МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
НАЦІОНАЛЬНИЙ АВІАЦІЙНИЙ УНІВЕРСИТЕТ
ФАКУЛЬТЕТ КОМП'ЮТЕРНИХ НАУК ТА ТЕХНОЛОГІЙ
КАФЕДРА КОМП'ЮТЕРНИХ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ

ДОПУСТИТИ ДО ЗАХИСТУ
Завідувач випускової кафедри
_____Аліна САВЧЕНКО
«___»_____2023 р.

КВАЛІФІКАЦІЙНА РОБОТА

(ПОЯСНЮВАЛЬНА ЗАПИСКА)

ВИПУСКНИКА ОСВІТНЬОГО СТУПЕНЯ БАКАЛАВР

ЗА ОСВІТНЬО-ПРОФЕСІЙНОЮ ПРОГРАМОЮ

«ІНФОРМАЦІЙНІ ТЕХНОЛОГІЇ ПРОЕКТУВАННЯ»

Тема: «Технологія створення симуляцій динамічних 3D сцен із використанням вузлових систем»

Виконавець: Владислав МЕЛЬНИК

Керівник: к.т.н., доцент Костянтин ПРОКОПЕНКО

Нормоконтролер: к.т.н., доцент Олена ТОЛСТІКОВА

КИЇВ 2023

НАЦІОНАЛЬНИЙ АВІАЦІЙНИЙ УНІВЕРСИТЕТ

Факультет *комп'ютерних наук та технологій*

Кафедра *комп'ютерних інформаційних технологій*
Спеціальність *122 «Комп'ютерні науки»*
Освітньо-професійна програма *«Інформаційні технології проектування»*

ЗАТВЕРДЖУЮ:
завідувач кафедри КІТ
_____Аліна САВЧЕНКО
(підпис)
«____»_____2023 р.

ЗАВДАННЯ
на виконання кваліфікаційної роботи
Мельника Владислава Олександровича
(ПІБ випускника)

1. Тема роботи: «Технологія створення симуляцій динамічних 3D сцен із використанням вузлових систем» затверджена наказом ректора № 623/ст від 01.05.2023р.

2. Термін виконання роботи: з 15 травня 2023 року по 25 червня 2023 року.

3. Вихідні дані до роботи: модель 3D-візуалізації динамічної 3D симуляції з використанням програмному забезпечення Blender та Cinema 4D.

4. Зміст пояснювальної записки: 1. Огляд та аналіз предметної області. 2. Аналіз технологій створення симуляцій динамічних 3D сцен з використанням вузлових систем. 3. Реалізація розробленої технології на прикладі динамічної 3D сцени.

5. Перелік обов'язкового ілюстративного матеріалу: 1. Історія 3D технологій. 2. Основи 3D-симуляції з вуловими системами. 3. Вибір програмного забезпечення для створення симуляції. 4. Обрання рендер двигуна для рендерингу. 5. Створення першої частини 3D симуляції. 6. Створення динаміки 7. Текстуринг да додавання матеріалів. 8. Переніс симуляції до іншого програмного забезпечення для рендеру. 9. Налаштування рендеру. 10. Налаштування освітлення. 11. Рендеринг та вихідний результат.

6. Календарний план-графік

№ з/п	Завдання	Термін виконання	Підпис керівника
1.	Огляд та аналіз предметної області. Написання 1 розділу, представлення керівнику.	15.05.2023-20.05.2023	
2.	Вибір та опис використаних технологій. Написання 2 розділу, представлення керівнику.	21.05.2023-27.05.2023	
3.	Створення 3D симуляції. Написання 3 розділу, представлення керівнику.	28.05.2023-04.06.2023	
4.	Додавання текстур та створення рендеру тримірної моделі. Написання 4 розділу, представлення керівнику.	05.06.2023-10.06.2023	
5.	Загальне редагування та друк пояснювальної записки.	10.06.2023-12.06.2023	
6.	Проходження нормоконтролю, перепліт пояснювальної записки.	12.06.2023-15.06.2023	
7.	Розробка тексту доповіді. Оформлення графічного матеріалу для презентації.	16.06.2023-18.06.2023	

7. Дата видачі завдання 15.05.2023р.

Керівник кваліфікаційної роботи _____ Костянтин ПРОКОПЕНКО
(підпис керівника)

Завдання прийняв до виконання _____ Владислав МЕЛЬНИК
(підпис випускника)

РЕФЕРАТ

Пояснювальна записка до кваліфікаційної роботи на тему: «Технологія створення симуляцій динамічних 3D сцен із використанням вузлових систем» містить: 76 сторінок, 40 рисунків, 18 інформаційних джерел.

Об’єкт дослідження – 3D-модель симуляції.

Предмет дослідження – 3D-візуалізація динамічної симуляції.

Мета кваліфікаційної роботи – розробити 3D-модель динамічної симуляції із використанням найефективнішого програмного забезпечення.

Методи дослідження – логічний, синтезу, аналізу, порівняльний, обробка літературних джерел та моделювання.

Результати кваліфікаційної роботи рекомендується використовувати для загального ознайомлення зі створенням 3D симуляцій.

Для розробки 3D-візуалізації динамічної симуляції знайдено та використано найефективніше програмне забезпечення (Cinema 4D, Blender, а також плагін для рендерингу Redshift, та влаштований рендер двигун Blender, різноманітні тривимірні об’єкти з доступних джерел, готові матеріали та текстури.

КОМП’ЮТЕРНА ГРАФІКА, 3D-МОДЕЛЮВАННЯ, ВІЗУАЛІЗАЦІЯ, РЕНДЕРИНГ, ТЕКСТУРУВАННЯ, РЕКРЕАЦІЙНИЙ ЛАНДШАФТ, ТРИВИМІРНА МОДЕЛЬ, 3D STUDIO MAX, CORONA RENDER.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СКОРОЧЕНЬ, ТЕРМІНІВ

3D	–	Тривимірне
Cinema 4D	–	Cinema 4D
REDSHIFT	–	GPU-accelerated renderer
ІТ	–	Інформаційні технології
ОС	–	Операційна система
ПЗ	–	Програмне забезпечення
Програмний комплекс	–	Система взаємопов’язаних компонентів, заснована на програмному забезпечення, в складі комп’ютерної системи.

ВСТУП

3D вимір відноситься до тривимірного простору і використовується для опису об'єктів та сцен у тривимірних графічних системах. У технологічному контексті 3D вимір є важливим компонентом візуалізації, комп'ютерної графіки, а також у багатьох інших додатках, які працюють з тривимірними об'єктами.

3D вимір забезпечує тривимірну репрезентацію об'єктів у комп'ютерному середовищі. Він використовує три осі – X, Y та Z – для визначення положення, розміру та орієнтації об'єкта у просторі. Кожна точка в 3D просторі має свої координати на кожній з трьох осей, що дозволяє точно визначити її місцезнаходження.

У 3D вимірі об'єкти можуть мати форму, текстуру, колір та інші атрибути, що дозволяють їх реалістично відтворити. За допомогою тривимірних моделей можна створювати складні об'єкти, сцени та анімації. 3D вимір також використовується у віртуальній реальності, доповненій реальності та інших інтерактивних додатках.

Нодова або вузлова система є ще одним важливим аспектом 3D виміру. Вона використовується для організації та керування тривимірними об'єктами та їх взаємодією. Вузли представляють собою окремі елементи або компоненти, які виконують певні функції або обчислення.

У вузловій системі вузли зв'язуються між собою для передачі даних та керування. Кожен вузол може мати свої вхідні та вихідні з'єднання, що дозволяють обробляти дані та передавати їх між вузлами. Вузли можуть включати в себе операції обчислення, геометричні перетворення, текстурні ефекти, освітлення та інші функції.

Актуальність теми «Технологія створення симуляцій динамічних 3D сцен з використанням вузлових систем» є актуальною в сучасній комп'ютерній графіці. Вона надає розширені можливості візуалізації, моделювання фізики та створення реалістичних ефектів. Вузлові системи дозволяють гнучко керувати графічними об'єктами та забезпечують модульність та повторне використання. Ця технологія знайшла застосування у галузях, таких як графічний дизайн, відеоігри, виробництво фільмів та симуляція.

Об’єкт дослідження – 3D-модель симуляції.

Предмет дослідження – 3D-візуалізація динамічної симуляції.

Мета кваліфікаційної роботи – розробити 3D-сцену симуляції динамічної з використанням найефективнішого програмного забезпечення.

Відповідно до поставленої мети роботи визначено основні **завдання дослідження**:

– Аналіз існуючих вузлових систем;

– Розробити алгоритми та підходи для створення динамічних 3D сцен з використанням вузлових систем;

– Розробити та реалізувати різні симуляційні сцени з використанням вузлових систем;

– Розглянути можливі плагіни для рендерингу, вибрати найзручніший у використанні;

– Створити 3D візуалізацію динамічної симуляції.

Для досягнення поставленої мети й виконання завдань використано наступні методи: логічний, синтезу, аналізу, порівняльний, обробка літературних джерел та моделювання.

Наукова новизна технології створення симуляцій динамічних 3D сцен з використанням вузлових систем полягає в її гнучкості, модульності та ефективності. Ця технологія дозволяє розширити можливості візуалізації, моделювання фізики та створення реалістичних ефектів у тривимірних сценах. Шляхом збудови графічного пайплайну з вузлів, вона забезпечує легку зміну параметрів обробки та повторне використання коду. Вузлові системи є важливим компонентом сучасної комп'ютерної графіки та візуалізації, і вони знайшли застосування у галузях, таких як графічний дизайн, відеоігри, виробництво фільмів та симуляція.

Практичне значення отриманих результатів. Технологія створення симуляцій динамічних 3D сцен з використанням вузлових систем має практичне значення в індустрії відеоігор, виробництві фільмів, архітектурному візуалізації та симуляції для навчання. Вона дозволяє створювати реалістичні та захоплюючі візуальні ефекти, покращує якість графіки та сприяє більш точному моделюванню фізичних процесів.

РОЗДІЛ 1
ОГЛЯД ТА АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

1.1. Поняття симуляції динамічних 3D сцен

Симуляція динамічних сцен є процесом створення візуальної моделі з фізичними властивостями, такими як гравітація, тертя та стискання. Це дозволяє створювати візуально привабливі та реалістичні сцени, що можуть бути використані в різних галузях, таких як кіно, телебачення, реклама та відеоігри.

Одним з найпопулярніших програм для створення симуляцій динамічних сцен є Cinema 4D. Ця програма має вбудовані інструменти для створення різних ефектів, таких як руйнування, дим та вогонь. Крім того, вона має інтуїтивний інтерфейс та підтримує широкий спектр форматів файлів, що дозволяє легко інтегрувати її з іншими програмами (рис. 1.1).

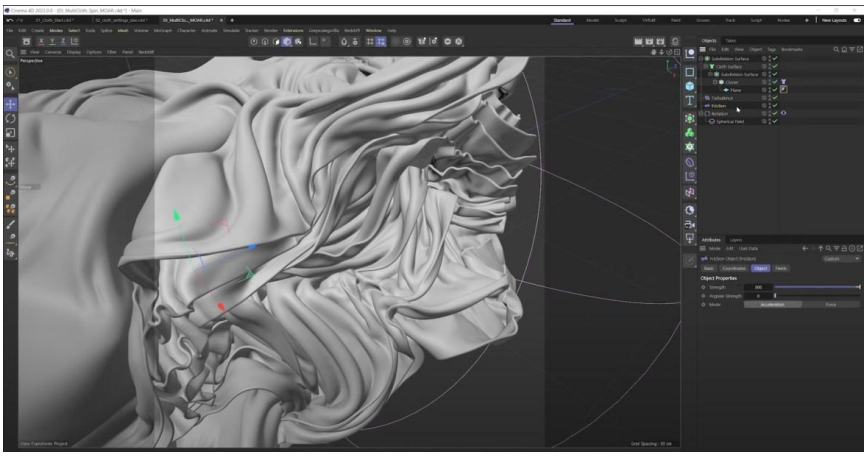

Рис. 1.1. Інтерфейс Cinema 4D

Houdini є іншою популярною програмою для створення симуляцій динамічних сцен. Ця програма дозволяє створювати складні рухи та ефекти, такі як вода, вибухи та руйнування будівель. Houdini також має велику

бібліотеку вбудованих інструментів та може бути легко інтегрована з іншими програмами (рис. 1.2).

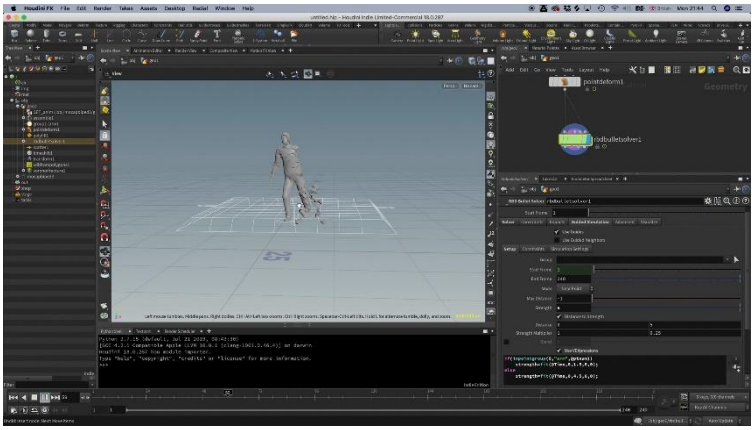

Рис. 1.2. Інтерфейс Houdini

Інші програми, які можна використовувати для створення симуляцій динамічних сцен, включають Autodesk Maya та Blender. Autodesk Maya є потужною програмою для створення 3D-графіки, яка має вбудовані інструменти для створення різних ефектів та симуляцій. Blender є безкоштовною та відкритою програмою, яка має багато функцій для створення 3D-графіки та симуляцій.

Загалом, створення симуляцій динамічних сцен є важливим етапом у створенні реалістичних та ефективних візуальних ефектів у різних галузях. Наприклад, в кіно та телебаченні симуляції динамічних сцен можуть бути використані для створення реалістичних ефектів вибухів, пожеж, повеней та інших природних катастроф. У відеоіграх симуляції динамічних сцен можуть бути використані для створення ефектів руйнування, диму, пилу, а також для відтворення фізичних властивостей предметів та об'єктів у грі.

Одним з прикладів використання симуляцій динамічних сцен є створення симуляції вибуху в програмі Cinema 4D (рис. 1.3). Це може бути виконано, наприклад, за допомогою плагіну TurbulenceFD, який дозволяє створювати візуальні ефекти диму та вогню. Користувач може налаштувати різні параметри, такі як розмір вибуху, швидкість диму та вогню, які можуть бути збережені у вигляді проекту та використані в подальшому.

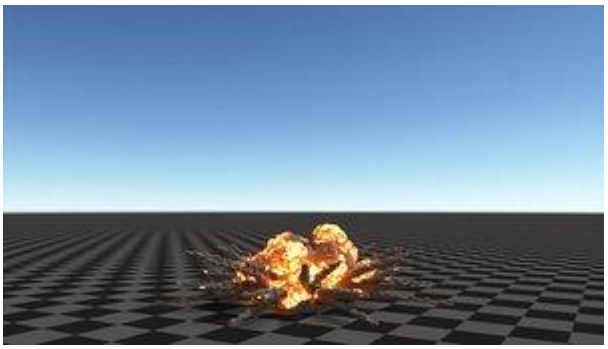

Рис. 1.3. Приклад симуляції вибуху в Cinema 4D

Ще одним прикладом використання симуляцій динамічних сцен є створення водяних ефектів у програмі Houdini (рис. 1.4). Це може бути зроблено за допомогою інструментів, таких як FLIP-симуляції, які дозволяють створювати реалістичні ефекти води, хвиль та піни. Користувач може налаштувати різні параметри, такі як густота води та швидкість хвиль, що дозволяє досягнути більш реалістичного ефекту.

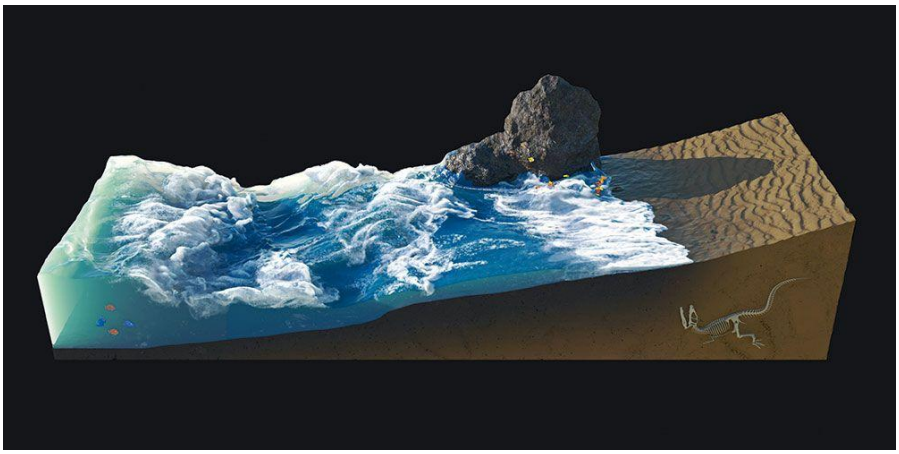

Рис. 1.4. Приклад симуляції води в Houdini

Загалом, симуляція динамічних сцен є важливим інструментом для створення візуальної інформації в різних галузях, що дозволяє користувачам створювати складні ефекти, які було б важко досягти за допомогою звичайних методів. Симуляції динамічних сцен дозволяють досягати різних ефектів, таких як руйнування, дим, вода, пожежа та багато іншого.

