Міністерство освіти і науки України
Тернопільський національний технічний університет імені Івана Пулюя

Факультет комп'ютерно-інформаційних систем і програмної інженерії
(повна назва факультету)
Кафедра комп'ютерних наук
(повна назва кафедри)

КВАЛІФІКАЦІЙНА РОБОТА

на здобуття освітнього ступеня

магістр
(назва освітнього ступеня)

на тему: Створення комбінованої багаторіневої нодової системи для накладання текстур матеріалів на Blender-об'єкти

Виконав: студент VI курсу, групи СНнм-61
спеціальності 122 Комп'ютерні науки
(шифр і назва спеціальності)

_____ Озіранець В.С.В.
(підпис) (прізвище та ініціали)

Керівник _____ Никитюк В.В.
(підпис) (прізвище та ініціали)

Нормоконтроль _____ Дуда О.М.
(підпис) (прізвище та ініціали)

Завідувач кафедри _____ Боднарчук І.О.
(підпис) (прізвище та ініціали)

Рецензент _____ Михалик Д.М.
(підпис) (прізвище та ініціали)

Тернопіль
2024

Міністерство освіти і науки України
Тернопільський національний технічний університет імені Івана Пулюя

Факультет комп'ютерно-інформаційних систем і програмної інженерії
(повна назва факультету)

Кафедра комп'ютерних наук
(повна назва кафедри)

ЗАТВЕРДЖУЮ
Завідувач кафедри
_____ Боднарчук І.О.
(підпис) (прізвище та ініціали)
« 13 » травня 2024 р.

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ

на здобуття освітнього ступеня Магістр
(назва освітнього ступеня)

за спеціальністю 122 Комп'ютерні науки
(шифр і назва спеціальності)

Студенту Озіранцю Віталію Степанові Володимировичу
(прізвище, ім'я, по батькові)

1. Тема роботи Створення комбінованої багаторіневої нодової системи для накладання текстур матеріалів на Blender-об'єкти

Керівник роботи Никитюк Вячеслав Вячеславович, к.т.н., доцент кафедри КН
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)

Затверджені наказом ректора від « 24 » листопада 2023 року № 4/7-1100

2. Термін подання студентом завершеної роботи «13» травня 2024р.

3. Вихідні дані до роботи Наукові публікації про системи накладання матеріалів у комп'ютерній графіці, веб-ресурси з оглядами та аналізом різних систем, документація до Blender, Blender Python API та інших програм для комп'ютерної графіки.

4. Зміст роботи (перелік питань, які потрібно розробити)
Вступ. 1 Аналіз сфери 3D графіки для проведення дослідження. 2. Дослідження гібридного підходу для створення матеріалу. 3. Обчислювальний експеримент ефективності розробленого з Blender API гібридного підходу. 4. Охорона праці та безпека в надзвичайних ситуаціях. Висновки. Перелік джерел. Додатки

5. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень, слайдів)
1 Титульна сторінка. 2 Тема, Мета, Об'єкт, Предмет дослідження. 3 Завдання дослідження. 4 Актуальність дослідження в комп'ютерній графіці. 5 Програмне забезпечення Blender 3D. 6. Структура сучасного PBR матеріалу. 7 Система вузлів. 8. Система рівнів. 9 Гібридний підхід. 10. Підходи реалізації системи. 11. Середовище розробки. 12. Реалізація системи. 13. Умови експерименту. 14. Результати експериментів. 15. Дослідження ефективності. 16 Висновки. 17 Завершальний слайд.

6. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
Охорона праці	Сенчишин В.С., доцент		
Безпека в надзвичайних ситуаціях	Клепчик В.М., ст. викладач		

7. Дата видачі завдання 24 листопада 2023 р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів роботи	Термін виконання етапів роботи	Примітка
1.	Ознайомлення з завданням до кваліфікаційної роботи	25.11.2023	*Виконано*
2.	Підбір наукових джерел про комп'ютерну графіку,	27.11.2023-01.12.2023	*Виконано*
	тривимірну графіку та програмне забезпечення		
	Blender, про системи накладання матеріалів.		
3.	Опрацювання наукових публікацій та збір даних по	04.12.2023-12.01.2024	*Виконано*
	системах накладання матеріалів та їх розробці.		
4.	Виконання дослідження згідно ефективності	15.01.2024-16.02.2024	*Виконано*
	гібридної системи накладання матеріалів.		
5.	Оформлення розділу «Аналіз сфери 3D графіки	19.02.2024-1.03.2024	*Виконано*
	для проведення дослідження»		
6.	Оформлення розділу «Дослідження гібридного	04.03.2024-15.03.2024	*Виконано*
	Підходу для створення матеріалу»		
7.	Оформлення розділу «Обчислювальний експеримент	18.03.2024-05.04.2024	*Виконано*
	ефективності розробленого з Blender API гібридного		
	підходу»		
8.	Виконання завдання до підрозділу «Охорона праці»	08.04.2024-12.04.2024	*Виконано*
9.	Виконання завдання до підрозділу «Безпека в	15.04.2024–19.04.2024	*Виконано*
	надзвичайних ситуаціях»		
10.	Оформлення кваліфікаційної роботи	22.04.2024-26.04.2024	*Виконано*
11.	Нормоконтроль	29.04.2024-03.05.2024	*Виконано*
12.	Перевірка на плагіат	03.05.2024	*Виконано*
13.	Попередній захист кваліфікаційної роботи	05.05.2024	*Виконано*
14.	Захист кваліфікаційної роботи	29.05.2024	

Студент _____ Озіранець В.С.В.
(підпис) (прізвище та ініціали)

Керівник роботи _____ Никитюк В. В.
(підпис) (прізвище та ініціали)

АНОТАЦІЯ

Створення комбінованої багаторівневої ходової системи для накладання текстур матеріалів на Blender-об'єкти. // Кваліфікаційна робота освітнього рівня «Магістр» // Озіранець Віталій Степан Володимирович // Тернопільський національний технічний університет імені Івана Пулюя, факультет комп'ютерно-інформаційних систем і програмної інженерії, кафедра комп'ютерних наук, група СНнм-61 // Тернопіль, 2024 // С. 74, рис. – 29, табл. – 0, кресл. – 17, додат. – 4, бібліогр. – 50.

Ключові слова: комп'ютерна графіка, python, blender, розробка системи, матеріал, тривимірна графіка, системи вузлів та шарів, метод аналізу ієрархій.

Мета кваліфікаційної роботи полягає у підвищенні ефективності розробки матеріалів у тривимірній графіці і присвячена власне розробці гібридної системи накладання матеріалів з використанням Blender Python API.

В першому розділі кваліфікаційної роботи описані галузь комп'ютерної графіки та актуальність досліджень у ній, після чого розглянуто програмне забезпечення для роботи з тривимірною графікою для проведення експерименту. Також висвітлено переваги та недоліки кожного з перелічених програмних продуктів. Розглянуто окремо програмне забезпечення Blender.

У другому розділі кваліфікаційної роботи описано системи формування матеріалів, досліджено їх переваги та недоліки, після чого подано інформацію про гібридну систему та її релевантність.

В третьому розділі кваліфікаційної роботи описано розробку доповнення до Blender на основі API, проаналізовано труднощі розробки та проведено експеримент з аналізу ефективності гібридної системі.

Об'єктом дослідження виступають процеси розробки та накладання матеріалів на тривимірний об'єкт з використанням гібридного підходу.

Предметом дослідження є методи накладання матеріалів на тривимірні об'єкти.

ANNOTATION

Creation of a combined multi-level running system for the imposition of textures of materials on Blender objects. // The educational level "Master" qualification work // Oziranets Vitalii Stepan // Ternopil Ivan Pulyuy National Technical University, Faculty of Computer Information Systems and Software Engineering, Department of Computer Science, SNnm-61 group // Ternopil, 2024 // P. 74, fig. - 29, tables - 0, posters - 17, annexes - 4, ref. – 50.

Key words: computer graphics, python, blender, system development, material, 3d graphics, node and layer systems, analytic hierarchy process.

The purpose of the thesis is to increase the efficiency of material development in three-dimensional graphics and is devoted to the development of a hybrid material blending system using the Blender Python API. The first chapter of the thesis describes the field of computer graphics and the relevance of research in it, followed by a discussion of the software for working with the three-dimensional graphics for the experiment. The advantages and disadvantages of each of these software products are also highlighted. The Blender software and its functionality are considered separately.

The second chapter of the qualification work describes the systems of material formation, examines their advantages and disadvantages, and then provides information about the hybrid system and its relevance.

The third section of the thesis describes the development of an API-based Blender add-on, analyzes the development difficulties, and conducts an experiment to analyze the effectiveness of the hybrid system.

The object of research is the processes of developing and applying materials to a three-dimensional object using a hybrid approach.

The subject of the study is the methods of applying materials to three-dimensional objects.

ПЕРЕЛІК СКОРОЧЕНЬ І ТЕРМІНІВ

3D (в контексті комп’ютерної графіки та даної роботи, від англ. three dimensions – три виміри) – тривимірний простір.

API (скорочення, від англ. application programming interface – прикладний програмний інтерфейс) – набір програм, протоколів взаємодії та інструментів для створення програмного забезпечення.

UV-розгортка – це є процес в 3D моделюванні, при якому двовимірне зображення з координатами (U, V) накладається на об’єкт з координатами (X, Y, Z).

ЕОМ – скорочення, що означає електронно-обчислювальна машина. Комп’ютер є цифровою ЕОМ.

ІТ – скорочення, що означає інформаційні технології. Це набір методів, засобів та інструментів використання ЕОМ для створення та редагування інформації.

Карта або від англ. Map – термін в 3D графіці, який означає проекцію двовимірного зображення на 3D об’єкт.

Полігон – термін в тривимірній графіці, який означає площину в 3D з мінімальною кількістю вершин рівною трьом.

СУОП – скорочення, що означає системи управління охороною праці.

Текстура – двовимірне зображення, яке проєктується на поверхню 3D об’єкта з використанням UV-розгортки.

Технологія AR (від англ. augmented reality – доповнена реальність) – технологія або сукупність засобів в ІТ індустрії, суть якої полягає в додаванні віртуальних об’єктів та елементів до реальності.

Технологія VR (від англ. virtual reality – віртуальна реальність) – технологія або сукупність засобів, суть якої полягає в перенесенні користувача у цифровий світ. Іншими словами створення ілюзії іншої реальності на основі відчуттів з використанням ЕОМ.

ЗМІСТ

ВСТУП

Актуальність теми. З розвитком ІТ росте все більше різноманітних сфер пересікаються з комп’ютерною графікою, наприклад для демонстрації продуктів. Однією з проміжних галузей є тривимірна графіка, яка на даний момент все більше впливає на реальне життя, включаючи її використання у VR та AR. Кожен тривимірний об’єкт вимагає повноцінного виконання кожного з пунктів для отримання бажаного результату, що вимагає часу. Якщо ж спробувати оптимізувати хоча б один крок, наприклад спростивши його або зробивши його уніфікованим та ефективним, наприклад накладання матеріалів, то можна застосовувати автоматизацію, і відповідно навіть підключати ШІ для роботи. Тому розробка гібридної системи накладання матеріалів на основі поєднання існуючих традиційних методів на основі вузлів та шарів є актуальним напрямком дослідження в галузі тривимірної комп’ютерної графіки.

Мета і задачі дослідження. Метою даної кваліфікаційної роботи освітнього рівня «Магістр» є покращення ефективності процесу розробки матеріалу. Для досягнення поставленої мети потрібно виконати ряд завдань, зокрема:

– Проаналізувати стан досліджень в області комп’ютерної графіці, а конкретно сефру тривимірної графіки.

– Дослідити існуючі на даний час методи або ж системи накладання матеріалів, виділити їх переваги та недоліки.

– Проаналізувати методи покращення або об’єднання традиційних систем з утворенням гібридного методу.

– Розробити робоче доповнення до Blender, яке знадобиться для проведення експериментів щодо релевантсноті гібридного підходу до створення матеріалів.

– Описати труднощі реалізації та оптимізації доповення, проаналізувавши проблеми що виникли та способи їх покращення.

– Виконати порівняння існуючих рішень з новим гібридним підходом з використанням методу аналізу ієрархій, виділивши основні критерії.

Об’єкт дослідження. Процеси розробки та накладання матеріалів на тривимірний об’єкт з використанням гібридного підходу.

Предмет дослідження. Методи накладання матеріалів на тривимірні об’єкти.

Наукова новизна одержаних результатів кваліфікаційної роботи полягає у тому, що отримав подальший розвиток системи накладання матеріалів, поєднуючи традиційний метод з використанням шарів та систему з вузлами.

Практичне значення одержаних результатів. Виконано порівняльний аналіз з використанням методу Сааті для ефективності гібридного підходу накладання матеріалів. Частково розроблено реалізацію гібридного підходу.

Апробація результатів магістерської роботи. Основні результати проведених досліджень обговорювались на VI Міжнародній студентській науково-технічній конференцій "Природничі та гуманітарні науки. Актуальні питання" Тернопільського національного технічного університету імені Івана Пулюя (м. Тернопіль, 2023 р.), на II міжнародній науковій конференції молодих учених та студентів “Філософські виміри техніки” Тернопільського національного технічного університету імені Івана Пулюя (м. Тернопіль, 2019р.) та на VII Міжнародній студентській науково-технічній конференцій "Природничі та гуманітарні науки. Актуальні питання" Тернопільського національного технічного університету імені Івана Пулюя (м. Тернопіль, 2024р.)

Публікації. Основні результати кваліфікаційної роботи опубліковано у чотирьох працях конференції (Див. додатки А, Б, В, Д).

Структура й обсяг кваліфікаційної роботи. Кваліфікаційна робота складається зі вступу, чотирьох розділів, висновків, списку літератури з 50 найменувань та 4 додатків. Загальний обсяг кваліфікаційної роботи складає 74 сторінки, з них 51 сторінка основного тексту, який містить 28 рисунків.

1 АНАЛІЗ СФЕРИ 3D ГРАФІКИ ДЛЯ ПРОВЕДЕННЯ ДОСЛІДЖЕННЯ

1.1 Аналіз галузі комп’ютерної графіки

Комп’ютерна графіка – це насамперед галузь знань, яка включає в себе створення та обробку графічного матеріалу, застосовуючи електронно-обчислювальні пристрої або ж ЕОМ [1]. Будь-яке зображення [2], будь-то простий відрізок в графічному редакторі Paint чи електронна картина пейзажу тощо, технології AR та VR [3]– це все приклади застосування комп’ютерної графіки (див. рисунок 1.1).

Рис. 1.1 – Графічне представлення частини сфер діяльності, в яких застосовується комп’ютерна графіка

Ця ІТ галузь поділяється за декількома різними категоріями, включаючи за конкретною сферою застосування, за методом відображення, за типом взаємодії та за характером стилізації. Почнімо з сфери застосування, так як це найбільш прямолінійна категорія, що включає в себе загальний перелік галузей та приклади використання, який виглядає наступним чином:

– Графіка у сфері науки.
– Ділова або ж корпоративна графіка.
– Конструкторська графіка.
– Художня графіка.

– Рекламна або ж ілюстративна графіка.

– Комп’ютерна анімація.

Прикладом використання у науковій сфері виступають програмні візуалізації та симуляції фізичних процесів, того ж розщеплення атому, не забуваючи про візуалізацію поверхонь, експериментів тощо. Додатково можна додати також і розробку дизайну для того чи іншого наукового відкриття або його представлення, яке у свою чергу дозволить краще презентувати вже людям або компанії, яка готова придбати патент на використання цього винаходу. Прямим прикладом використання наукового представлення є бібліотека matplotlib [4], яка застосовується при побудові графіків на основі заданих користувачем параметрів з використанням мови програмування Python (див. рисунок 1.2).

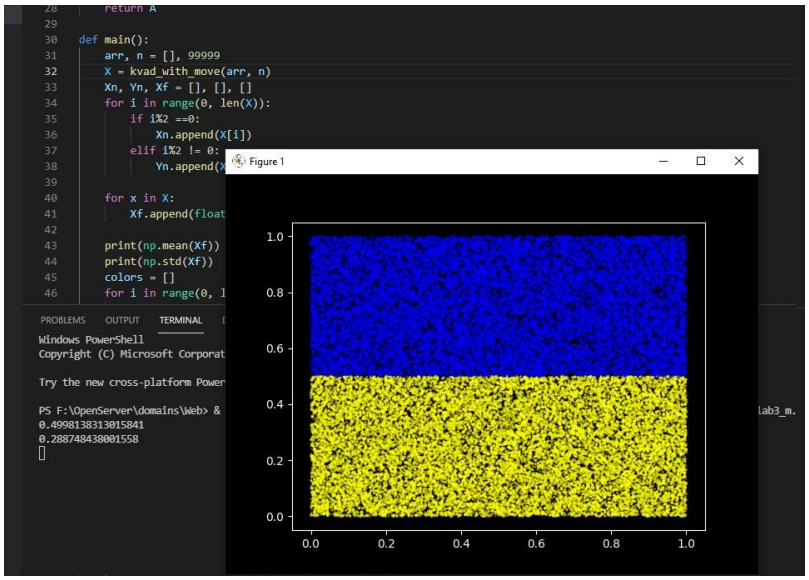

Рис. 1.2 – Приклад застосування комп’ютерної графіки в науковій сфері на основі генерації базового білого шуму

Різниця ж між науковою та ілюстративною графікою полягає у тому, що остання включає в себе більш рекламну або ж ознайомлювальну функцію. До ілюстративної графіки відносяться як банери, листівки, так і графіки, креслення тощо.

У свою чергу недалеко від даних категорій знаходиться ділова графіка, що включає використання різноманітних графів та графіків для подачі інформації з подальшим аналізом останньої для розробки стратегій збуту товару та планів подальших дій компанії тощо. Додатковим представлення даного типу є використання електронних таблиць як наприклад в Microsoft Excel (див. рисунок 1.3).

