об'єкти.

На кінець хочу зазначити що створення волосся, яке може бути як частиною одягу, так і саме волоссям на голові персонажа, чи покривати його повністю в разі потреби, якщо це істота з повним покриванням моделі хутром, є великою проблемою, так як це вимагає багато зусиль, але в той же час забезпечує максимально реалістичну модель. Цей процес створює необхідну кількість ключових волосків вздовж кривих поверхонь NURBS.

Алгоритм генерації волосся має два необов'язкових аргументи: кількість кривих волосся (n) і кількість контрольних точок, які має мати кожна крива волосся (c). Ці дві змінні, встановлені користувачем, використовуються разом з параметричним діапазоном (значення u і v) поверхні для визначення контрольних точок ключових кривих волосся. Інкрементні розміри кроків для u та v обчислюються та використовуються для циклу через параметричний діапазон для генерації ключових кривих волосся.

2.4. Вибір мови скриптів в середовищі графічного редактору

Оскільки Maya є основним редактором для майбутнього проєкту, робота зі створенням скриптів відбувається в редакторі сценаріїв, який відомий як Script Editor. При запуску Maya фактично виконується кілька сценаріїв MEL (Maya Embedded Language), які формують всі вікна. По суті, Maya не має жодного власного інтерфейсу. Навіть можна запустити Maya з командного рядка операційної системи, просто введіть команду "Maya -prompt"! Практично все, що створюється в Maya, виконується за допомогою сценаріїв MEL. Ця мова чудово підходить для виконання майже 95% всіх завдань, вона містить понад 600 команд і приблизно 75 функцій. MEL забезпечує швидку та гнучку взаємодію з усіма інструментами всередині Maya. Ось приклад команди MEL: sphere – radius 10 – name ball.

Також при роботі буде застосовуватись мова Python, яка є автономною мовою сценаріїв, незалежною від системи Maya, і вона є дуже популярною в галузі

візуальних ефектів. У іграх зазвичай потрібно працювати з форматами, такими як JSON, або з власними форматами даних, і Python має неймовірну підтримку для цього.

Python використовується для створення серій текстур для конкретної моделі, які потім можна пакетно перетворити в інший формат. Крім того, Python має модулі для безпосереднього доступу до API C++ через відкритий модуль API. Коли потрібно створити новий деформер, можна швидко розпочати та створити прототип через Python API, а потім перенести концепції на C++.

Наприклад, команда Python може виглядати так:

python

import maya.cmds as cmds

cmds.sphere(radius=10, name='ball')

Префікс `maya.cmds` повідомляє Python, що викликається команда MEL. Потім параметри `radius` і `name` передаються команді `sphere` як аргументи, відповідно до стандартного синтаксису Python.

2.5. Висновки другого розділу

Висновком до розділу хочу зазначити:

У даному розділі проаналізовано інструменти та методи для втілення системи анімації персонажів, і було зроблено такі висновки:

1. Під час аналізу графічних редакторів 3D було обрано Autodesk Maya. Цей редактор володіє необхідним набором функцій та інструментів для реалізації проекту.

2. У якості методів моделювання було обрано полігонне та NURBS моделювання. Ці підходи важливі для створення майбутньої моделі, оскільки дозволяють досягти потрібних результатів.

3. Серед розглянутих методів анімації найбільш відповідними для системи є анімація за ключовими кадрами та анімація траєкторії. Ці методи найкраще підходять для імітації різних рухів персонажа.

4. Ще одним етапом реалізації системи є написання скриптів анімації. Оскільки в редакторі Autodesk Maya є можливість використовувати дві мови скриптів — MEL та Python, раціональним вибором буде використання Python, яка є більш універсальною і незалежною від редактора.

РОЗДІЛ 3
РЕАЛІЗАЦІЯ СТВОРЕННЯ 3D МОДЕЛІ ТА АНІМАЦІЇ

3.1. Структура 3D моделі

Наразі буде проводитись розробка та налагодження безпосередньо моделі персонажу, саму розробку, для зручності опису, потрібно розбити на частини, та продемонструвати структуру розробки, що, та після чого буде йти, спочатку потрібно розібратись з структурою моделі.

Структура тривимірної моделі складається з полігональної сітки (mesh), що формує основу для розгортки (UV) з об'єктами (Objects). Текстури (Textures) мають різноманітні картки, включаючи дифузні (Diffuse) та нормальні (Normal). Дифузна карта визначає базовий колір (Base color), відблиск (Metallic) і текстурну шорсткість (Roughness). Обидві карти піддаються впливу матеріалів (Materials), які налаштовані за допомогою конкретних шейдерів (Shaders).

Структурована схематика 3D моделі зображена на рисунку 3.1.

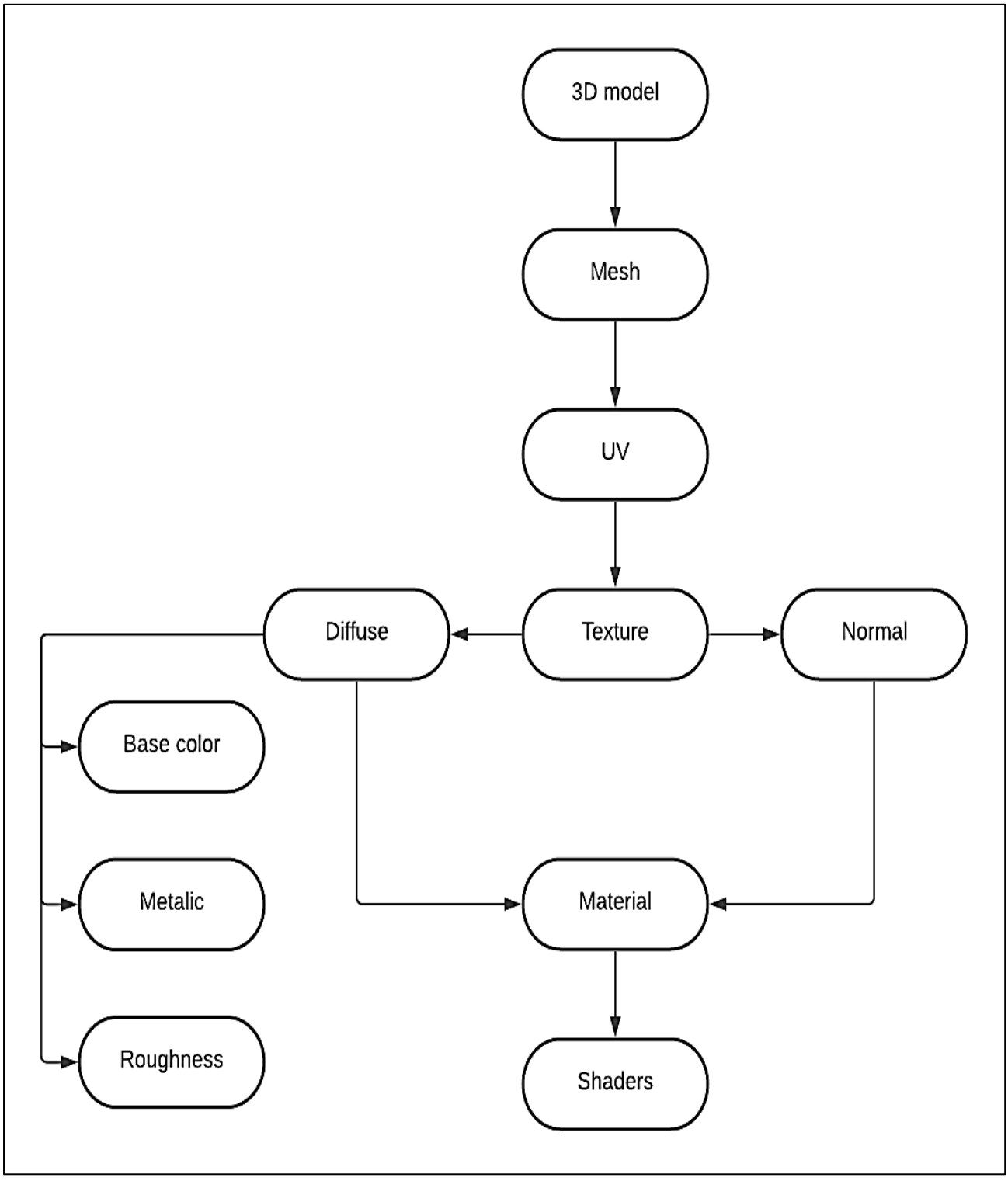

Рис. 3.1. Структура основних компонентів моделі.

Ключові особливості компонетів структури 3D моделі вже було висвітлено в попередніх розділах, тому саме головне що потрібно продемонструвати по роботі це:

Текстурні карти, фактично це особливості дифузної картки (Diffuse) які чітко визначають базовий колір (Base color) матеріалу [2]. Під час роботи з карткою Metallic, яка заснована на карті Base color, враховується інформація з дифузної карти. Ця картка також визначає, яка частина карти Base Color сприймається як дифузний вихід. Чим більше металевих областей на карті, тим більше вона буде відображати кольору.

Карти шорсткості регулюють кількість світла, що відбивається від тривимірної моделі. У PBR-орієнтованих ігрових рушіях/рендерах (render), чим вищий рівень шорсткості, тим менше світла відбивається від поверхні, що означає меншу кількість блисків.

Карти нормалей (Normal) представляють собою RGB-зображення, де кожен канал (червоний, зелений, синій) відповідає за координати нормалей поверхні X, Y та Z відповідно. Червоний канал відповідає за вісь X (нормалі, спрямовані вправо або вліво), зелений канал за вісь Y (нормалі, спрямовані вниз або вгору), а синій канал за вісь Z (нормалі, спрямовані прямо від поверхні). Одиничні вектори Normal, які відповідають текстурним координатам u, v, зображаються на картах нормалей. Вони представлені тільки векторами, спрямованими на глядача (z: від 0 до -1 для орієнтації зліва направо), оскільки вектори на геометріях, спрямованих від глядача, не відображаються. Порівняння наведено на рисунку 3.2.

```
X: -1 to +1 :  Red:     0 to 255
Y: -1 to +1 :  Green:   0 to 255
Z:  0 to -1 :  Blue:  128 to 255
```

Рис. 3.2. Зіставлення векторів та їхнього відображення в RGB

Також , шейдер (Shader) [4] необхідний для коректного відображення та рендерингу моделі. Якщо шейдер не призначений для поверхні, об’єкт не буде відображений під час рендерингу.

Шейдери (Shaders) побудовані на основі вузлів. Кожен вузол має атрибути, які визначають властивості шейдера. Для 3D моделей створюються мережі шейдерів з взаємопов’язаних вузлів, що подібні до ієрархічних структур та груп моделей.

Вершинний шейдер (Vertex Shader) зв’язаний з вершинами полігонів моделі та їх текстурними координатами. Цей шейдер містить атрибути для позиції та кольору. На рисунку 3.3 наведено опис Вершинного шейдера.

```
layout (location = 0) in vec3 position;


out vec4 vertexColor;

void main()
{
    gl_Position = vec4(position, 1.0);
    vertexColor = vec4(0.5f, 0.0f, 0.0f, 1.0f);

}
```

Рис. 3.3. Опис вершинного шейдеру

Standard Surface Shader - це типовий фізичний шейдер, який може створювати різноманітні типи матеріалів. Він включає в себе дифузний шар та поверхневе розсіювання, які особливо корисні для реалістичного відображення шкіри персонажу.