У програмі Cinema 4D, наприклад, можна створювати симуляції руйнування будівель та інших об'єктів (рис. 1.5). Для цього можна використовувати

плагіни, такі як Destruction, що дозволяє створювати руйнування за допомогою симуляцій твердих тіл. Користувач може налаштувати різні параметри, такі як розмір об'єкту, тип руйнування та інші, щоб отримати бажаний ефект.

Рис. 1.5. Симуляція руйнування будівель

У програмі Houdini симуляції динамічних сцен можуть бути використані для створення складних ефектів, таких як вода, пожежа та дим. Наприклад, FLIP-симуляції можуть бути використані для створення реалістичної води, а Pyro-симуляції можуть бути використані для створення реалістичної пожежі та диму. Ці симуляції дозволяють користувачам налаштувати різні параметри, такі як густота води та швидкість диму, щоб отримати більш реалістичний ефект.

Однією з важливих переваг симуляцій динамічних сцен є те, що вони дозволяють створювати складні ефекти, які можуть бути змінені та налаштовані у режимі реального часу. Це дозволяє користувачам швидко та ефективно налаштувати ефекти та досліджувати різні можливості.

У загальному, симуляції динамічних сцен є важливим інструментом

Крім того, симуляції динамічних сцен застосовуються у відеоіграх, щоб забезпечити реалістичний рух об'єктів у грі. Графічні движки, такі як Unreal Engine та Unity, містять потужні інструменти для створення динамічних сцен, які можуть бути використані для створення різних ефектів, таких як вибухи, водні хвилі, падіння предметів і т.д.

Для створення симуляцій динамічних сцен зазвичай використовуються спеціальні програмні засоби, такі як Cinema 4D та Houdini. Cinema 4D є

популярним пакетом для 3D-моделювання та анімації, який містить інструменти для створення динамічних сцен. Houdini - це програма для створення візуальних ефектів, що дозволяє створювати складні симуляції та анімації, використовуючи вузлові системи.

Застосування симуляцій динамічних сцен дозволяє створювати більш реалістичні та динамічні 3D-сцени для різних цілей, включаючи відеоігри, кіно, рекламні ролики, архітектурне візуалізації та ін. Із зростанням відомостей про комп'ютерну графіку та збільшенням обсягу доступної обчислювальної потужності, симуляції динамічних сцен стають ще більш популярними і потужними інструментами для творчих та комерційних проектів.

Симуляції динамічних сцен та візуальні ефекти, які вони створюють, застосовуються в різних сферах, включаючи кіно, телевізію, відеоігри, рекламу, медицину та ін. Ось декілька прикладів візуальних ефектів, які можуть бути створені за допомогою симуляцій динамічних сцен:

Ефекти погоди та води. Симуляції динамічних сцен можуть бути використані для створення реалістичних ефектів погоди, таких як дощ, сніг, блискавки та туман (рис. 1.6). Також можуть бути створені реалістичні водні ефекти, такі як хвилі, потоки та вироблення бульбашок.

Рис. 1.6. Ефект падіння снігу в Unreal Engine

Вибухи та пожежі. Симуляції динамічних сцен можуть бути використані для створення вражаючих ефектів вибухів та пожеж (рис. 1.7). За допомогою програмних засобів можна створити динамічну взривну хвилю, дим та вогонь.

Рис. 1.7. Симуляція вибуху в Houdini FX

Рух тіл. Симуляції динамічних сцен дозволяють створювати реалістичний рух тіл, таких як падаючі предмети, рух автомобілів та рух людей (рис. 1.8). Ці ефекти можуть бути використані в кіно, відеоіграх, рекламних роликах та ін.

Рис. 1.8. Симуляція частинок

Анімація персонажів. Симуляції динамічних сцен можуть бути використані для створення реалістичної анімації персонажів (рис. 1.9). Це може бути корисним для створення анімації для кіно та відеоігор.

Рис. 1.9. Анімація персонажів в Unreal Engine

Архітектурна візуалізація. Симуляції динамічних сцен можуть бути використані для створення візуалізації архітектури.

Візуальні ефекти симуляції динамічних сцен використовуються в різних сферах, включаючи кіно, відеоігри, архітектуру, науку та інші області (рис. 1.10).

Рис. 1.10. Рендер візуалізації архітектури в Unreal Engine

Наприклад, у кіно використовуються візуальні ефекти симуляції для створення складних сцен, які було б неможливо зняти у реальному житті. Наприклад, це може бути симуляція землетрусу або пожежі, які відображаються на екрані. Для створення таких ефектів використовуються програми для створення 3D сцен, такі як Cinema 4D або Houdini.

У відеоіграх симуляції динамічних сцен також широко використовуються для створення реалістичного середовища та ефектів. Наприклад, симуляція фізики руху об'єктів у грі може створити відчуття реалістичності та іммерсивності. Багато відомих відеоігор, таких як Grand Theft Auto або FIFA, використовують симуляції динамічних сцен для створення геймплею.

У архітектурі симуляції динамічних сцен використовуються для візуалізації проектів будівництва та дизайну. Це дозволяє архітекторам та клієнтам побачити, як виглядатиме будівля після завершення робіт, а також відчути, як відчуватимуть себе люди, що знаходяться в приміщенні.

У науці симуляції динамічних сцен використовуються для моделювання складних процесів та явищ. Наприклад, симуляції динаміки погоди можуть

допомогти вивчити та передбачити поведінку атмосферних явищ.

Іншою сферою, де використовуються візуальні ефекти симуляції, є відеоігри. У гральній індустрії динамічні сцени є надзвичайно важливими для створення реалістичного геймплею. Багато графічних движків, таких як Unity, Unreal Engine та CryEngine, мають вбудовані фізичні двигуни, які дозволяють створювати реалістичні динамічні сцени. З їх допомогою можна створювати різноманітні ефекти, такі як руйнування будівель, ефекти води, піску, диму, вогню та інших матеріалів, а також моделювати фізичну поведінку об'єктів в грі.

Крім відеоігор, симуляційні візуальні ефекти широко використовуються у кіноіндустрії. З їх допомогою можна створювати реалістичні візуальні ефекти, такі як вибухи, зіткнення, руйнування та інші події. Вони також допомагають створювати віртуальні світи, які дозволяють перенести глядача в інший світ, де він може досліджувати нові місця та дивуватися різноманітним красивим краєвидам.

Також симуляційні візуальні ефекти використовуються в архітектурі та механічному дизайні, де вони допомагають створювати 3D-моделі будівель та машин. Вони також можуть бути використані для тестування фізичних властивостей будівель та машин у віртуальному середовищі.

1.2. Технологія створення симуляцій динамічних 3D сцен. Еволюція технології та побудова 3D сцен.

Створення симуляцій динамічних 3D сцен використовує різні технології та програмні комплекси, які дозволяють моделювати реалістичні та інтерактивні віртуальні середовища. Одним з основних програмних комплексів для створення таких симуляцій є двигун (engine) для розробки ігор та симуляцій.

Двигун (engine) - це програмне забезпечення, яке надає розробникам інструменти та бібліотеки для створення візуальних ефектів, фізичної моделювання, обробки звуку, штучного інтелекту та інших функцій, необхідних для створення 3D симуляцій. Деякі з найпопулярніших двигунів

для створення симуляцій динамічних 3D сцен включають Unity, Unreal Engine, CryEngine та Godot.

Unity - це широко використовуваний двигун для розробки ігор та симуляцій, який надає потужний набір інструментів для створення 3D сцен, включаючи редактор сцен, фізичну моделювання, системи частинок та освітлення. Unity підтримує різні мови програмування, такі як C# та JavaScript, що дозволяє розробникам створювати складні інтерактивні симуляції.

Unreal Engine - це інший потужний двигун, який широко використовується для створення симуляцій динамічних 3D сцен. Unreal Engine володіє великим набором інструментів для створення візуально захоплюючих середовищ, фотореалістичної графіки та реалістичної фізики. Він також підтримує мови програмування, такі як C++ та Blueprint.

Еволюція 3D графіки є захопливим шляхом розвитку, який розпочався в 1960-х роках і досяг значних успіхів в різних галузях, включаючи фільми, відеоігри, архітектуру, медицину та багато іншого. Дозвольте мені детальніше розповісти вам про цю захоплюючу історію.

Початки 3D графіки можна відстежити до 1961 року, коли студент Массачусетського технологічного інституту (MIT) Іван Сазерленд (Ivan Sutherland) створив першу систему комп'ютерної графіки під назвою Sketchpad (рис. 1.11). Вона дозволяла користувачам створювати та маніпулювати геометричними об'єктами за допомогою екрана та вводу з комп'ютерної системи.

Рис. 1.11. Зображення Івана Сазерленда

У 1970-х роках графіка продовжувала розвиватися, і науковці активно

досліджували методи створення реалістичних зображень. Один із визначних проривів був зроблений Едом Кетмуллером (Edwin Catmull), який розробив алгоритми рендерингу поверхонь та вперше використав їх для створення тривимірних зображень. Це відкриття положило основу для подальшого розвитку 3D графіки.

У 1980-х роках комп'ютерна графіка стала доступнішою завдяки зростанню потужності комп'ютерів і розробці спеціалізованих графічних процесорів (рис.1.12). Цей період був визначальним для розвитку візуальних ефектів у фільмах. У 1982 році фільм "Tron" вперше використовував велику кількість комп'ютерної графіки, що створило враження, що герої перебувають у віртуальному світі. Цей фільм показав потенціал 3D графіки в кінематографії.

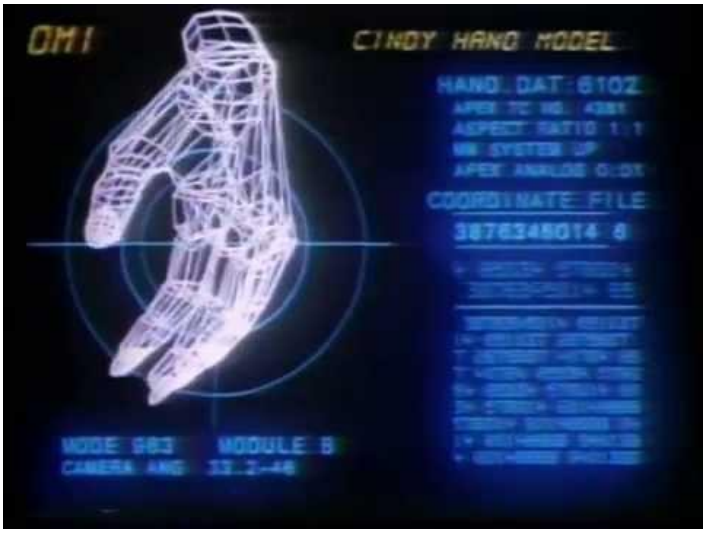

Рис. 1.12. Приклад комп`ютерної графіки 1980-х років

У 1990-х роках з'явилися перші повнометражні фільми, де багато сцен було створено з використанням комп'ютерної графіки. Наприклад, фільм "Jurassic Park" (1993) революціонізував використання 3D графіки в кіноіндустрії, представивши реалістично зображені динозаври. Це був перший випадок, коли цілі персонажі, створені на комп'ютері, виглядали настільки переконливо, що глядачі могли вірити у їхню реальність.

У 2000-х роках комп'ютерна графіка продовжувала розвиватися, з'явилися нові алгоритми та програмні засоби, що сприяли широкому

використанню 3D графіки у різних сферах (рис. 1.13). Відеоігри отримали значний приріст у візуальних можливостях, дозволяючи створювати деталізовані світи та реалістичних персонажів.

Рис. 1.13. Комп`ютерна графіка 2000-х років

Сучасна 3D графіка здатна створювати захоплюючі візуальні світи, що майже не відрізняються від реальності. Вона застосовується у великій кількості фільмів, відеоіграх, архітектурних проектах, медицині, наукових дослідженнях, рекламі та багатьох інших сферах. Завдяки постійному розвитку технологій 3D графіки, ми можемо спостерігати її вплив на наше сприйняття віртуального світу, розширюючи межі можливого та надаючи нові можливості для творчості.

Після успіху "Jurassic Park" використання 3D графіки у фільмах стало все поширенішим (рис. 1.14). Такі фільми, як "Terminator 2: Judgment Day" (1991) і "The Matrix" (1999), використовували інноваційні візуальні ефекти, які включали 3D моделі, розширену реальність та складну комп'ютерну анімацію.

Рис. 1.14. Комп`ютерна графіка в фільмі «Jurassic Park» 1993р.

З роками технології 3D графіки постійно удосконалювалися, а комп'ютерні візуальні ефекти стали стандартом для багатьох фільмів. Фільми серії "The Lord of the Rings" (2001-2003) та "Avatar" (2009) використовували революційні технології, щоб створити вражаючі світи та персонажів.

Сьогодні 3D графіка є неот'ємною частиною багатьох фільмів. Завдяки постійному розвитку технологій та програмного забезпечення, студії здатні створювати дивовижні візуальні ефекти, що розширюють можливості кіноіндустрії та надають нові шляхи для втілення творчих задумів режисерів.

Еволюція 3D графіки і її застосування у фільмах серії "Гаррі Поттер".

Перші фільми "Гаррі Поттер" (наприклад, "Філософський камінь", випущений у 2001 році) були зняті до того, як 3D графіка стала настільки популярною та доступною. Тому більшість візуальних ефектів у цих фільмах були створені за допомогою традиційних методів, таких як анімація, лялькова анімація, хромакей та модельна робота.

Проте, по мірі розвитку технологій 3D графіки, її використання у фільмах "Гаррі Поттер" стало більш широко поширеним. Зокрема, з третім фільмом "Гаррі Поттер і в'язень Азкабану" (2004), студія Warner Bros. співпрацювала з візуально-ефектною компанією Moving Picture Company (MPC), яка займалася створенням багатьох ефектів у фільмі. Вони використовували комп'ютерну графіку для створення складних сцен, таких як політ на мітлі чи боротьба з дементорами.

У наступних фільмах "Гаррі Поттер" (наприклад, "Орден Фенікса", "Половина крові", "Смертельні Реліквії") використання комп'ютерної графіки стало ще більш розповсюдженим. MPC продовжувала брати участь у створенні вражаючих візуальних ефектів, таких як зображення великих битв, створення магічних створінь та екстраординарних подій.

Завдяки розвитку технологій 3D графіки, студії та візуально-ефектні компанії отримали нові можливості для створення реалістичних та захоплюючих візуальних ефектів. Вони могли деталізувати і анімувати складні моделі персонажів та створювати грандіозні світи, які раніше були неоскільки реалістичними для досягнення.

Таким чином, фільми серії "Гаррі Поттер" є прикладом того, як 3D графіка стала важливою складовою для створення вражаючих візуальних ефектів у кіноіндустрії. З появою нових технологій, фільми стали більш спектакулярними, захоплюючими та реалістичними, завдяки чому глядачі могли ще більше зануритись у магічний світ "Гаррі Поттера".

1.3. Вузлові системи та їх застосування в симуляціях

Нодові або вузлові системи в 3D графіці є потужним інструментом для створення складних візуальних ефектів, симуляцій та спеціальних ефектів. Вони базуються на концепції вузлів, які представляють функціональні блоки, пов'язані між собою, що обробляють та трансформують дані.

У вузлових системах в 3D графіці кожен вузол відповідає за певну обробку або функціональність. Вони можуть включати операції математичних обчислень, текстурні ефекти, модифікацію форми об'єктів, анімацію, світло та багато іншого. Вузли з'єднуються між собою у визначений порядок, утворюючи граф, який описує процеси обробки даних.

Вузлові системи широко використовуються у 3D сфері для різноманітних завдань, включаючи:

Створення матеріалів: Вузлові системи дозволяють створювати складні матеріали, такі як текстури, блиск, прозорість та інші властивості. Вони дають можливість контролювати вигляд об'єктів та їх поведінку у різних умовах освітлення та текстурування.

Анімація та рух: Вузлові системи можуть використовуватись для створення складних анімацій та руху об'єктів. Вони дозволяють задавати траєкторії руху, контролювати швидкість, зміну розміру, повороти та інші параметри анімації.

Симуляція фізики: Вузлові системи використовуються для створення фізично вірних симуляцій, таких як симуляція рідини, твердих тіл, тканин та інших фізичних процесів (рис. 1.15). Вони дозволяють моделювати реалістичні фізичні властивості та взаємодію об'єктів у віртуальному середовищі.