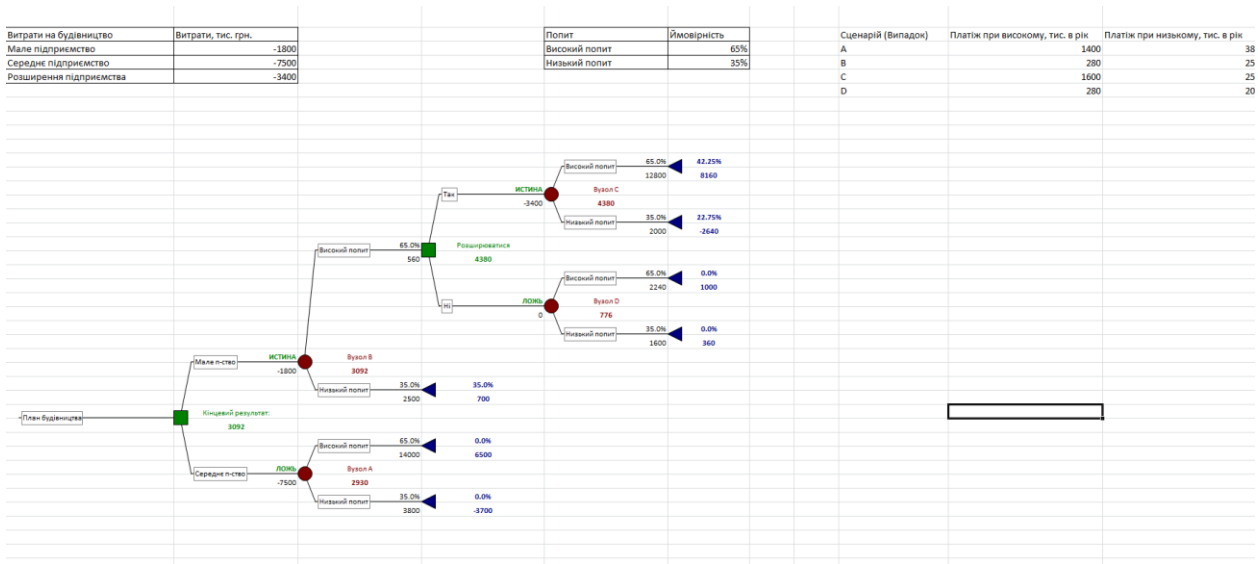

Рис. 1.3 – Приклад застосування комп’ютерної графіки на прикладі побудови дерева рішень

Конструкторська графіка використовується для планування та візуалізації не тільки будинків та інтер’єрів як може здатися на перший погляд, але й в цілому при дизайні нових виробів, включаючи ювелірні вироби або навіть проєкти для тривимірного друку, такі як фігурки або в стоматології навіть зуби, кістки. Прикладом програм для роботи з конструкторською графікою є AutoCAD, Revit, SketchUp, Blender [5] тощо.

Останніми та не менш важливими є галузі анімації [2] та художня графіка. Комп’ютерна анімація виокремлюється з усіх перекислених, так як являє собою потік зображень, що змінюють один одного з часом і фактично ця галузь може бути залучена і з іншими при виконанні поставленої задачі. Так наприклад при розробці рекламної кампанії на розгляд висуваються не тільки класичні паперові варіанти, але й доцільність створення рекламного ролику або

презентації для більш оптимального способу донести вадливість товару для користувача. Якщо ж додати анімацію до наукової сфери, то можна окрім як виводити результат симуляції показати власне процес, що теж може бути не менш цікавим. Існують цілі Інтернет ресурси, які окрім як подачі інформації займаються ще й приємним візуальним оформленням використанням інформації, так як візуальна подача буде приємним доповненням до слухової, які не мають сильної різниці при впливі на запам’ятовування об’єкта [6]. Програмне забезпечення для анімації – Blender, Maya, Adobe Premier Pro, так як візуальні ефекти теж є частиною комп’ютерної анімації, Animation Desk тощо. На рисунку 1.4 зображено приклад використання комп’ютерної анімації.

Рис. 1.4 – Приклад застосування комп’ютерної графіки для створення анімації логотипу кафедри комп’ютерних наук

До художньої сфери відноситься створення ілюстрацій, які стараються розповісти певну історії або викликати відчуття від простого погляду на них. Сюди входять як створені художниками зображення, так і повноцінні мультфільми, серіали, фільми тощо.

За методом відображення комп’ютерна поділяється на такі під категорії:

– Растрова графіка – побудова на основі невеличких точок названих пікселями [2], кожен із яких відображає колір на основі однієї з основних систем, будь-то RGB, HSV, HEX та інших. Вона є частиною двовимірної

комп’ютерної графіки і також одним із результатів візуалізації у тривимірній графіці.

– Векторна графіка – побудована на основі примітивних геометричних фігур [2], будь-то точка, крива, пряма, вектор, відрізок тощо, які формують чітке зображення. Порівняно з растровою графікою дозволяє не втрачати якість при зміні розмірів об’єкта або зображення, але є складнішою для внесення змін до готової роботи. В цілому використовується для логотипів компаній, рекламних банерів. Також є частиною галузі двовимірної комп’ютерної графіки. Приклад двовимірної графіки зображено на рисунку 1.5.

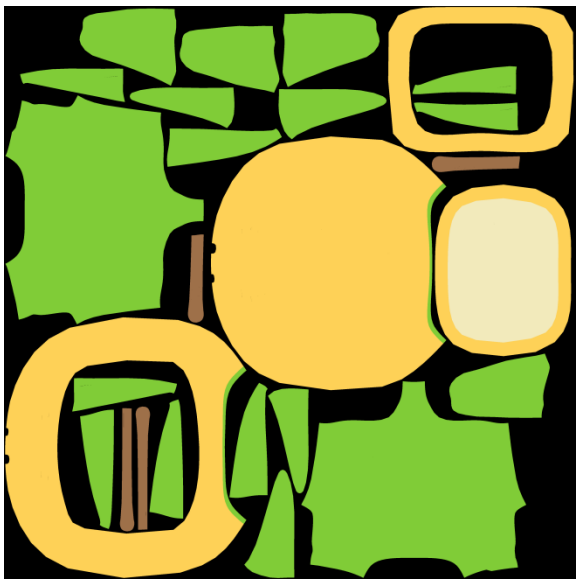

Рис. 1.5 – Використання растрової двовимірної комп’ютерної графіки для створення текстур

– Графіка з використанням фракталів – доволі рідкісний тип відображення, оснований на фракталах, які представляють собою “геометричні фігури та візерунки, які можуть описати шорсткість (або нерівність), присутню майже в кожному об’єкті природи. Багато фракталів можуть повторювати свою геометрію в менших або більших масштабах”[7]. В основному використовується для процедурної генерації на ЕОМ різних реалістичних матеріалів або візерунків (див. рисунок 1.6).

Рис. 1.6 – Використання фракталів для створення зображення підводної фауни

– Тривимірна графіка, яка займає особливе місце в IT індустрії, являє собою динамічне поле, де інновації постійно змінюють парадигми та визначають нові стандарти якості. Загальна концепція тривимірної графіки включає в себе моделювання, текстурування, анімацію, освітлення та рендеринг на ЕОМ, що у поєднанні створюють ілюзію реалістичного світу.

Кожен з цих аспектів [8] відіграє свою унікальну роль, доповнюючи та покращуючи загальний візуальний досвід користувача. Моделювання, як основна складова [9], дозволяє створювати форми та структури об'єктів, а текстурування додає реалізму шляхом застосування карт та матеріалів, або дозволяє стилізацію через малювання на тривимірному об’єкті з використанням UV-розгортки [10]. Анімація, у свою чергу, оживляє статичні моделі, роблячи їх життєвими та динамічними [2]. Освітлення впливає на атмосферу сцени, створюючи тіні, відблиски та настрій, а рендеринг конвертує тривимірні об'єкти в двовимірні зображення з відмінною якістю. Такий комплексний підхід до тривимірної графіки відкриває безліч можливостей для творчого вираження та інноваційних рішень у різних сферах, починаючи від розваг і закінчуючи науковими дослідженнями і професійними додатками. Приклад застосування такої графіки показано на рисунку 1.7.

Рис. 1.7 – Приклад персонажа, створеного з використанням тривимірної графіки

За характером стилізації комп’ютерну графіку можна поділити на дві основі категорії – реалістична та відповідно стилізована. Реалістична графіка включає в себе як архітектурну візуалізацію з зазначеною точністю, так і представлення продукту в промо-ролику. Цей вид графіки вимагає неабияких знань користувача у галузях фізики та математики, не кажучи вже про знання архітектури, анатомії тощо, і це лише для того, щоб розробити модель [1], а ще ж треба розробити матеріали, бажано у форматі PBR [11], щоб вони могли відповідати напряму фізичним властивостям реально-існуючих матеріалів, а не старим стандартам з псевдо-матеріалами [12], налаштувати освітлення відповідно до логічного розміщення, так як не існує невидимих джерел світла, фізику, якщо така представлена в проєкті, що включає не тільки базові відношення такі як гравітація чи взаємодія двох об’єктів, будь-то зіткнення чи відштовхування, а ще й рух води, симуляцію певного затуманення тощо. На рисунку 1.8 показано приклад розробки ванної кімнати у реалістичному стилі.

Рис. 1.8 – Приклад майже реалістичної візуалізації

Як можна побачити на рисунку, частина об’єктів сцени не відповідає реалізму. Так, якщо глянути на дзеркало, то можна побачити у верхній частині якийсь невідомий білий об’єкт, який насправді являє собою точку освітлення лампи, збільшену в декілька разів для створення невеликого контрасту та освітлення в сцені. У свою чергу, стилізована графіка дозволяє упустити деякі деталі. Остання категорія на розгляд – тип взаємодії, під яким розуміють динаміку в графіці, що самим собою означає поділ на два основних види – динамічний та статичний. Різниця між ними лише в зміні зображення з часом.

Усе це необхідно було для конкретизації області, в якій проводитиметься дослідження, яка звучить наступним чином: дослідження проводитиметься в області статичної та динамічної, стилізованої та реалістичної тривимірної комп’ютерної графіки для усіх видів діяльності.

1.2 Огляд програмного забезпечення для проведення дослідження

Обравши область, у якій буде проведено дослідження наступним логічним кроком буде оглянути програмне забезпечення, в якому проводитимуться експерименти. Всього є п’ять програм для проведення саме цього дослідження, кожна з яких має власні переваги та недоліки, які необхідно розглянути для вибору найоптимальнішого:

– Autodesk (3ds Max, Maya).

– Cinema4D.

– ZBrush.

– Blender.

3ds Max, вироблений компанією Autodesk, є визнаним лідером у сфері тривимірної графіки [1] та архітектурної візуалізації і виділяється надзвичайною потужністю та розмаїттям функцій, що робить його улюбленим інструментом для багатьох професіоналів у цій галузі, надаючи величезний набір інструментів для створення складних тривимірних об'єктів, сцен та анімацій [13]. Крім того, програма має дуже широкий функціонал, і для повного оволодіння всіма його можливостями може знадобитися тривалий час і практика.

Інтерфейс програми організований хоч і разом з тим засмічений функціями та меню, які можна виконувати чи знайти у інших панелях, не кажучи про виділення окремої кнопки під кожну з них, що зменшує робочу область. Варто ще відзначити, що використання 3ds Max може бути також фінансово витратним та має високі системні вимоги, і для роботи з великими проєктами може знадобитися потужний комп'ютер. На рисунку 1.9 зображено інтерфейс 3ds Max, взятий з документації [13].

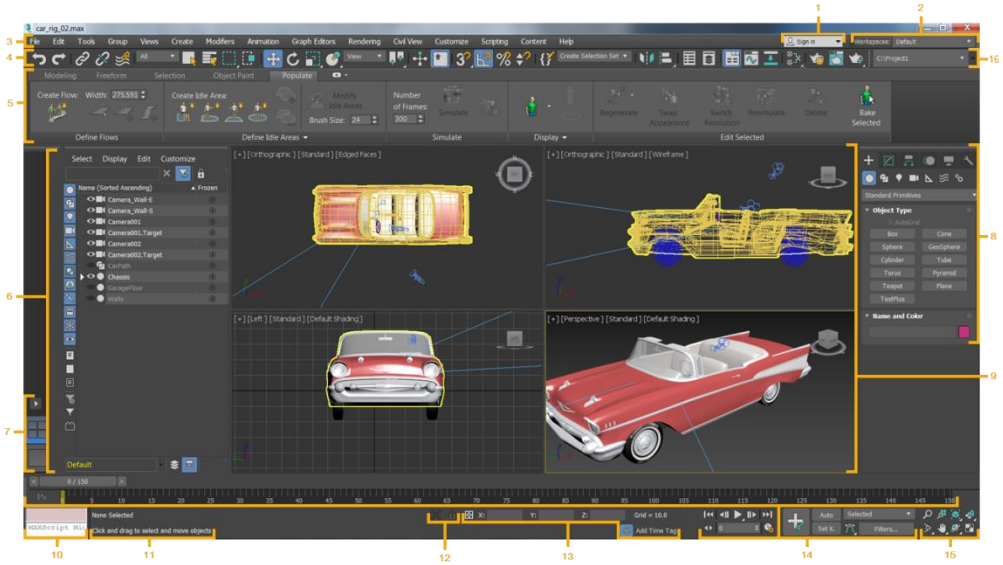

Рис. 1.9 – Інтерфейс програми 3ds Max

У свою чергу програма Maya відома своєю гнучкістю та потужністю у створенні складних анімаційних ефектів. Фактично це 3Ds Max в сфері тривимірною анімації [1]. Основна область застосування програми Maya полягає в індустрії тривимірної анімації, візуальних ефектів та графічного дизайну, включаючи моушн-дизайн [14]. Також Maya використовується в архітектурній візуалізації, для створення тривимірних моделей будівель та просторів, щоб допомогти дизайнерам візуалізувати свої ідеї перед тим, як реалізувати їх у життя. та у науково-дослідних областях, таких як медицина та біологія, для створення візуалізацій та анімацій для показу різних процесів та явищ.

Програма Cinema 4D – це вже комплексне програмне рішення для 3D-моделювання та візуалізації, розроблене Maxon [15]. Він широко відомий своїм потужним набором інструментів і гнучкістю, особливо в моушн-дизайні (див. рисунок 1.10).

Рис. 1.10 – Приклад моуш-дизайну в Cinema 4D

Що стосується реалізації матеріальної системи, Cinema 4D пропонував стандартну систему матеріалів, яка використовує підхід на основі шарів. Ця система нагадує традиційне програмне забезпечення для редагування двовимірних зображень, де користувачі накладають шари текстур, шейдерів і ефектів для створення складних матеріалів. У нових версіях [15] даного

програмного забезпечення вже присутній редактор матеріалів на основі вузлів, який пропонує більш гнучкий та інтуїтивно зрозумілий підхід до створення матеріалу.

Також важливо буде поговорити ще й про ZBrush [16], програму для цифрового скульптингу, що став галузевим стандартним інструментом у таких сферах, як кіно, відеоігри та цифрове мистецтво, завдяки своєму потужному набору функцій та зрозумілому і близькому багатьом художникам робочому процесу (див. рисунок 1.11).

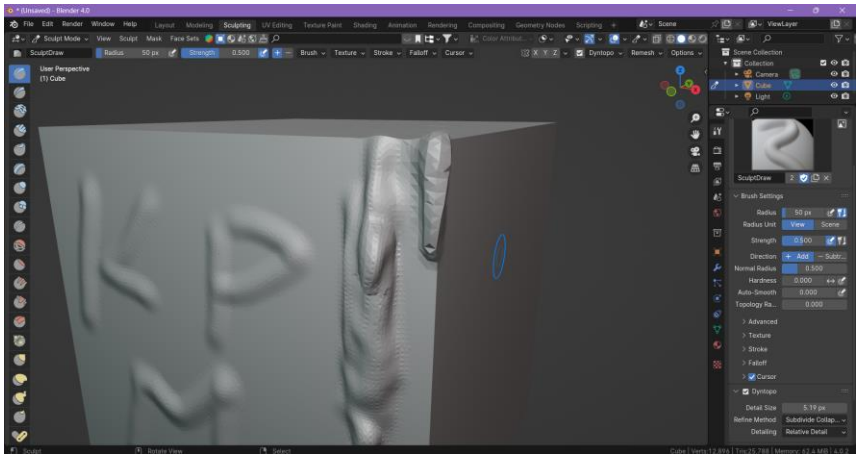

Рис. 1.11 – Приклад цифрової ліпки

Основою ZBrush є його функція цифрового ліплення, яка дозволяє художникам ліпити віртуальну глину за допомогою пензлів, які імітують інструменти в реальному світі. Ці пензлі пропонують широкий спектр ефектів, від ліплення дрібних деталей до вирізання складних форм. Унікальна технологія ZBrush, яка називається DynaMesh [16], дозволяє редагувати поверхні моделі, дозволяючи художникам додавати та видаляти геометрію на льоту, не турбуючись про обмеження топології.

Окрім ліплення, ZBrush надає потужні інструменти малювання для додавання кольору та текстури моделям безпосередньо у вікні 3D. Художники можуть малювати безпосередньо на поверхні моделей, використовуючи різні пензлі та режими малювання. ZBrush також підтримує поліфарбування, що дозволяє малювати безпосередньо на вершинах моделі, хоч і дуже сильно

залежить від кількості останніх, забезпечуючи неруйнівний спосіб додавання кольору та деталей.