Цей шейдер створює ефект, де матеріал розбивається на десять компонентів, які розподіляються і змішуються у ієрархічному порядку, як зображено на рисунку 3.4. Параметри кожної окремої компоненти можуть варіюватися по всій поверхні. Горизонтальне розташування компонентів представляє статистичну змішану структуру, тоді як вертикальне розташування - це їх взаємне шарування.

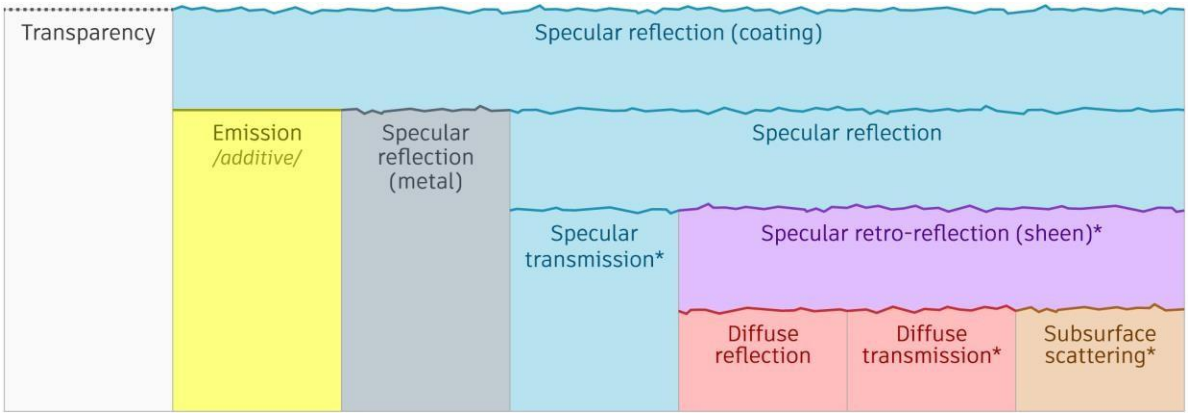

Рис. 3.4. Схематична ілюстрація моделі матеріалу

На рисунку 3.5 представлено візуалізацію взаємозв'язку між шейдером та текстурними картами, а також його структуру.

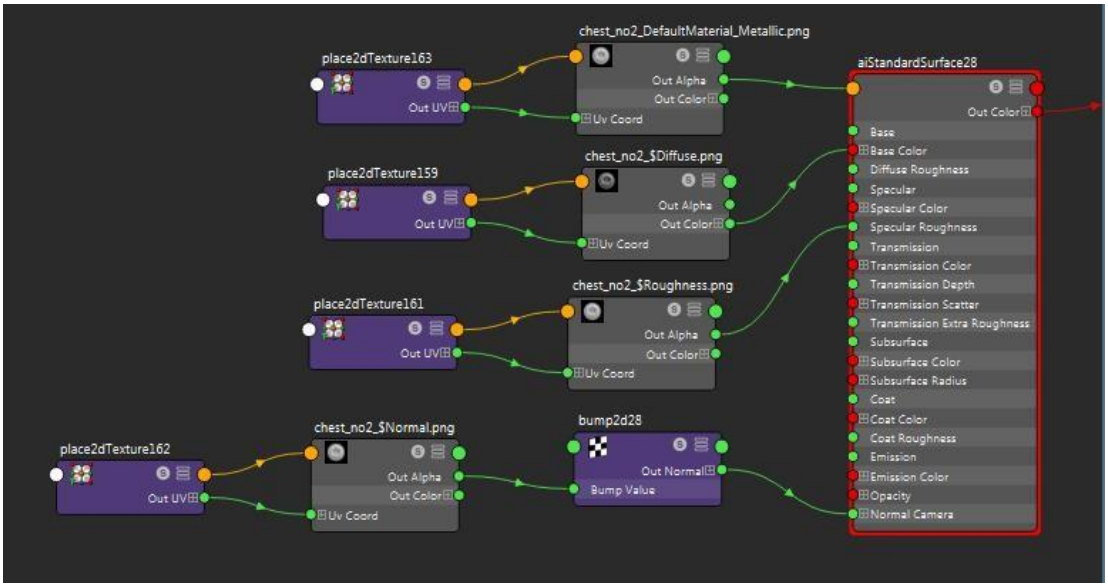

Рис. 3.5. Структура Standard Surface Shader та зв’язок з текстурними Картами

Standard Surface Shader - це стандартний шейдер, який дозволяє моделювати різноманітні типи матеріалів. Він включає дифузний шар та можливість розсіювання світла для імітації шкіри персонажу.

Рис. 3.6. Структура шейдеру Traditional Game Surface та прив’язка до матеріалу

3.2. Структура керуючих компонентів для анімації моделі

Завдяки належно налаштованій системі анімації, майбутня 3D-модель може бути інтегрована в ігровий двигун та піддаватися різним маніпуляціям. Ця інтеграція та налаштування здійснюються за допомогою керуючих компонентів для скелетної анімації. Кожен з цих компонентів має свою власну ієрархію, що спрощує процес створення анімацій і уникне необхідності встановлення та збереження анімації для кожної окремої кістки. Структурна схема таких компонентів для анімації 3D-моделі представлена на рисунку 3.7.

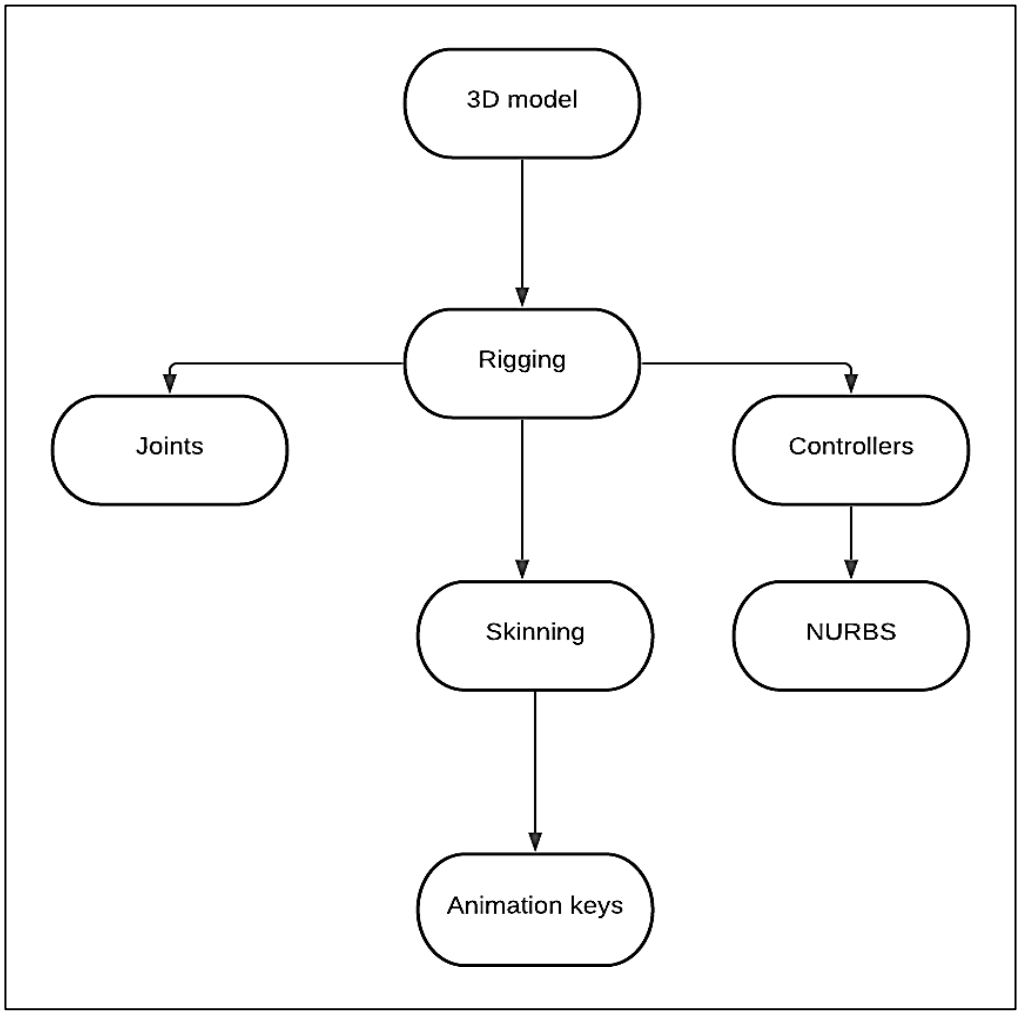

Рис. 3.7. Структурна схема компонентів для анімації 3D моделі

Rigging є ключовим компонентом у процесі анімації, оскільки він встановлює основну ієрархію для майбутніх анімаційних дій. Він включає в себе повну ієрархію кісток (Joints), а також ієрархію контролерів (Controllers) з кривими NURBS, які визначають напрям руху для кісток. На основі рігу (Rigging) створюється Skinning, що дозволяє "прикріпити" полігональну сітку 3D-моделі до скелету. Анімаційні ключі (Animation keys) містять інформацію про анімаційні рухи персонажу.

3.3. Ключові особливості компонентів структури анімації

При використанні Joints, хочу зауважити що днією з його особливостей є їхнє функціональне призначення у створенні конструктивних блоків для кісток та точок артикуляції. Вони не мають конкретної форми і не піддаються візуальному відображенню (rendering). Кожен з'єднувальний елемент (joint) може мати прикріплену до себе одну або кілька кісток. Кістки (bones) не мають окремих вузлів і не існують як фізичні або обчислювальні об'єкти в сцені. Вони представляють лише візуальні сигнали, які демонструють взаємозв'язок між joints. Процес побудови joints показано на рисунку 3.8..

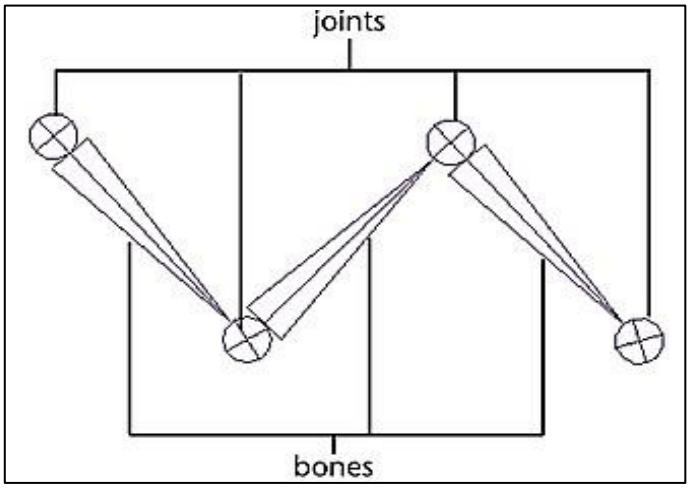

Рис. 3.8. Принцип побудови joints

Для створення joint можна використовувати скрипти (наприклад, MEL або Python) або вбудовані команди у графічному редакторі Autodesk Maya.