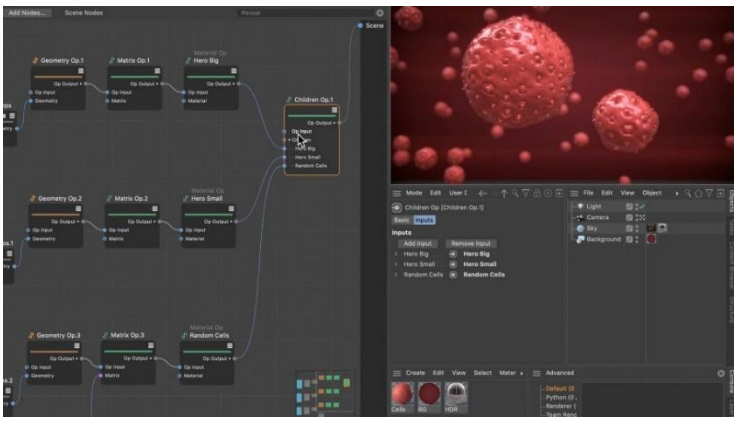

Рис. 1.15. Приклад нодової системи в Cinema 4D

Спеціальні ефекти: Вузлові системи дозволяють створювати різноманітні спеціальні ефекти, такі як вогонь, дим, вибухи, частинки, магічні ефекти та інші. Вони дають можливість контролювати властивості та вигляд спеціальних ефектів для створення вражаючих сцен.

Вузлові системи забезпечують велику гнучкість та контроль над візуальними ефектами та симуляціями у 3D графіці. Вони дозволяють артистам та візуальним ефектним спеціалістам ефективно створювати складні та реалістичні сцени, використовуючи широкий спектр параметрів та опцій, щоб досягти бажаного результату.

Вузлова система в 3D графіці будується на основі графа, де вузли представляють функціональні блоки, а ребра визначають зв'язки між вузлами (рис. 1.16). Кожен вузол має вхідні та вихідні порти, через які відбувається передача даних між вузлами. Програмний код, який виконується в кожному вузлі, визначає операції та обробки, які потрібно здійснити з даними.

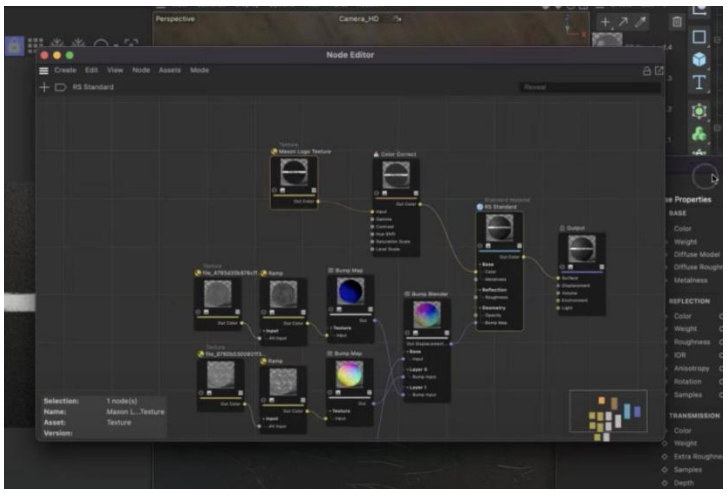

Рис. 1.16. Приклад графів в Cinema 4D

Основний принцип вузлової системи полягає в тому, що ви можете створювати складні візуальні ефекти та обробки, комбінуючи різні вузли та з'єднуючи їх між собою. Наприклад, вузол для генерації текстури може бути підключений до вузла для зміни кольору, а потім до вузла для накладання текстури на об'єкт. Таким чином, ви можете створити послідовність операцій, що обробляють та модифікують дані, щоб досягти бажаного візуального ефекту.

Для реалізації вузлової системи у програмному коді можна використовувати спеціалізовані бібліотеки або фреймворки, які надають готові компоненти для побудови та керування вузловими системами. Наприклад, у багатьох програмах для створення 3D графіки, таких як Blender, Unity або Unreal Engine, існують вбудовані інструменти для вузлових систем.

Програмний код вузлової системи може бути написаний мовами програмування, які підтримують роботу з графами та операціями з даними. Зазвичай вузли представляють собою класи або функції, які виконують певні обчислення або маніпуляції з даними. З'єднання між вузлами може бути реалізовано за допомогою звичайних змінних, об'єктів або спеціальних структур даних, які забезпечують передачу даних між вузлами.

Використання вузлових систем у створенні 3D графіки допомагає спростити процес розробки та контролювати складність обробки великих обсягів даних. Вона дає змогу артистам та розробникам ефективно експериментувати з візуальними ефектами, вносити зміни та налаштовувати їх у реальному часі. Вузлові системи також дозволяють перевикористовувати компоненти та швидко змінювати конфігурацію ефектів, що збільшує продуктивність та творчий потенціал розробників.

ВИСНОВКИ ДО РОЗДІЛУ 1

У даному розділі було розглянуто поняття симуляції динамічних 3D сцен із використанням вузлових систем. Була проаналізована еволюція технологій у створенні симуляцій та побудові 3D сцен, що призвело до використання вузлових систем як потужного інструменту для створення реалістичних і вражаючих візуальних ефектів.

Технологія створення симуляцій динамічних 3D сцен з використанням вузлових систем має велике значення в різних галузях, таких як відеоігри, виробництво фільмів, архітектурна візуалізація та навчання. Вузлові системи дозволяють створювати складні сцени з реалістичними ефектами, деталізацією та фізичною моделлю об'єктів. Вони забезпечують гнучкість, модульність та ефективність у розробці та настройці сцен, що дає можливість швидкої зміни параметрів та повторного використання коду.

Застосування вузлових систем в симуляціях динамічних 3D сцен відкриває широкі перспективи для поліпшення візуалізації, моделювання та створення реалістичних ефектів. Отримані результати досліджень дають підстави для подальшого розвитку цієї технології та застосування її у різних сферах, що сприятиме покращенню віртуального досвіду користувачів та розвитку сучасної комп'ютерної графіки.

РОЗДІЛ 2
АНАЛІЗ ТЕХНОЛОГІЙ СТВОРЕННЯ СИМУЛЯЦІЙ ДИНАМІЧНИХ 3D СЦЕН З ВИКОРИСТАННЯМ ВУЗЛОВИХ СИСТЕМ

2.1. Визначення потреби у вузлових системах

Як дизайнеру, вузлові системи в 3D графіці можуть бути надзвичайно корисними інструментами для досягнення бажаного візуального результату.

Нижче наведено декілька способів, які вузлові системи можуть задовольнити твої потреби у процесі дизайну.

Гнучкість та контроль: Вузлові системи надають велику гнучкість та контроль над візуальними ефектами. Вони дозволяють тобі змінювати параметри та налаштовувати вузли, щоб досягти бажаного результату. Ти можеш експериментувати з різними комбінаціями вузлів, їх порядком та параметрами, щоб створити унікальні та вражаючі візуальні ефекти.

Швидкість та ефективність: Вузлові системи дозволяють тобі швидко програвати, змінювати та налаштовувати ефекти у реальному часі. Це дозволяє тобі бачити миттєві результати твоїх змін та швидко вносити корективи. Завдяки цьому процес творчої роботи стає більш ітеративним та продуктивним.

Перевикористання компонентів: Вузлові системи дозволяють тобі перевикористовувати та зберігати свої створені компоненти. Ти можеш створити складні ефекти або обробки і зберегти їх як шаблони, які можна використовувати у майбутніх проектах. Це зекономить твій час та дозволить швидко створювати нові ефекти на основі вже перевірених та оптимізованих компонентів.

Візуалізація процесу: Вузлові системи надають візуальний спосіб представлення процесу обробки даних. Ти можеш бачити зв'язки між

вузлами, потік даних та зміни параметрів у реальному часі. Це допомагає тобі краще розуміти, як працюють твої ефекти та як впливають на кінцевий результат.

В цілому, вузлові системи в 3D графіці дозволяють дизайнерам експериментувати, контролювати та створювати різноманітні візуальні ефекти з великою гнучкістю та швидкістю. Вони спрощують процес творчої роботи, допомагають досягти бажаних результатів та відкривають нові можливості для виразності та інновацій у дизайні (рис. 2.1).

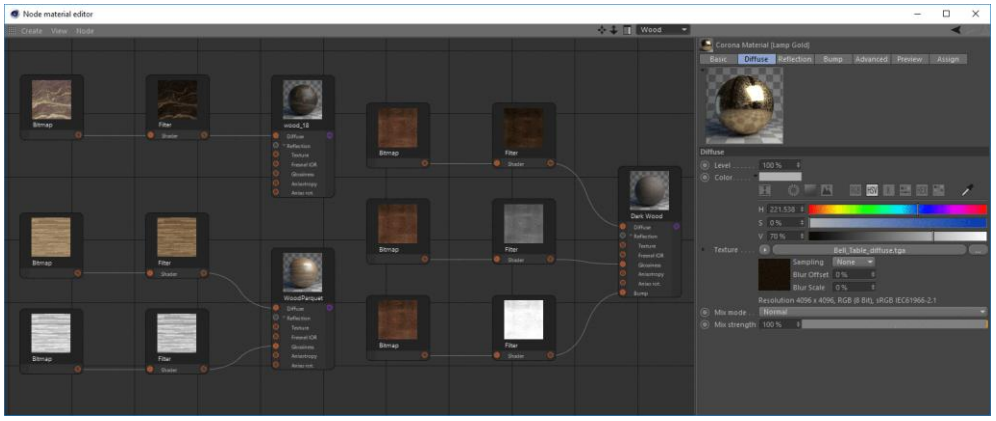

Рис. 2.1. Створення матеріалів за допомогою нодової системи в Cinema 4D

Вузлові системи в 3D графіці можуть бути взаємопов'язані з програмним кодом у різних аспектах. Ось кілька способів, як вони можуть бути пов'язані:

Реалізація вузлових систем: Самі вузлові системи можуть бути побудовані за допомогою програмного коду. Наприклад, вузли можуть бути реалізовані у вигляді класів або функцій, які виконують певні обчислення та маніпуляції з даними. Ти можеш створювати власні вузли або використовувати готові компоненти, що надаються фреймворками чи бібліотеками.

Взаємодія з кодом: Вузлові системи можуть дозволяти взаємодіяти з програмним кодом. Це означає, що ти можеш редагувати або налаштовувати програмний код вузлів, щоб досягти певних ефектів або змінити їх поведінку. Наприклад, ти можеш змінювати параметри вузлів, виконувати операції на основі вхідних даних або створювати власні функції для спеціальних обробок.

Редагування програмного коду: Як дизайнер, ти можеш мати

можливість редагувати програмний код, пов'язаний з вузловою системою, для досягнення бажаного результату (рис. 2.2). Це може включати зміну параметрів вузлів, модифікацію логіки або додавання власних функцій та операцій. Таким чином, ти маєш можливість налаштувати вузлову систему під свої потреби та створити унікальні візуальні ефекти.

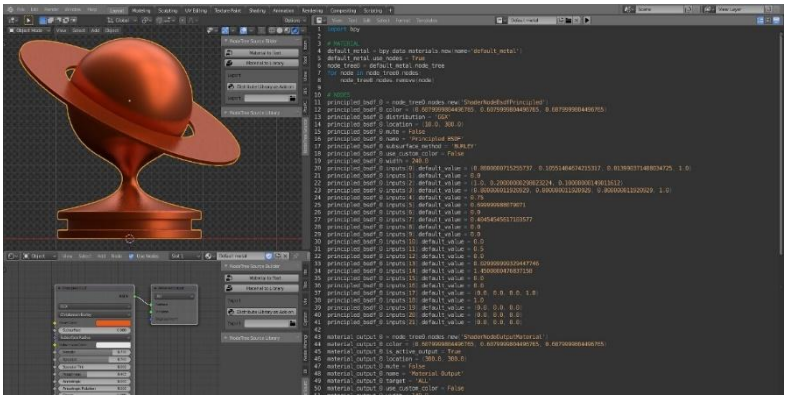

Рис. 2.2. Приклад створення програмного коду

Важливо враховувати, що редагування програмного коду вузлової системи може вимагати розуміння програмування та специфіки використовуваного фреймворку чи редактора. Якщо ти не маєш достатнього досвіду в програмуванні, співпраця з програмістами може бути корисною, щоб отримати необхідну допомогу та експертні поради при редагуванні коду.

Загалом, вузлові системи в 3D графіці можуть забезпечувати можливість редагування програмного коду, що дозволяє дизайнерам налаштовувати та адаптувати ефекти до своїх потреб. Це дозволяє створювати унікальні та індивідуальні візуальні рішення, а також розширює можливості дизайнера у створенні 3D графіки.

3D софт і вузлові системи можуть бути написані з використанням різних мов програмування, залежно від конкретного програмного забезпечення. Ось кілька прикладів популярних мов програмування, які використовуються у розробці 3D софту та редагуванні програмного коду вузлових систем:

1) C++: Це одна з найпоширеніших мов програмування, яка широко використовується у розробці 3D софту. Вузлові системи та редактори, що підтримують вузловий підхід, можуть бути реалізовані з

використанням C++.

2) Python є популярною мовою програмування, яка знаходить широке застосування у розробці 3D софту. Багато фреймворків та бібліотек, які використовуються для вузлових систем, підтримують Python як мову розширення або скриптінгу.
3) JavaScript є мовою програмування, яка широко використовується для розробки веб-додатків та інтерактивної 3D графіки. Вузлові системи, що працюють у веб-середовищі, можуть використовувати JavaScript для реалізації вузлів та взаємодії з ними.
4) Shader-мови: У сфері 3D графіки широко використовуються спеціальні мови програмування для створення шейдерів, які контролюють візуальний вигляд графічних об'єктів. Прикладами таких мов є GLSL (OpenGL Shading Language) та HLSL (High-Level Shading Language), які використовуються в різних програмних середовищах.

Процес редагування програмного коду вузлової системи може відрізнятися залежно від конкретного софту та його реалізації. Однак, загальний підхід може включати наступні кроки:

- Відкриття вузлової системи: Відкриємо програму або редактор, який має вузлову систему для редагування;
- Навігація до вузлів: Знайдемо конкретні вузли, які бажаєш редагувати. Це можуть бути вузли для обчислень, модифікації параметрів, зміни поведінки тощо;
- Редагування параметрів: Змінюємо параметри вузлів, які впливають на їхню роботу та результат. Це може включати зміну числових значень, текстових параметрів, підключення/відключення з'єднань між вузлами тощо;
- Модифікація логіки: У деяких випадках, можливо, знадобиться змінити саму логіку роботи вузла. Це може включати додавання нових операцій, функцій, умовних виразів та інших елементів програмного коду для досягнення бажаного результату;
- Перевірка результату: Після внесення змін у програмний код вузлової

системи, перевіряйте зміни в реальному часі або після виконання симуляції/рендерингу. Це допоможе переконатися, що внесені зміни працюють належним чином та дають бажаний результат.

Важливо мати розуміння програмного коду вузлової системи та можливості, які надаються редактором чи програмним забезпеченням. Також, бажано мати певний досвід у програмуванні, щоб ефективно редагувати програмний код і досягати бажаних змін у вузловій системі.

2.2. Переваги використання вузлових систем для симуляцій динамічних 3D сцен

Вузлові системи є потужним інструментом для створення симуляцій динамічних 3D сцен, що має безліч переваг.

1) Гнучкість: Вузлові системи дозволяють користувачам розробляти складні симуляції, використовуючи гнучкий графічний інтерфейс. Завдяки цьому користувачі можуть швидко змінювати параметри симуляції та ефекти.
2) Легкість використання: Вузлові системи дозволяють користувачам зосередитися на розробці ідеї, а не на кодуванні. Користувачі можуть легко встановлювати та з'єднувати вузли, що дозволяє значно скоротити час розробки.
3) Масштабованість: Вузлові системи дозволяють легко масштабувати симуляцію, додавати нові функції та змінювати налаштування без необхідності змінювати код симуляції.
4) Візуалізація: Вузлові системи зазвичай мають вбудовані візуалізатори, що дозволяють користувачам побачити результати симуляції у режимі реального часу (рис. 2.3). Це дозволяє користувачам швидко реагувати на зміни та вдосконалювати симуляцію.

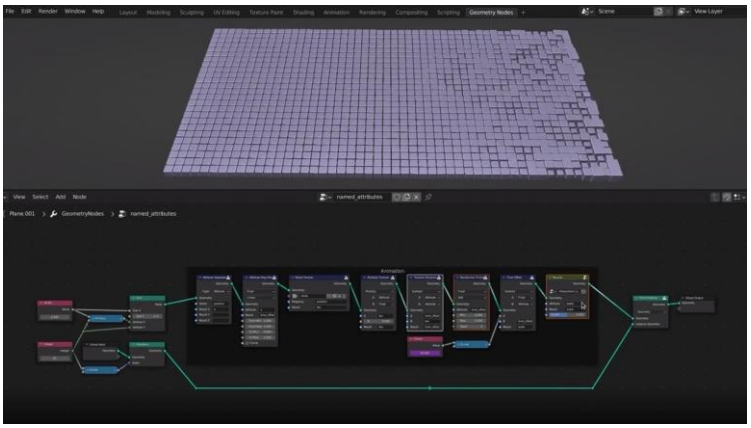

Рис. 2.3. Приклад відображення симуляції в реальному часі

5) Підтримка плагінів: Вузлові системи зазвичай мають велику кількість плагінів, що дозволяють розширити функціональність системи та додати нові можливості.

Одним з найпопулярніших програм для створення симуляцій динамічних 3D сцен з використанням вузлових систем є Houdini. Houdini має потужні інструменти для створення складних симуляцій, включаючи симуляції рідин, газів, диму та пожежі (рис. 2.4).