Залишається Blender – програмне забезпечення з відкритим кодом для створення 3D-графіки з широким спектром функціоналу, включаючи моделювання, анімацію скелетів, рендеринг, композицію та захоплення руху [1], інтерфейс якого подано на рисунку 1.10. Крім цього, воно має вбудований відеоредактор та потужний фізичний двигун [5]. Останнє оновлення двигуна зробило програму більш оптимізованою для роботи з невеликою кількістю полігонів.

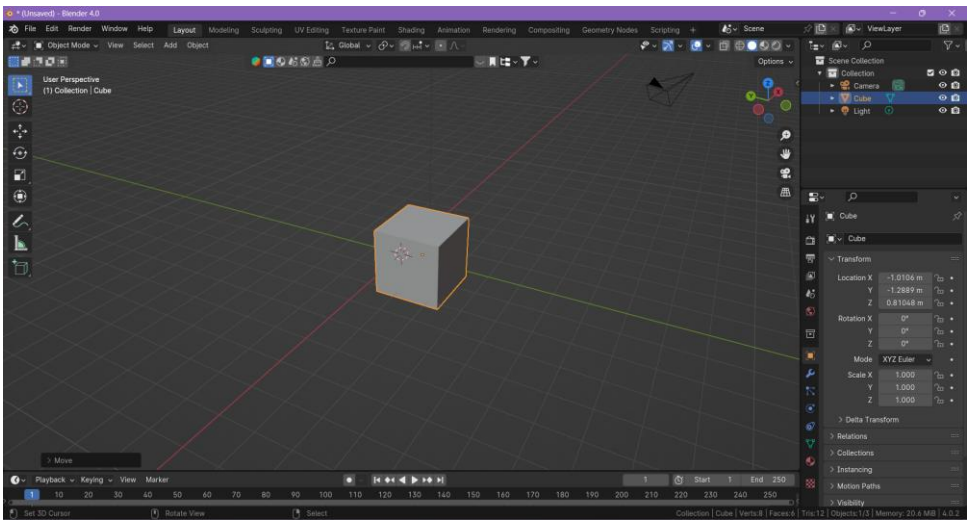

Рис. 1.12 – Інтерфейс програми Blender

Програма Blender фактично об'єднує можливості кількох редакторів, таких як 3Ds Max та ZBrush, в одному програмному забезпеченні, при цьому маючи відкритий код і будучи безкоштовною.

1.3 Опис основних текстур та параметрів матеріалу в Blender

Матеріал насамперед представляє собою підсистему, що з точки зору програми розуміється як скомпонований шейдер [2], в якому прописана уся необхідна інформація для двигуна, коли як для користувача ця структура виглядає як комбінації кількох властивостей [17], які потім просто проходять через шейдер. До основних властивостей відносять [11]:

– Колір (Diffuse, Albedo, Base Color Map) – це текстурна карта або параметр, що визначає кольори поверхні без урахування світла, що попадає на неї. Основна функція – відображенням кольорів матеріалу, задається в колірних просторах RGB, HSV, HEX тощо.

– Визначення властивості металевості (Metallic Map) – це текстура або параметр від 0 до 1, який вказує, яка частина поверхні або в цілому чи поверхня є металевою, а які ні. В основному чорний колір, який відповідає нулю – це неметал, коли як білий та 1 – метал.

– Шорсткість поверхні (Roughness Map) – текстура чи параметр від 0 до 1, що вказує на ступінь розсіювання або заломлення світла на поверхні, фактично відтворює нерівності на поверхні. Білий – шорстка поверхня, чорний – гладка.

– Карта випромінювання світла (Emissive, Emission Map) – текстура, що дозволяє поверхні матеріалу виділяти або ж випромінювати світло, незалежно від навколишнього оточування та освітлення. В цілому складається з двох параметрів – кольору та сили випромінювання.

– Карта затінення (AO Map, Ambient Occlusion Map) – це текстурна карта, яка відповідає за те, які саме частини повинні бути затіненими через обмежене світло до них.

– Карта нормалей (Normal Map) – це карта, що використовується для створення деталей та рельєфу на поверхні без зміни власне поверхні та її геометрії [18]. Фактично імітує випуклості та опуклості, різноманітні подряпини тощо.

Крім основних властивостей також часто використовують і додаткові для отримання того чи іншого матеріалу, або покращення вже існуючого. До цих властивостей відносять наступні:

– Карта висот (Height, Bump Map), яка змінює нормалі об'єкта, створюючи ілюзію висоти поверхні об'єму [18], не змінюючи форми та геометрії. Також, як і карти металевості має два основних значення, при яких або змінюється висота, або не вноситься ніяких змін.

– Карта прозорості (Opacity, Alpha Map), суть якої полягає у описі двигуну програми, які з частин поверхні необхідно зробити прозорими. Дуже часто застосовується у розробці матеріалів листя та скла.

– Карта під поверхневого розсіювання (Subsurface Scattering Map) – це карта, яка використовується для імітації просвічування світла крізь такі поверхні, як шкіра та віск. Фактично показує, як саме світло повинне розсіюватися всередині об’єкту.

– Карта зміщень або ж карта розміщення (Displacement Map) – це текстура, що використовується для заміни геометрії моделі на основі математичних значень, закодованих власне у ній. З використанням модифікатору Displacement карта дозволяє повністю змінити реальну форму поверхні.

– Карта проходження світла (Transmission Map), що вказує на пропускання світла через матеріал, такий як скло або прозорі текстильні матеріали. Вона використовується для імітації заломлення світла всередині поверхні.

– Карта відблисків (Specular Map), суть якої полягає у відображенні яскравості відблисків світла на поверхні об’єкта. В цілому застосовується для демонстрації чистоти об’єкта, по типу пилу тощо.

1.4 Висновок до першого розділу

Дослідження у тривимірній графіці є актуальними завдяки постійному розвитку технологій та появі нових методів інтерактивної візуалізації, а розробка більш реалістичних та ефективних алгоритмів відображення світла, обробки текстур та інших аспектів графіки постійно привертає увагу людей у галузі графіки, та навіть розважального контенту. У свою чергу Blender завдяки активному ком'юніті [19], а також відкритому коду дозволяє реалізувати будь-яке дослідження, що дозволяє розробляти та вдосконалювати інструменти для створення графічного контенту.

2 ДОСЛІДЖЕННЯ ГІБРИДНОГО ПІДХОДУ СТВОРЕННЯ МАТЕРІАЛУ

2.1 Традиційна система вузлів: переваги та недоліки

Системи матеріалів на основі вузлів [20] стали візитною карткою сучасного програмного забезпечення для тривимірної графіки завдяки гнучкості у керуванні, неруйнівному робочому процесі, візуальному представленні та модульності.

Гнучке керування дозволяє створювати складні мережі взаємопов'язаних елементів, що відповідають за найменші фізичні властивості матеріалу об'єкту, які дають змогу користувачам досягати бажаного ефектів і поведінки – від простих дифузних шейдерів для простого відображення кольору або стилізації, до складних процедурних текстур дерева, металу, скла тощо.

Неруйнівний робочий процес у свою чергу означає, що зміни при внесені в будь-яку точку мережі вузлів, не впливатимуть на базову геометрію чи інші аспекти сцени. Така неруйнівна природа дозволяє вільно експериментувати, швидко повторювати та з легкістю налаштовувати матеріали, не кажучи про швидку адаптацію матеріалу при додаванні нового вузла.

Наступним йде візуальне представлення зв'язків, яке забезпечує більш чітке бачення кожного аспекту матеріалу і на відміну від інших традиційних підходів полегшує для художників розуміння фізичних властивостей матеріалу та керування ними, створюючи більш інтуїтивно зрозумілий та інтерактивний робочий процес. Так, наприклад, для керування прозорістю елемента надається окремий вузол IOR, значення якого і відповідає за той чи інший реальний матеріал.

Залишається ще модульність та рандомізація. Користувачі мають можливість "інкапсулювати" типові шейдери або процедурні текстури в групи вузлів, що дозволяє легко повторно використовувати їх у кількох проєктах або ділитися ними з іншими. Ця модульність підвищує ефективність робочого процесу та сприяє співпраці при роботі над великими проєктами [21].

Як приклад можна виділити систему нодів у програмному забезпечені Blender, Houdini, Substance Designer тощо [11]. Розберемо для початку використання нодів у кожному з вище перелічених продуктах оглянувши першим чином програму від Adobe – Substance Designer. Можна розглянути формування матеріалу у останньому на рисунку 2.1, взятому з статті [11].

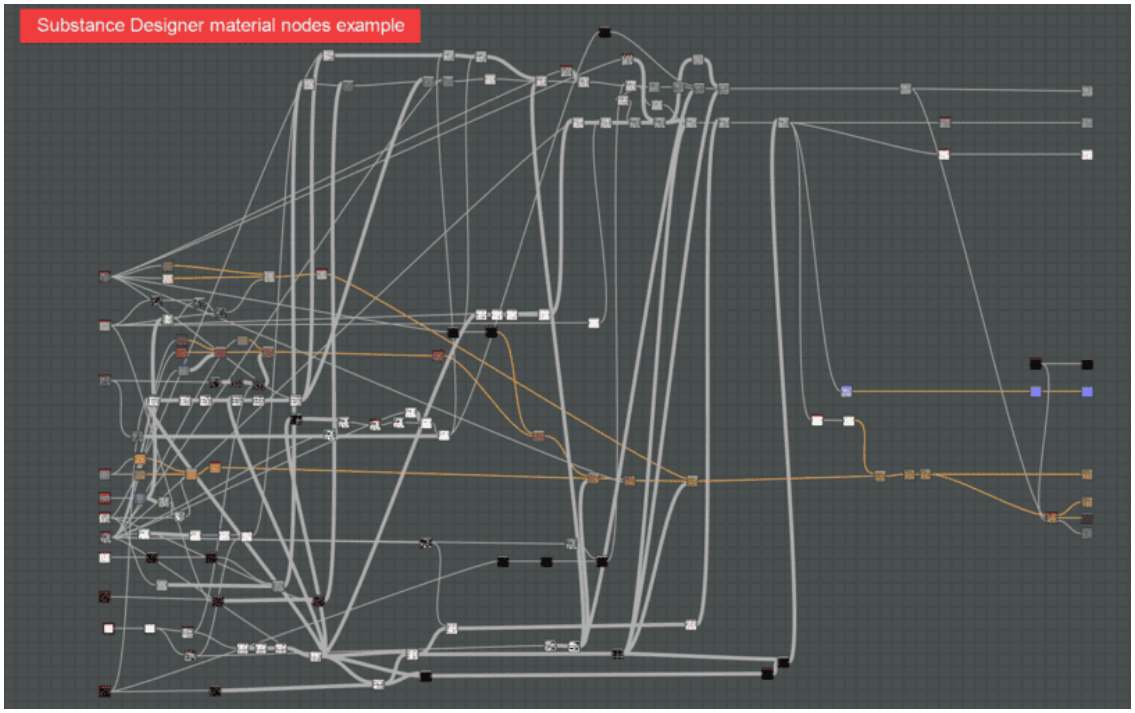

Рис. 2.1 – Приклад використання нодової системи для реалізації PBR матеріалу в Substance Designer

Програмне забезпечення Substance Designer, разом з Substance Painter [22], є одним із стандартів в індустрії створення матеріалів основаних на PBR або ж фізичному рендерингу і тому його розгляд є необхідним для виділення основних елементів у нодовій системі. Іншою відомою програмою з нодовою системою є Houdini, в якій ноди використовуються вже не для формування матеріалу, а для розробки симуляцій та формування геометрії [23]. Принцип аналогічний з PBR нодовою системою Designer. Приклад такого використання можна побачити при генерації дерева [24], один з прикладів процесу якого можна побачити на рисунку 2.2 взятого з відео під номер п’ять. Як можна побачити, система вузлів в Houdini представлена послідовним деревом, коли як в Substance Designer це швидше граф.

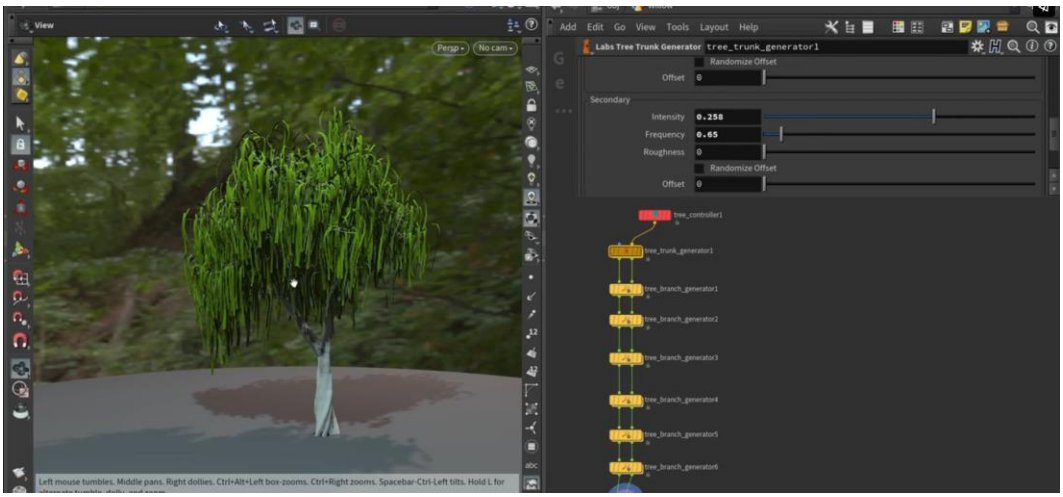

Рис. 2.2 – Приклад використання нодової системи для реалізації дерева в Houdini

На рисунку 2.3 зображено приклад модулю у рамці (Frame) в програмному забезпеченні Blender як ще один вид модульності.

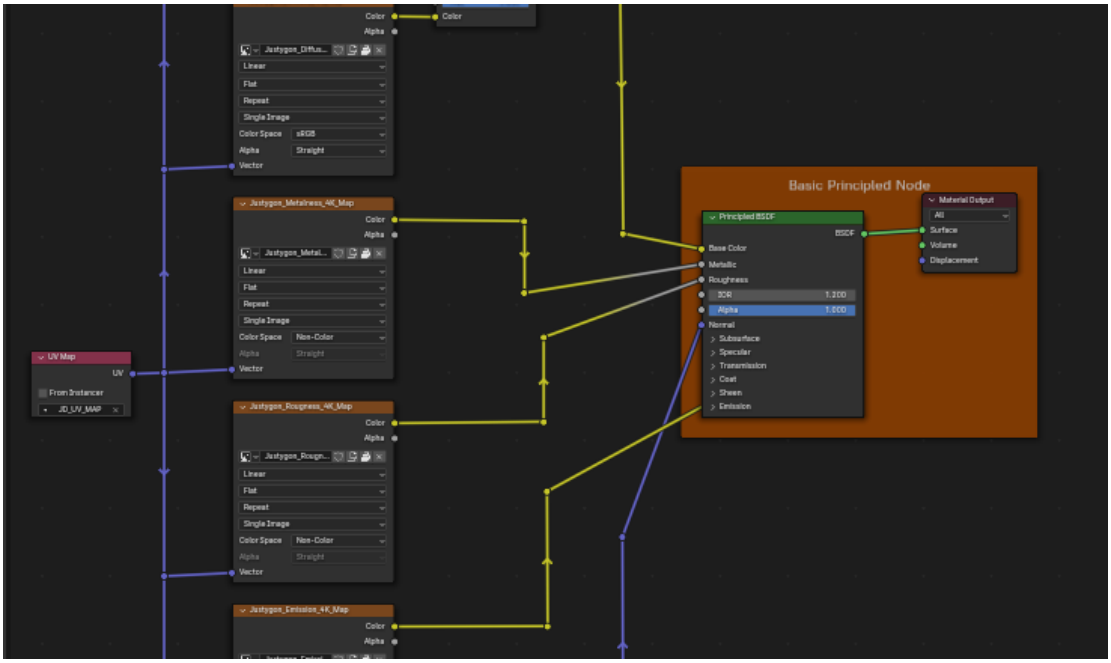

Рис. 2.3 – Приклад модульності матеріалу в Blender

Під рандомізацією розуміється процедурна генерація або випадкова вибірка конкретно властивостей матеріалу [25]. Використовуючи математичні функції та шумові алгоритми, система вузлів дозволяє створювати органічні текстури, візерунки та варіації, які було б важко або неможливо досягти

малюванням. Цей процедурний підхід не тільки економить час, але й дозволяє створювати динамічні та адаптивні матеріали, які реагують на зміни освітлення, кута огляду чи інших факторів сцени, не кажучи вже про зміни геометрії.

І хоча системи на основі вузлів надають численні переваги для створення матеріалів, вони також мають ряд обмежень і проблем, які перешкоджатимуть ефективності робочого процесу та взаємодії з користувачем, так як часто мають високий поріг входження, особливо для користувачів-початківців або тих, хто переходить від програмного забезпечення з традиційною двовимірною графіки, будь-то векторна чи растрова. Розуміння функціональності різних вузлів, їхніх входів і виходів, а також того, як вони взаємодіють один з одним, потребує часу та зусиль. Ця складність може лякати і заважати використовувати систему повністю. До цього додається візуальний безлад, який у міру того, як матеріальні мережі стають все складнішими, може стати неймовірним, що призводить до труднощів в організації, навігації та налагодженні. Користувачам буде важко підтримувати ясність і читабельність у своїх матеріалах, що перешкоджатиме продуктивності та творчості, не кажучи вже про роботу кількох людей над одним матеріалом. Складні вузлові мережі також можуть призвести до значного накладення на продуктивність, особливо під час візуалізації або попереднього перегляду в реальному часі. Кожен вузол у мережі створює свій внесок у обчислювальне навантаження, і в міру того, як мережа зростає в розмірі та складності, вимоги до обчислень відповідно зростають. Це призводить до довшого часу візуалізації, сповільнення продуктивності вікна перегляду та збільшення споживання ресурсів, особливо в системах з обмеженим апаратним забезпеченням.