Основними параметрами для створення joint є його позиція або розташування, яке визначається трьома координатами X, Y, Z, а також кут, який вказує на його поворот у вибраній площині.

Також важливо зазначити, що групи joints для кінцівок (наприклад, руки чи ноги) можна створювати лише для однієї сторони. Це обумовлено тим, що кінцівки персонажа є симетричними, і пізніше ці групи будуть автоматично дубльовані відносно основного root joint.

Ця обмеженість виникає через симетричну будову кінцівок персонажа. Пізніше, коли модель ригується, кінцівки автоматично віддзеркалюються відносно основного root joint, що допомагає забезпечити правильну симетрію і рухомість.

На рисунку 3.9 показано інструкції та послідовність команд для скрипту, який створює joint.

```
import maya.cmds as cmds

# Create a 3-joint chain
#
cmds.select( d=True )
cmds.joint( p=(0, 0, 0) )
cmds.joint( p=(0, 4, 0)  )
cmds.joint( 'joint1', e=True, zso=True, oj='xyz' )
cmds.joint( p=(0, 8, -1) )
cmds.joint( 'joint2', e=True, zso=True, oj='xyz' )

# Create a fourth joint with z joint limits of -90 deg for
# the lower limit and 90 deg for the upper limit.  The
# joint will be positioned at (0, 0, 4) in world
# coordinates.
#
cmds.joint( lz=('-90deg', '90deg'), p=(0, 8, 4) )

# Set the joint limits but leave them disabled.
cmds.joint( edit=True, lz=('-90deg', '90deg'), lsz=False )
```

Рис. 3.9. Опис та порядок задання команд для joints(Python)

Проблема ,та одночасно особливість joints виявляється в їхній орієнтації. Ця особливість унікальна тим, що лише joints можна зорієнтувати відносно координатних осей.

Для того щоб в подальшому анімувати модель, кожну кістку необхідно правильно орієнтувати. Це важливо, оскільки деякі кістки мають обмеження у напрямку згину порівняно з іншими. Для досягнення цієї мети можна використати скрипт OrientJoint. Його опис та функціонал наведено на рисунку 3.10.

```
def addRotToOri(joint):
    orientation = pm.getAttr(joint.jointOrient)
    rotation = pm.getAttr(joint.rotate)
    for i in range(3):
        orientation[i] += rotation[i]
    nullVector = [0, 0, 0]
    pm.setAttr(joint.rotate, nullVector)
    pm.setAttr(joint.jointOrient, orientation)
```

Рис. 3.10. Опис скрипту OrientJoint(Python)

Для забезпечення коректної анімації важливо дотримуватись чіткої ієрархії кісток (joints). У цій ієрархії завжди присутній основний (Root) joint, який виступає головним відносно усіх інших кісток. Цей основний joint має вирішальне значення для забезпечення симетрії рухів персонажа в подальшому. На рисунку 3.11 зображено структуру ієрархії joints

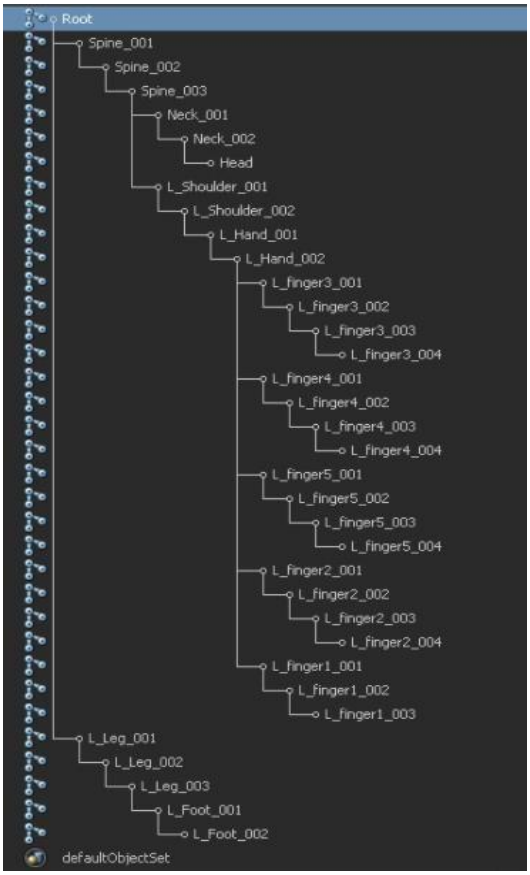

Рис. 3.11. Правильна ієрархія joints

3.4. Контролери(Controllers) та їхнє використання

Контролери дотримуються тієї самої ієрархії, що й кістки (joints), тому кожен контролер створюється для конкретного joint. Зазвичай контролери створюються за допомогою NURBS кривих, оскільки ці криві не рендеряться і легко піддаються редагуванню їх форми. Крива назначається так, що її центр збігається з центром joint.

Для створення контролера потрібно обрати один або кілька joints і використати відповідний скрипт. Опис процесу створення контролера для групи joints подано на рисунку 3.12.

```
import maya.cmds as cmds
sl = cmds.ls(sl=1)
for s in sl:
ctrlName = s.replace("_jnt", "_ctrl")
ctrl = cmds.circle( nr=(0, 1, 0), r=3,  n=ctrlName)[0]
group = cmds.group(ctrl, n=ctrl + "_auto")
offset = cmds.group(group, n=ctrl + "_offset")
cmds.parentConstraint(s, offset, mo=0)
cmds.delete(cmds.parentConstraint(s, offset))
cmds.parentConstraint(ctrl, s, mo=0)
```

Рис. 3.12. Опис скрипту для контролера

Скрипт генерує криву з заданим радіусом (r) та копіює назву joint для цієї кривої, замінюючи індекс joint на індекс контролера в ієрархії. Потім ці криві групуються двічі для створення ієрархій нод (node) "auto" і "offset". Нода "offset" прив'язується до відповідного joint за допомогою Parent Constraint, щоб зайняти відповідне положення, а сам Parent Constraint видаляється. В результаті joint стає прив'язаним до контролера.

Також для контролерів для IK-joints, потрібно розуміти, так як під час роботи з ригом персонажа для певних груп joints (таких як хребет, плечі, ноги) застосовуються контролери з інверсійною кінематикою (IK). Коли створюються інтерполяції пози, редактор не може розпізнати joint як рушій пози. Тому атрибут керування встановлюється для контролера, і застосовуються інтерполятори пози для контрольованих joints. Це зазвичай створює відображення з'єднання, де пози визначаються атрибутами контролера.

```
import maya.cmds as cmds

# Will create a handle from Joint-1 to an end-effector at
# the location of Joint-5 with a priority of 2 and a
# weight of 0.5
#
cmds.ikHandle( sj='joint1', ee='joint5', p=2, w=.5 )

# Create a handle called leg from the start joint
# named hip to the end-effector named Ankle.
#
cmds.ikHandle( n='Leg', sj='Hip', ee='Ankle' )
```

Рис. 3.13. подано опис скрипту для створення IK handle.

Для створення інверсійної кінематики joint використовується спеціальний скрипт, який можна застосовувати тільки після створення контролерів. Опис цього скрипту наведено на рисунку 3.13, що знаходиться вище.

Спочатку обирається конкретне з’єднання, а потім викликається команда IK handle. Ця команда дозволяє контролювати орієнтацію групи joints. Крім того, IK handle має значення прапорців (flags), які встановлені за замовчуванням.

- -name "ikHandle#"
- -priority 1
- -weight 1.0
- -positionWeight 1.0
- -solver "ikRPsolver"
- -forceSolver on
- -snapHandleFlagToggle on
- -sticky off
- -createCurve true

- -simplifyCurve true
- -rootOnCurve true
- -twistType linear
- -createRootAxis false
- -parentCurve true
- -snapCurve false

Тепер переходимо до скінінгу, так як під час редагування позицій та орієнтації joints для анімації об'єктів у полігональних моделях, часто виникає проблема неправильної деформації, що може породжувати артефакти або небажані зміни форми. Особливо це стає актуальним при використанні інструментів для зміни розмірів, обертання або згинання joints. Для контролю цих деформацій та досягнення потрібного результату використовується вага (weight), яка визначає ступінь впливу кожного joint на конкретну вершину полігональної сітки. Кожна вершина повинна мати інформацію про те, яким чином кожен joint впливає на неї та в якій мірі. Ця інформація представлена коефіцієнтом, який встановлюється для кожного joint. Важливо пам'ятати, що сума всіх коефіцієнтів повинна дорівнювати одиниці, щоб забезпечити правильну деформацію моделі під час анімації.

Для контролю орієнтації групи joints обирається конкретне з'єднання, і застосовується команда IK handle, яка дозволяє керувати орієнтацією групи joints. Крім того, IK handle має прапорці (flags), які встановлені за замовчуванням.

За допомогою цих коефіцієнтів відбувається змішування обчислених координат вершин (vertex blending). Можна відредагувати вплив joints для однієї групи кінцівок у вікні редактору та віддзеркалити його на ідентичні групи за допомогою команди copySkinWeights. Віддзеркалення відбувається відносно root joint.

Опис структури команди зображено на рисунку 3.14.

```
# Create plane and a skeleton. Bind the skin.
#
cmds.file( f=True,new=True )
cmds.polyPlane( ch=1, w=10, h=10, sx=5, sy=5, ax=(0,1,0) )
cmds.select( d=True )
cmds.joint( p=(0, 0, -6) )
cmds.joint( p=(0, 0, -4) )
cmds.joint( 'joint1', e=True, zso=True, oj='xyz')
cmds.joint( p=(2, 0, -4) )
cmds.joint( 'joint2', e=True, zso=True, oj='xyz')
cmds.joint( p=(5, 0, -3) )
cmds.joint( 'joint3', e=True, zso=True, oj='xyz')
cmds.select( 'joint2', r=True )
cmds.joint( p=(-2, 0, -4) )
cmds.joint( 'joint4', e=True, zso=True, oj='xyz')
cmds.joint( p=(-5, 0, -3) )
cmds.joint( 'joint5', e=True, zso=True, oj='xyz')
cmds.select( 'joint2', r=True )
cmds.joint( p=(0, 0, 3) )
cmds.joint( 'joint6', e=True, zso=True, oj='xyz')
cmds.joint( p=(5, 0, 5) )
cmds.joint( 'joint7', e=True, zso=True, oj='xyz')
cmds.select( 'joint7', r=True )
cmds.joint( p=(-5, 0, 5) )
cmds.joint( 'joint8', e=True, zso=True, oj='xyz')
cmds.select( 'pPlane1', 'joint1', r=True )
maya.mel.eval('createSkinCluster "-mi 5 -dr 4"' )
# Modify some weights on the -x side of the character
#
cmds.skinPercent( 'skinCluster1', 'pPlane1.vtx[30]', tv=('joint2',0.200000) )
cmds.skinPercent( 'skinCluster1', 'pPlane1.vtx[31]', tv=('joint2',0.200000) )
cmds.skinPercent( 'skinCluster1', 'pPlane1.vtx[24]', tv=('joint5',0.550000) )
cmds.skinPercent( 'skinCluster1', 'pPlane1.vtx[25]', tv=('joint5',0.550000) )
# Mirror the skin weights to the other side of the character
# Mirror inverse is chosen since we want to go from -x to +x, not +x to -x.
#
cmds.copySkinWeights( ss='skinCluster1', ds='skinCluster1', mirrorMode='YZ', mirrorInverse=True )
# Now create a second plane and bind it as skin
#
cmds.polyPlane( ch=1, w=10, h=10, sx=5, sy=5, ax=(0,1,0) )
cmds.select( 'pPlane2', r='joint1' )
maya.mel.eval('createSkinCluster "-mi 5 -dr 4"' )
# Copy the skin weights from the first plane onto the new plane.
# The -noMirror flag is used since we want to copy directly, not mirror.
#
cmds.copySkinWeights( ss='skinCluster1', ds='skinCluster2', noMirror=True )
```