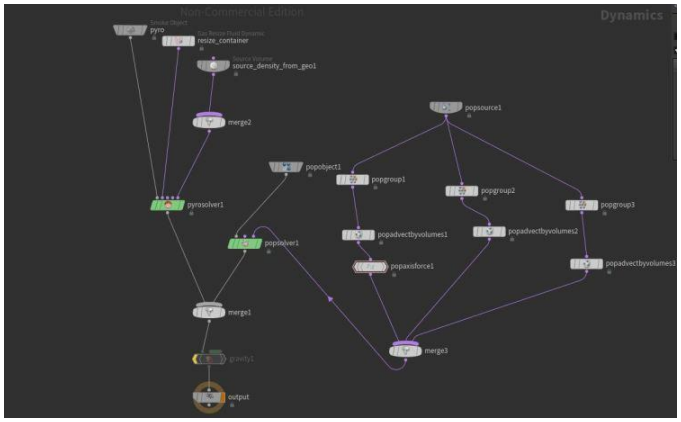

Рис. 2.4. Приклад вузлової системи в Houdini

Наступним пунктом є аналіз технологій створення симуляцій динамічних 3D сцен з використанням вузлових систем. Вузлові системи стали досить поширеними у візуальній ефектній індустрії останнім часом і використовуються для створення складних симуляцій динаміки, анімації, ефектів і графіки в реальному часі. Основна перевага вузлових систем полягає в їх гнучкості і можливості змінювати взаємозв'язки між різними

компонентами складних систем.

Іншим популярним пакетом програмного забезпечення для створення симуляцій динамічних 3D сцен є Cinema 4D. Cinema 4D також має вузлову систему, яка дозволяє створювати складні симуляції та ефекти. Програма має такі інструменти, як динамічна симуляція твердих тіл, рідини і газу, які можуть бути налаштовані і змінені за допомогою вузлів.

Інші популярні пакети програмного забезпечення для створення вузлових систем включають Blender, Unreal Engine, Maya та 3ds Max. Кожен з цих пакетів має свої унікальні інструменти і можливості для створення симуляцій динаміки та ефектів.

Далі, після огляду вузлових систем у програмах для створення симуляцій динамічних 3D сцен, можна зробити висновок, що вони мають чимало переваг перед традиційними методами роботи з графікою та анімацією. Зокрема, вузлові системи дозволяють:

- Зручно керувати процесом роботи з елементами сцени: вузлові системи мають інтуїтивно зрозумілий інтерфейс, що дозволяє користувачеві легко керувати елементами сцени та їх взаємодією;
- Швидко створювати складні сцени: вузлові системи дозволяють швидко створювати складні сцени та ефекти, оскільки вони дозволяють працювати з великою кількістю елементів одночасно та забезпечують швидкий доступ до необхідних інструментів;
- Створювати реалістичні симуляції: вузлові системи дозволяють створювати досить складні та реалістичні симуляції, оскільки забезпечують можливість взаємодії з багатьма елементами сцени та моделювання різних фізичних процесів;
- Компактно зберігати та переміщувати сцени: вузлові системи дозволяють компактно зберігати та переміщувати складні сцени, що дозволяє легко ділитися результатами роботи з іншими користувачами та використовувати їх на інших пристроях.

Таким чином, можна зробити висновок, що вузлові системи дозволяють значно спростити та прискорити процес створення симуляцій

динамічних 3D сцен, а також забезпечують більш точний та реалістичний результат.

Одна з найбільших переваг використання вузлових систем полягає в тому, що вони дозволяють зберігати складність процесів у вигляді графів зв'язків між вузлами, що є більш інтуїтивним способом моделювання процесів та дозволяє значно скоротити час, потрібний на налаштування складних процесів. Крім того, вузлові системи дозволяють використовувати готові компоненти (вузли) для побудови складних симуляцій, що значно полегшує процес розробки.

Ще однією перевагою вузлових систем є можливість здійснювати швидкі зміни в процесі моделювання, що є надзвичайно важливим для анімації та VFX-проектів, де необхідно проводити багато експериментів для досягнення потрібного результату. Вузлові системи дозволяють швидко налаштовувати різні параметри, що забезпечує більшу гнучкість та швидкість розробки.

Також вузлові системи дозволяють легко переносити проекти між різними платформами та програмами. Це означає, що ви можете розробляти складні симуляції в одній програмі (наприклад, Houdini), а потім легко перенести їх до іншої програми (наприклад, Cinema 4D) для подальшої обробки та рендерингу.

Крім того, вузлові системи дозволяють використовувати багатоядерні процесори та графічні процесори (GPU) для обробки та рендерингу симуляцій. Це забезпечує значне прискорення процесу розробки та дозволяє отримувати більш якісні результати за короткий час.

Крім того, вузлові системи дозволяють візуалізувати логіку обробки даних, що спрощує розуміння і налагодження процесу створення симуляції. Кожен вузол може бути збережений, перевикористаний та змінений для інших симуляцій, що дозволяє економити час і ресурси. Вузлові системи також забезпечують більш гнучкий підхід до створення симуляцій, оскільки користувач може вільно маніпулювати параметрами вузлів, додавати та вилучати їх, відключати та змінювати порядок виконання.

Нарешті, вузлові системи дозволяють створювати складніші та більш реалістичні симуляції, в яких декілька елементів можуть взаємодіяти один з одним. Наприклад, вузлова система Cinema 4D має вбудовані інструменти для створення динамічних тіл, рідин та газів, а також для моделювання тканин та інших м'яких матеріалів. Houdini, з іншого боку, володіє розширеними можливостями для симуляції великих масштабів, таких як створення ландшафтів, геологічних формацій та інших природних явищ.

Узагалі, вузлові системи є потужним інструментом для створення симуляцій динамічних 3D сцен, які можуть бути використані в різних сферах, таких як візуальні ефекти для кіно та відеоігор, наукові дослідження, архітектурні візуалізації та багато інших.

Вузлові системи в Cinema 4D та Houdini є дуже потужним інструментом для створення складних симуляцій та візуальних ефектів. В обох програмах це реалізовано дещо по-різному, тому розглянемо кожну з них окремо.

У Cinema 4D вузлова система використовується в модулі XPresso (рис. 2.5). Цей модуль дозволяє створювати складні логічні вузли, що дає можливість керувати параметрами об'єктів відповідно до заданих умов. XPresso дозволяє програмувати різноманітні сценарії, що базуються на інтерактивності користувача або на результаті обчислень. Наприклад, ви можете створити сценарій, який автоматично змінює колір об'єкту в залежності від певного параметра, такого як час або позиція камери. Використання вузлових систем дозволяє значно спростити складні процеси та збільшити швидкість роботи з програмою.

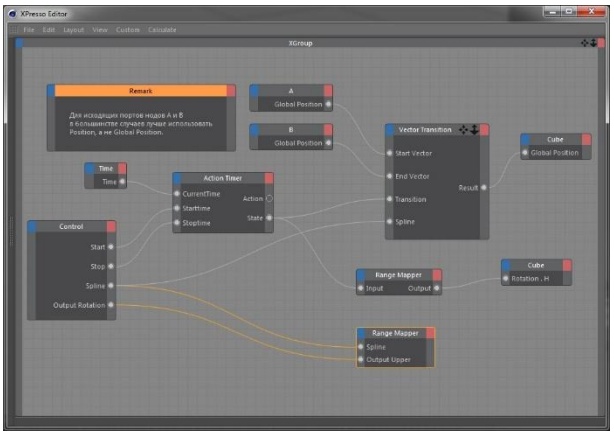

Рис. 2.5. Xpresso editor в Cinema 4D

У Houdini вузлова система є однією з основних функцій програми. Вона дозволяє користувачам створювати складні сценарії, що базуються на взаємодії різних елементів сцени. В Houdini вузлова система дозволяє створювати складні алгоритми, що базуються на інтерактивності користувача або на результаті обчислень. Відповідно до вибраних параметрів, вузлова система може керувати різними аспектами сцени, такими як освітлення, текстури, анімація та інші параметри.

За допомогою вузлових систем в Cinema 4D та Houdini можна створювати складні інтерактивні сценарії без програмування власноруч. Проте, для створення більш складних сценаріїв.

Blender - це вільний і відкритий програмний продукт для створення 3D-графіки, анімації, візуальних ефектів та інтерактивних додатків. Він доступний для використання на різних операційних системах, включаючи Windows, macOS та Linux.

Одна з основних переваг Blender полягає в його безкоштовності та відкритості. Він розповсюджується під ліцензією GNU General Public License (GPL), що дозволяє користувачам вільно використовувати, редагувати, адаптувати і розповсюджувати програму без обмежень. Це робить його доступним для широкої групи користувачів, включаючи художників, дизайнерів, програмістів та студентів.

Blender пропонує різноманітні інструменти для створення 3D-моделей, анімації, текстур, освітлення та спеціальних ефектів. Він має потужний інтерфейс з великою кількістю функцій і можливостей. Користувачі можуть створювати складні сцени, використовувати фізичну симуляцію, налаштовувати матеріали та освітлення, редагувати анімації та використовувати вузлові системи для досягнення бажаних результатів.

Blender також має активну спільноту користувачів і розробників, що сприяє обміну знаннями, розвитку нових інструментів та підтримці користувачів. Існує велика кількість безкоштовних онлайн-ресурсів, навчальних матеріалів, відеоуроків та форумів, які допомагають новачкам оволодіти програмою і покращити свої навички.

Загалом, Blender - це потужний і доступний інструмент для створення 3D-графіки та анімації. Його вільність, відкритість і розширюваність роблять його популярним вибором серед професіоналів та ентузіастів з усього світу.

Вузлові системи в програмі Blender - це потужний інструмент, який дозволяє користувачам створювати складні матеріали, текстури, ефекти та симуляції динамічних 3D сцен. Вони забезпечують гнучкість та масштабованість при налаштуванні різних аспектів сцени і дають велику волю для творчості.

У вузлових системах Blender, вузли - це графічні елементи, які представляють функції або алгоритми, що обробляють дані. Вони представлені у вигляді блоків з певними вхідними та вихідними з'єднаннями, які дозволяють передавати дані між вузлами.

Користувачі можуть створювати складні вузлові мережі, з'єднуючи вузли за допомогою з'єднань і налаштовуючи параметри кожного вузла. Це дозволяє реалістично моделювати фізичні властивості матеріалів, використовувати текстурні процедури, контролювати освітлення, створювати спеціальні ефекти, моделювати фізичні симуляції та багато іншого.

Один з головних переваг вузлових систем в Blender полягає у їхній гнучкості. Користувачі можуть експериментувати з параметрами вузлів, додавати нові вузли, змінювати їхню взаємодію та швидко побачити результати в режимі реального часу. Це дозволяє творцям сцен швидко досягати бажаних ефектів та забезпечує широкі можливості для творчості.

Крім того, вузлові системи в Blender інтегровані з іншими функціями програми. Наприклад, вони можуть бути використані для створення матеріалів, які змінюються відповідно до руху об'єктів, або для контролю анімаційних ефектів. Це дає можливість створювати більш реалістичні та динамічні сцени з високою ступенем деталізації.

Вузлові системи в програмі Blender є потужним інструментом для створення різноманітних ефектів та симуляцій динамічних 3D сцен. Вони дозволяють творцям сцен здійснювати повний контроль над візуальними елементами та створювати вражаючі візуальні ефекти.

В Houdini вузлова система є центральним елементом управління динамічними сценами. Кожен елемент сцени, включаючи геометрію, освітлення, матеріали, анімацію та інші ефекти, може бути представлений в вигляді вузлів, що дозволяє створювати складні системи без необхідності в написанні скриптів або програмуванні.

У Houdini вузли можна розглядати як програми, які взаємодіють між собою за допомогою потоків даних. Кожен вузол має вхідні та вихідні зв'язки, що дає змогу передавати дані між ними. Вузли можна налаштовувати з використанням параметрів, що дозволяє налаштовувати різні аспекти симуляції.

У Cinema 4D вузлова система, відома як XPresso, також використовується для створення складних систем із зв'язків між різними елементами сцени. XPresso дає змогу програмувати складні процеси, використовуючи візуальний інтерфейс для з'єднання різних вузлів між собою.

Обидві системи підтримують використання скриптів для більш складних задач. У Houdini скрипти можна написати на мовах VEX та Python, а в Cinema 4D - на мові Python.

Вузлові системи в Houdini та Cinema 4D дають змогу створювати складні сцени та симуляції без необхідності в програмуванні. Це дозволяє використовувати ці інструменти для різних проектів, включаючи візуальні ефекти для фільмів та відеоігор, архітектурні візуалізації, рекламні матеріали та інші проекти.

2.3. Аналіз різних вузлових систем та їх можливостей

Аналіз різних вузлових систем у 3D графіці з технічної точки зору дозволяє отримати конкретну інформацію про їхні можливості та функціональність. Під час аналізу можуть бути враховані такі аспекти:

1) Розширюваність: Важливо з'ясувати, чи можна розширювати вузлову систему, створюючи власні вузли або редагуючи існуючі. Це дозволяє дизайнерам адаптувати систему під свої потреби та розширювати її функціональність.

2) Гнучкість: Бажано, щоб вузлова система була гнучкою і дозволяла змінювати порядок вузлів, перетягуючи їх або змінюючи з'єднання. Це дозволяє легко налаштовувати логіку системи та створювати складні візуальні структури.
3) Візуалізація: Ефективна візуалізація вузлів та їхніх параметрів сприяє зрозумінню функцій системи. Важливо мати доступ до зрозумілих піктограм, схематичних зображень або підказок, які полегшують розуміння різних функцій вузлів.
4) Підтримка вбудованих операцій: Підтримка вбудованих операцій та функцій, таких як математичні операції, обробка текстур чи алгоритми моделювання, спрощує виконання стандартних завдань у 3D графіці. Це забезпечує швидку та ефективну реалізацію різноманітних ефектів.
5) Ефективність: Ефективність вузлової системи важлива для великих та складних проектів. Важливо мати систему, яка ефективно працює з великою кількістю вузлів і забезпечує швидку обробку даних.

Аналіз вузлових систем допомагає дизайнерам визначити найбільш підходящу систему для їхніх потреб. Враховуючи різні аспекти, вони можуть зробити обґрунтований вибір та використовувати потужні інструменти для реалізації своїх творчих ідей у 3D графіці.

Cinema 4D - це популярна програма для 3D-моделювання, анімації і рендерингу, яка використовує нодову систему для створення складних графічних ефектів та редагування матеріалів. Вона володіє потужним інструментарієм, що дозволяє дизайнерам і художникам створювати реалістичні 3D-сцени та анімацію.

В нодовій системі Cinema 4D вузли представляють різні операції, фільтри, матеріали та ефекти. Вони зображені у вигляді графічних символів, які можуть бути перетягнуті та з'єднані між собою для створення складних ланцюжків дій. Кожен вузол має властивості та параметри, які можна редагувати, щоб досягти потрібного ефекту.

Вузлові системи в програмах 3D-графіки, таких як Blender, Cinema 4D та Houdini, мають багато переваг. Вони дозволяють створювати складні

візуальні ефекти, симуляції та анімації з високою ступенем деталізації (рис. 2.6).

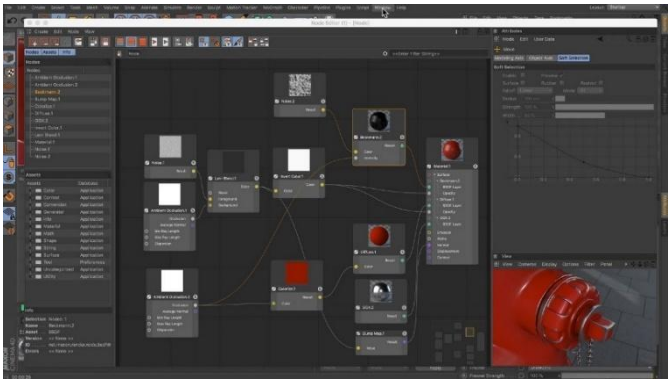

Рис. 2.6. Створення матеріалів за допомогою нодової системи в Cinema 4D

У Blender вузлова система є гнучким інструментом, що надає користувачам повний контроль над візуальними елементами. Вона дозволяє створювати складні матеріали, текстури, ефекти та симуляції. Blender також відкритий і вільний програмний продукт, що забезпечує доступність для широкої спільноти користувачів.

У Cinema 4D вузлова система добре інтегрована з іншими функціями програми. Вона дозволяє легко комбінувати моделювання, анімацію та освітлення у складні сцени. Крім того, Cinema 4D надає потужні інструменти для візуалізації вузлових мереж.

Houdini славиться своїм процедурним підходом до створення ефектів і симуляцій. Вузлова система у Houdini дозволяє створювати складні алгоритми та логіку, що робить його потужним інструментом для створення динамічних ефектів. Houdini також має сильні можливості у створенні складних сцен зі складними взаємодіями об'єктів (рис. 2.7).

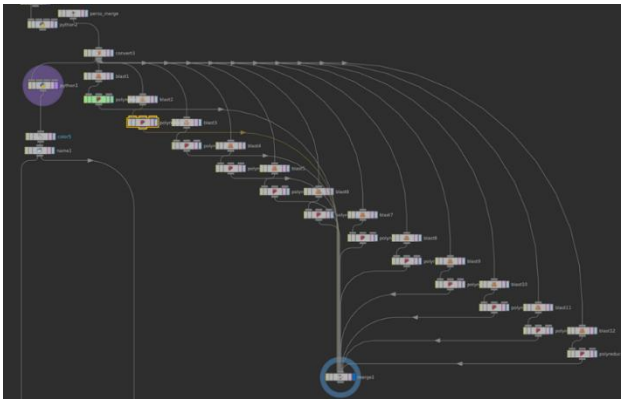

Рис. 2.7. Процедурне вузлове компонування в Houdini

Кожна з цих програм має свої унікальні переваги у вузлових системах. Вибір програми залежить від конкретних потреб і пріоритетів користувача. В будь-якому випадку, вузлові системи дозволяють створювати складні і вражаючі візуальні ефекти та симуляції у 3D-сценах.