Останнім виділимо труднощі в досягненні реалістичних результатів, оскільки системи на основі вузлів пропонуючи величезну гнучкість роблять задачу досягнення фотореалістичних або фізично точних результатів складним завданням, оскільки доведеться покладатися на спроби й помилки, повторне вдосконалення або зовнішні посилання, щоб точно налаштувати властивості та поведінку матеріалу. Без глибокого розуміння основоположних принципів взаємодії світла та фізики матеріалу буде важко досягти переконливих

результатів, особливо в складних сценаріях освітлення або з фізично-відповідними візуалізаціями.

2.2 Традиційна система шарів: переваги та недоліки

На відміну від систем на основі вузлів, системи на основі шарів для матеріалів пропонують більш структурований та інтуїтивно зрозумілий підхід до створення матеріалів, так як вони будуються в ієрархічній формі, причому кожен шар представляє окремий компонент або атрибут матеріалу. Ці шари можуть включати дифузний колір, відблиски, рельєфні карти та інші властивості, кожна з яких сприяє загальному вигляду матеріалу. У робочому процесі зазвичай починають із базового шару, що представляє основний колір або текстуру матеріалу. Потім додаються наступні шари, щоб додати деталі або ефекти, такі як відблиски, шорсткість або прозорість.

Зазвичай зустрічаються в програмному забезпеченні для редагування двовимірних зображень, наприклад Adobe Photoshop [26] та Krita [27], де вони використовуються для компонування та обробки зображень. На рисунку 2.4 зображено стек рівнів в Krita.

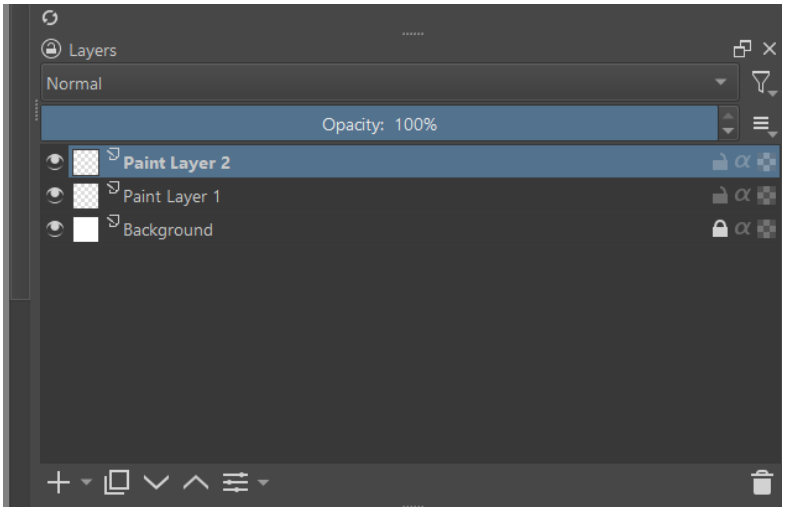

Рис. 2.4 – Приклад систем рівнів на основі стеку в Krita

Однак їх також було адаптовано для використання в програмному забезпеченні для тривимірної графіки, щоб спростити процес створення та

редагування матеріалу, так як системи на основі шарів для матеріалів пропонують насамперед інтуїтивно зрозумілий інтерфейс, схожий на програмне забезпечення для редагування 2D-зображень. Це знайомство полегшує, особливо тим, хто має досвід графічного дизайну чи цифрового мистецтва, перехід до створення 3D-сцен, що дозволить користувачам зосереджуватися на одному аспекті матеріалу за раз, наприклад кольорі, текстурі чи прозорості, не перевантажуючись складністю мережі на основі вузлів, отримавши ефективну організацію та управління матеріальними компонентами. Крім цього, користувачі можуть легко переставляти шари, групувати їх у папки та застосовувати маски або коригувати, щоб контролювати видимість і вплив кожного елемента. Така організаційна гнучкість підвищує ефективність робочого процесу та дозволяє отримувати структурований матеріал. Порівняно зі складними мережами на основі вузлів, системи на основі рівнів можуть запропонувати ще й переваги продуктивності, особливо в сценаріях відтворення в реальному часі.

До спільних із системою вузлів переваг можна віднести співпрацю між художниками, що можуть легко ділитися файлами матеріалів і обмінюватися ними, знаючи, що інші зможуть зрозуміти та працювати з тією самою структурою шарів.

На відміну ж від систем на основі вузлів, які дозволяють створювати складні мережі, які використовують фізичні властивості об’єктів напряму, системи рівнів обмежені їхнім лінійним стеком, що може ускладнювати розробку типів матеріалів і не підтримувати розширені методи затінення або процедурні ефекти.

Також зберігаються труднощі в досягненні реалістичності, так як цим системам незважаючи на те, що вони чудово відображають основні властивості матеріалу, такі як колір, прозорість, їх майже не впоратися з імітацією складних явищ, таких як підповерхневе розсіювання, анізотропні відбиття або деталі мікроповерхні. Художникам необхідно вдаватися до методів обходу або зовнішніх інструментів, щоб досягти реалістичних результатів, що займе багато часу та буде неефективним як з точки зору часу, так і фінансової сторони. На

відміну ж від систем на основі вузлів, які пропонують повністю неруйнівний робочий процес, системи на основі шарів накладатимуть обмеження. Хоч коригування, такі як зміни прозорості або режими змішування шарів, зазвичай можна змінити без проблем, інші, наприклад зміна розміру або трансформація шарів, роблять неможливими зміни у матеріалі без повного перероблення шарів. Також до цього списку можна добавити повторне використання та модульність. Окремі шари можна зберігати та повторно використовувати в інших проєктах, але ієрархічна природа стеків шарів ускладнює входження компонентів складних матеріалів у багаторазові модулі.

Підсумовуючи вище надану інформацію системи на основі шарів для матеріалів пропонують зручний та інтуїтивно зрозумілий підхід до створення матеріалів, хоча також мають обмеження щодо складності, реалістичності, неруйнівного редагування та можливості повторного використання. Та проблеми на цьому не закінчуються, оскільки окрім недоліків є ще обмеження, які випливають напряму з недоліків:

– Обмежене представлення складних властивостей матеріалів, тобто системи на основі шарів розроблені в основному для обробки простих властивостей матеріалів і їм важко відобразити більш складні властивості.

– Труднощі під час процедурної генерації, так як системи рівнів дозволяють укладання та змішування вже існуючих шарів і не мають можливості генерувати процедурні текстури чи візерунки безпосередньо в системі і користувачам необхідно покладатися на зовнішні генератори процедурних текстур або системи на основі вузлів для досягнення бажаних результатів.

2.3 Концепція гібридного підходу для створення матеріалу

Гібридний підхід, який поєднує в собі інтуїтивну простоту робочих процесів із гнучкістю та потужністю мереж на основі вузлів має на меті використовувати сильні сторони обох систем, одночасно пом’якшуючи їхні відповідні обмеження, надаючи більш універсальний і ефективний набір

інструментів для створення матеріалів у програмному забезпеченні тривимірної графіки.

В основі гібридного підходу зберігається знайомий інтерфейс на основі рівнів для організації та керування компонентами матеріалу, одночасно вводячи концепцію “шарів вузлів”, щоб забезпечити більш складні та динамічні взаємодії матеріалів [28]. На рисунку 2.5 зображено приклад власної реалізації гібридного підходу в Blender.

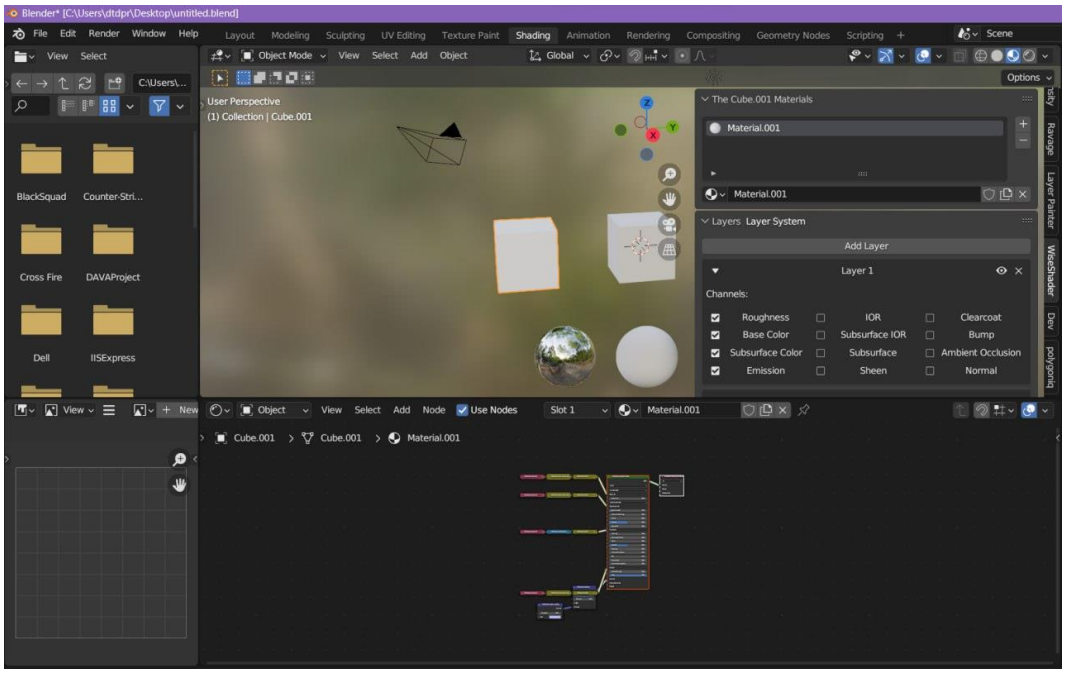

Рис. 2.5 – Приклад гібридного підходу

На рисунку стек шарів розташований у верхній частині екрану з правої сторони, коли як у редакторі вузлів ці шари представлені трьома чи більше вузлами. Фактично, шари діють як контейнери для мереж вузлів, дозволяючи художникам вкладати складні налаштування шейдерів, процедурні ефекти та властивості матеріалів у структурі ієрархічного шару:

– Інтерфейс редактора матеріалів складається з традиційного стека шарів, подібного до того, що є в Photoshop, де користувачі створюватимуть та впорядковуватимуть рівні, що представляють різні компоненти матеріалу, такі як колір, текстури, дзеркальність, прозорість тощо.

– У кожному шарі можна створювати складні мережі вузлів, що можуть включати шейдери, текстури, математичні операції та процедурні ефекти,

надаючи гнучкість для створення дуже деталізованих і динамічних матеріалів. Мережі вузлів працюють незалежно один від одного, дозволяючи художникам ізолювати та керувати різними аспектами матеріалу в межах окремих контейнерів.

– Виходи мереж вузлів у межах шару інтегруються в стек за допомогою режимів змішування та коригування непрозорості. Користувачі можуть змішувати результати мереж вузлів з іншими шарами в стеку, використовуючи звичні режими множення, накладання та екранування. Коригування непрозорості дозволяє додатково контролювати видимість і вплив шарів у ієрархії матеріалів.

– Робиться акцент на неруйнівному редагуванні та попередньому перегляді змін матеріалу в реальному часі, при чому можна коригувати як традиційні шари, так і шари вузлів, і зміни миттєво відображаються в попередньому перегляді матеріалу. Цей ітеративний робочий процес заохочує експерименти та тонке налаштування, дозволяючи досягати бажаних матеріальних ефектів з точністю та контролем.

– Підхід забезпечує ефективну організацію, оскільки користувачі можуть легко переставляти шари та вузли, групувати їх у папки та застосовувати маски чи шари коригування, щоб контролювати видимість і вплив кожного компонента. Ця ієрархічна структура підвищує ефективність робочого процесу та полегшує співпрацю між художниками.

В цілому гібридний підхід пропонує унікальний набір переваг, які використовують сильні сторони обох методологій:

– Підтримує знайомий інтерфейс на основі шарів, що робить його доступним для художників із різним рівнем досвіду і у той же час представляє розширену функціональність, що дозволяє створювати складні матеріальні взаємодії та процедурні ефекти в рамках знайомої організаційної структури. Таке поєднання простоти та потужності підвищує продуктивність і креативність.

– Поєднуючи ієрархічну організацію шарових систем із гнучкістю мереж на основі вузлів, гібридний підхід дозволяє художникам створювати дуже деталізовані та динамічні матеріали.

– Організація, як наслідок, сприяє ефективному робочому процесу та співпраці між художниками, надаючи єдиний інтерфейс для створення та редагування матеріалів.

– Гібридний підхід є універсальним і адаптованим до широкого спектру проєктів і робочих процесів, незалежно від того, чи працюють над стилізованою анімацією [29], фотореалістичною візуалізацією чи проєктами розробки ігор, художники можуть використовувати гібридний підхід для створення матеріалів, які відповідають їхнім конкретним потребам і вимогам.

Тому гібридний підхід до створення матеріалів пропонує збалансоване поєднання простоти, гнучкості та функціональності, що дає змогу створювати переконливі та візуально приголомшливі матеріали з легкістю та ефективністю. Та будь-яка система не ідеальна, і в цього підходу також є свої недоліки, про які піде мова далі.

Почнемо з трохи конфліктного недоліку – складності і кривої навчання. Інтеграція систем на основі шарів і вузлів вносить додаткову складність у процес створення матеріалу, так як може знадобитися ознайомлення з обома парадигмами, що призведе до крутішої кривої навчання, особливо для тих, хто новачок у програмному забезпеченні 3D-графіки. Керування як традиційними шарами, так і шарами вузлів в одному інтерфейсі може вимагати додаткового навчання та налаштування для користувачів, які звикли до простіших робочих процесів, хоча з іншого боку це дозволить хоча б трохи орієнтуватися в програмах знаючи одну з концепцій і досягати непоганих результатів не знаючи повністю усієї системи.

Також гібридний підхід може призвести до додаткових витрат на продуктивність, особливо під час роботи зі складними мережами вузлів у межах рівнів. Візуалізація та попередній перегляд у режимі реального часу матеріалів, що містять складні налаштування шейдерів і процедурні ефекти, можуть потребувати більше обчислювальних ресурсів порівняно з простішими

матеріалами на основі шарів. Це веде також до того, що при включенні вузлів у процес створення матеріалу на основі рівнів або навпаки створюватиме додаткову складність в управлінні проєктами та організації файлів.

Звісно ж, якщо є можливість в погіршенні продуктивності, то також є можливість несумісності. Залежно від програмного забезпечення, яке використовується, можуть виникнути проблеми під час обміну матеріалами, створеними за допомогою гібридного підходу, з іншими виконавцями або на різних програмних платформах. Користувачам знадобиться запікання текстур, щоб можна було передавати проєкти до іншого програмного забезпечення.

2.4 Методи імплементації гібридного методу в Blender

Реалізація гібридної системи, яка поєднує робочі процеси на основі шарів і вузлів для матеріалів у Blender може бути досягнута за допомогою різних підходів, серед яких можна виокремити групування вузлів, композиції вузлів, розробка окремого доповнення та використання інтеграції або сторонніх засобів.

Спеціальні групи вузлів у редакторі шейдерів – це підхід, при якому користувачі мають можливість створювати групи, які містять складні налаштування шейдерів, процедурні ефекти або властивості матеріалів. Потім ці спеціальні групи організовуватимуться та керуватимуться в ієрархічній структурі, подібній до шарів у традиційних системах, користувачі зможуть використовувати функцію групування вузлів Blender і впорядковувати групи у редакторі. Це є найпростіший серед методів, так як він не змінює нічого внутрішньо, лише відображення для власне користувача.

Композиційні вузли для шарів матеріалу в свою чергу є підходом, що полягає у використанні вузлів Compositing для створення шарів матеріалу. Користувачі можуть використовувати вузли зображення або вузли з процедурними текстурами для представлення різних компонентів матеріалу в Редакторі композиції. Розташувавши ці вузли в стек, можна реалізувати багатошарову структуру, подібну до традиційних систем на основі шарів.

Функції ж маскування та змішування Blender у редакторі Compositing можна використовувати для керування видимістю та змішуванням різних шарів матеріалу. Даний підхід вже фактично вимагає спроби перенесення частини функціоналу з одного модуля в інший, хоч і не вирішує проблему, так як в Blender API [30] це дві окремі зони впливу – Shader та Compositing.

Крім вище вказаних методів, розробники, знайомі з Blender API та роботою програми, можуть створювати власні сценарії та доповнення Python [31], щоб розширити можливості Blender щодо створення матеріалів. Ці сценарії можуть представити нові інструменти та функціональні можливості для керування шарами матеріалу та мережами вузлів в інтерфейсі Blender. Наприклад, розробити спеціальні панелі інтерфейсу користувача або оператори, які полегшують створення, організацію та редагування шарів матеріалів і мереж вузлів. Доповнення також можуть автоматизувати повторювані завдання та оптимізувати процес створення матеріалу [32], надаючи попередньо визначені шаблони або попередні налаштування для типових налаштувань матеріалів. Прикладом схожої реалізації, але системи рівнів, є Layer Painter [33] та Ravage [34], які одразу ж надають базовий шаблон матеріалу і дозволяють одним натиском добавити шар з налаштуваннями, масками, фільтрами тощо. Цей підхід є найважчим з усіх, але у винагороду дає найбільше можливостей для реалізації гібридного підходу до створення матеріалів.