Рис. 3.14. Опис структури команди copySkinWeights

І фінальна стадія роботи, робота з анімаційними ключами, цей інструмент дозволяє створювати анімацію, встановлюючи значення або атрибути персонажа у будь-який визначений момент часу. Коли потрібно, ви встановлюєте характеристики персонажа (розмір, положення, кут повороту) і робите поточний кадр ключовим. Редактор фіксує всю інформацію про властивості 3D моделі на цьому ключовому кадрі. Подібні маніпуляції виконуються для наступних обраних ключових кадрів. Пізніше редактор автоматично визначає, як персонаж буде змінюватись між цими ключовими кадрами, забезпечуючи плавну анімацію. На рисунку 3.15 показаний опис структури скрипту для ключових кадрів (Keyframes).

```
import maya.cmds as cmds
cmds.keyframe( 'surface1', attribute='translateX', query=True, keyframeCount=True )
cmds.keyframe( 'surface1', time=(0,20), query=True, valueChange=True, timeChange=True);
cmds.keyframe('surface1.translateX',index=(0,0),query=True);
cmds.keyframe(edit=True,relative=True,timeChange=1,time=(10,20))
cmds.keyframe(time=(10,10),timeChange=12)
cmds.keyframe('surface1.translateX',edit=True,index=(1,1),timeChange='1.5sec',valueChange=10.25)
cmds.keyframe('nurbsCone1',at='tx',t=(3,3),q=True,eval=True)
cmds.keyframe( 'nurbsCone1', at='tx', sl=True, q=True, tc=True )
cmds.keyframe( 'nurbsCone1', sl=True, q=True, kc=True )
myCone = cmds.cone()
cmds.setKeyframe( myCone[0], t=[0,5,10], at='tx', v=5 )
cmds.setKeyframe( myCone[0], t=[2,7,12], at='ty', v=10 )
cmds.setKeyframe( myCone[0], t=[4,9,14], at='tz', v=15 )
cmds.selectKey( t=[(5,5),(12,12),(4,4)] )
cmds.selectKey( animation='objects', add=True, t=(14,14) )

nodes = cmds.keyframe(myCone,query=True,name=True)
for node in nodes:
   keyTimes = cmds.keyframe(node,sl=True,query=True,tc=True)
   print "Node: %s" % node
   print keyTimes
cmds.keyframe( query=True, lastSelected=True, name=True )
cmds.keyframe( query=True, lastSelected=True, timeChange=True )
cmds.keyframe( query=True, lastSelected=True, valueChange=True )
```

Рис. 3.15. Опис скрипту для ключових кадрів

Параметр кривої (paramCurve), підключається до певного об’єкту(surface1.translateX) після чого надсилається запит на всі ключові кадри об’єкту в діапазоні від 0 до 20 після чого вони зміщуються відповідно до діапазону.

Індекс ключа на animCurve(крива анімації) рівний нулю, що означає перший ключ для кожної кривої

3.5. Висновки третього розділу

У даному розділі було розглянуто структурні схеми 3D моделі персонажа та його анімації, а також представлено специфіку кожного з компонентів структур моделі та анімації. Також було визначено необхідні інструменти для реалізації майбутньої системи.

Щодо створення 3D моделі, важливим етапом є реалізація якісної розгортки, з урахуванням особливостей даного компоненту, оскільки на його основі створюються текстурні карти. Обрано відповідні шейдери для матеріалів, що дозволить надати персонажу необхідний колір та текстурність.

Щодо анімації моделі, можна зазначити наступне:

1. Кожен joint може мати прикріплену до нього одну або кілька кісток.

2. Необхідно правильно зорієнтувати кожен joint.

3. Всі joints повинні складати правильну ієрархію.

4. Важливо створити кореневий joint.

5. Ієрархія контролерів має відповідати ієрархії joints.

6. Кожен контролер повинен мати NURBS криву.

7. Оскільки joints не є рушіями позиції, для них створюються контролери з інверсійною кінематикою.

8. Важливо враховувати ступінь деформації полігональної сітки під час маніпуляцій з joints.

РОЗДІЛ 4
АНАЛІЗ РЕАЛІЗОВАНОЇ СИСТЕМИ

4.1. Аналіз розроблених компонентів 3D моделі персонажу

4.1.1. Модель персонажа

При розробці моделі персонажу були задіяні знання вивчені, та розписані і продемонстровані в минулих розділах, тепер суто до демонстрації проекту та пояснення роботи, стосовно персонажу далі будуть продемонстровані частини моделі, по причині великої кількості фото, це буде розбито на 3 сторінки для демонстрації роботи, там присутні голова персонажа (Рис. 4.1), кінцівки персонажа (Рис. 4.2), верхній одяг (Рис. 4.3), загальний вигляд моделі з двох ракурсів (Рис. 4.4), також UV розгортки тіла персонажа (Рис. 4.5), та одягу персонажа (Рис. 4.6). І звичайно карта текстур зображена на рисунку 4.7 та карта кольору і блиску зображена на рисунку 4.8.