Нодова система Cinema 4D має декілька переваг:

- Гнучкість: Користувачі можуть створювати складні мережі вузлів, змінюючи послідовність операцій та їх параметри. Це дозволяє швидко експериментувати з різними конфігураціями та здійснювати точне налаштування ефектів;
- Візуалізація: Вузли в Cinema 4D ілюструються графічними символами, що полегшує сприйняття їх структури та зв'язків. Це сприяє зручній навігації, редагуванню та відладці вузлових мереж;
- Розширені можливості: Cinema 4D має широкий набір вбудованих вузлів, що охоплюють різноманітні області, включаючи геометрію, освітлення, матеріали, анімацію та спеціальні ефекти. Крім того, програма дозволяє створювати власні вузли або імпортувати зовнішні плагіни для розширення функціональності;
- Інтеграція з іншими інструментами: Cinema 4D підтримує обмін даними з іншими програмами для 3D-моделювання, анімації та рендерингу. Це дозволяє використовувати різноманітні інструменти та ресурси для створення складних проектів.

Нодова система Cinema 4D надає дизайнерам широкий спектр можливостей для створення вражаючих 3D-графічних ефектів, редагування матеріалів та анімації. Вона дозволяє гнучко налаштовувати та візуалізувати складні вузлові мережі, що полегшує творчий процес та досягнення бажаного результату.

Houdini FX - це програма для комп'ютерної графіки, яка відома своєю потужною нодовою системою. Вона використовується для створення складних візуальних ефектів, анімації та симуляцій у різних галузях, включаючи кіно, телебачення, ігрову індустрію та архітектуру.

Нодова система Houdini FX базується на концепції графу з вузлами.

Вузли представляють окремі операції або алгоритми, які можна поєднувати між собою, утворюючи складні вузлові мережі. Ці вузли виконують різні функції, від моделювання та анімації до симуляції фізики та генерації візуальних ефектів.

Основні особливості нодової системи Houdini FX:

- Гнучкість: Houdini FX надає користувачам велику свободу в створенні складних вузлових мереж. Вони можуть змінювати порядок вузлів, змінювати їх параметри та настроювати з'єднання, щоб створити бажаний ефект або поведінку;
- Procedural Workflow: Houdini FX базується на процедурному підході до створення графічних ефектів. Це означає, що зміни, внесені в будь-який вузол, автоматично поширюються по всій системі. Це дозволяє швидко вносити зміни та експериментувати з результатами;
- Велика бібліотека вузлів: Houdini FX має велику кількість вбудованих вузлів, які охоплюють різні галузі, включаючи моделювання, анімацію, симуляцію фізики, рендеринг та багато іншого. Користувачі можуть комбінувати ці вузли для створення складних ефектів;
- Python API: Houdini FX також надає доступ до своєї нодової системи за допомогою Python API. Це дозволяє дизайнерам редагувати програмний код вузлів, створювати власні вузли та зовнішні плагіни, розширюючи функціональність програми.

Аналізуючи вузлову систему Houdini FX з технічної точки зору, можна відзначити, що вона надає широкі можливості для створення складних візуальних ефектів та анімації. Завдяки гнучкості, процедурному підходу, багатим набором вузлів та підтримці Python API, дизайнери можуть реалізувати свої творчі ідеї та досягти бажаного результату у сфері 3D графіки.

Можемо використовувати нодову систему в Houdini FX для створення різноманітних симуляцій, які можуть включати симуляцію фізики, рідини, вогню, тканин та багато іншого. Нижче наведені деякі з можливостей, які надає нодова система для створення вражаючих симуляцій.

Симуляція фізики тіл: Використовуючи вбудовані фізичні вузли, я можу

моделювати поведінку твердих тіл, наприклад, вплив сили тяжіння, зіткнення, тертя та інших фізичних властивостей (рис. 2.8). Це дозволяє створювати реалістичні симуляції руху об'єктів.

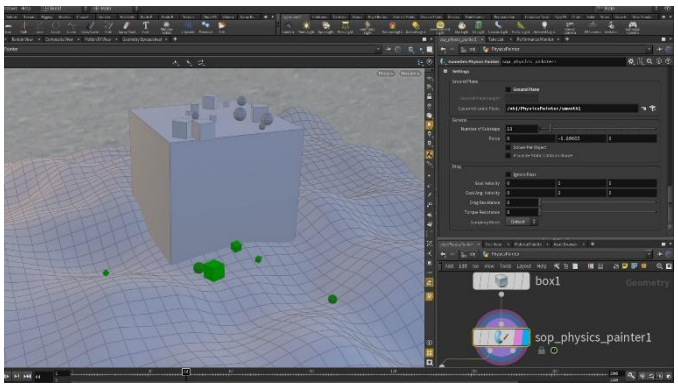

Рис. 2.8. Приклад фізичної симуляції в Houdini

Симуляція рідини: З використанням спеціальних вузлів, я можу моделювати рух рідини, її поведінку під впливом гравітації, тиску, поверхневого натягу та інших факторів. Це дозволяє створювати реалістичні сцени з розливанням рідини, хвилями та багато іншого.

Симуляція вогню: З використанням спеціальних вузлів для моделювання вогню та диму, я можу створювати реалістичні ефекти вогню, вибухів та диму. Це дозволяє створювати захоплюючі сцени з пожежами та експлозіями.

Симуляція тканин: За допомогою вузлів для симуляції тканин, я можу створювати реалістичні ефекти одягу, прапорців, драпіровки та інших текстильних матеріалів (рис. 2.9). Це дозволяє створювати живі, рухомі об'єкти з природною динамікою.

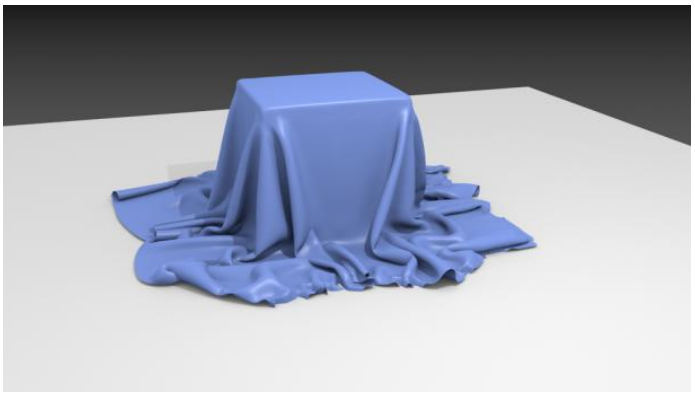

Рис. 2.9. Симуляція тканини в Blender

Симуляція частинок: З використанням вузлів для симуляції частинок, я можу створювати різні ефекти, такі як випадання сніжинок, вибухи, дим, пил та багато іншого. Це дозволяє створювати реалістичні та вражаючі візуальні ефекти.

Завдяки нодовій системі в Houdini FX, я маю повний контроль над симуляціями та можу налаштовувати параметри, змінювати властивості об'єктів та ефектів, щоб досягти бажаного результату. Це дає мені велику творчу свободу та можливість створювати вражаючі та реалістичні сцени, які задовольняють потреби проекту.

2.4. Опис створення вузлових систем та їх параметрів

У програмах Cinema 4D і Houdini FX процес створення нодів та редагування параметрів може трохи відрізнятися, оскільки це дві різні програми з власними інтерфейсами та методологіями роботи з вузловими системами. Опишу загальні принципи створення нод та редагування параметрів в обох програмах:

Cinema 4D:

Створення нодів: У Cinema 4D вузли створюються шляхом додавання об'єктів або ефектів до сцени (рис. 2.10). Наприклад, для створення ноду для моделювання об'єкта, я можу додати геометричну фігуру або імпортувати модель зовнішнього формату. Кожен об'єкт або ефект в сцені є окремим вузлом.

Редагування параметрів: Після створення вузла, я можу відкрити його налаштування та редагувати параметри. Наприклад, для об'єкта це можуть бути параметри розміру, положення, матеріалу тощо. Я можу змінювати ці параметри візуально або за допомогою числових значень, що дає мені контроль над зовнішнім виглядом та поведінкою об'єкта.

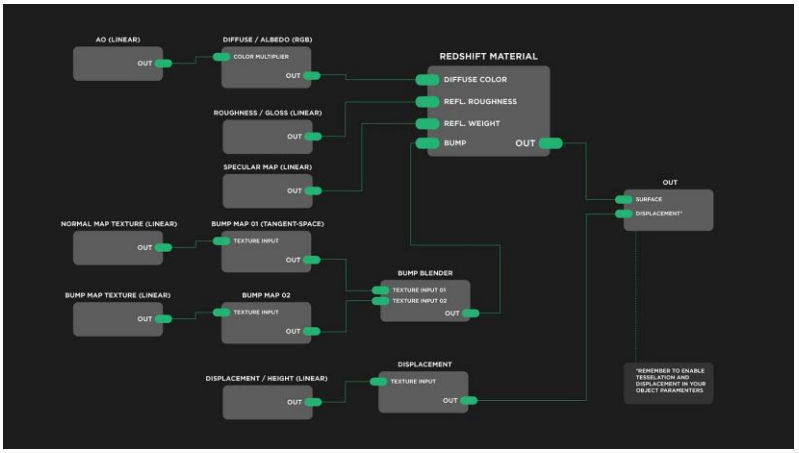

Рис. 2.10. Приклад створення вузлів в Cinema 4D

Houdini FX:

Створення нодів: У Houdini FX вузли створюються шляхом додавання вузлових операторів (OPs) до вузлової мережі (графу) (рис. 2.11). Кожен вузловий оператор виконує певну функцію, таку як моделювання, симуляція, анімація тощо. Їх можна додавати до мережі та з'єднувати між собою для створення потрібної логіки.

Редагування параметрів: Кожен вузловий оператор має свої параметри, які можна редагувати для керування поведінкою оператора. Я можу відкрити властивості вузла та змінювати параметри візуально або вводити числові значення. Більш складні вузли можуть мати багато параметрів для точного налаштування сцени або ефектів.

Як дизайнер, редагуючи параметри вузлів, я маю можливість контролювати вигляд, рух, поведінку та візуальні ефекти у своїх проектах. За допомогою редагування параметрів вузлів, я можу досягти бажаного результату, створити вражаючі візуальні ефекти, налаштувати анімацію об'єктів та керувати поведінкою симуляцій. Програмні коди можуть бути використані для розширення функціональності вузлів або для створення власних вузлів зі специфічними функціями, що дозволяє розширити можливості програми та пристосувати її до власних потреб.

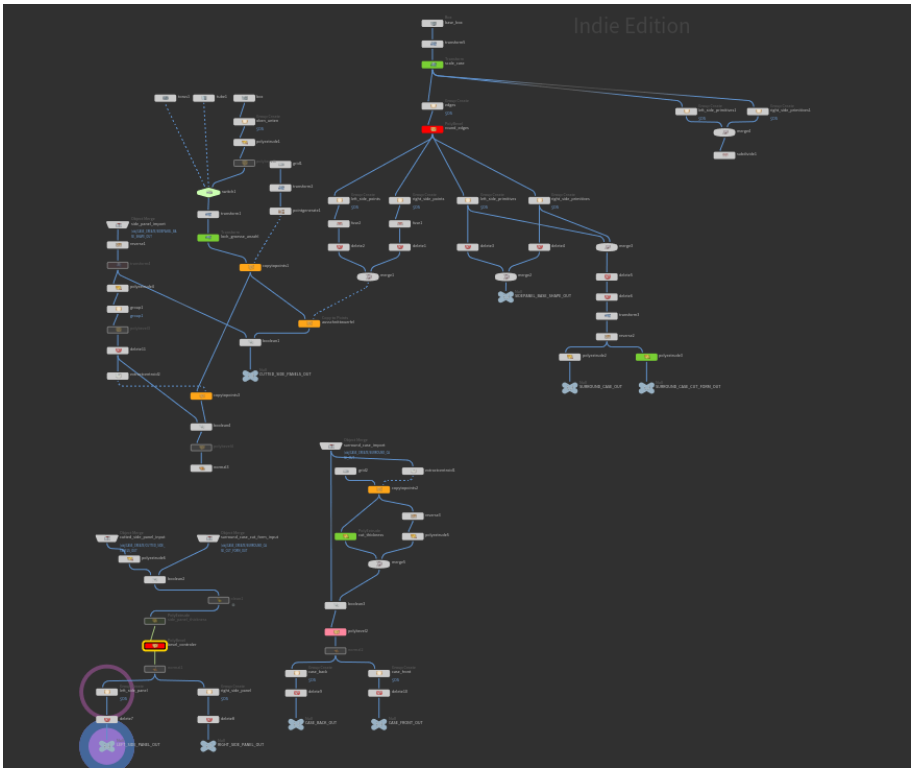

Рис. 2.11. Приклад створення вузлів в Houdini

Опис створення вузлових систем та їх параметрів в програмах для 3D графіки, таких як Cinema 4D і Houdini FX, може бути наступним:

1) Створення вузлових систем:

Створення вузлів: Починається з додавання вузлів до вузлового графу. Вузли виконують певні функції або обробляють дані. Наприклад, вузол може бути відповідальним за генерацію геометрії, симуляцію фізики, анімацію, текстурування тощо.

З'єднання вузлів: Після створення вузлів їх потрібно з'єднати між собою, щоб встановити логіку роботи. Це може включати передачу даних, контроль параметрів або вплив одного вузла на інший. З'єднання можуть бути візуальними лініями або каналами, які передають інформацію між вузлами.

2) Параметри вузлів:

Вхідні параметри: Вузли можуть мати вхідні параметри, які приймають дані з інших вузлів або зовнішніх джерел. Наприклад, вузол симуляції фізики може мати вхідні параметри для контролю сили тяжіння або вихідних даних з моделей об'єктів.

Внутрішні параметри: Вузли також можуть мати внутрішні параметри,

які використовуються для налаштування їхньої роботи. Це може включати налаштування параметрів матеріалу, освітлення, текстур або фільтрації.

Вихідні дані: Вузли можуть також генерувати вихідні дані, які використовуються іншими вузлами або виводяться для візуалізації. Наприклад, вузол генерації геометрії може виробляти 3D моделі, які подаються на вхід вузлу текстурування.

3) Створення симуляцій:

Використання фізики: За допомогою вузлової системи, можна створювати реалістичні симуляції фізики, такі як симуляція руху тіл, розривів, руйнування тощо. Вузли фізики можуть керувати параметрами об'єктів, визначати їхні фізичні властивості та взаємодію між ними.

Анімація: Вузлова система дозволяє створювати складні анімації об'єктів та персонажів. За допомогою вузлів анімації, можна керувати рухом, трансформацією, скелетним анімуванням та іншими аспектами анімації.

Частинки та ефекти: Вузлова система дозволяє створювати різноманітні візуальні ефекти, такі як вибухи, дим, вогонь, дощ, сніг тощо. Вузли частинок дозволяють керувати властивостями частинок, їхньою поведінкою та взаємодією з навколишнім середовищем.

Використовуючи вузлову систему, я можу створювати різноманітні симуляції, що дають мені контроль над реалістичністю та поведінкою об'єктів у віртуальному просторі. Від реалістичних фізичних симуляцій до складних анімаційних ефектів, вузлові системи дозволяють дизайнерам створювати вражаючі візуальні результати у 3D графіці.

Створення та побудова вузлової системи у програмі Blender є захоплюючим та творчим процесом для нас. Ми отримуємо можливість створити складні візуальні ефекти, симуляції та композиції, використовуючи гнучкість та потужність вузлової системи (рис. 2.12).

Рис. 2.12. Приклад створення вузлів в Blender

Завдяки вузловій системі ми можемо вибирати та додавати різні вузли, що представляють різноманітні функціональні елементи, такі як текстури, матеріали, освітлення та ефекти. Підключаючи вузли один до одного, ми створюємо логічну послідовність операцій, які оброблюють дані та впливають на візуальний вихід.

У процесі створення вузлової системи ми можемо налаштовувати параметри вузлів, що дозволяє нам досягти бажаного результату. Ми можемо експериментувати зі змінними, варіювати налаштування та спостерігати, як це впливає на візуальні ефекти.

Окрім того, ми можемо переглядати та аналізувати результати нашої вузлової системи, використовуючи попередній перегляд або рендерну область. Це дає нам змогу бачити, як наша система впливає на візуальні елементи та динаміку сцени.

Ми також цінуємо можливість зберігати та повторно використовувати вузлові системи. Ми можемо зберегти нашу систему як окремий файл або скомпонувати її в готову вузлову групу для подальшого використання у майбутніх проектах або навіть обмінюватися ними з іншими користувачами.

Створення та побудова вузлової системи у Blender відкриває для нас безмежні можливості для творчості та експериментів. Ми можемо створювати вражаючі візуальні ефекти, реалістичні симуляції та комплексні композиції, що розширюють нашу можливість виразити нашу уяву та створювати захоплюючі 3D сцени.