Можна було б досліджувати сторонні інтеграції, які розширюють можливості Blender для створення матеріалів. Доступно декілька таких доповнень, які забезпечують розширені робочі процеси на основі вузлів та інструменти керування матеріалами, при чому вони пропонують такі функції, як вкладені групи вузлів, процедурні генератори текстур, а також розширену організацію та візуалізацію матеріальних компонентів. Проблема лиш в тому, що немає повністю гібридної системи, відповідно просто взяти для дослідження стороннє рішення немає змоги, тому цей варіант відпадає.

Ну і звісно ж метод очікування, або ж розробки Blender і внесків спільноти, оскільки природа Blender полягає у програмі з відкритим кодом, що дозволяє робити внески спільноти та розробляти до цього небачені системи,

спрямовані на покращення робочих процесів створення матеріалів чи інших сфер роботи у тривимірній графіці. Так в Blender з’явилися системи геометричних вузлів [35], які дозволили створювати частину моделей просто змінюючи кілька налаштувань, наприклад кількості вікон для будівель, структуру даху тощо. Будь-які художники та розробники можуть вносити ідеї, відгуки та код у процес розробки Blender, щоб впливати на реалізацію систем створення комплексних матеріалів. Більше того, розробники програми планували спробувати реалізувати вузол [36], який би відповідав за рівень матеріалу, що і є фактично гібридним підходом, просто замість того, щоб вузол входив до рівня, усе навпаки. Беручи активну участь у спільноті розробників Blender, художники можуть допомогти сформувати майбутній напрямок інструментів створення матеріалів і робочих процесів у програмному забезпеченні.

Усі ці підходи можна комбінувати та налаштовувати на основі конкретних вимог проєктів і робочих процесів використовуючи гнучку архітектуру Blender і широкі можливості налаштування. Розглянемо детальніше деякі з підходів.

Підхід користувацьких вузлів передбачає створення груп вузлів у редакторі шейдерів Blender. Ось як можна реалізувати цей підхід – користувачі починають із розробки спеціальних груп вузлів, що містять мережу, яка створює бажаний матеріальний ефект або поведінку. Далі користувачі організовують ці групи ієрархічно в редакторі шейдерів. Вони можуть створити спеціальну рамку для кожного компонента матеріалу та впорядкувати їх у багатошарову структуру. Наприклад, створити базову кольорову групу вузлів у нижній частині ієрархії, за якою йдуть групи вузлів, що представляють шорсткість та інші властивості матеріалу, розташовані одна на одній. На рисунку 2.6 зображено приклад користувацького або спеціального вузла.

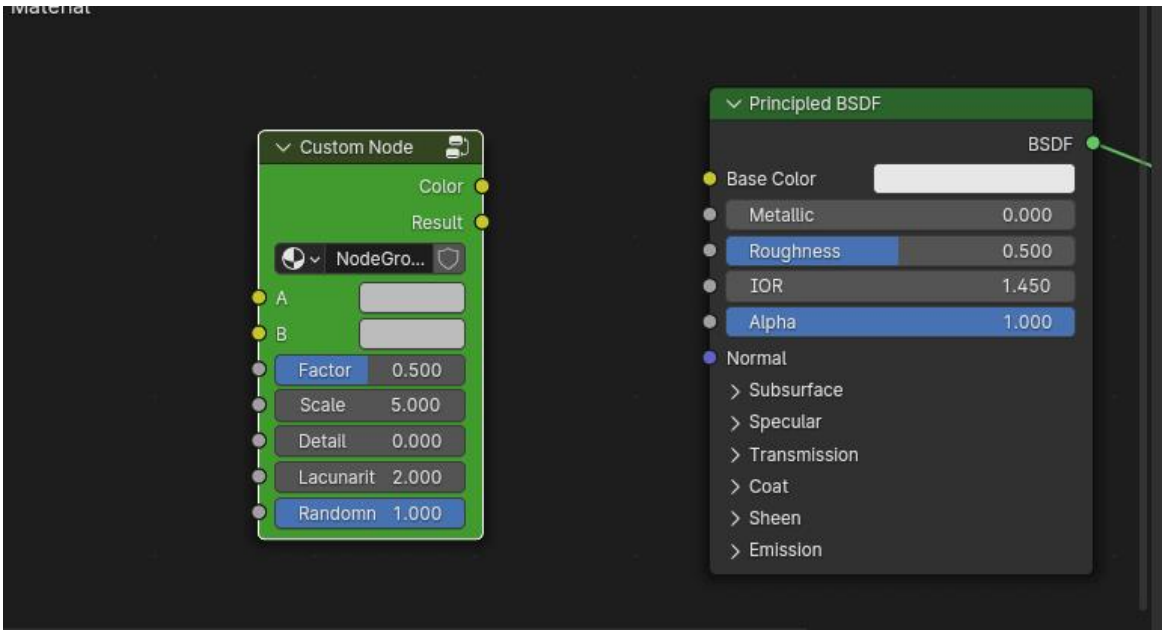

Рис. 2.6 – Користувацький вузол

Як можна побачити з рисунку 2.6, даний вузол представляє собою окрему властивість об’єкта. В основі такого ноду лежить групування, про яке піде мова пізніше. Фактично через ці ноди можна реалізувати ту ж систему рівнів, яку зробили розробники Layer Painter [33].

Додатково окрім вказаного методу через користувацький вузол можна просто використати функцію групування вузлів Blender, щоб накладати вузли один на одного в редакторі шейдерів для отримання тієї чи іншої властивості. Вони можуть контролювати змішування та взаємодію між різними компонентами матеріалу за допомогою різних режимів та вузлів коригування, таких як вузол Mix Color [37]. Кожна користувацька група або проста група вузлів в ієрархії має входи та виходи, які дозволяють підключатися до інших груп вузлів і зовнішніх компонентів. Вхідні дані представляють параметри, які можна регулювати або контролювати ззовні, а вихідні дані забезпечують кінцеві дані про матеріал. Найкраще обирати вузли, що дозволяють математично все налаштовувати. Для прикладу візьмемо налаштування шорсткості поверхні на основі текстури металу. Її можна змінювати як за допомогою кольорового вузлу Color Ramp [37], так як текстура шорсткості чорно-біла (тільки в Grey-scale діапазоні), або можна застосувати Map Range [37], який дозволяє контролювати максимальне і мінімальне значення кольорів

через просте число з плаваючою комою. Входи та виходи кожної групи вузлів належним чином підключені для підтримки узгодженості та безперервності в ієрархії матеріалів, що допускає ітеративне тестування. Художники періодично вдосконалюють спеціальні групи вузлів, щоб переконатися, що вони створюють бажані ефекти та поведінку, ті ж процедурні текстури, приклад якої взято з Blender Kit [38], який зображено на рисунку 2.7.

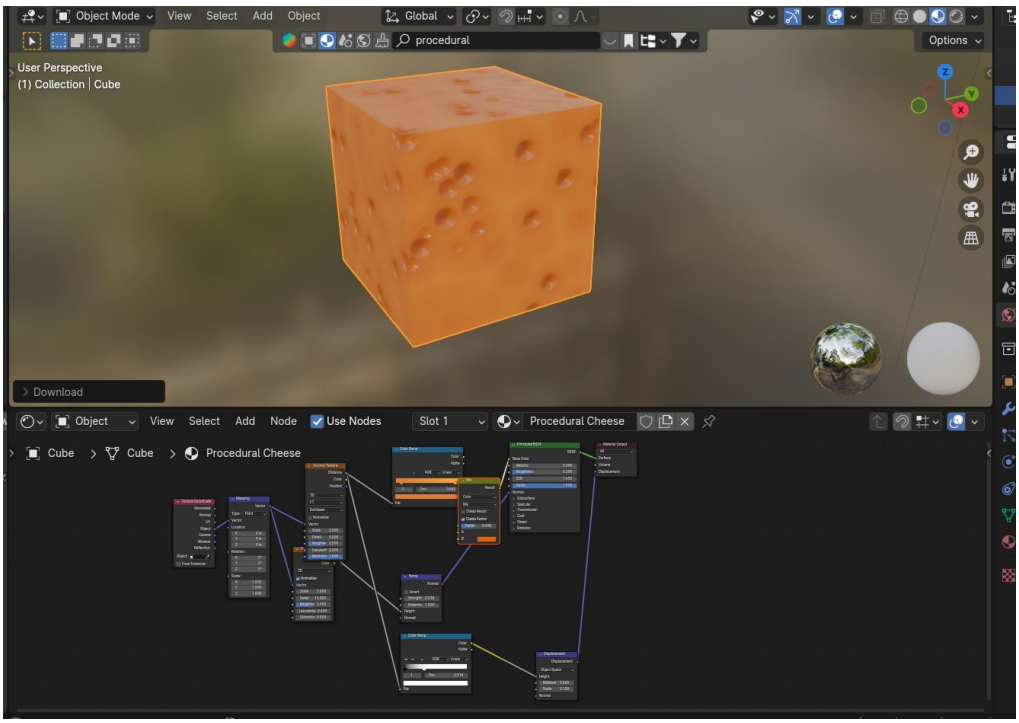

Рис. 2.7 – Приклад процедурного матеріалу сиру, взятого з Blender Kit

Завдяки цьому повторюваному процесу можна точно налаштовувати користувацькі групи вузлів відповідно до конкретних вимог свого проєкту, будь то для фотореалістичного рендерингу, стилізованої анімації чи розробки ігор, або як в даному випадку під будь-яку форму сиру.

Та у цього методу є декілька недоліків, а саме:

– Сумісність, оскільки користувацькі групи вузлів можуть бути неповністю сумісними або переносними між різними версіями Blender або до іншого програмного забезпечення для тривимірної графіки, тим паче якщо є серйозні відмінності в реалізації вузлів, можливостях шейдерів або механізмах відтворення. Вони призведуть до проблем із втрати точності під час передачі

матеріалів між програмами, пошкодивши або навіть не завантаживши матеріалу.

– Налагодження, яке включає усунення несправностей складних мереж вузлів у користувальницьких групах, що може бути складним завданням, особливо під час неочікуваної поведінки або помилок. Художникам може знадобитися додатковий час на налагодження та тестування, щоб виявити та вирішити проблеми в налаштуваннях матеріалу, які можуть перешкоджати ефективності робочого процесу та продуктивності.

У свою чергу підхід з розробкою доповнень із використанням мови програмування Python [39] передбачає створення власних сценаріїв для розширення можливостей створення матеріалів у Blender, надаючи користувачам додаткові інструменти та функції для керування шарами матеріалів і мережами вузлів. Щоб реалізувати гібридний підхід використовуючи Blender API [30] необхідно для початку визначити вимоги до робочого процесу, фактично зробити концепцію того, як виглядатиме і які задачі виконуватиме програмний продукт. Для систем накладання матеріалів важливим буде огляд інтерфейсу та перевірка його зручності для користувача, а також чи є необхідність в певних функціях, таких як маски, фільтри тощо. Крім цього, це включає оцінку звичайних завдань, повторюваних процесів або обмежень у вбудованих інструментах редагування матеріалів Blender. Наприклад, можна розробити інструменти для швидкого створення та керування матеріальними шарами з використанням системи Drag and Drop для автоматизації повторюваних завдань, або створити механізм для генерації процедурних текстур і візерунків.

Наступною буде інтеграція функцій редагування вузлів, що дозволяє користувачам створювати та керувати мережами вузлів безпосередньо в інтерфейсі з шарами, не переходячи в панель для роботи з ними. Так замість того, щоб шукати параметр кольору можна просто вивести його як окрему опцію шару, тим самим змінюючи параметр вузла не проводячи його пошуки. Вже не кажучи про можливість зробити навпаки – щоб при додаванні вузла до рівня його параметри можна було редагувати безпосередньо в інтерфейсі

керування шарами, додавши до цього інструменти організації вузлів, керування групами вузлів і виявлення помилок.

Після реалізації основного функціоналу та базового інтерфейсу настає етап, коли необхідно оптимізувати продуктивність і зручність користувальницьких інструментів та доповнень для забезпечення плавної та ефективної роботи в інтерфейсі Blender. Це може включати оптимізацію коду, мінімізацію обчислювальних витрат і надання інтуїтивно зрозумілих інтерфейсів для користувача із системою з чітким відгуком і документацією.

Останніми двома етапами можна виділити:

– Тестування та ітераційне вдосконалення

– Створення документації або посібника користувача.

На першому з вище вказаних етапів ретельно тестують спеціальні інструменти, що були створені щоб забезпечити функціональність, стабільність і сумісність, при чому з різними версіями Blender. На останньому ж етапі залишається створення документації, в якій розписано кожну функцію та задачу, яку вона виконує, можливо із додаванням секцій з кодом для подальшої зміни під себе.

2.5 Висновок до другого розділу

Підсумовуючи, хоча гібридний підхід до створення матеріалів пропонує переконливе поєднання гнучкості та функціональності, він також створює додаткову складність, потенційні витрати на продуктивність і сумісність. Художники повинні ретельно оцінити свої конкретні вимоги до проєктів та вподобань робочого процесу, щоб визначити, чи переваги гібридного підходу переважають його недоліки. Крім того, постійна розробка програмного забезпечення та відгуки користувачів можуть з часом усунути деякі з цих недоліків, ще більше підвищивши зручність використання та ефективність підходу.

3 ОБЧИСЛЮВАЛЬНИЙ ЕКСПЕРИМЕНТ ЕФЕКТИВНОСТІ РОЗРОБЛЕНОГО З BLENDER API ГІБРИДНОГО ПІДХОДУ

3.1 Реалізація гібридного підходу з використанням Blender API

Для початку встановлюємо з офіційного сайту Visual Studio Code [40] та інтерпретатор Python [39], щоб підготувати в цілому середовище розробки доповнення.

Наступним роком знадобиться ознайомитися та встановити основне доповнення – bpy, що дозволяє додавати основі елементи з Blender Python API та в цілому його застосовувати в розробці. Дане доповнення не тільки дозволить взаємодіяти, але й створювати за шаблонами структурну схему. Для створення стандартного шаблону доповнення необхідно в Visual Studio Code натиснути комбінацію клавіш CTRL+SHIFT+P та обрати з випадаючого списку Create addon. Програма автоматично створить один файл з кодовою назвою __init__.py. Даний файл відповідає за ініціалізацію та реєстрацію усіх елементів в межах однієї папки. Далі створюємо чотири наступні папки або ж директорії:

– UI – містить прописаний інтерфейс програми.

– TOOLS – включає загальні інструменти або виведені в окрему категорію функції або класи, щоб не нагромаджувати одноманітним кодом програму.

– PROPERTIES – включає розроблені звичайні класи, які представляють рівні, канали та вузли як певні абстракції.

– OPERATIONS – містить оператори програми.

В директорії UI розміщено буде три файли, першим з яких обов’язково повинен бути __init__.py, в якому необхідно зареєструвати Blender елементи, які в даному випадку представленні панелями [41]. Панель в Blender Python API представляє собою область в редакторі, на якій розміщуються інші UI елементи, такі як Label, Textbox, Button тощо. Програма вимагатиме дві основні панелі, одну для роботи з матеріалами на об’єкті і другу власне під редагування утворених матеріалів. Для першої використаємо уже готові шаблони, аналогічні

при створенні матеріалу в традиційній системі, а другу пропишемо так, щоб вона містила вибір рівнів та каналів відповідно до них з властивостями кожного вузла. На рисунку 3.1 можна побачити кінцевий вигляд кожної з панелей, а у додатку Д код власне панелі без шаблонів.

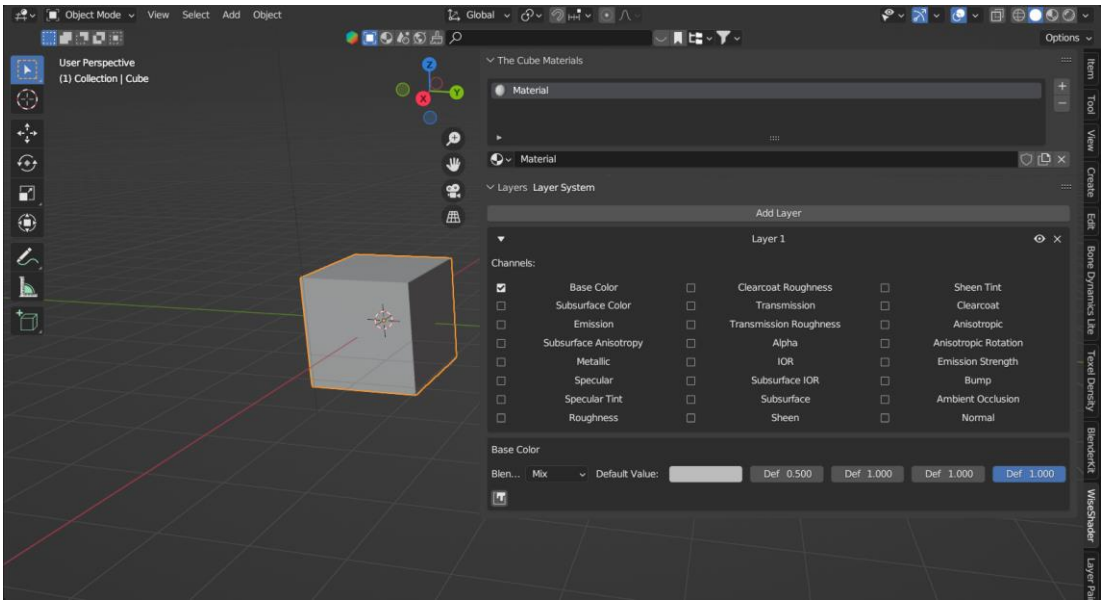

Рис. 3.1 – Основні дві панелі доповнення

Наступним кроком буде задача прописування операторів. Оператор в Blender API [42] – це насамперед функція, яка буде застосовуватися при спрацьовуванні того чи іншого тригера, наприклад натискання кнопки, реалізація списку тощо. Для кожної логічної функції необхідно свій оператор. Наприклад, для рівня можна зробити так, що його можна додати, його можна видалити, розширити та сховати, відповідно буде чотири оператори на усі ці дії – add_layer, remove_layer, expand_layer, visible_layer. Усе залежить лише від функцій, які для простої реалізації виглядатимуть наступним чином:

– Маніпуляції з шарами або рівнями, описані вище.