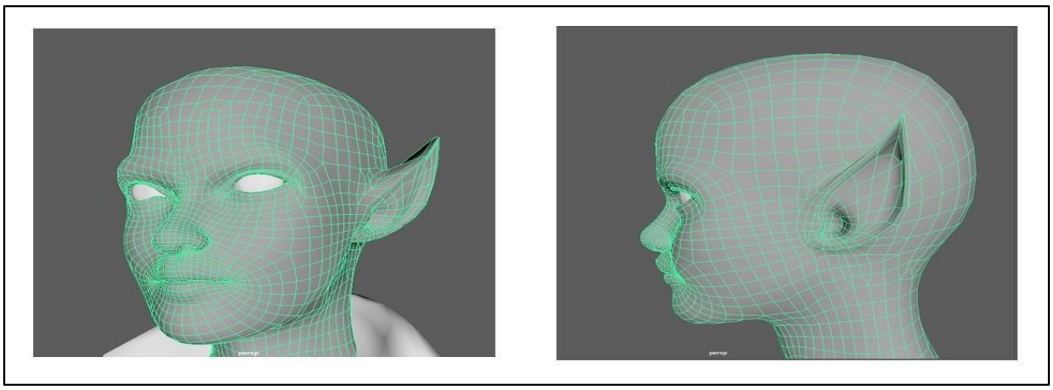

Рис. 4.1 – Голова персонажу

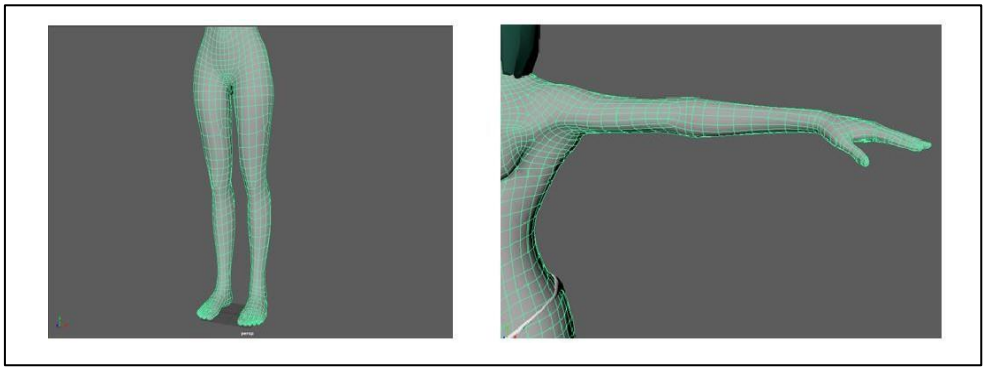

Рис 4.2. Нижні та верхні кінцівки персонажу

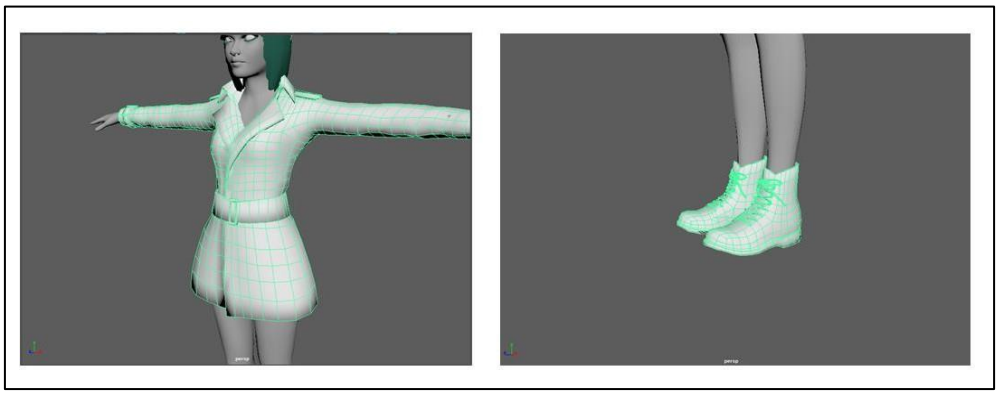

Рис. 4.3. Одяг персонажа

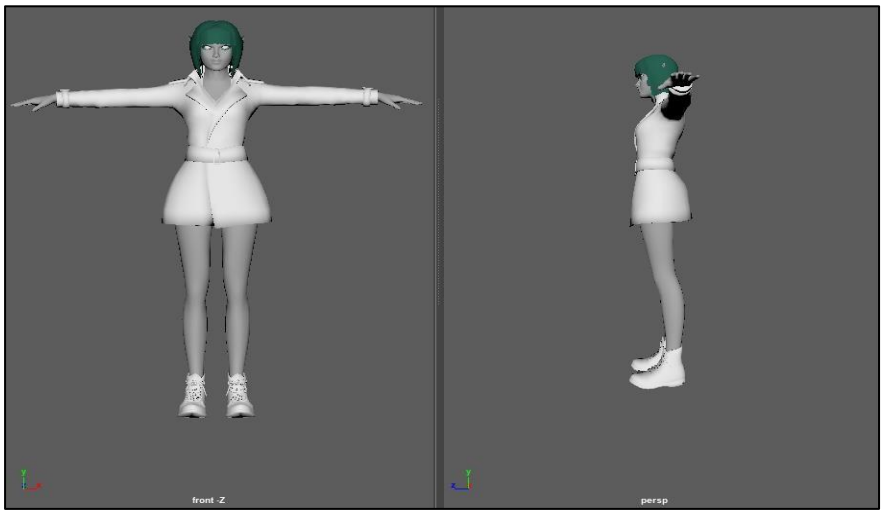

Рис. 4.4. Загальний вигляд 3D моделі персонажу

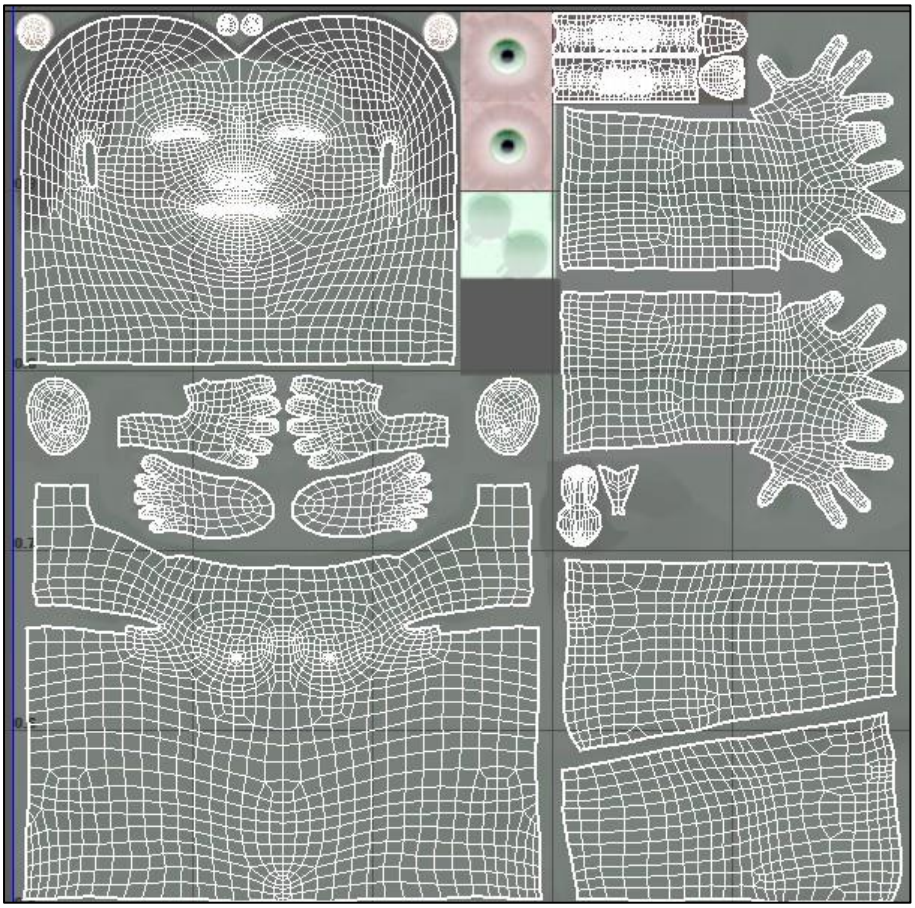

Рис. 4.5. UV розгортка тіла 3D моделі персонажа

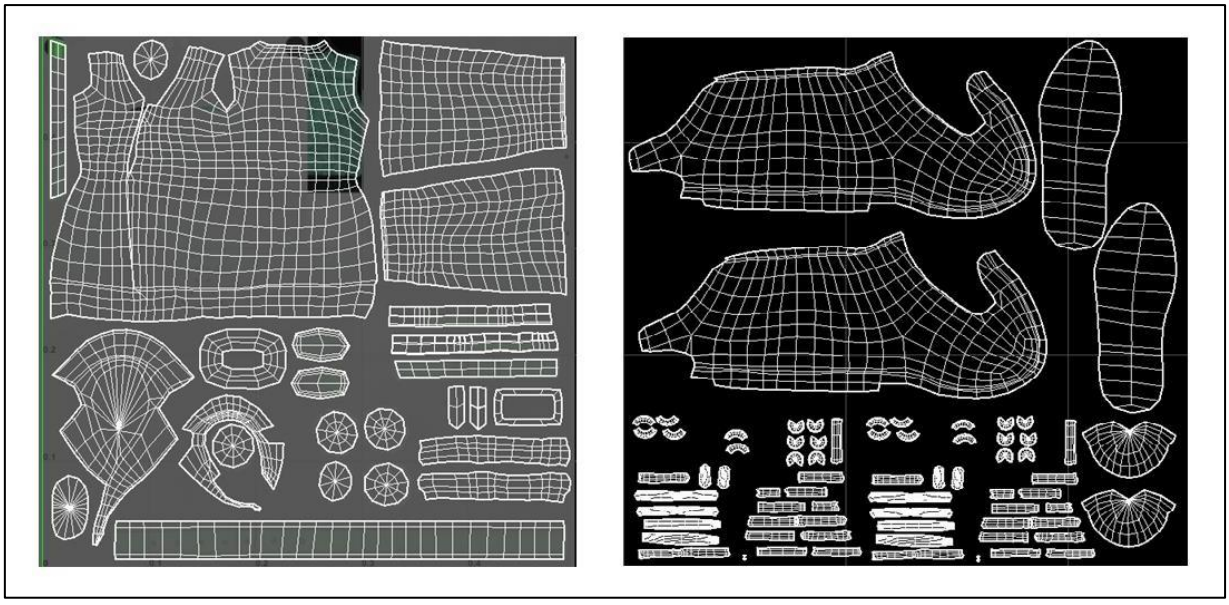

Рис. 4.6. UV розгортки одягу 3D моделі персонажа

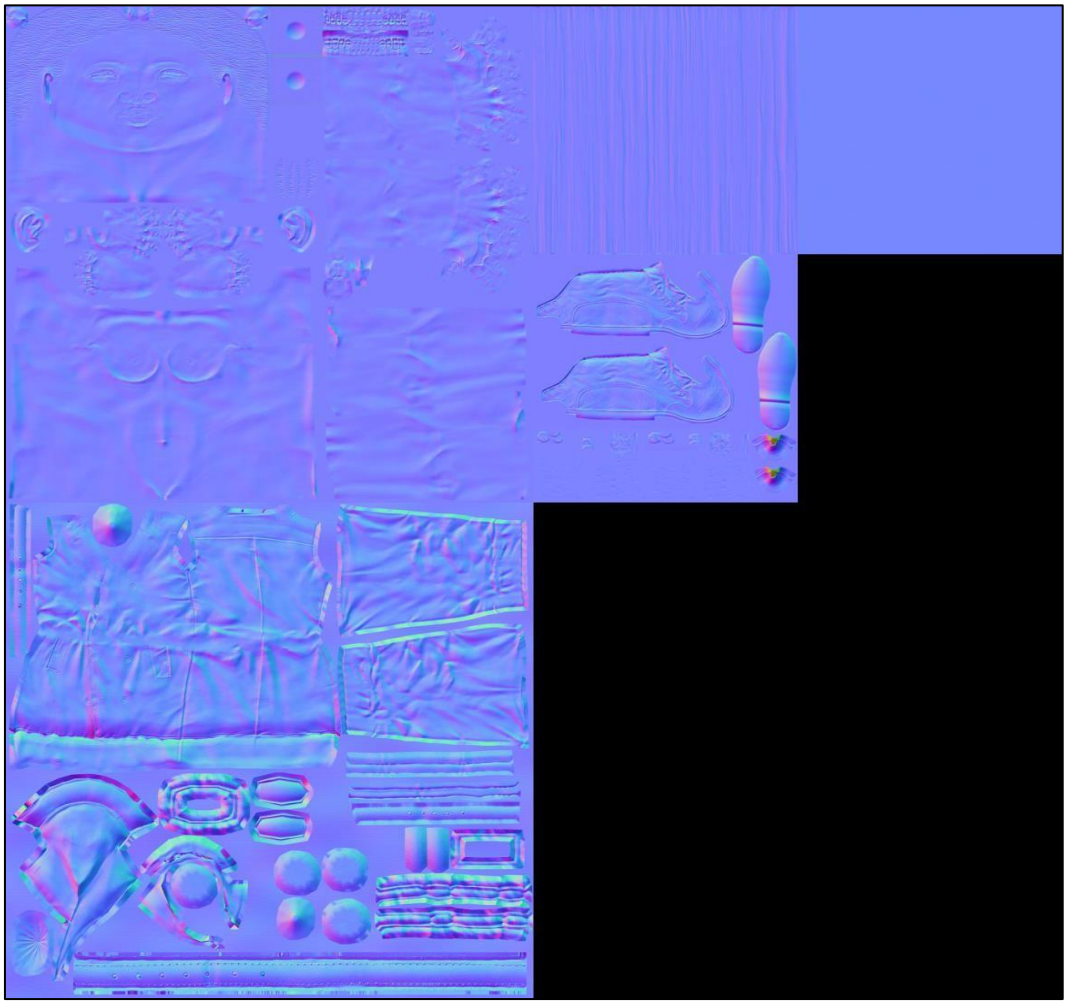

Рис. 4.7. Карта нормалей 3D моделі

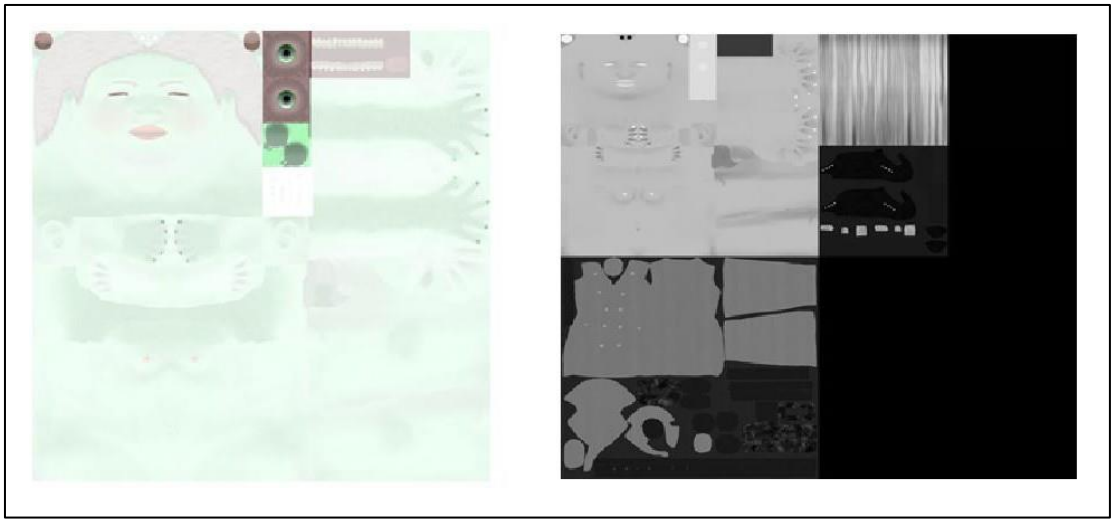

Рис. 4.8. Карта кольору(зліва) та карта блиску(зправа)

4.2.1. Шейдери

На рисунку 4.9 зображено структуру розробленого шейдеру ShaderFX. До створеної ноди текстурної карти(TextureMap) підключено ноду UV розгортки, через ноду MulOp, яка передає значення розгортки. Текстурна карта зв'язує атрибут кольору до атрибуту кольору дифузної карти шейдеру, а він в свою чергу підключає матеріал до поверхні моделі.