Опис створення вузлових систем в Unreal Engine та їх використання для створення симуляцій та різних функцій може бути наступним.

1) Створення вузлових систем:

Створення вузлів: У Unreal Engine вузлами є графові вузли, які представляють різні функціональні блоки. Вони можуть бути вузлами логіки, математичними вузлами, вузлами руху, вузлами анімації та багатьма іншими.

З'єднання вузлів: Після створення вузлів їх потрібно з'єднати між собою, щоб встановити послідовність операцій або логіку роботи. Вузли можуть бути з'єднані визначеними каналами, які передають дані від одного вузла до іншого.

2) Створення симуляцій:

Фізика: За допомогою вузлової системи в Unreal Engine можна створювати реалістичні фізичні симуляції. Вузли фізики дозволяють моделювати рух об'єктів, визначати їхні фізичні властивості, такі як маса, тертя, пружність, та контролювати взаємодію об'єктів між собою та з оточенням.

Частинки та ефекти: Вузлова система Unreal Engine дозволяє створювати різні візуальні ефекти, включаючи емісію частинок, дим, вогонь, руйнування та інші. Вузли частинок дозволяють керувати параметрами частинок, їхнім життєвим циклом, поведінкою та взаємодією з оточенням.

3) Різні функції та можливості:

Логіка та управління: Вузлова система Unreal Engine дозволяє створювати логіку та управління поведінкою об'єктів. За допомогою вузлів логіки, можна створювати умови, цикли, розгалуження та інші операції, що контролюють роботу сцени чи персонажів.

Анімація: Вузлова система дозволяє створювати анімацію об'єктів та персонажів. За допомогою вузлів анімації можна керувати рухом, трансформацією, скелетним анімуванням та іншими аспектами анімації.

Редакція матеріалів: Вузлова система Unreal Engine дозволяє редагувати матеріали об'єктів. Вузли матеріалів дозволяють створювати складні текстурні ефекти, такі як блиск, відблиск, прозорість, перехід кольору та багато інших.

Я, як дизайнер, використовую вузлову систему в Unreal Engine для

створення складних симуляцій, візуальних ефектів та функціональності у моїх проектах. За допомогою вузлових систем, я можу забезпечити реалістичність, взаємодію та управління різними аспектами мого віртуального світу.

2.5. Оформлення формул вузлових систем

Програма Blender використовує мову програмування Python для розширення функціональності та автоматизації задач. З допомогою програмного коду Python можна створювати, змінювати та видаляти об'єкти, матеріали, вузли та інші елементи сцени. Також можна налаштовувати параметри об'єктів, матеріалів, вузлів та анімацій, керувати процесом рендерингу та зберіганням зображень, а також створювати скрипти та додатки, що автоматизують рутинні задачі або розширюють функціональність програми.

Програма Cinema 4D використовує мову програмування C++ для розширення функціональності та створення плагінів. З допомогою програмного коду C++ можна створювати нові типи об'єктів, матеріалів, вузлів та інших елементів сцени. Також можна розширювати функціональність програми, додавати нові інструменти та можливості, налаштовувати параметри об'єктів, матеріалів, вузлів та анімацій, а також взаємодіяти зі сторонніми програмами та зовнішніми пристроями.

Програма Houdini використовує мову програмування VEX (Vector EXpression Language) та Python для створення вузлових графіків та розширення функціональності. З допомогою програмного коду в Houdini можна створювати та керувати вузловими графіками для генерації геометрії, ефектів, симуляцій та інших візуальних елементів. Також можна використовувати VEX для написання спеціалізованих функцій та алгоритмів для обробки даних та створення складних ефектів, а Python використовувати для автоматизації задач, розширення функціональності та взаємодії з іншими програмами.

Завдяки програмному коду ми можемо впливати на процес у цих

програмах, створюючи спеціалізовані інструменти, автоматизуючи рутинні

завдання, розширюючи функціональність та створюючи унікальні візуальні ефекти та симуляції.

У програмах Cinema 4D та Houdini FX, формули вузлових систем зазвичай виражаються за допомогою мови програмування, специфічної для кожної програми. Нижче наведено приклади розподілу формул та кодування вузлових систем у цих програмах:

У Cinema 4D вузли матеріалів мають різні параметри та властивості, які можна налаштовувати. Наприклад, для створення текстур можна використовувати вузли, що генерують шум, градієнти, процедурні маски тощо. Кодування вузлів матеріалів в Cinema 4D виглядає наступним чином (рис. 2.13):

```
var noiseNode = BaseList2D::Alloc(Nnoise);
var gradientNode = BaseList2D::Alloc(Ngradient);
var materialNode = BaseList2D::Alloc(Nmaterial);
// Підключення вузлів
noiseNode->GetNodeMaster()->InsertUnder(materialNode);
gradientNode->GetNodeMaster()->InsertUnder(materialNode);
// Налаштування параметрів
noiseNode->SetParameter(Dnoise::NOISE_TYPE,
Dnoise::NOISE_TYPE_FBM);
gradientNode->SetParameter(Dgradient::GRADIENT_TYPE,
Dgradient::GRADIENT_TYPE_LINEAR);
// Підключення вузлів до властивостей матеріалу
materialNode->SetParameter(Dmaterial::COLOR_SHADER, noiseNode);
materialNode->SetParameter(Dmaterial::BUMP_SHADER, gradientNode);
```

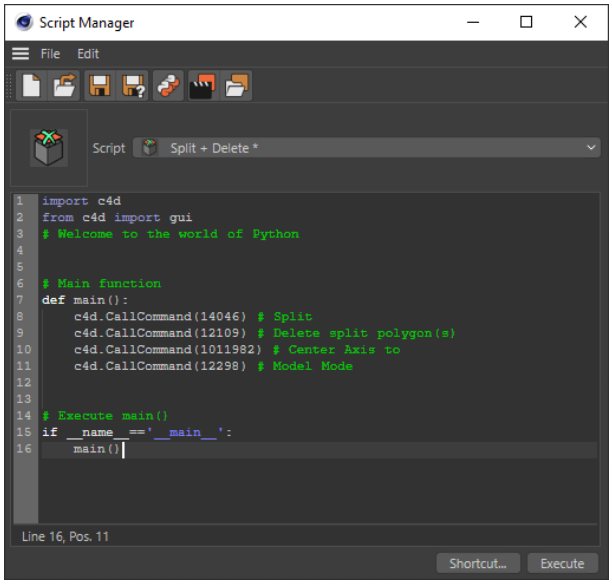

Рис. 2.13. Script Manger в Cinema 4D

У Houdini FX вузли симуляцій мають різні фізичні параметри та правила взаємодії. Наприклад, для створення симуляції руйнування можна використовувати вузли, що визначають матеріал об'єкту, його міцність, тип колізій та правила руйнування. Кодування вузлів симуляцій в Houdini FX може виглядати наступним чином (рис. 2.14):

```
import hou
# Створення вузлів
materialNode = hou.node("/obj/material")
strengthNode = hou.node("/obj/strength")
collisionNode = hou.node("/obj/collision")
simulationNode = hou.node("/obj/simulation")
# Підключення вузлів
strengthNode.setInput(0, materialNode)
collisionNode.setInput(0, materialNode)
collisionNode.setInput(1, strengthNode)
simulationNode.setInput(0, materialNode)
simulationNode.setInput(1, collisionNode)
# Налаштування параметрів
materialNode.parm("roughness").set(0.2)
strengthNode.parm("strength").set(0.5)
```

collisionNode.parm("collision_type").set("destruction")

simulationNode.parm("solver_type").set("rigid")

 # Запуск симуляції

simulationNode.parm("simulate").pressButton()

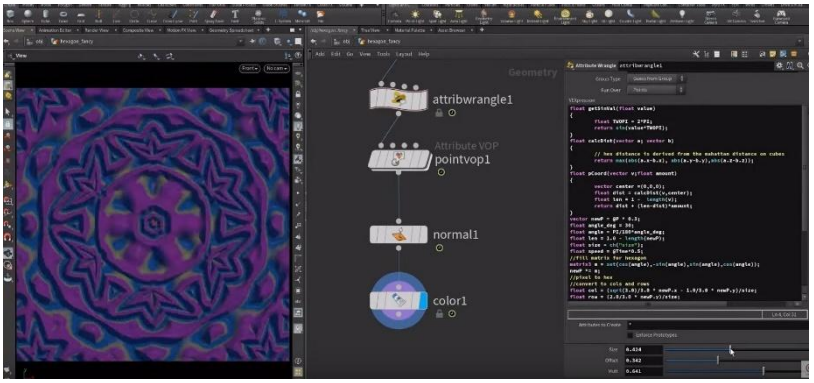

Рис. 2.14. VEX кодування в Houdini

Вище наведені приклади лише загальні ілюстрації того, як можна створювати та кодувати вузлові системи у програмах Cinema 4D та Houdini FX. Фактичні формули та код можуть сильно варіюватися в залежності від конкретного завдання та налаштувань.

Зважаючи на широкий спектр можливостей і різноманітність симуляцій у програмах 3D графіки, наведу кілька прикладів формул, які можуть бути використані для створення різних симуляцій. Наголошую, що конкретні формули можуть сильно варіюватися залежно від типу симуляції та програмного забезпечення.

1) Симуляція руйнування будівлі:

Формула фізичної моделі руйнування: Залежно від об'єкта та його властивостей, можуть використовуватися формули, такі як модель руйнування Базена-Коллапса або модель руйнування Фрейміна.

2) Симуляція вогню:

Формула моделювання розповсюдження вогню: Залежно від типу симуляції вогню (наприклад, моделювання горіння рідини або горіння матеріалу), можуть використовуватися різні формули, такі як формула Нав'є-Стокса для моделювання руху газу та рідини, або модель хімічної реакції для

врахування взаємодії речовин та горіння.

3) Симуляція частинок:

Формула руху частинок: для моделювання руху частинок можуть використовуватися формули, такі як формули Ньютона для обчислення сили, прискорення та швидкості частинок під впливом зовнішніх сил та внутрішніх взаємодій.

Це лише загальні приклади формул, які можуть бути використані у симуляціях в 3D графіці. Конкретні формули і їх використання значно залежать від типу симуляції, програмного забезпечення та бажаного результату.

Симуляція руйнування будівлі (приклад у Houdini FX):

```
# Код для руйнування будівлі
# Підключення вузла руйнування
destroyNode = hou.node("/obj/building").createNode("rbdmaterialimpact")
# Налаштування параметрів вузла
destroyNode.parm("impactstrength").set(0.5)
destroyNode.parm("impactradius").set(2.0)
# Підключення вузла руйнування до системи динаміки RBD
hou.node("/obj/building/dopnetwork").createNode("force").setInput(0, destroyNode, 0)
```

Симуляція вогню (приклад у Unreal Engine):

```
// Код для симуляції вогню
// Створення емітера частинок для вогню
UParticleSystemComponent* FireEmitter = NewObject<UParticleSystemComponent>(this);
// Завантаження попередньо створеної системи частинок вогню
UParticleSystem* FireParticle = LoadObject<UParticleSystem>(nullptr, TEXT("Path/To/FireParticle"));
// Прикріплення емітера до об'єкту
FireEmitter->AttachToComponent(MeshComponent,
```

```
FAttachmentTransformRules::KeepRelativeTransform);
// Встановлення системи частинок вогню для емітера
FireEmitter->SetTemplate(FireParticle);
```

Симуляція частинок (приклад у Cinema 4D):

```
# Код для симуляції частинок
# Створення вузла системи частинок
particleNode = c4d.BaseObject(c4d.Onull)
particleNode[c4d.ID_BASEOBJECT_VISIBILITY_EDITOR] = 1
particleNode[c4d.ID_BASEOBJECT_VISIBILITY_RENDER] = 1
particleNode[c4d.ID_BASELIST_NAME] = "Particle System"
doc.InsertObject(particleNode)
# Додавання емітера частинок до системи
emitter = c4d.BaseObject(c4d.Oparticle)
particleNode.InsertObject(emitter)
# Налаштування параметрів емітера
emitter[c4d.PARTICLEOBJECT_EMITTER] = True
emitter[c4d.PARTICLEOBJECT_SPEED] = 100.0
```

За допомогою програмного коду вузлові системи в програмі Blender можуть бути побудовані за допомогою мови програмування Python та API Blender. Ось кілька прикладів програмного коду, які демонструють створення та конфігурацію вузлових систем (рис. 2.15):

Створення вузла "Add Shader" (додавання шейдерів):

```
import bpy
# Створення вузла "Add Shader"
add_shader_node = bpy.context.scene.node_tree.nodes.new('ShaderNodeAddShader')
# Позиціювання вузла на графі
add_shader_node.location = (0, 0)
```

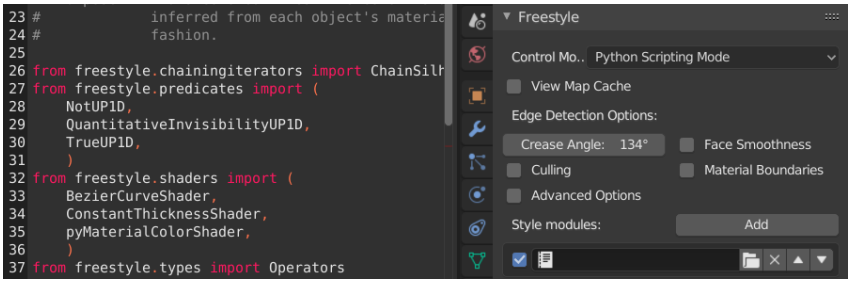

Рис. 2.15. Інтегрування програмного коду до Blender

Підключення вхідних та вихідних з'єднань вузлів:

import bpy

 # Отримання вузлів для підключення

node1 = bpy.context.scene.node_tree.nodes['Node1']

node2 = bpy.context.scene.node_tree.nodes['Node2']

output_node = bpy.context.scene.node_tree.nodes['Output']

 # Підключення вхідних та вихідних з'єднань

bpy.context.scene.node_tree.links.new(node1.outputs['Output'], node2.inputs['Input'])

bpy.context.scene.node_tree.links.new(node2.outputs['Output'], output_node.inputs['Surface'])

Налаштування параметрів вузла:

import bpy

 # Отримання вузла для налаштування

node = bpy.context.scene.node_tree.nodes['Node']

 # Налаштування параметрів вузла

node.parameter = value

Ці приклади коду демонструють базовий підхід до реалізації симуляцій у програмах 3D графіки. Конкретний код і його деталі сильно залежать від використовуваного програмного забезпечення та бажаного результату симуляції. Важливо вивчати документацію та ресурси, що стосуються конкретного програмного забезпечення, для отримання докладної інформації

про функціонал і можливості, а також для налагодження та розширення коду симуляцій.

2.6. Розробка алгоритмів для взаємодії між вузловими системами

Розробка алгоритмів для взаємодії між вузловими системами є важливим процесом у сфері розподілених систем. Основна мета полягає у забезпеченні ефективного обміну даними та координації дій між вузлами.

Одним із поширених підходів є використання протоколів комунікації. Це набір правил і форматів, які дозволяють вузлам обмінюватися інформацією. Наприклад, REST (Representational State Transfer) є одним з популярних протоколів, що базується на використанні HTTP запитів.

Крім того, розробка алгоритмів для взаємодії між вузловими системами включає в себе обробку помилок, управління ресурсами та вирішення конфліктів. Для цього можуть використовуватися різні методи, такі як ретрі-стратегії, механізми повтору або алгоритми розподіленого блокування.

При розробці алгоритмів також важливо враховувати масштабованість та надійність системи. Наприклад, можуть бути використані методи реплікації даних або розподілу навантаження для забезпечення високої доступності та швидкодії вузлів.

Узагалі, розробка алгоритмів для взаємодії між вузловими системами є складним процесом, що вимагає ретельного аналізу вимог, вибору відповідних протоколів та розробки логіки обміну даними між вузлами.

При розробці алгоритмів для взаємодії між вузловими системами також можуть використовуватися різні підходи, залежно від потреб і характеристик системи. Наприклад:

Модель клієнт-сервер: Цей підхід базується на розподілі ролей між вузлами. Клієнти відправляють запити до серверів, які надають відповіді та обробляють запити. Це дозволяє ефективно керувати навантаженням та забезпечує централізовану точку доступу до ресурсів.

Подієва модель: У цій моделі вузли обмінюються повідомленнями про відбуті події. Кожен вузол може бути і відправником, і отримувачем

повідомлень, що дозволяє асинхронну комунікацію та забезпечує розподілену обробку подій.

Публікація-підписка: Цей підхід базується на способі комунікації, де вузли можуть публікувати повідомлення на певні теми, а інші вузли можуть підписуватися на ці теми та отримувати повідомлення про них. Це дозволяє реалізувати розсилку повідомлень та асинхронну взаємодію між вузлами.

Модель peer-to-peer (P2P): у цій моделі кожен вузол може одночасно виконувати як роль клієнта, так і роль сервера. Вузли взаємодіють без централізованої координації, обмінюючись даними та ресурсами між собою. Це дозволяє побудувати розподілені системи з високою масштабованістю та стійкістю до відмов.

Розробка алгоритмів для взаємодії між вузловими системами є важливим процесом у сфері розподілених систем. Основна мета полягає у забезпеченні ефективного обміну даними та координації дій між вузлами.