– Вибір активного каналу та маніпуляції з ними, заміна базового кольору на зображення в кольорових каналах. Відповідно це буде active_channel.py та image_channel.py.

Звісно ж як і з панелями необхідно в папку до цих шести файлів добавити файл ініціалізації __init__.py. Після готових інтерфейсу та операторів наступним кроком конкретно для даної розробки є переписування вузлів під

потреби гібридної системи, що включає додавання власних налаштувань, виведення властивостей кожного вузла та каналу в панелі та звісно ж логічне з'єднання з основними вузлами, серед яких виділяються Principled BSDF, Material Output, Bump та Normal Map. Ця конкретна реалізація є не дуже оптимізованою з точки зору Blender API, оскільки використовує звичайні класи та методи [43] замість заготовлених розробниками інструментів, через що можуть виникнути проблеми та нестабільна поведінка у програмі. Для реалізації вузлів необхідно створити два класи – загальний Node та його наслідників. Перший буде батьківським та міститиме загальні властивості для груп вузлів, серед яких виділяються:

– Вузли накладання матеріалів та створення маски (Bevel, Mapping).
– Загальні вузли (Mix, RGB).
– Основні вузли (Normal Map, Principled BSDF).

Це зроблено для того, щоб зменшити кількість повторень та оптимізувати код програми ще на ранніх етапах розробки. Аналогічна ситуація з каналами (див. рисунок 3.2), кожен яких відповідає за певну властивість, розглянуто ще вкінці першого розділу.

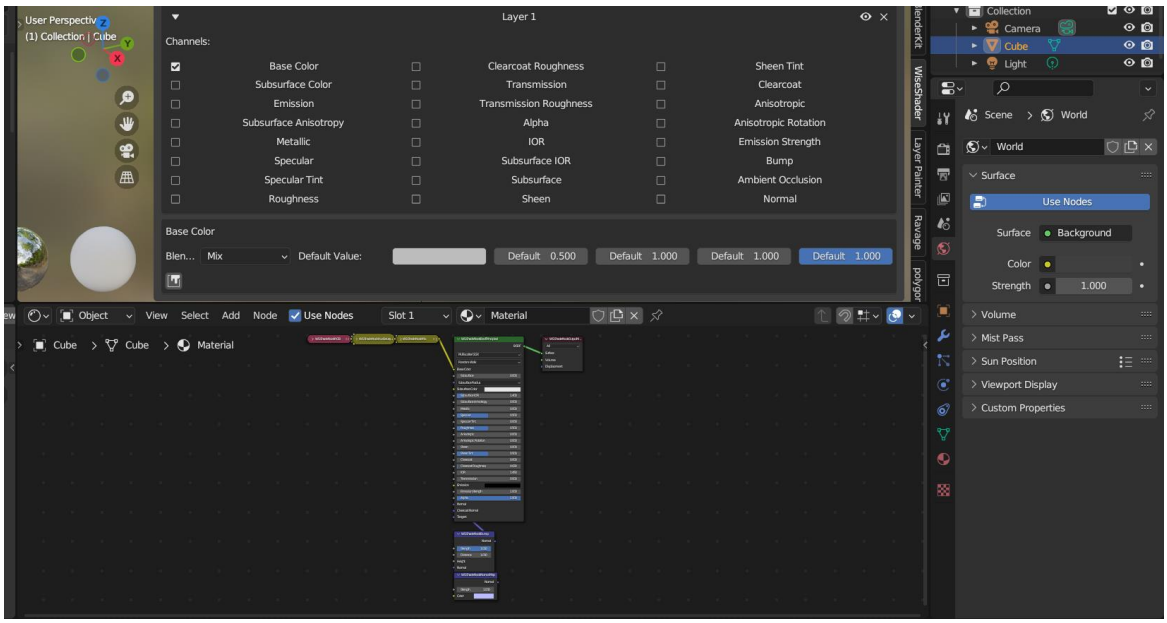

Рис. 3.2 – Активний канал базового кольору першого рівня

Після панелей, операторів та вузлів залишається останній крок – додаткові функції, такі як наприклад авто-виправлення матеріалу після якоїсь помилки видалення тощо. Наприклад, як показано в додатку Д, є файл default_node.py, суть якого полягає в відновлені основних вузлів або додаванні до матеріалу при їх відсутності, тим самим забираючи проблему, коли можна добавити канал Normal Map, а вузла, в який це повинно підключатися, немає. Щодо з'єднань, то за це відповідає layers.py, який з'єднує усі ноди відповідно до каналів на Principled BSDF або інших проміжних вузлах.

Файлову структуру розробленого доповнення подано на рисунку 3.3.

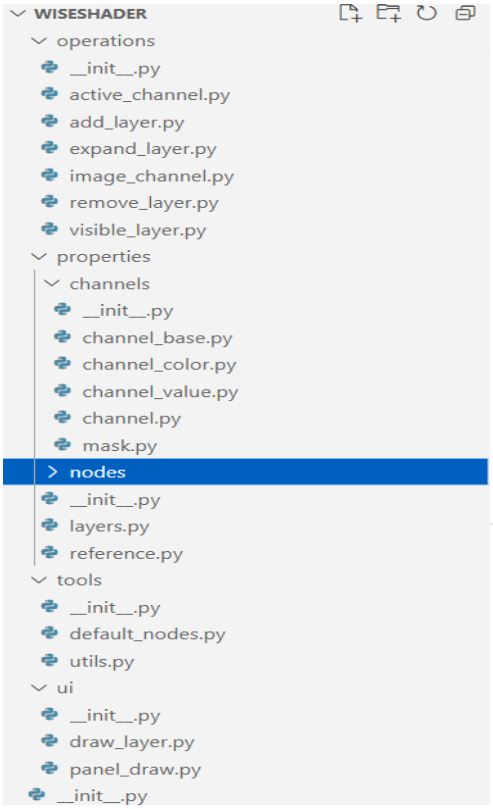

Рис. 3.3 – Фінальна файлова структура доповнення (з закритою папкою з нодами)

У кожному файлі ініціалізації необхідно також прописати реєстрацію та вивільнення кожного ініціалізатора директорії, оператора, панелі тощо. Так, для того, щоб основний ініціалізатор міг звертатися наприклад до панелей, в ньому необхідно прописати ui.register(). Ця функція фактично підтягує усе, що

прописано в __init__.py UI папки. Крім цього, для кожної структури та оператори необхідно прописувати аналогічні механізми реєстрації.

Також необхідно згадати про застосування додаткового доповнення до Visual Studio Code під назвою Fake BPY [44], суть якого полягає у спрощенні роботи з Blender Python API завдяки додаванню підказок та розписуванні кожної функції усіх модулів при наведенні на них, так як за його відсутності необхідно постійно мати відкритою документацію, яка для деяких функцій майже не дає інформації, а то й взагалі лише посилається на них, що вони просто існують і виконують, а як з ними взаємодіяти не описує. Тому на цій ноті можна перейти до труднощів, які виникли при розробці додатку та які проблеми має ця реалізація.

Перш за все основною і найбільшою з проблем була слабка документація, на основі чого для того, щоб реалізувати доповнення необхідно шукати форуми, на яких оглядати схожі проблеми, які виникали у інших користувачів.

Щодо вирішення проблем з продуктивністю та стабільністю, які виникли при реалізації гібридної системи на основі Python API [30], можна застосувати кілька стратегій для оптимізації та покращення його функціональності:

– Оптимізація коду, яка включає в себе розгляд ефективніших алгоритмів, зменшення непотрібних обчислень і мінімізацію ресурсомістких операцій для покращення загальної продуктивності, таких як додаткові конструкції поверх вже існуючих рішень тощо.

– Запровадження механізмів кешування для зберігання та повторного використання проміжних результатів або обчислень, зменшуючи потребу в надлишкових обчисленнях, оскільки кешовані дані можна зберігати в пам’яті або на диску, щоб підвищити продуктивність і зменшити витрати на обробку, особливо для складних налаштувань матеріалу.

– Використання багатопотокових та асинхронних методів обробки для розпаралелювання обчислювальних завдань і підвищення загальної пропускної здатності. Якщо намагатися реалізовувати власні класи, то над цим треба більше задумуватися, коли як використовуючи вбудовані механізми не

вимагатимуть від користувача власноручної розробки та прописування багато поточності.

– Впровадження надійних механізмів обробки помилок і налагодження, щоб ефективно виявляти й усувати проблеми. Це може включати використання журналу помилок, обробки винятків і інструментів діагностики.

– Оптимізації моделей користувачем теж можуть допомогти в покращені продуктивності, наприклад можливо використовувався один із більш ресурсоємних методів, таких як скульптинг, про що можна почитати в додатку Б та в [9]. Таким чином можна створити модель з меншим полігонажем використовуючи ретопологію або зменшуючи кількість непотрібних полігонів, які можуть не давати жодного ефекту або об’єму.

Окрім вище вказаних проблем виникла і ще одна, пов’язана вже напряму з розумінням структури доповнення та його функціями – це обмеженість в розумінні інтерфейсу та задачах, які повинно виконувати воно.

До обмеженості входять проблеми з реалізацією UI частини з нерозумінням як один елемент інкапсулюється в іншому, або як одна властивість може повністю зламати програму, коли як інша навіть помилки не виведе. Так наприклад при реалізації функції використання кількості рівнів відмінної від одного деякі текстури пере накладалися на один і той же рівень через помилку в адресації.

Якщо ж повернутися до проблеми функціоналу, то хоч взаємодія між рівнями створена, але її використання обмежене через проблему у відслідковуванні параметрів. При наприклад зміні режиму змішування шарів необхідно перейти в стандартний інтерфейс Node Editor, де можна вже поміняти цей параметр вручну замість того, що робити це в інтерфейсі меню доповнення. Додатково при тестування необхідно вручну було додавати маски, так як не було розроблено даної можливості, що фактично додавало більші затрати часу на реалізацію одного і того ж об’єкту, не кажучи про комфорт редагування матеріалу.

Для виправлення вище зазначених проблем можна:

– Додати функціонал масок, фільтрів та можливо навіть керування UV-розгортками або накладання матеріалу на об’єкт в цілому.

– Переробити інтерфейс повністю змінивши нагромадження параметрів кожного з вузлів на упорядкований список або вивівши їх у окрему категорію, як це зробив Ravage.

– Додати унікальний вузол, що дозволило б мати редагування в різних режимах одночасно. Так, наприклад, додавши вузол з текстурою до каналу рівня можна отримати усі його властивості у інтерфейсі рівня, що дозволяє незалежно від області маніпулювати ними.

Впроваджуючи ці стратегії, можна оптимізувати та покращити реалізацію підходу до створення гібридного матеріалу на основі Python для забезпечення кращої продуктивності, стабільності та зручності використання.

3.2 Практичні результати дослідження гібридного підходу

Основними критеріями для порівняння ефективності методу обрано загальну кількість елементів, необхідних для отримання бажаного результату, використання пам’яті, затрачений час на виконання завдання, стабільність роботи імплементації системи та звісно ж гнучкість. Для тесту взято Layer Painter v 2.0, Ravage Lite, Node Editor в Blender та розроблена власноруч система. Постановка задачі звучить наступним чином – у нас є геометрично простий об’єкт, наприклад куб або сфера, на який необхідно накласти матеріал, що містить два різних фізичних матеріали. На основі тестування, яке включало перезавантаження кілька разів комп’ютера для чистки кешу та створення одного і того ж матеріалу декілька разів, отримано наступні результати:

– Доповнення Layer Painter (безкоштовне, на основі шарів) – кількість елементів (вузлів) 104, використання пам’яті найбільше серед усіх інших у списку, має багато проблем зі стабільністю, але займає менше часу, ніж традиційний.

– Доповнення Ravage Lite (безкоштовна версія, на основі шарів) – кількість елементів 27, використання пам'яті на рівні традиційного, без проблем зі стабільністю, дуже швидкий, швидший за традиційну систему вузлів.

– Традиційна система (на основі вузлів, вбудована в Blender) – кількість елементів 16, використання пам'яті середнє поміж усіх інших варіантів, стабільна, але займає багато часу.

– Гібридна система на основі Blender Python API (безкоштовна, гібридна) – кількість елементів 23, використання пам'яті найменше (але це тому, що вона не повністю реалізована), нестабільна, як і Layer Painter, швидша традиційної системи.

Про власне проведення експерименту описано в наступному підрозділі.

3.3 Проведення експерименту

Розпочнемо проведення експерименту на основі методу аналізу ієрархій [45] з першого пункту, а саме постановки задачі, якою виступатиме визначення найефективнішого методу створення матеріалу конкретно в програмному забезпеченні Blender. На основі вже задачі формується перелік основних категорій, за якими проводитиметься прийняття рішення, а також їх проміжних категорій:

– Ефективність – наскільки оптимізованою є та чи інша система або рішення, та скільки необхідно компонентів для реалізації одного і того ж результату.

– Функціональність – які можливості для створення матеріалу та його редагування, скільки це займе по часу та наскільки стабільно працює система або рішення.

На даному етапі необхідно розглянути, як саме будемо ділити або порівнювати саме за цими параметрами, так як якщо у часу наприклад є повноцінне значення в секундах, хвилинах, годинах тощо, то чим заміряти оптимізацію наприклад, або стабільність. Для цього найоптимальнішим серед

усіх варіантів буде розробити шкалу для кожного з них, щоб можна було потім побудувати матриці попарних порівнянь.

Отож, для усіх параметрів обрано наступні одиниці вимірювання та шкали:

– Стабільність – шкала від 0 до 1 з дробовими значеннями, поділена на 10 станів або ж значень. Значення від 0 до 0.4 в залежності від складності та критичності проблем відповідає системам, які постійно вилітають при виконанні певних дій або постійно вилітають. Значення від 0.5 до 0.7 відповідає рішенням, що не мають майже проблем або мають некритичні, що не пошкодять робочий процес, максимум трішки сповільнять. Вже від 0.8 до 1 йдуть стабільні реалізації, які можуть вилітати рідко і в цілому показують себе з кращої сторони, не впливаючи взагалі на комфорт робочого процесу.

– Час на створення матеріалу – також шкала від 0 до 1 з дробовими значеннями, поділена на 10 значень. Від 0 до 0.3 входитиме найповільніше виконання, тобто створення простого матеріалу займатиме більше 30 хвилин. Вже починаючи від 0.4 до 0.6 входять рішення, в яких час на розробку зайняв від 10 до 30 хвилин. Від 0.7 до 0.9 йдуть рішення, які дозволили реалізувати матеріал від 5 до 10 хвилин, ну і звісно ж 1 відповідає за створення матеріалу до 5 хвилин.

– Різноманіття дій, яке буде останньою шкалою серед усі проміжних критеріїв. Значення змінюється теж від 0 до 1, при чому від 0 до 0.4 система працює лише як обмежена, тобто надає базовий функціонал, не надаючи глибшого редагування. Вже починаючи з 0.5 і закінчуючи 0.7 з’являється більш глибоке редагування з додаванням наприклад масок, фільтрів або в цілому більше можливостей для створення саме необхідного матеріалу. Ну і починаючи з 0.8 до 1 можна редагувати будь-який параметр незалежно і маніпулювати ним як завгодно.

– Оптимізація, для якої будемо брати параметр завантаження оперативної пам’яті, так як він постійний і його можна відстежувати. Вимірюватиметься все у Мегабайтах. Чим менше використовується пам’яті, тим краще.

– Кількість компонентів для створення матеріалу. Тут все просто – чим більше необхідно умовних одиниць, наприклад вузлів, для реалізації матеріалу, тим гірше.

Виділивши основні та проміжні категорії перейдемо до альтернатив, між якими і необхідно провести вибір або які треба порівняти один з одним:

– Доповнення Layer Painter на основі традиційної систем шарів. На GitHub є версія з відкритим кодом, яку можна редагувати як завгодно під свої потреби.

– Доповнення Ravage 2 з використанням системи рівнів. Є безкоштовна пробна версія, що дозволяє використовувати основний функціонал без додаткових можливостей, таких як запікання текстур.

– Стандартна класична система вузлів, вбудована у Blender.

– Доповнення, реалізоване для проведення експерименту на основі Blender API з використанням мови програмування Python.

Вкінці отримаємо ієрархію, зображену на рисунку 3.1.

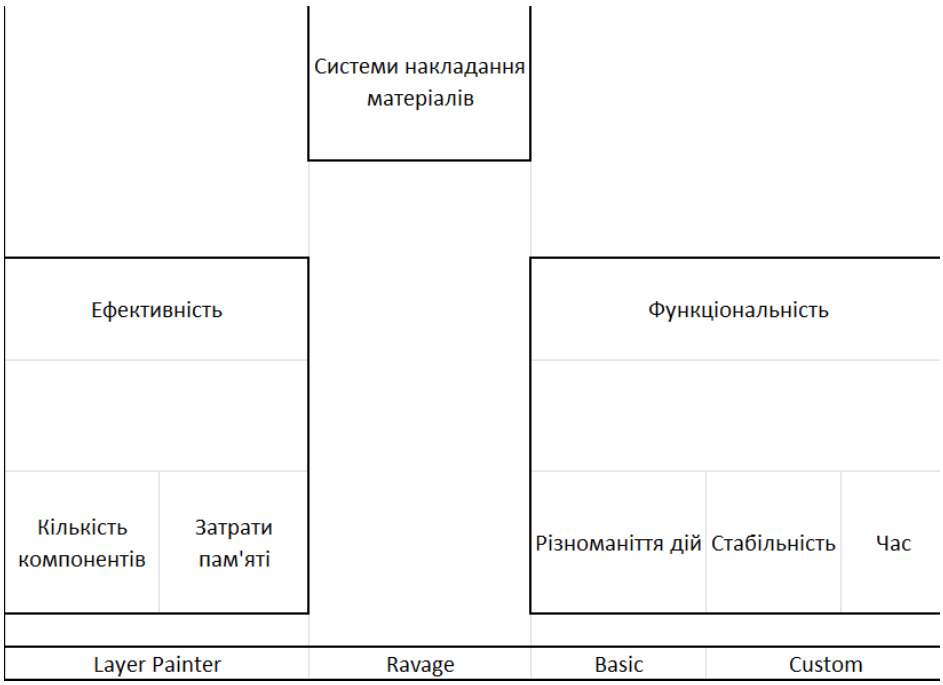

Рис. 3.1 – Зображення графу для методу аналізу ієрархій

Далі необхідно визначити важливість кожного критерію, як основних так і проміжних. Наприклад, якщо для нас ефективність є найважливішим

критерієм, необхідно призначити йому більше ваги, ніж іншому критерію. Власне ваги розподіляються експертом або кількома людьми з використанням шкали Сааті, що зображена на рисунку 3.2.