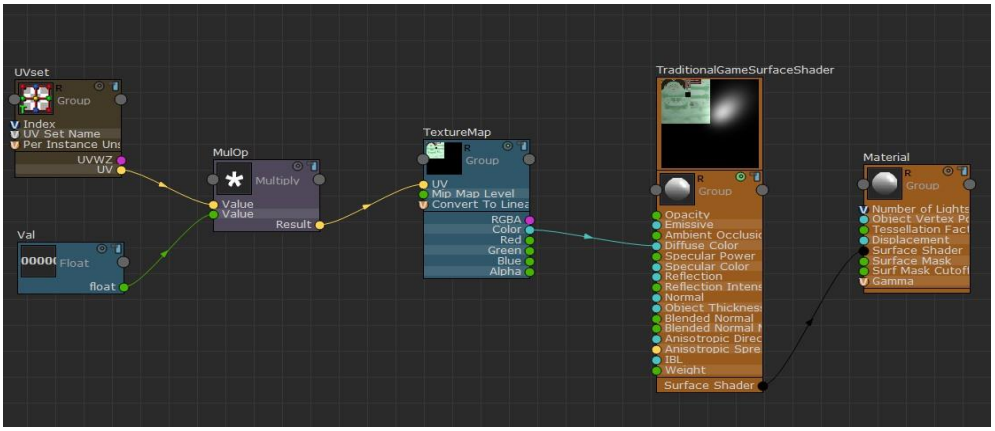

Рис. 4.9. Шейдер ShaderFX

На рисунку 4.10 зображений розроблений шейдер StandartSurface.

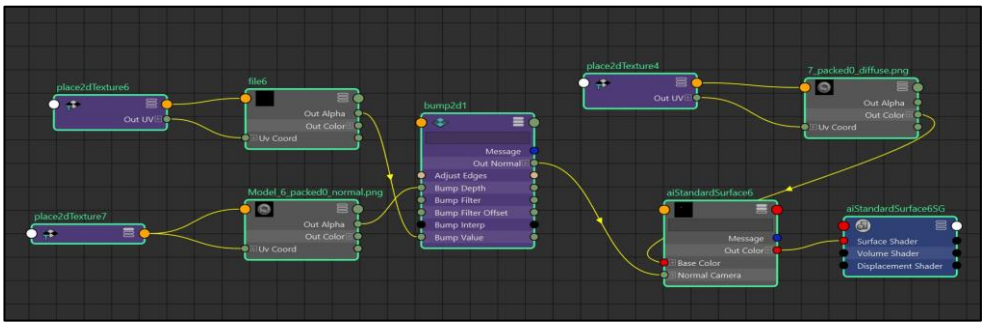

Рис. 4.10. Шейдер ShaderFX

Нода текстур (place2dTexture) включає ноду файлу з текстурною картою. Потім ці елементи підключаються до ноди шейдеру, зокрема до атрибутів кольору, які з'єднуються з поверхневим шейдером, та дають відображення, яке зображено на рисунку 4.11.

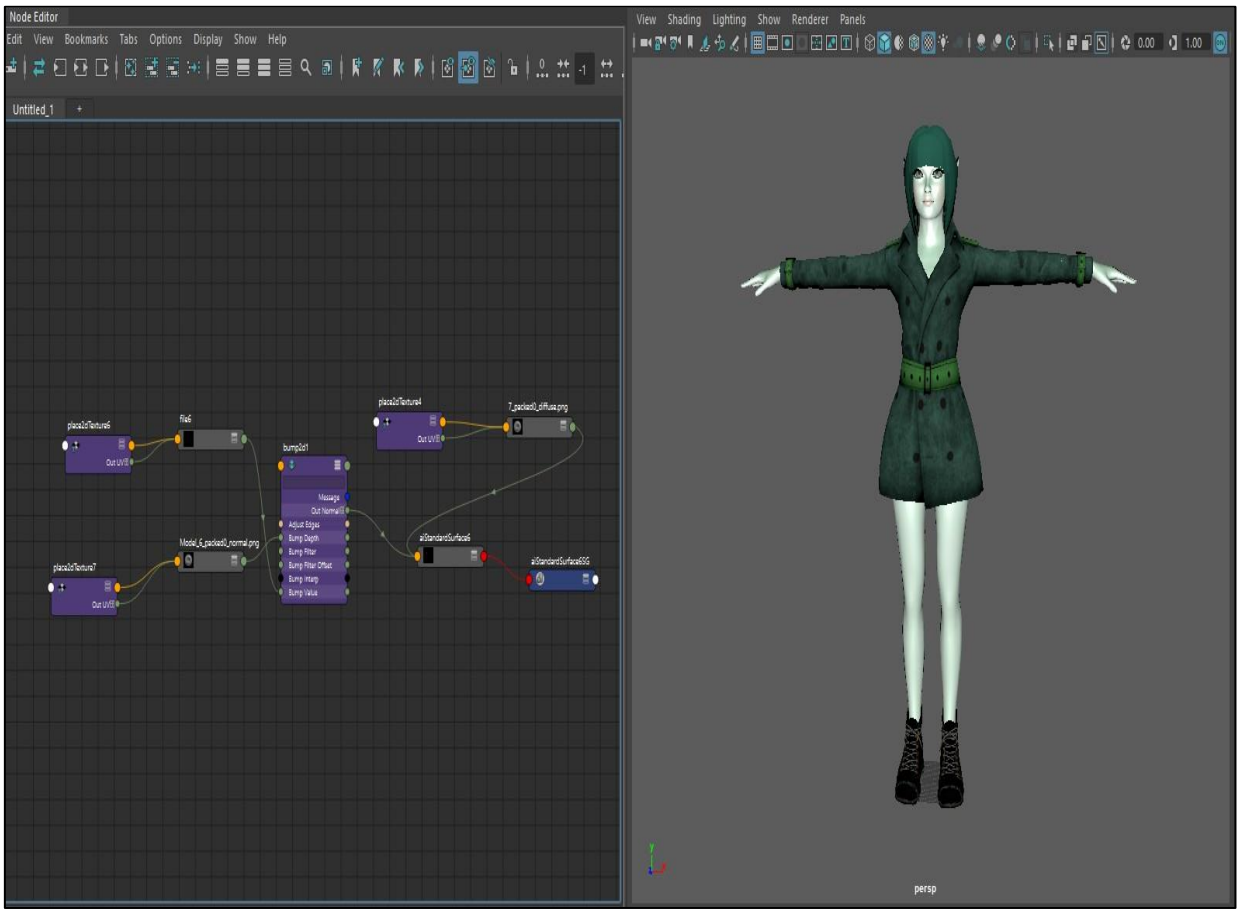

Рис. 4.11. Відображення шейдеру ShaderFX на моделі

Загальний вигляд персонажа починає набирати якоїсь людяності, і вже стає схожим на хорошого персонажа, але це тільки початок, далі ще є немаленька частка роботи.

4.3. Аналіз розроблених компонентів для анімації 3Dмоделі

4.3.1. Joints

Ієрархічна структура joints включає основний joint (Central_Joint) і наступні joints:

- Pelvis (таз) — joint, до якого прив'язані групи L_Femur та R_Femur.

- Ridgle та Ridgle_1 (хребет) — joints, від яких залежать групи Chest та Shoulders.

- Chest (грудна клітка).

- L_Femur (ліва нога).

- R_Femur (права нога).

- Shoulders (плечі) — включають групу Neck (шия), яка містить Head (голову). Також включають дві групи L_Shoulder та R_Shoulder, що містять групи кінцівок L_Hand та R_Hand, з підгрупами L_Finger(n) та R_Finger(n).

Структуру joints можна побачити на рисунку 4.13, зображеному на наступній сторінці, тим часом як її створений варіайнт зображено на рисунку 4.12.

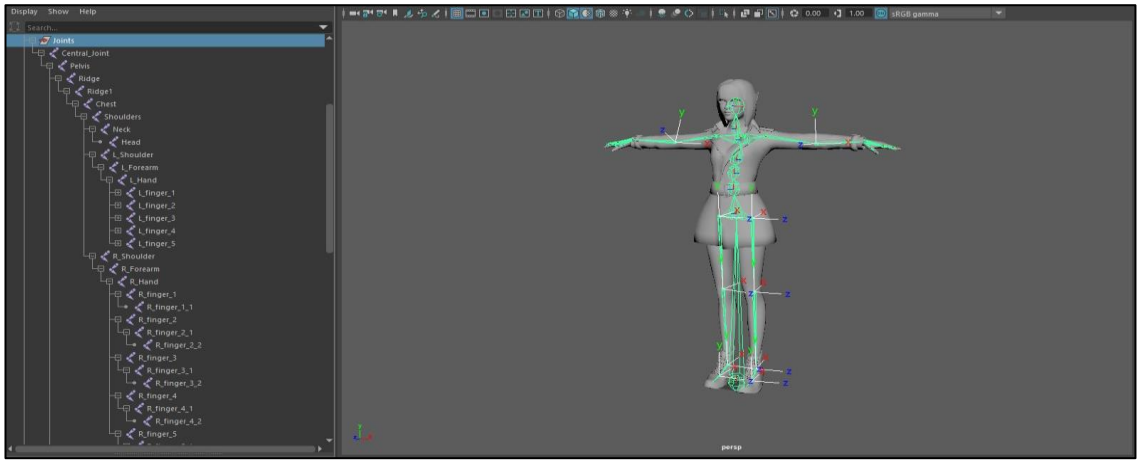

Рис. 4.12. Створені joints

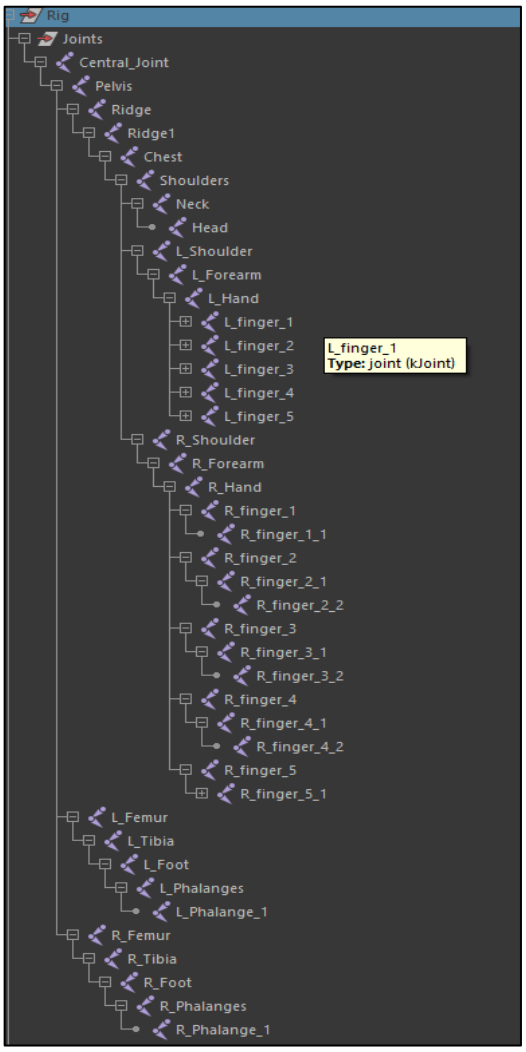

Рис. 4.12. Структура joints

4.3.2. Контролери

Створені відповідно до ієрархії joints. Варто зазначити, що кінцеві joints не мають контролерів, оскільки вони зазвичай залишаються нерухомими і їхня анімація не помітна.