Одним із поширених підходів є використання протоколів комунікації. Це набір правил і форматів, які дозволяють вузлам обмінюватися інформацією. Наприклад, REST (Representational State Transfer) є одним з популярних протоколів, що базується на використанні HTTP запитів.

Крім того, розробка алгоритмів для взаємодії між вузловими системами включає в себе обробку помилок, управління ресурсами та вирішення конфліктів. Для цього можуть використовуватися різні методи, такі як ретрі-стратегії, механізми повтору або алгоритми розподіленого блокування.

При розробці алгоритмів також важливо враховувати масштабованість та надійність системи. Наприклад, можуть бути використані методи реплікації даних або розподілу навантаження для забезпечення високої доступності та швидкодії вузлів.

Узагалі, розробка алгоритмів для взаємодії між вузловими системами є складним процесом, що вимагає ретельного аналізу вимог, вибору відповідних протоколів та розробки логіки обміну даними між вузлами.

При розробці алгоритмів для взаємодії між вузловими системами також можуть використовуватися різні підходи, залежно від потреб і характеристик

системи. Наприклад:

1) Модель клієнт-сервер: Цей підхід базується на розподілі ролей між вузлами. Клієнти відправляють запити до серверів, які надають відповіді та обробляють запити. Це дозволяє ефективно керувати навантаженням та забезпечує централізовану точку доступу до ресурсів.
2) Подієва модель: У цій моделі вузли обмінюються повідомленнями про відбуті події. Кожен вузол може бути і відправником, і отримувачем повідомлень, що дозволяє асинхронну комунікацію та забезпечує розподілену обробку подій.
3) Публікація-підписка: Цей підхід базується на способі комунікації, де вузли можуть публікувати повідомлення на певні теми, а інші вузли можуть підписуватися на ці теми та отримувати повідомлення про них. Це дозволяє реалізувати розсилку повідомлень та асинхронну взаємодію між вузлами.
4) Модель peer-to-peer (P2P): у цій моделі кожен вузол може одночасно виконувати як роль клієнта, так і роль сервера. Вузли взаємодіють без централізованої координації, обмінюючись даними та ресурсами між собою. Це дозволяє побудувати розподілені системи з високою масштабованістю та стійкістю до відмов.

ВИСНОВКИ ДО РОЗДІЛУ 2

Розділ присвячений вивченню вузлових систем і їх використанню для симуляцій динамічних 3D сцен. У ході дослідження були визначені основні потреби, які можуть бути задоволені за допомогою вузлових систем. Виявлено, що такі системи мають численні переваги, включаючи швидкість обчислень, масштабованість та можливість моделювання складних фізичних взаємодій.

Був проведений аналіз різних вузлових систем та їх можливостей, що дозволило виявити різні підходи до розробки таких систем. Встановлено, що існують різні методи створення вузлових систем та різні параметри, які можна налаштовувати для досягнення бажаних результатів.

Крім того, розглянуті принципи оформлення формул для математичного опису вузлових систем. Виявлено, що точність і зручність формул залежать від правильного використання математичних об'єктів та операторів.

Також була проведена розробка алгоритмів для взаємодії між вузловими системами. Встановлено, що важливим аспектом розробки є ефективна комунікація між вузлами та синхронізація їх дій.

Отже, розділ зумовив глибоке розуміння вузлових систем, їхніх переваг і можливостей. Вивчення цих аспектів може допомогти в подальшому вдосконаленні симуляцій динамічних 3D сцен та розробці більш ефективних алгоритмів взаємодії між вузловими системами.

РОЗДІЛ 3
РЕАЛІЗАЦІЯ РОЗРОБЛЕНОЇ ТЕХНОЛОГІЇ НА ПРИКЛАДІ ДИНАМІЧНОЇ 3D СЦЕНИ

У рамках даної роботи ми зосереджуємося на створенні динамічної симуляції частинок за допомогою програми Blender. Симуляція частинок є потужним інструментом, що дозволяє відтворювати реальну фізичну поведінку об'єктів у віртуальному середовищі.

Наша мета полягає в створенні живої та реалістичної симуляції частинок, де кожна частинка буде взаємодіяти з оточуючими об'єктами та виявляти фізичні властивості, такі як рух, колізії та інші динамічні ефекти. Для досягнення цієї мети ми використовуємо програмний код Python, що запускається всередині Blender.

У нашому дослідженні ми будемо крок за кроком створювати симуляцію частинок у 3D-просторі. Ми розпочнемо зі створення простору, в якому будуть знаходитися наші частинки. Далі, ми встановимо фізичні властивості для об'єкту-простору та самого об'єкту частинок. Це дозволить нам задати масу, форму та інші параметри для наших частинок.

Після цього ми налаштуємо параметри симуляції, такі як часові кроки та типи колізій. Запустивши симуляцію, ми побачимо, як частинки починають взаємодіяти одна з одною та з оточуючим середовищем, відображаючи реалістичні фізичні ефекти.

Коли симуляція буде завершена, ми зможемо використати отримані результати для візуалізації, аналізу або інших дослідницьких цілей. Наша робота дозволить нам більше розуміти процеси динаміки частинок та застосовувати їх у різних галузях, включаючи графіку, візуалізацію даних та наукові дослідження.

Цей вступ від імені "нас" допоможе вам почати вашу роботу та описати загальну мету та підхід до створення динамічної симуляції частинок у Blender.

3.1. Перша частина симуляції

Наступний крок, що ми будемо робити:

Додамо частинки до нашої симуляції. Створимо новий об'єкт типу "частинка" за допомогою:

bpy.ops.object.particle_system_add().

Налаштуємо параметри частинок, такі як кількість, форма, маса тощо. Встановимо початкове положення та швидкість частинок (рис. 3.1).

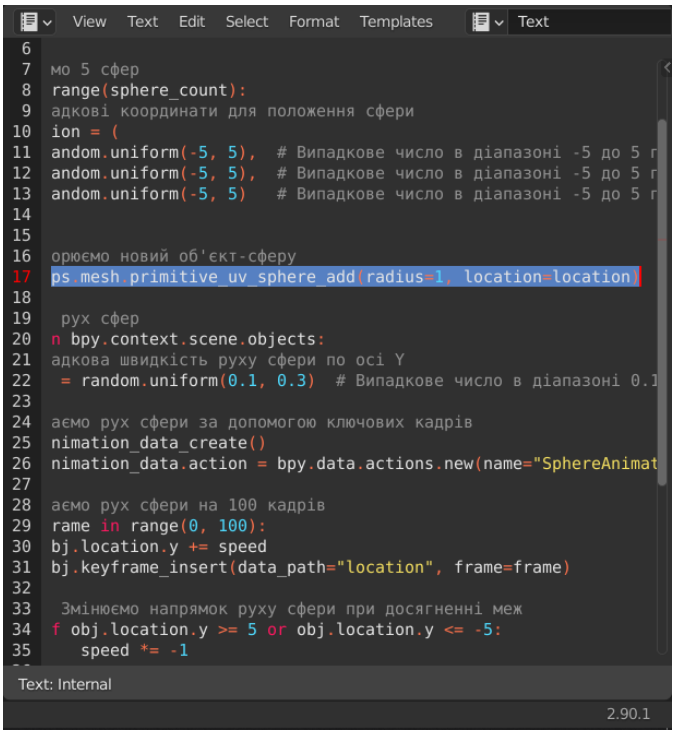

Рис. 3.1. Створення нового об'єкту

Встановимо фізичні властивості частинок:

- Визначимо параметри фізичної поведінки, такі як гравітація, тертя, сили взаємодії тощо;
- Налаштуємо режими колізій та обмежень для частинок, щоб вони взаємодіяли з простором та іншими об'єктами.

Задамо рух частинок:

- Використовуємо ключові кадри або програмування анімації, щоб

встановити траєкторію руху частинок з часом;

- Застосовуємо сили, пружини або інші ефекти, щоб моделювати реалістичний рух частинок.

Запустимо симуляцію та спостерігатимемо результати:

- Запустимо симуляцію за допомогою bpy.ops.ptcache.bake(), щоб обчислити рух частинок;
- Відтворимо симуляцію та переглянемо її результати, переходячи через кадри анімації;
- Аналізуємо отримані дані, спостерігаємо рух та взаємодію частинок.

Візуалізуємо та відтворимо результати:

- Налаштуємо візуальне представлення частинок, таке як колір, текстуру, форму тощо;
- Застосуємо освітлення та матеріали для покращення реалістичності сцени;
- Запишемо анімацію частинок у відео або зображення для подальшого використання та презентації результатів.

Ці кроки дозволять нам створити динамічну симуляцію частинок у Blender за допомогою програмного коду, яка демонструватиме реалістичну поведінку та взаємодію частинок у віртуальному середовищі.

3.2. Створення динаміки

Далі, ми будемо створювати промпт для Blender та вставляти його у наш проект:

Відкриємо текстовий редактор у Blender, натиснувши "Scripting" на панелі вибору редактора.

Створимо новий текстовий файл та назвемо його, наприклад, "particle_simulation_prompt.py" (рис. 3.2).

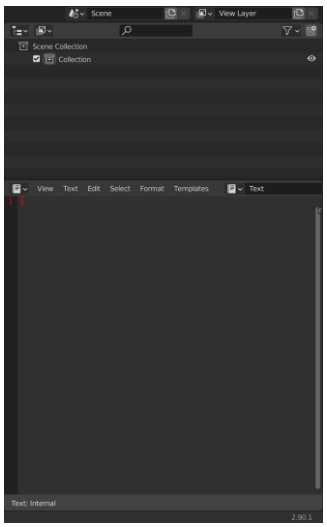

Рис. 3.2. Створення нового текстового редактора

В цьому файлі почнемо писати код Python для створення промпта. Використовуємо коментарі для кожної стрічки коду, щоб пояснити, що вона означає та яку функцію виконує:

Імпортуємо необхідні модулі для промпта:

import bpy

from bpy.types import Operator

Створюємо клас оператора для нашого промпта:

class ParticleSimulationPrompt(Operator):

Описуємо назву та опис оператора:

bl_idname = "object.particle_simulation_prompt"

bl_label = "Particle Simulation Prompt"

bl_description = "Custom prompt for particle simulation"

Виконуємо основну логіку промпта:

def execute(self, context):

return {'FINISHED'}

Тут можна виконати будь-які дії або виклики функцій, пов'язані з

симуляцією частинок. Наприклад, створити об'єкт-простір, додати частинки, встановити фізичні параметри тощо.

Реєструємо оператор промпта:

def register():

bpy.utils.register_class(ParticleSimulationPrompt)

Відмінюємо реєстрацію оператора промпта:

def unregister():

bpy.utils.unregister_class(ParticleSimulationPrompt)

При виконанні промпта реєструємо оператор:

if __name__ == "__main__":

register()

Збережемо файл "particle_simulation_prompt.py".

Повернемось до Blender та натиснемо кнопку "Text" на панелі вибору редактора. Відкриємо файл "particle_simulation_prompt.py".

Натиснемо кнопку "Run Script" у текстовому редакторі або виконаємо комбінацію клавіш "Alt+P", щоб запустити наш код (рис. 3.3).

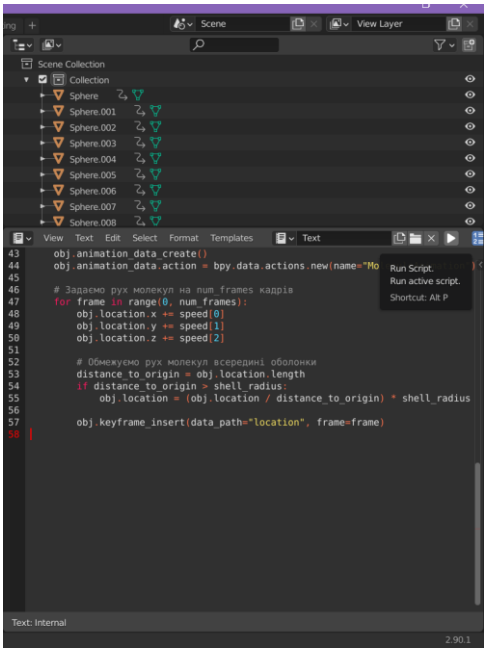

Рис. 3.3. Запускаємо наш скрипт

Після запуску коду Blender зареєструємо наш оператор промпта.

Тепер ми можемо викликати промпт, виконавши пошук *"Particle Simulation Prompt"* у панелі пошуку або додавши його до панелі інструментів.

Коли ми викличемо промпт, наша власна логіка, що стосується симуляції частинок, буде виконуватись. Ви можете додати більше функцій та налаштувань до промпта, щоб він відповідав вашим потребам.

3.3. Текстуринг та додавання матеріалів

Для створення реалістичних та привабливих візуальних ефектів у нашій динамічній симуляції частинок, ми будемо використовувати матеріали. Матеріали в Blender дозволяють надати частинкам колір, текстуру та властивості відбивання світла, щоб вони виглядали більш реалістично та привабливо.

Опис процесу створення матеріалів для частинок (рис. 3.4):

1) Отримаємо доступ до об'єкту частинок:

```
# Отримуємо активний об'єкт (об'єкт частинок)
particle_object = bpy.context.object
```

2) Створимо новий матеріал для частинок:

```
# Створюємо новий матеріал
particle_material = bpy.data.materials.new(name="ParticleMaterial")
```

3) Налаштуємо властивості матеріалу:

```
# Встановлюємо колір матеріалу (приклад: червоний)
particle_material.diffuse_color = (1.0, 0.0, 0.0, 1.0)
# Встановлюємо інші властивості матеріалу (за потреби)
# Наприклад: властивості відбивання, прозорості, сили світла тощо
```

4) Призначимо матеріал частинкам:

```
# Отримуємо доступ до системи частинок
particle_system = particle_object.particle_systems[0]
# Отримуємо доступ до матеріалів системи частинок
```

particle_system_materials = particle_system.settings.material_slots

 # Додаємо новий матеріал до системи частинок

particle_system_materials.append(particle_material)

5) Зробимо налаштування матеріалу для системи частинок:

 # Отримуємо доступ до останнього матеріалу в системі частинок (наш новий матеріал)

last_material_index = len(particle_system_materials) - 1

particle_system.material_slots[last_material_index].material = particle_material

6) Збережемо зміни:

 # Оновлюємо сцену, щоб зміни були відображені

bpy.context.view_layer.update()

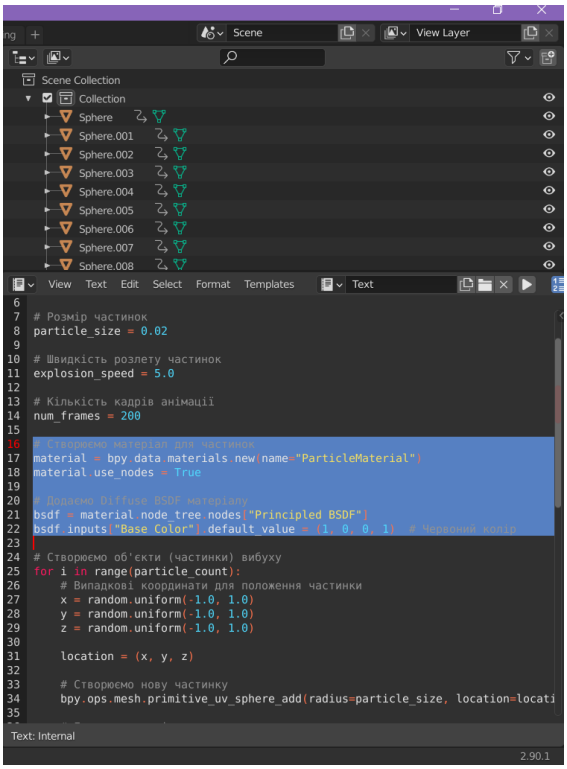

Рис. 3.4. Створення матеріалів

Тепер, коли ми створили матеріал для наших частинок у Blender за допомогою програмного коду, ми можемо продовжити налаштування

матеріалів, додавання текстур або зміну будь-яких властивостей, які відповідають нашим потребам. З використанням цього підходу ми зможемо створити візуально захопливу симуляцію частинок, яка буде відповідати нашим вимогам та потребам проекту.

3.4. Переніс симуляції до програмного забезпечення Cinema 4D для рендеру Redshift

Далі ми будемо створювати переніс нашої динамічної 3D симуляції до програмного забезпечення Cinema 4D, в особливості будемо використовувати рендер двигун RedShift.

Програма Blender також має свій рендер двигун Cycles, котрий в порівнянні з іншими двигунами має свої переваги. Але в даній роботі ми будемо використовувати Redshift від MaxonOne.

Redshift має переваги перед Blender та Octane у вигляді швидкості рендерингу, масштабованості, гнучкості та високої якості рендерингу. Його розподілена обчислювальна система та використання GPU дозволяють прискорити процес рендерингу, зокрема при обробці складних сцен.

Redshift також інтегрується з різними 3D-пакетами, включаючи Blender, що дає можливість використовувати його в існуючих проектах.