Definition	Standard values	Inverse values
The same importance	1	1
Weak dominance	3	1/3
Strong dominance	5	1/5
Very strong dominance	7	1/7
Absolute dominance	9	1/9
Intervalues	2, 4, 6, 8	1/2, 1/4, 1/6, 1/8

Рис. 3.2 – Шкала Сааті

Розглянемо перші два критерії – ефективність та функціональність. Якщо розглядати їх, то сама основна відмінність гібридного підходу, про яку постійно наголошувалося в другому розділі – це розширення функціоналу через об’єднання традиційних систем, а тому вони практично рівні, тобто значення буде рівне 2 за шкалою. Матрицю попарних порівнянь першого рівня зображено на рисунку 3.3.

Системи накладання матеріалів	Ефективність	Функціональність
Ефективність	1	0,5
Функціональність	2	1

Рис. 3.3 – Матриця попарних порівнянь основних критеріїв

Далі перейдемо до рівня з проміжними критеріями і надамо їм ваги аналогічні до попередньої матриці. Почнемо з матриці попарних порівнянь для критерію Ефективності. Найважливішим з двох цих критеріїв буде затрати пам’яті на підтримку матеріалу, при чому це досить суттєвий фактор, який майже повністю відповідає ефективності. Він буде переважати над кількістю

компонентів, хоч і не сильно, тобто за шкалою на рисунку 3.2 це буде значення проміжне, яке рівне 4. Пов’язано це з тим, що кількість компонентів впливає швидше на пряму швидкість роботи користувача з мережевою системою, а якщо використовуватиметься дуже простий матеріал, який вимагає лише створення одного або навіть двох рівнів, то даний критерій не матиме впливу взагалі. В більш комплексних матеріалах, де необхідно більш гнучке налаштування шейдеру та параметрів знадобиться щось схоже на групування з використанням Frame тощо.

Щодо проміжних критеріїв для Функціональності, то тут основним виділятиметься все таки стабільність та час, затрачений на задачу або проєкт. Без стабільності роботи немає сенсу намагатися створювати матеріал, так як вона не тільки збільшує затрати часу, але ще й може пошкодити цілий файл, в якому не тільки знаходиться інформація про матеріали, але й об’єкти та їх структура. Цей параметр має слабку перевагу над різноманіттям дій, та майже слабку перевагу над часом, відповідно значення за шкалою 3 та 2. Щодо різноманіття дій, то цей параметр є менш суттєвий порівняно із затраченим часом, а тому він трохи менш важливий, відповідно значення в матриці буде рівне 2. Матриці для проміжних критеріїв можна побачити на рисунку 3.4.

Функціональність	Різноманіття дій	Стабільність	Час	Ефективність	Кількість компонентів	Затрати пам'яті
Різноманіття дій	1	0,33333333	0,5	Кількість компонентів	1	0,25
Стабільність	3	1	1,5	Затрати пам'яті	4	1
Час	2	0,66666667	1			

Рис. 3.4 – Матриці попарних порівнянь проміжних критеріїв

Для побудови останніх п’яти матриць попарних порівнянь використаємо дані, про які вже йшла частково мова у попередньому підрозділі. Для початку, як отримано ці дані:

– Для проведення дослідження використовувався ноутбук з процесором i5-8250U та вбудованою графікою, 16 гігабайт DDR4 оперативки з частотою 2400 Мегагерц та на накопичувачі SSD, який заповнений на 70%.

– Для керування та навігації використовувався Touchpad ноутбуку та клавіатура з Numpad.

– Відкривається Blender з включеними усіма доповненнями, включаючи власне розроблене для того, щоб усі робочі області були в однакових умовах, бо включення навіть одного модуля під час тестування змінювало затрати пам’яті, не кажучи про те, що це може наприклад вплинути на час, затрачений на матеріал.

– Видаляються лишні об’єкти камери та освітлення, і використовується стандартний куб, в якому стандартну UV-розгортку замінено розгорткою на основі Cube Projection. Крім того для кубу створюється новий базовий матеріал, старий видаляється. Будь-яке додавання іншого об’єкту або матеріалів дасть навантаження на пам’ять.

– Для усіх тестів заготовлюються два набори текстур, спочатку тільки з Base Color, потім з Base Color, Roughness та Normal Map OpenGL для тесту багатошаровості з використанням маски у формі букви “T”, що є скороченням від тест, з числом ітерації експерименту. Це зроблено для того, щоб не було жодних відмінностей, окрім власне підходів та їх ефективності роботи.

– Вигляд сцени змінюється на Shader Node Editor з Material Preview з однаковим оточенням, вікно доповнення розширюється до половини екрану, особливо актуально для доповнення Ravage та розробленого на сонові гібридної системи.

– Для доповнення, що використовує гібридний підхід, додається запис про використання текстур вже всередині програми, додавши їх перед тим. Це зумовлено кривизною реалізації та сирим інтерфейсом, які сильно вплинуть при розробці комплексних матеріалів або матеріалів, що використовують більше двох текстур.

– Початком створення матеріалу є старт використання доповнення з додаванням рівнів, або додаванням вузлів. Він необхідний буде для розрахунку загального часу виконання, та й прибирає лишнє нагромадження з повторним створенням сцени та її налаштуванням. Кінцем створення вважатиметься повністю завершений матеріал, у якого всі області кольору виставлені

правильно і маска застосована успішно. Приклад зображений на рисунку 3.5 з початком тесту пам’яті.

Рис. 3.5 – Приклад готового матеріалу з першої ітерації експерименту

Усього було проведено десять основних ітерацій, починаючи з нульової і закінчуючи дев’ятою з урахуванням помилок та похибок при розрахунках та замірах часу, витраченому на реалізацію матеріалів. Для кожної ітерації використовувалося по дві текстури кольору та маска. Результати усіх ітерацій для вибору значень відповідно до шкал подані на рисунку 3.6.

Час (E1)	Ravage2Lite	Blender	LayerPainter	WiseShader
0	00:42.73*	00:40.84**	00:40.49*	00:39.59*
1	00:41.82	00:39.40*	00:40.15	00:40.09*
2	00:43.80	00:37.94*	00:39.08	00:38.12*
3	00:43.75	00:39.84*	00:38.62	00:37.68*
4	00:40.12	00:39.57*	00:39.34	00:38.22*
5	00:40.85	00:38.24*	00:38.86	00:37.48*
6	00:40.46	00:39.09*	00:38.05	00:38.00*
7	00:39.15	00:39.05*	00:40.52	00:38.36*
8	00:39.20	00:38.89*	00:37.24	00:37.18*
9	00:38.69	00.39.74*	00:38.33	00:37.70*
	174.1 MB	173.5 MB	175.3 MB	173.6 MB
		Знаючи, що треба лише три текстури скопійовано рівно три		У програму вже додано текстури, без них + 5 секунд

Рис. 3.6 – Час та пам’ять, необхідні для створення матеріалу з двома базовими текстурами та маскою

Як можна побачити, оранжевим виділені самі перші тести, так як після цього можна було на автоматі відкривати текстури, нажимати на створення рівнів тощо. Зеленим виділено ті заміри, в яких не було жодних додаткових проблем та умов. Для класичного підходу з використанням вбудованого редактору Blender такою умовою було копіювання основної системи кілька разів з заміною лише текстур, коли як при використанні готового рішення дві текстури кольору були вже додані в редактор, що займало в середньому 5 секунд. Тут також важливо уточнити про пам’ять. Під час тестування додатковим параметром, який дуже сильно впливав на її використання виявився перед показ текстури, який вбудований у Shade Node Editor. Ще одним параметром було відкриття додаткових вікон або перехід в інше меню. Тож дані бралися лише після перезапуску програми. В цілому використання пам’яті найбільше залежало саме від текстур, що було перевірено з використанням матеріалів деревини, які можна було побачити на рисунку 3.5 і результати тесту подані були в пункті 3.3 та зображені на рисунку 3.7.

	Тест E2*
Layer Painter (Layer)	к. 104, з. 218.7, р. 0.4, с. 0, ч. 0.7.
Ravage (Layer)	к. 27, з. 153.0, р. 0.4, с. 1, ч. 1.
Basic (Node)	к. 16, з. 165.7, р. 1, с. 1, ч. 0.4.
Custom (Node + Layer)	к. 23, з. 141.3, р. 0.7, с. 0.4, ч. 0.7.

Рис. 3.7 – Результат тестування з текстурами дерева

Що ж до кількості елементів, то найбільше з них було використано в Layer Painter, який для кожного рівня зразу додавав усі можливі ноди, а для параметрів просто ховав ті, що не використовувалися, через що використання пам’яті було найбільшим серед усіх систем. Наступними йшли Ravage та

Custom, перший з яких створював в цілому лише основні властивості, додаючи вузол комбінації. Розроблений Custom у свою чергу використав менше надаючи більш широкі можливості до редагування, хоч і вимагав механічного втручання через нереалізованість частини функціоналу, як наприклад маски. Найкращими у плані застосування елементів і найефективніший по пам’яті виявився вбудований в Blender редактор, який хоч і не надавав великої свободи маніпулювання властивостями, але він дозволяв це зробити при необхідності, чим зачіпаємо пункт різноманіття дій.

Найбільш обмеженим з усіх є Ravage, який при додаванні власного вузла або видаленні вже існуючого видавав помилку, щоб користувач повернув його. Це зумовлено тим, що Ravage ще на етапі створення матеріалу просить застосувати його властивості, щоб можна було працювати з ним. Далі йде Layer Painter, який не ламався при додаванні нових вузлів, але при видаленні вже існуючих міг просто вилетіти. На другому місці розташувався Custom, в якому можна спокійно додавати ноди, видаляти вже створенні та міняти їх на інші. Єдине, що при поверненні назад до рівневої частини підходу програма вилітала при спробі видалити вже неіснуючий вузол. Ну і звісно ж найкраще себе проявив гнучкий підхід на основі вузлів в Blender.

Після проведення цього експерименту виникло питання – наскільки сильно вплине недороблений інтерфейс на можливості програми та час, необхідний для досягнення результату, і наскільки стабільно вони себе поводитимуть, не кажучи про міни в затратах пам’яті та можливостях. Для того, щоб перевірити дану теорію було зроблено тест уже з трішки складнішим завданням:

– Усі початкові умови зберігаються.

– Замість двох текстур кольору і однієї текстури маски тепер використовуватимуться шість текстур та одна текстура маски, при чому текстури Normal Map та Roughness мають іншу область кольору – Non-color.

– Намагатися видаляти утворені вузли, додавати нові, використовувати відміну або повернення до минулої дії тощо для перевірки стабільності роботи доповнень та альтернатив.

Результати нового експерименту можна побачити на рисунку 3.8.

Час (E2)	Ravage2Lite	Blender	LayerPainter	WiseShader
0	01:57.01**	02:23.85**	01:11.11	02:50.07***
1	01:54.25*	02:24.41*	01:14.53	02:48.41**
2	01:55.59*	02:23.81*	01:07.69	02:48.84**
	430.3 MB	429.8 MB	432.1 MB	430.1 MB
	Не застосував автоматично область, з нею ще + 15 с	Не застосував область, з нею ще + 15 с	Автоматично виставив область	Не застосував область, так як не виніс її в програмі

Рис. 3.8 – Час та пам’ять при другому експерименті

На цей раз найгірше показав себе Custom, в якому вже через додавання кожної текстури витрачалося приблизно 25 секунд, не кажучи про кривизну реалізації, яка не дозволила змінити область застосування. У свою чергу Blender та Ravage теж вимагали додаткового застосування областей кольорів, не кажучи про різке зростання кількості вузлів у останнього. Найкраще ж себе проявив Layer Painter, який виставляв автоматично області відповідно до типу текстури та дозволяв додавати маску найкраще з усіх, надаючи цілий список можливих варіантів. Єдине, що не змінилося порівняно з попереднім експериментом – це розподілення між чотирма альтернатива за кількістю елементів, використанням пам’яті та різноманіттям дій. Щодо стабільності, то вона ще більше змінилася та стала більш явною проблема в Layer Painter та Custom в Blender. Перший вилітав при будь-якій спробі відмінити дію або при додаванні третього рівня і його видаленні. Розроблений Custom ж при спробі видалити редаговані вузли постійно вилітав або напрям відмовлявся прибирати їх з області Node Editor. Єдине, що Ravage та Blender ніяк в цьому плані не змінилися і працювали взагалі без збоїв.

На основі отриманих даних можна робити матриці попарних порівнянь для кожного з проміжних критеріїв, почавши з різноманіття дій. Найбільшу свободу в редагуванні та доповненні матеріалу надавав Blender, після якого

йшов Custom, а тому перший дуже сильно переважає над усіма іншими, коли як Custom трохи краще за Ravage та Layer, а тому відповідно значення 7 та значення 2 за шкалою.

Наступним критерієм візьмемо кількість елементів, в якій найменше використовував знову ж таки Blender, після якого йде Custom, ну і далі Ravage та Layer відповідно, а тому перші три будуть досить вагомо переважати над останнім.

Затрати пам’ять вже цікавіші, так як дуже сильно впливають які текстури використовувалися та скільки вузлів застосовано. Якщо огляди в цілому картину, то найкраще себе проявили Custom та Blender з найкращою ефективністю, так як Custom використовував вже стандартні методи імплементації на Python, які непогано себе показали, хоч і при повній реалізації можливі просадки, як наприклад при додаванні елементів інтерфейсу, які будуть засмічувати пам’ять. Найгірше з усіх за цим критерієм є Layer Painter, який постійно старається створити величезну кількість вузлів, застосовуючи лише 15-20% з них, що веде до ще більшого засмічення пам’яті. Може здатися, що ці пару мегабайт це ніщо, але це були тести з простими матеріалами, а що ж буде при застосуванні десятка текстур та різних вузлів.

За стабільністю найгіршим є Layer Painter, який що не крок – це можливість вильоту. Трохи краща ситуація спостерігається у Custom, хоч він так само сирий, особливо в плані інтерфейсу. Найкращими знову ж таки є Ravage та Blender, а тому вони будуть переважати над попередніми альтернативами.

Останнім серед проміжних критеріїв є час, з яким найбільше суперечностей в тестуванні. З одного боку при використанні простих матеріалів, або навіть одного, найшвидшим з усіх буде Ravage та Layer Painter, так як застосовується підхід на сонові рівнів, який не вимагає гнучкості. На рівні з ними також розташувався Custom, оскільки він комбінує обидва підходи в собі, не враховуючи те, що необхідно додавати текстури через кривизну реалізації. Та як тільки йде хоча б якесь ускладнення, або збільшення кількості наборів текстур в матеріалі, то різко що Ravage, що Custom просідають, коли як

Layer Painter у випадку з кількома наборами має заготовлений функціонал, який майже в один клік підтягує увесь матеріал як рівень. У випадку ще й з Custom дуже сильно проявляється нестача функціоналу та некомфортний інтерфейс, через які необхідно було фактично боротися з системою. Забравши цю проблему та створивши підтягування текстур замість просто вибору з вже наявних можна скоротити цей час до рівня Ravage. Так, якщо забрати додавання текстур до програми, економія йде майже в пів хвилини. Тому, щоб дані були найбільш відповідні, візьмемо результати експериментів з текстурами дерева та двома текстурами кольору. Матриці попарних порівнянь подано на рисунку 3.9.

Кількість компонентів	Layer Painter	Ravage	Basic	Custom
Layer Painter	1	0,2	0,14	0,166666667
Ravage	5	1	0,25	0,5
Basic	7	4	1	3
Custom	6	2	0,33	1

Різноманіття дій	Layer Painter	Ravage	Basic	Custom
Layer Painter	1	1	0,14	0,5
Ravage	1	1	0,14	0,5
Basic	7	7	1	5
Custom	2	2	0,2	1

Час	Layer Painter	Ravage	Basic	Custom
Layer Painter	1	0,5	2	1
Ravage	2	1	3	2
Basic	0,5	0,33333333	1	0,5
Custom	1	0,5	2	1

Затрати пам'яті	Layer Painter	Ravage	Basic	Custom
Layer Painter	1	0,3333	0,5	0
Ravage	3	1	2	1
Basic	2	0,5	1	0
Custom	4	2	3	1

Стабільність	Layer Painter	Ravage	Basic	Custom
Layer Painter	1	0,1667	0,17	1
Ravage	6	1	1	4
Basic	6	1	1	4
Custom	2	0,25	0,25	1

Рис. 3.9 – Матриці попарних порівнянь альтернатив

На основі цих даних використовується метод аналізу ієрархій, результат якого подано у аналізі даного експерименту. Якщо врахувати результати експерименту зі складнішими матеріалами, то Custom проявить себе гірше всіх, а Layer Painter наздожене Ravage через недоліки та труднощі, описані в пункті 3.2.