Joints, на які не встановлюються контролери, показані на рисунку 4.14. Жовтими та зеленими кільцями позначено криві контролера, а кін

Створені відповідно до ієрархії joints. Варто зазначити, що кінцеві joints не мають контролерів, оскільки вони зазвичай залишаються нерухомими і їхня анімація не помітна.

Joints, на які не встановлюються контролери, показані на рисунку 4.14. Жовтими та зеленими кільцями позначені криві контролера, а крайні joints (кінцівки фалангів пальців) їх не мають.

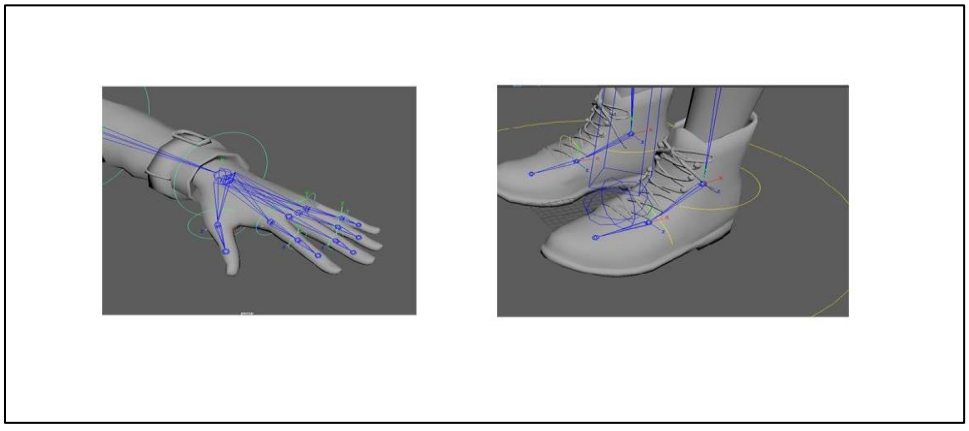

Рис. 4.14. Joints без контролерів

Так як вищеописані joints не мають контролерів в ієрархії контролерів, що зображено на наступній сторінці в Рис.у 4.15, вони відсутні. Але всі інші контролери ідентичні joints.

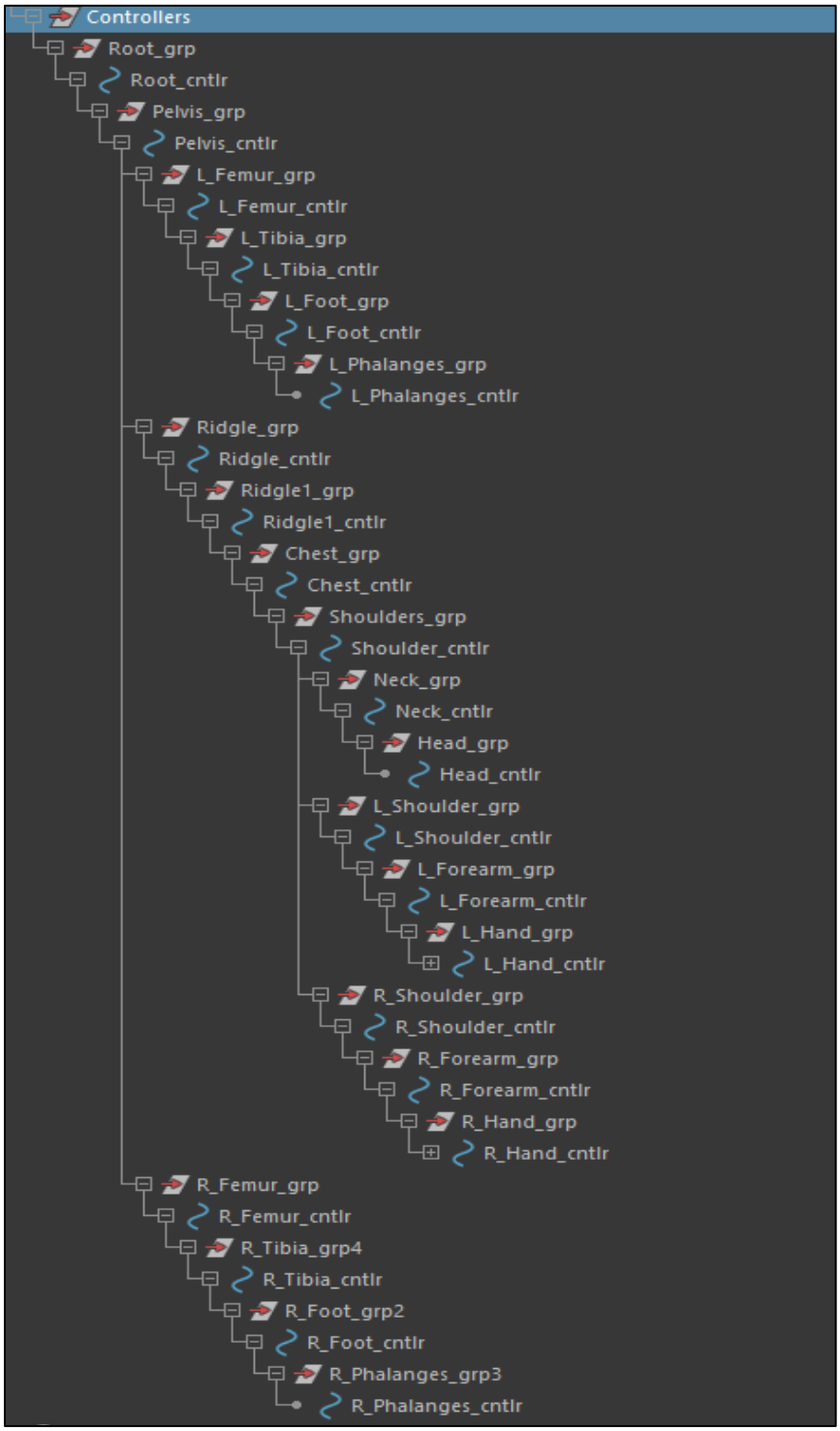

Рис. 4. 15. Структура контролерів

На рисунку 4.16 показані всі контролери та їхнє розташування відносно joints. Зверніть увагу на контролер, закріплений на Head joint (кільце над головою персонажа). Центр цього контролера зміщений вгору, щоб покращити анімацію рухів голови. Крім того, внизу моделі знаходиться корінний контролер (Root_cntlr), який забезпечує симетрію по центру моделі та виконує функцію «дзеркала».

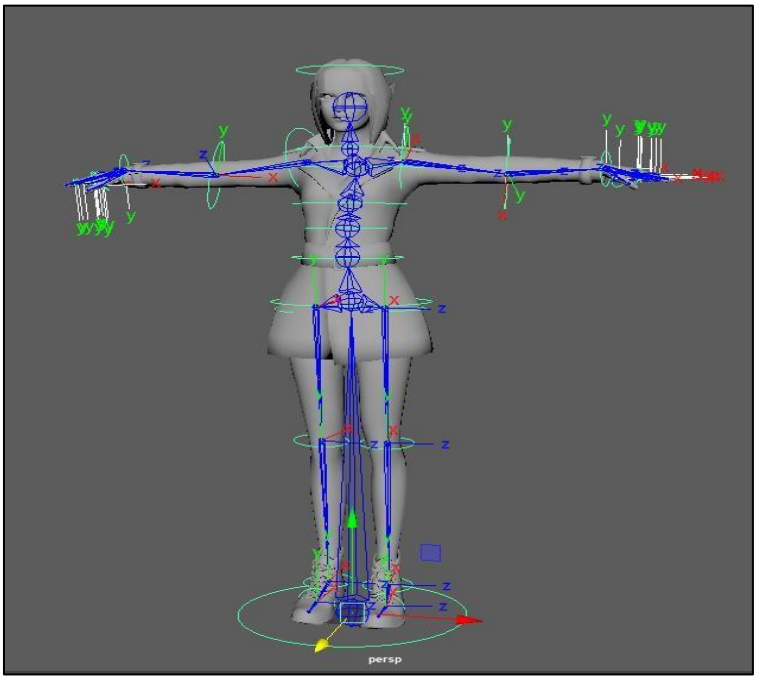

Рис. 4.16. Розташування контролерів на 3D моделі

4.3.3. Тестування контролерів

Тестування контролерів дозволяє оцінити ієрархію та поведінку контролерів. Контролери протестовані за наступними критеріями:

1. Критерій №1: При виборі групи контролерів та їх маніпулюванні зміни відбуваються усіма елементами групи.

2. Критерій №2: Вибравши дві однакові групи (наприклад, руки або ноги) та здійснивши маніпуляції над ними, вони зміщуються, збільшуються або повертаються симетрично.

3. Критерій №3: Під час вибору та маніпуляції Root контролера, виділяються всі контролери, і маніпуляція здійснюється відносно всіх контролерів.

Нижче наведені зображення, що ілюструють виконання критеріїв тестування контролерів.