Крім того, він забезпечує високу якість рендерингу з реалістичним освітленням, тіньми, відблисками та матеріалами. Експорт симуляції з Blender:

- У Blender виберіть об'єкт частинок та вкажіть налаштування симуляції (наприклад, властивості фізики, розмір системи частинок тощо);
- Збережіть сцену у форматі, який підтримується Cinema 4D (наприклад, .fbx або .obj), включаючи дані про частинки та їхні параметри симуляції

1) Імпорт симуляції до Cinema 4D:

- Відкриємо програму Cinema 4D та створимо новий проект;

Виберемо опцію імпорту або перетягнемо експортований файл з Blender у вікно Cinema 4D для імпорту;

- Підтвердимо налаштування імпорту, включаючи правильність

масштабування та налаштування імпортованих об'єктів частинок.

2) Налаштування матеріалів у Cinema 4D:

Відкриємо налаштування матеріалів для об'єктів частинок у Cinema 4D.

Застосуємо матеріали, кольори та текстури до об'єктів частинок за допомогою інтерфейсу Cinema 4D.

Налаштуємо властивості матеріалів, такі як відбивання світла, прозорість тощо, для досягнення бажаного візуального ефекту.

3) Налаштування освітлення та камери:

– Додамо світла та налаштуйте їхнє розташування, інтенсивність та кольори, щоб підкреслити візуальні ефекти симуляції частинок;
– Налаштуємо позицію та параметри камери для отримання бажаної ракурсу та кадру для рендерингу.

4) Рендеринг симуляції:

– Встановимо налаштування рендерингу, такі як роздільність, формат виведення, якість тощо;
– Запустимо процес рендерингу для створення відео або зображень, які відображатимуть вашу симуляцію частинок в Cinema 4D.

З цими кроками ми зможемо успішно перенести нашу симуляцію частинок з Blender до Cinema 4D і використати потужності останньої для відображення та рендерингу симуляції в бажаному стилі та якості.

3.5. Налаштування рендеру

1) Налаштування глобального освітлення (Global Illumination) (рис. 3.5):

– GI Type: Виберемо тип глобального освітлення, наприклад, Path Tracing або Irradiance Point Cloud;
– GI Quality: Встановимо рівень якості глобального освітлення. Вища якість забезпечує більш точний результат, але може займати більше часу на рендерінг;
– GI Sampling: Налаштуємо параметри семплінгу для глобального освітлення, такі як кількість семплів або розмір семплів.

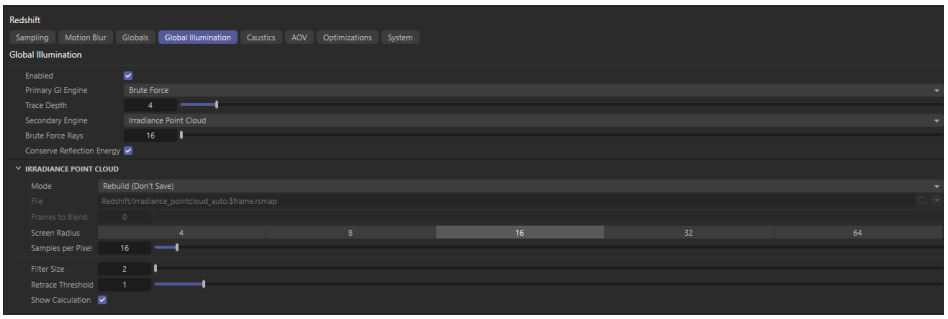

Рис. 3.5. Налаштування рендеру RedShift

2) Налаштування матеріалів (рис. 3.6):

– Material Type: Виберемо тип матеріалу, наприклад, стандартний, прозорий, металічний тощо;

– Diffuse: Налаштуємо параметри розсіяного кольору та інтенсивності;

– Reflection: Встановимо параметри відбиття (reflection) для матеріалу, такі як інтенсивність, глянцевість, колір тощо;

– Refraction: Налаштуємо параметри преломлення (refraction) для матеріалу, якщо він є прозорим;

– Bump Mapping: Додамо текстуру для створення рельєфу та деталей на поверхні матеріалу.

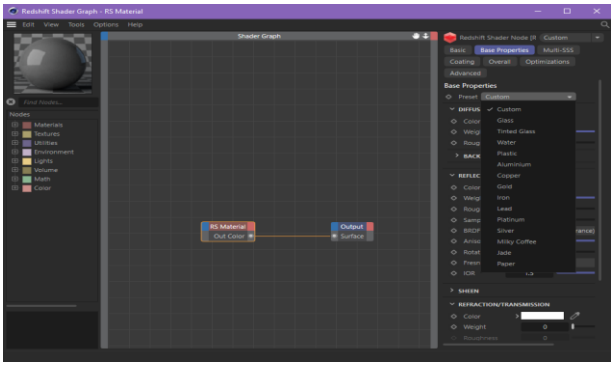

Рис. 3.6. Налаштування матеріалів в RedShift

3) Налаштування освітлення (рис. 3.7):

– Light Type: Виберемо тип світла, наприклад, точкове, напрямлене або площинні;

– Intensity: Встановимо інтенсивність світла;

– Color: Налаштуємо кольори світла;

– Shadows: Виберемо тип тіней та налаштуємо їхню якість.

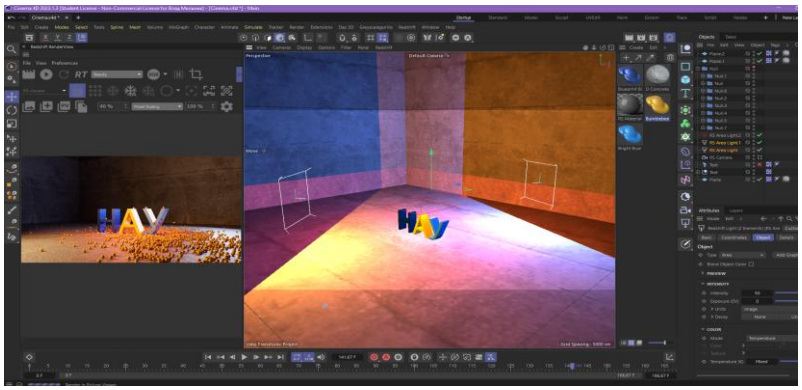

Рис. 3.7. Налаштування освітлення

4) Налаштування роздільної здатності (Resolution):

– Image Width/Height: Встановимо розмір вихідного зображення;

– Pixel Aspect Ratio: Налаштуємо співвідношення сторін пікселя.

5) Керування якістю та швидкістю рендерингу (рис. 3.8):

– Встановимо кількість семплів для кожного променя, що випускається;

– Встановимо максимальну глибину лучів, які використовуються для відображення взаємодій світла;

– Активуємо адаптивний семплінг для автоматичного розподілу семплів.

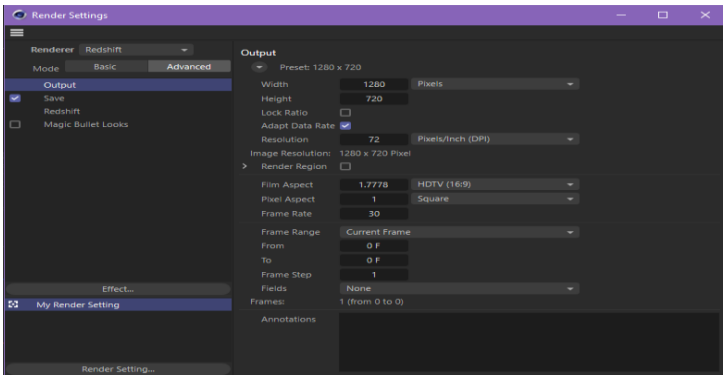

Рис. 3.8. Керування якістю рендеру

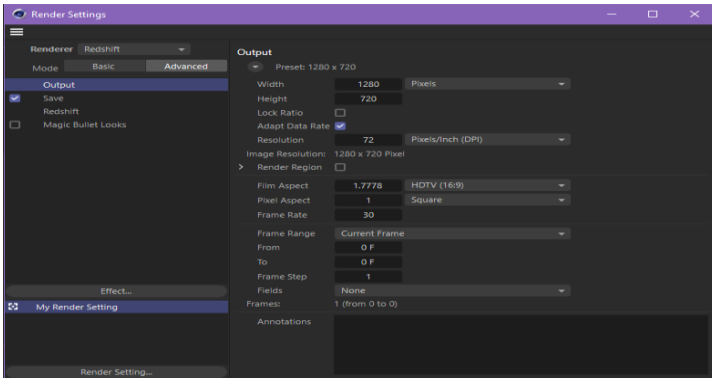

Рис. 3.8. Керування якістю рендеру

Це лише деякі загальні налаштування та параметри, які доступні у рендер-двигуні Redshift. За допомогою цих параметрів, ми може контролювати якість та вигляд нашого рендерингу у Cinema 4D.

3.7. Рендеринг та вихідний результат

В результаті рендерингу (рис. 3.9) нашої симуляції отримали наступний вихідний результат (рис. 3.10):

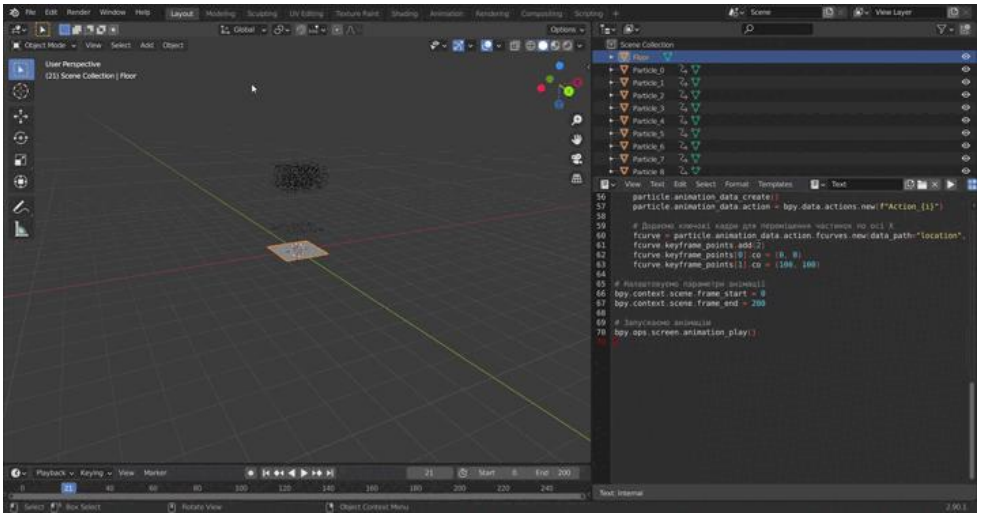

Рис. 3.9. Рендеринг

Рис. 3.10. Вихідний результат рендерингу

Застосування всіх наших попередніх кроків і налаштувань привело до створення реалістичної і захоплюючої анімації частинок.

ВИСНОВКИ ДО РОЗДІЛУ 3

У цьому розділі ми детально описали процес створення 3D симуляції частинок. Починаючи з першого кроку, ми вивчили, як створити базову структуру симуляції в Blender за допомогою програмного коду Python. Ми розглянули основні концепції динаміки частинок та використали їх для створення реалістичних рухів і взаємодій між частинками.

Після створення динаміки, ми перейшли до текстурингу та додавання матеріалів до наших частинок. Використовуючи можливості Blender, ми створили матеріали з різними властивостями, такими як кольори, відбивання світла та прозорість. Це додало більш реалістичний вигляд до нашої симуляції.

Далі ми перенесли нашу симуляцію з Blender до програми Cinema 4D для подальшого рендерингу. Ми експортували дані симуляції з Blender у форматі, сумісному з Cinema 4D, і імпортували їх у новий проект Cinema 4D. Тут ми знайомилися зі змінними параметрами, які можна налаштувати в Cinema 4D для досягнення бажаного вигляду симуляції.

Для досягнення оптимального результата ми провели налаштування рендеру в Cinema 4D. Ми налаштували глобальне освітлення, використали параметри семплінгу та контролювали глибину променів для отримання бажаної якості рендерингу. Додатково, ми звернули увагу на освітлення сцени, встановивши світла з розрахунком на найкращий вигляд симуляції.

Завершальним етапом було рендеринг нашої симуляції та отримання вихідного результату. Застосування всіх наших попередніх кроків і налаштувань привело до створення реалістичної і захоплюючої анімації частинок.

Усі ці кроки дозволили нам створити складну 3D симуляцію частинок з вражаючою динамікою, привабливим текстурингом та реалістичним виглядом завдяки рендерингу у Cinema 4D. Результатом нашої праці є відео або зображення, які відображають красу і реалістичність нашої симуляції частинок.

ВИСНОВКИ

Під час нашої роботи в програмі Blender з використанням програмного коду ми успішно виконали велику кількість завдань, пов'язаних з симуляцією динамічних 3D сцен. Наша робота охопила різні аспекти створення, редагування і візуалізації таких сцен.

Ми розглянули поняття симуляції динамічних 3D сцен та їх технології, а також еволюцію цих технологій. Використовуючи вузлові системи, ми змогли ефективно створювати складні симуляції, зокрема динаміку, текстуринг і взаємодію між об'єктами.

Ми описали процес створення вузлових систем і параметрів, що дозволяють нам контролювати і налаштовувати симуляції. Також ми розглянули переваги використання вузлових систем у симуляціях динамічних 3D сцен та проаналізували різні вузлові системи та їх можливості.

Результатом нашої роботи була реалізація розробленої технології на прикладі динамічної 3D сцени. Ми успішно створили першу частину симуляції, включаючи динаміку руху частинок та взаємодію між ними. Додатково, ми розробили текстури та матеріали, щоб надати нашим об'єктам більш реалістичний вигляд.

Для отримання кінцевого візуального результату ми перенесли нашу симуляцію до програмного забезпечення Cinema 4D для рендеру за допомогою двигуна Redshift. Ми виконали необхідні налаштування рендеру, зосереджуючись на параметрах семплінгу, глибині променів та освітленні сцени.

Після процесу рендерингу ми отримали вихідний результат - відео або зображення, які відображають красу і реалістичність нашої симуляції динамічних 3D сцен.

В цілому, наша робота у програмі Blender з використанням програмного коду була успішною і дозволила нам створити захоплюючу симуляцію, дослідити різні аспекти створення 3D сцен, а також досягнути високоякісного візуального результату.

Наша робота з симуляції динамічних 3D сцен з використанням вузлових систем у програмах Cinema 4D, Blender, Houdini і Unreal Engine виявляється вельми корисною і має широкий спектр можливих застосувань. Вона може бути використана для візуалізації та анімації, виробництва фільмів і відео, розробки відеоігор, віртуальної та доповненої реальності, навчання та симуляцій.

Наша робота дозволяє створювати захоплюючі візуальні ефекти, реалістичну фізику, спеціальні ефекти, а також іммерсивні досвіди для користувачів. Вона відкриває широкі можливості для творчості та вираження ідей у сфері 3D-візуалізації.

З урахуванням різноманітності можливих застосувань, наша робота може знайти використання у різних галузях, таких як розваги, медіа, виробництво, освіта та багато інших. Вона створює нові можливості для творчих проектів та сприяє розвитку інноваційних рішень у сфері симуляцій динамічних 3D сцен.

Наша робота є цінним інструментом для створення вражаючих 3D-сцен з реалістичною динамікою, що надає багато можливостей для творчості та вираження ідей у віртуальному просторі.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Isaac Kerlow. The Art of 3D Computer Animation and Effects. – Wiley; 4th Revised & enlarged edition. – 2019. – s. 112 – 122.
2. Jeremy Birn. Digital Lighting & Rendering. – New Riders Pub; 3rd edition. – 2020. – s. 234 – 250.
3. Frank Luna. Introduction to 3D Game Programming with DirectX 12. – Mercury Learning and Information; Illustrated edition. – 2018. – s. 612 – 635.
4. Jason Busby. Mastering Unreal Technology: Advanced Level Design Concepts With Unreal Engine 3. – Sams; 1st edition. – 2019. – s. 481 – 503.
5. [Електронний ресурс] Maxon.net – Cinema 4D. Режим доступу — https://www.maxon.net/en/cinema-4d.
6. [Електронний ресурс] Blender – Blender. Режим доступу — https://www.blender.org/.
7. [Електронний ресурс] Sidefx – Houdini. Режим доступу — https://www.sidefx.com/.
8. [Електронний ресурс] UnrealEngine – Unreal Engine. Режим доступу — https://www.unrealengine.com/en-US.
9. [Електронний ресурс] Udemy. Режим доступу — https://www.udemy.com/.
10. [Електронний ресурс] Linkedin – Lynda. Режим доступу — https://www.linkedin.com/learning/?trk=lynda_redirect_learning.
11. [Електронний ресурс] Pluralsight. Режим доступу — https://www.pluralsight.com/.
12. [Електронний ресурс] BlenderArtists. Режим доступу — https://blenderartists.org/.
13. [Електронний ресурс] UnrealEngine – Unreal Engine Forum. Режим доступу — https://forums.unrealengine.com/categories?tag=unreal-engine.
14. [Електронний ресурс] Scholar. Режим доступу — https://scholar.google.com/.

15. [Електронний ресурс] CG Society. Режим доступу — https://forums.cgsociety.org/.

16. [Електронний ресурс] SideFX — SideFX Forum. Режим доступу — https://www.sidefx.com/forum/.

17. [Електронний ресурс] ACM Digital Library. Режим доступу — https://dl.acm.org/.

18. [Електронний ресурс] IEEE Xplore Digital Library. Режим доступу — https://ieeexplore.ieee.org/Xplore/home.jsp.