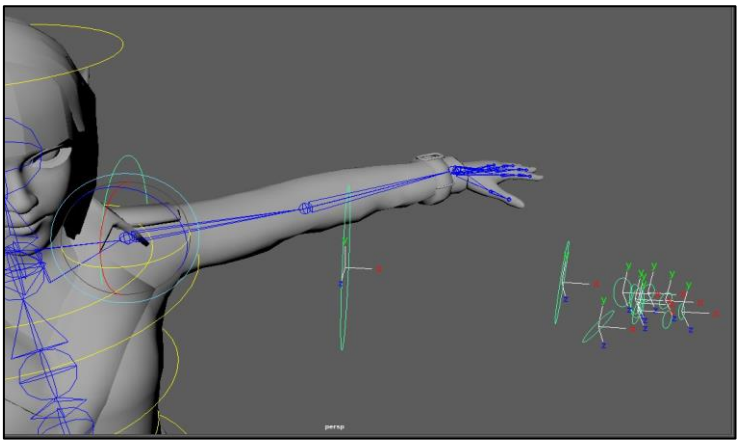

Рис. 4.17. Тестування конролерів по критерію №1

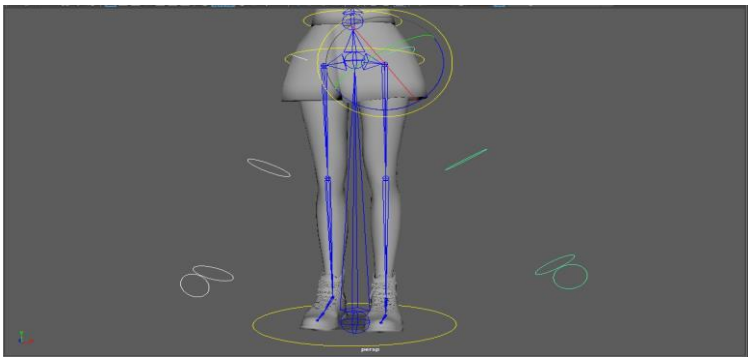

Рис. 4.18. Тестування конролерів по критерію №2

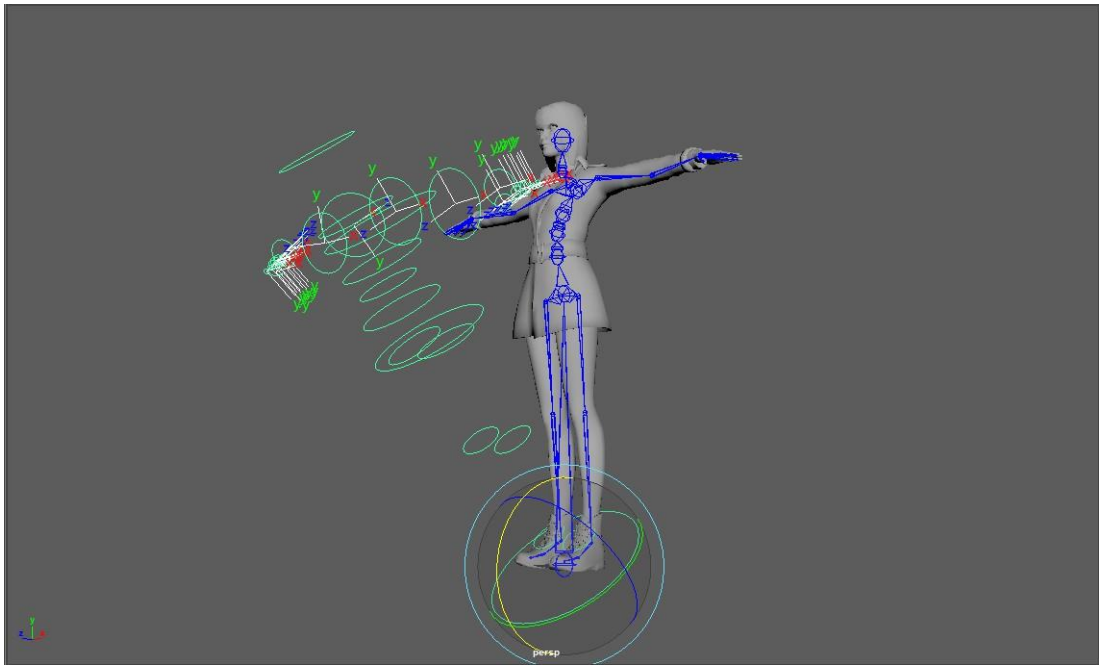

Рис. 4.19. Тестування конролерів по критерію №3

4.4. Тестування системи анімації, порівняння системи з аналогами

Так як відобразити анімаційний процес в ПЗ неможливо. Тому зроблено опис тестування реалізованої системи. Всі тести описані в таблиці на наступній сторінці (Таблиця 4.1) стосовно питання тестування компонентів моделі та артефактів, та також в таблиці 4.2 є інформація стосовно тестування анімацій та експорту моделей.

Також на рисунку продемонстровано експорт текстури в ігровий рушій (Рис. 4.20) для перевірки роботоспроможності текстури при застосуванні напряму в ігровому двигуні.

Таблиця 4.1

Тестування компонентів моделі та артефактів

Тестування коректної роботи всіх компонентів моделі персонажу під час анімації.	Полігональна сітка моделі не деформується, відображення тестур та кольору присутнє і не деформується. Всі текстурні карти правильно прикріплені до моделі.
Тестування деформації полігональної сітки на артефакти (Skinning)	Встановлено правильні коефіцієнти ваги(weights) для лівих груп joints(L_Soulder, L_Femur) . Віддзеркалення за допомогою команди copySkinWeights пройшло успішно. Відповідно деформації полігональної сітки не відбуваються.

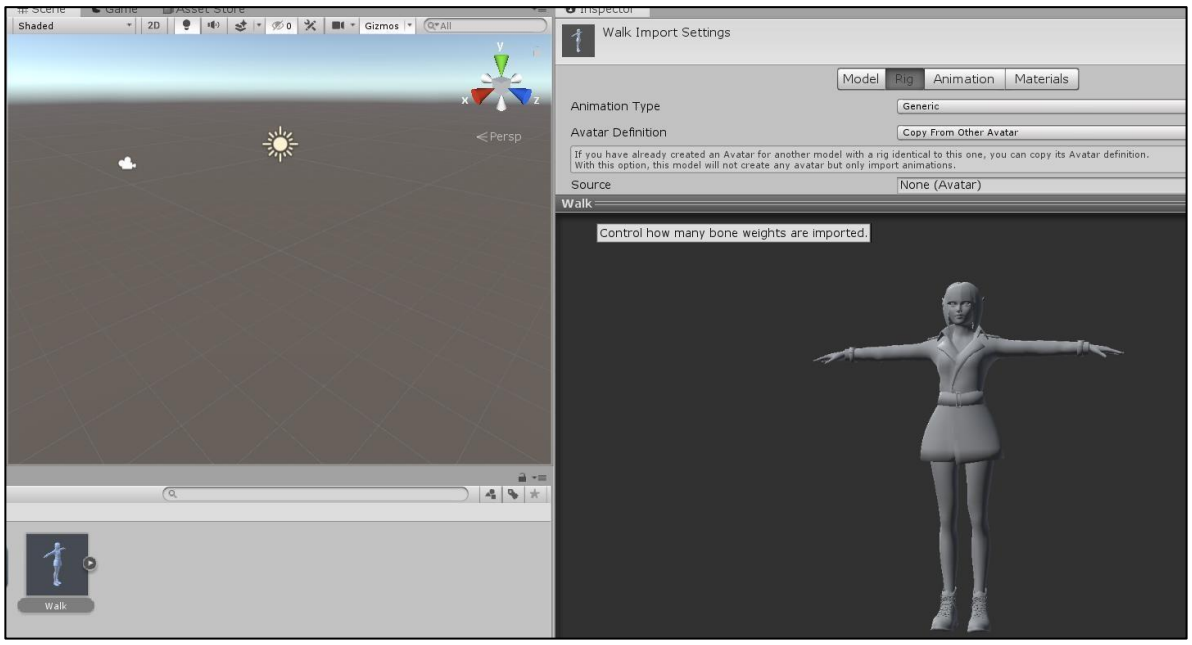

Рис. 4.20. Тестування моделі на експорт в ігровий рушій

Виконуючи практичну частину я хотів би зазначити, що порівнюючи реалізовану систему з аналогами, що описані в розділі 2, створена система має складнішу анімацію, але в той же час модель системи анімації не є так деталізована, як в аналогах.

4.5. Висновки четвертого розділу

У цьому розділі проведено глибокий аналіз створеної 3D моделі, включаючи всі її компоненти та розгорнуту систему анімації. Піддається увазі не лише технічна реалізація моделі, але й важливість її відтворення, деталізація, анімаційні можливості та зручність інтеграції з ігровими рушіями.

Ретельно розглядається кожен компонент 3D моделі, починаючи від її основних структурних елементів, таких як Joints та контролери, і закінчуючи найменш важливими деталями, які впливають на реалізацію анімаційних ефектів. Окрема увага приділяється процесу створення текстурних карт та їхньому використанню для досягнення бажаного ефекту візуалізації.

Аналізується продуктивність та ефективність реалізованої анімаційної системи порівняно з аналогами на ринку. Особлива увага приділяється функціональності системи, рівню контролю, стабільності та можливості швидкого інтегрування з іншими програмними рішеннями.

Завершуючи розділ, детально описуються результати тестування, включаючи роботу анімаційних ключів, точність відтворення рухів, а також процес експорту моделі в ігрові рушії з метою перевірки сумісності та продуктивності.

ВИСНОВКИ

Об'єктом цього дослідження є вивчення розробки 3D моделей та анімацій. Метою роботи є аналіз різних методів розробки 3D моделей та анімацій для них та їхнє вивчення, також розробка програмної реалізації стосовно анімацій 3D моделей, і демонстрація виконання.

У даній кваліфікаційній роботі розглянуто потребу у використанні 3D моделей, можливості використання, та вивчення галузі розробки моделей загалом. Проаналізовано різні методи розробки, та ресурси для кращого виконання роботи з 3D моделями і анімаціями для них.

Детально розглянуті етапи розробки моделей, їхня структура, склад, можливості застосування, та загальний вигляд під час виконання.

За результатами дослідження створено 3D модель на основі попередньо складеного плану роботи, з чітким виконанням заздалегідь поставлених правил та задач щодо виконання, також вивчена сфера розробки та створення моделей, можливості застосування, варіації розробки, та кінцевим результатом вивчення та застосування на практиці отриманих навичок. Розроблена візуальна частина моделі, її сітка, система текстурингу та шейдеризації моделі, і подальше скелетування та створення анімацій на основі попередньо створеної моделі, використовуючи програмні ресурси та мови програмування.

Таким чином, мета роботи, якою є розробка 3D моделі і анімацій для неї з застосуванням комп’ютерного коду, яка в подальшому може використовуватись в багатьох галузях починаючи з галузі розробки ігор, закінчуючи медичною галуззю чи галуззю маркетингу або дизайну, на мою думку успішно досягнута. Кінцевим результатом стала створена 3D модель та анімації для неї, які в суммі працюють коректно та відповідають заздалегідь поставленій задачі, результатом роботи задоволений.

СПИСОК БІБЛІОГРАФІЧНИХ ПОСИЛАНЬ ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Character Modeling with Maya and ZBrush: Professional Polygonal Modeling Techniques, Jason Patnode, 2008.
2. Mapping and UVs URL: https://www.spiria.com/en/blog/desktop- software/understanding-uv-mapping-and-textures/
3. 3D Art Essentials: The Fundamentals of 3D Modeling, Texturing, and Animation, Ami Chopine, 2011.
4. Shader materials URL: https://knowledge.autodesk.com/support/maya/learnexplore/caas/CloudHelp/cloudhelp/2016/ENU/3PP-MAYA-INTRO- Wiley/files/GUID-6468D54C-8D5B-40AD-9349-88F328E0BC2B-
5. MEL Scripting for Maya Animators, Mark R. Wilkins and Chris Kazmier, 2002.
6. Rython in Maya[Електронний ресурс URL: https://download.autodesk.com/us/maya/2011help/index.html?url=./files/Glossary_R_root_joint.htm,topicNumber=d0e207856
7. Rigging URL: https://3dpapa.com/human-body-rigging-in-maya/.
8. Joints URL: https://download.autodesk.com/us/maya/2011help/commandspython/join t.html
9. Controllers URL: https://inspireme.blog/create-auto-rig-controls-in-maya-with-free- python-script/
10. IKHandle URL: https://download.autodesk.com/us/maya/2011help/commandspython/ikH

11. Skinning URL: https://knowledge.autodesk.com/support/maya/learn-explore/caas/CloudHelp/cloudhelp/2018/ENU/Maya- CharacterAnimation/files/GUID-66922B35-A848-4161-BFCF- 4EB75413702E-htm.html

12. Autodesk Maya keyframes URL: https://help.autodesk.com/cloudhelp/2016/ENU/Maya- Tech-Docs/CommandsPython/keyframe.html

13. Eric G. Character animation crash course!. Beverly Hills, CA : Silman-James Press, 2008.

14. Williams R. E. Animator's Survival Kit. Faber & Faber, Incorporated, 2015.

15. Ahearn L. 3D Game Art f/x & Design. Coriolis Group Books, 2001. 408 p

Maestri G. Digital Character Animation 2, Volume II: Advanced Techniques. Waite Group Press, 2001. 240 p.