Для покращення системи необхідною умовою буде розробити таку систему в доповненні, яка б могла відслідковувати кожен вузол в конкретному каналі, для чого є кілька варіантів реалізації, а саме через користувацький

вузол, а також прописану власну структуру. Та даного рішення як мінімум достатньо для підведення загального підсумку щодо роботи такого підходу.

3.4 Висновок до третього розділу

Вибір між гібридною системою для нанесення матеріалів і традиційними системами на основі вузлів або шарів залежить від різних факторів, включаючи конкретні вимоги проєкту, уподобання та досвід користувачів, а також бажаний баланс між гнучкістю, простотою, і ефективність. Гібридна система поєднує в собі найкраще з обох світів, пропонуючи гнучкість систем на основі вузлів з організаційною простотою систем на основі рівнів. Це дозволяє налаштовувати складні матеріали, зберігаючи інтуїтивно зрозуміле шарування та змішування. Залежно від реалізації, добре спроектована гібридна система може запропонувати також покращену ефективність і зручність використання порівняно з іншими системами.

З іншого боку, користувачі, які вже володіють традиційними робочими процесами на основі вузлів або рівнів, можуть віддати перевагу використанню знайомих інструментів та методологій. Перехід на гібридну систему може вимагати додаткового навчання та налаштування. Крім цього, слід враховувати сумісність з існуючими робочими процесами, конвеєрами та інструментами. Якщо проект вже значною мірою покладається на системи на основі вузлів або рівнів, перехід до гібридного підходу може спричинити проблеми сумісності. Додати до цього щ складність і масштаби проєктів, які відіграють важливу роль у визначенні придатності різних систем застосування матеріалів. Для простих проєктів може бути достатньо традиційних вузлових або шарових систем. Однак для більш складних проектів із складною взаємодією матеріалів і процедурними ефектами гібридна система може запропонувати переваги з точки зору контролю.

4 ОХОРОНА ПРАЦІ ТА БЕЗПЕКА В НАДЗВИЧАЙНИХ СИТУАЦІЯХ

4.1 Основні положення, стан електробезпеки в Україні та дія електричного струму на людину

Згідно з чинними нормативно-правовими актами, електробезпека полягає в системі організаційних та технічних заходів та засобів, спрямованих на захист людей від шкідливого та небезпечного впливу електричного струму, електричної дуги, електричного поля та статичної електрики. Травми, які виникають в результаті дії електричного струму або електричної дуги на організм людини, відомі як електротравми, при чому вони можуть виникати при проходженні або без проходження струму через тіло людини, наприклад, через опіки або засліплення електричною дугою.

Термін "електротравматизм" використовується для опису сукупності електротравм [46]. За статистичними даними, українські промислові травми, пов'язані з електричним струмом, становлять близько 1% від загальної кількості виробничих травм, при цьому до 20% є смертельними. Варто відзначити, що до 80% смертей від електротравм стаються внаслідок нещасних випадків з напругою до 1000 В, оскільки низьковольтні системи електропостачання доступні практично всім працівникам. У виробничих умовах кількість електротравм становить близько 500 випадків щорічно, з них приблизно 150 призводять до смерті. Оскільки електроенергія широко використовується у всіх сферах господарства, проблема електробезпеки набуває особливого значення. Порівняно з іншими видами травм, електротравми мають кілька особливостей, так як людина не може визначити наявність напруги безпосередньо, тому дія струму зазвичай є раптовою. Крім цього, струм впливає на всі органи та системи організму, що може призвести до серйозних порушень їх функціонування.

Вплив електричного струму на організм має різноманітні прояви, включаючи термічні, електролітичні та біологічні наслідки. Термічний ефект виявляється у нагріванні тканин, випаровуванні вологи та може призводити до

опіків та розриву тканин. Електролітичний ефект полягає в розкладанні органічних рідин та порушенні їх складу. Біологічна реакція полягає в збудженні та порушенні внутрішніх процесів організму, що може виявлятися у випадковому скороченні м'язів та порушенні діяльності важливих органів, таких як серце та легені.

Електричні травми можна умовно розділити на місцеві, загальні та комбіновані [47]. До місцевих належать опіки, механічні ушкодження та електрофтальмія. Опіки є найбільш поширеними видами електротравм, які, залежно від умов, бувають контактними, дуговими або комбінованими. Контактні опіки, як правило, виникають в установках з невеликою напругою через тепловий вплив струму і пошкоджують прилеглі тканини. Дугові опіки можуть виникати внаслідок виникнення електричної дуги внаслідок короткого замикання між елементами установки або між елементами та тілом людини, що може призвести до серйозних ушкоджень і навіть смерті потерпілого. Механічні ушкодження виникають внаслідок неконтрольованих судорожних скорочень м'язів, які виникають під впливом струму. Електрофтальмія - це запалення зовнішніх оболонок очей, викликане ультрафіолетовим випромінюванням електричної дуги. Це запалення зазвичай починається через кілька годин після опромінення і характеризується почервонінням шкіри та слизових оболонок повік, сльозами, гнійними виділеннями та підвищеною чутливістю до світла. Його тривалість зазвичай становить 3-5 днів.

Загальні електричні травми включають в себе електричний удар, який може привести до судорог, зупинки дихання і серцевої діяльності через порушення роботи різних груп м'язів, що може призвести до втрати свідомості та навіть клінічної смерті. Комбіновані об’єднують місцеві та загальні.

4.2 Ергономічні вимоги до організації робочих місць користувачів комп’ютерів

Організація робочого простору для користувачів комп'ютерів має відповідати стандартам, визначеним у НПАОП 0.00-1.31-10 та НПАОП 0.00-

7.15-18 [47]. Наприклад, площа одного робочого місця з використанням відеодисплейного терміналу (ВДТ) повинна бути не менше 6,0 м², а об'єм приміщення - не менше 20 м³. Робочі місця з ВДТ слід розміщувати на відстані не менше 1 м від стіни з вікнами; відстань між бічними поверхнями ВДТ має бути не менше 1,2 м; відстань між задньою поверхнею одного ВДТ та екраном іншого - не менше 2,5 м; прохід між рядами робочих місць має бути не менше одного метра. Також важливо враховувати розміри меблів (див. рисунок 4.1, взятий з [48]) для комп'ютеризованих робочих місць, таких як стіл для ВДТ, який має ширину 1200 мм і глибину 800 мм [48]. Щоб уникнути відбиття природного світла на екрані ВДТ, рекомендується розміщувати їх вздовж стіни з вікнами. Для зменшення впливу шуму з сусідніх робочих місць і забезпечення кращої концентрації під час виконання складних завдань, рекомендується використовувати перегородки висотою 1,5 - 2 м для відокремлення робочих зон.

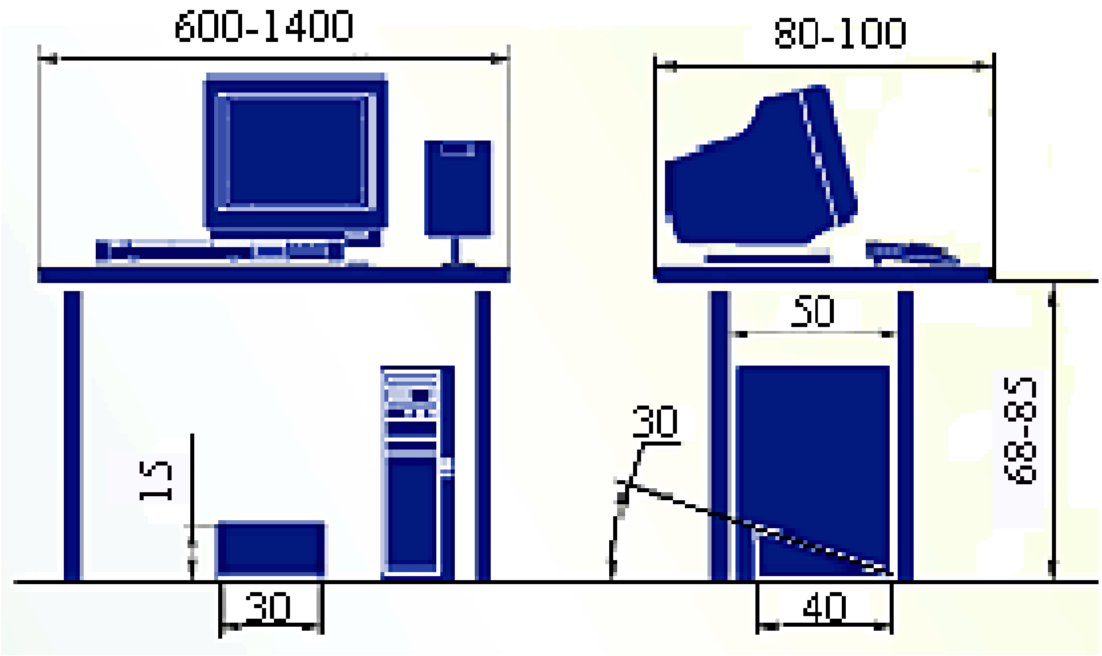

Рис. 4.1 – Організація робочого місця

Ефективне розташування відео термінала, клавіатури та принтера на робочому місці є ключовим чинником для забезпечення безпеки та зручності користувачів комп'ютерів. Правильне розташування екрана (дисплея) повинно

забезпечувати комфортне спостереження зором у вертикальній площині під кутом ± 30° від лінії зору оператора. Найкращі умови для зорового сприйняття досягаються, коли верхній край екрана знаходиться на рівні очей, а погляд спрямований вниз на центр екрана. Оскільки робота за комп'ютером найбільш комфортна з нахилом голови вперед на близько 20° від вертикалі (що сприяє розслабленню м'язів шиї), то екран також повинен бути нахилений назад на 20° від вертикалі. Рекомендована оптимальна відстань від очей користувача до екрана та клавіатури повинна бути не менше 600 мм, з урахуванням розміру символів. Наприклад, при діагоналі екрана 35 см, відстань до очей має бути 60–70 см, при діагоналі 43 см – 70 см, при діагоналі 48 см – 80 см. Для клавіатури комп'ютера на робочому столі має бути виділений достатній простір для переміщення та поворотів. Положення клавіатури і кут її нахилу повинні відповідати індивідуальним побажанням користувача. Кут нахилу клавіатури може змінюватись від 5 до 10 градусів. Якщо клавіатура не має вбудованого простору для долонь, їх слід розташовувати на відстані не менше 100 мм від краю столу в зоні оптимального моторного поля. Також можливе розташування клавіатури на спеціальній робочій поверхні, окремо від основного столу. Розташування принтера або іншого пристрою введення-виведення інформації на робочому місці повинно забезпечити зручний доступ до екрана комп'ютера і можливість комфортного керування пристроєм в межах моторного поля (висота від 900 до 1300 мм, глибина від 400 до 500 мм).

4.3 Організація і функціонування системи управління охороною праці

Україна має складну систему управління охороною праці [49], яка включає різні рівні державної влади, управлінські структури підприємств і організацій, а також трудові колективи. Власне СУОП [50] поділяється на дві основні групи, а саме ланки, що забезпечують вирішення загальних питань охорони праці, включаючи законодавчо-нормативні, науково-технічні, соціально-економічні аспекти тощо, та ланки, які забезпечують безпеку праці

на рівні конкретних організацій і підприємств. До перших відноситься Кабінет Міністрів України, який відповідає за впровадження державної політики у сфері охорони праці, орієнтуючись на стан безпеки праці в країні, організує розробку загальнодержавних програм для покращення цього стану, затверджує їх і контролює виконання, а також визначає функції органів виконавчої влади з управління питаннями охорони праці та нагляду за нею. Крім даної установи ще до першої групи входить Верховна Рада України, яка співпрацює з відповідними органами державної виконавчої влади у визначенні державної політики у галузі охорони праці. Вона також розглядає питання про удосконалення та розвиток законодавства щодо охорони праці, а також соціальні аспекти, пов'язані з умовами та безпекою праці. На останок ще виділяють Держгірпромнагляд України, Фонд соціального страхування від нещасних випадків та центральні з місцевими органами влади. Перша організація втілює державну стратегію в сфері гірничого нагляду та промисловості, і співпрацює з міжнародними організаціями щодо охорони праці. Рішення установи обов'язкові для виконання всіма міністерствами, центральними органами державної влади, місцевими державними адміністраціями, радами народних депутатів і підприємствами. Фонд соціального страхування від нещасних випадків координує страхову діяльність у сфері охорони праці. Серед центральних органів влади, Міністерство праці і соціальної політики здійснює державну експертизу умов праці та контроль за якістю атестації робочих місць, коли як інші міністерства визначають політику конкретної галузі, розробляють заходи щодо безпеки праці та навчають працівників правилам охорони праці. Для забезпечення виконання вказаних завдань в апаратах міністерств і інших центральних органів державної виконавчої влади формуються служби охорони праці. Останніми з першої групи розглянемо місцеві державні адміністрації та органи місцевого самоврядування, які розробляють місцеві програми для покращення безпеки та гігієни праці, здійснюють контроль за дотриманням нормативних актів на місцевому рівні.

До другої групи належать управлінські органи підприємств, які забезпечують виконання вимог законів та нормативних актів для створення безпечних умов праці та запобігання травм і захворювань серед працівників, вирішуючи всі питання, пов'язані з виробництвом. У своїй діяльності вони взаємодіють з комісією з охорони праці підприємства, якщо така є, профспілками та представниками трудового колективу.

Правова основа системи управління охороною праці включає в себе Конституцію України, Кодекс законів про працю України, а також законодавство про охорону праці та державне соціальне страхування від нещасних випадків на виробництві та професійних захворювань, які призвели до втрати працездатності. Крім того, до складу правової бази входять накази та розпорядження Президента України, а також акти управління, такі як розпорядження та постанови Кабінету Міністрів, Держгірпромнагляду, Міністерства охорони здоров'я, Міністерства праці і соціальної політики, а також інших органів, які мають директивні повноваження з питань охорони праці.

Вихідні параметри СУОП визначаються на основі вимог нормативних документів, аналізу виробничої ситуації та інших факторів, тому їх можна віднести до категорії багатоконтурних систем, які піддаються програмуванню. Це пояснюється складністю об'єкта управління та його великою інерційністю.

При роботі в Blender з системою створення матеріалів, яка розглянута у результаті дослідження, користувачі використовуватимуть комп’ютер, тому необхідно обговорити питання електробезпеки при роботі з цим обладнанням. Головна мета електробезпеки полягає в зменшенні ризику негативного впливу електричного струму на людину. Цій меті можна досягти шляхом:

– створення безпечних та надійних конструкцій електроустановок;

– застосування організаційних та технічних заходів для безпечної експлуатації електроустановок та використання електричної енергії;

– використання технічних засобів захисту.

Залежно від засобів електробезпеки, усі електротехнічні вироби поділяються на 5 класів:

– Клас 0 – електрична установка має лише робочу ізоляцію як засіб захисту.

– Клас I – має як ізоляцію, так і заземлення. Клас 0I проміжний.

– Клас II забезпечує подвійну ізоляцію.

– Клас III передбачає використання лише низької напруги (до 42 В) для живлення установки.

Комп’ютер швидше представляє собою конструкцію з кількох таких пристроїв, а тому його не можна віднести до однієї групи в цілому. Але якщо розглянути конкретно саму найнебезпечнішу частину, а саме блок живлення, то його можна віднести до класу II. Стан ізоляції провідних частин системи забору повинен відповідати вимогам "Правил використання електроустановок". Згідно з цими правилами, передбачено регулярне перевірка ізоляції: двічі на рік у приміщеннях зі складними умовами і підвищеною вологістю, та один раз на рік у приміщеннях з нормальним середовищем. Ізоляція створює великий опір, що перешкоджає протіканню струму через неї. Опір ізоляції кожної системи повинен бути не менше 0,5 МОм (500 000 Ом). У разі зниження опору ізоляції на 50% від початкового, ізоляцію потрібно замінити.

4.4 Висновок до четвертого розділу

В цілому підсумовуючи про системи управління охороною праці, то на конкретній організації вона завжди є багаторівневою, де верхній рівень представлений державним управлінням, а нижній – управлінням на конкретному об'єкті. Крім цього, описано ергономічні вимоги щодо комп’ютерного робочого місця та як правильно його облаштувати в залежності від пристроїв. Так наприклад чим більшою йде діагональ монітора, тим на більшу відстань необхідно сидіти для збереження здоров’я очей. Також, в цьому розділі піднято питання та подано інформацію щодо електробезпеки при роботі з комп’ютерним устаткуванням, та про можливі електротравми, які можуть виникнути.

ВИСНОВКИ

Підсумовуючи, хоча гібридна система для створення та накладання матеріалів пропонує переконливі переваги з точки зору гнучкості, ефективності та налаштування, її придатність залежить від багатьох різних факторів, характерних для кожного проєкту та потреб користувачів, які шукають власне баланс між гнучкістю систем на основі вузлів для комплексного налаштування з отриманням бажаного результату і простотою розуміння та навчання систем на основі шарів. Для художників конкретно у програмі Blender добре спроєктована гібридна система дійсно може запропонувати краще рішення для застосування матеріалів та їх редагування, над чим навіть розробники програмного рішення вже задумувалися. Однак важливо ретельно оцінити вимоги, робочі процеси та досвід користувачів в цілому, можливо навіть розробити опитування, щоб визначити найбільш прийнятний підхід до створення матеріалу в будь-якому контексті,. Завдяки ж отриманим з дослідження даним на основі чотирьох програмних рішень вдалося визначити релевантність створення гібридної системи.

В першому розділі було подано інформацію про стан галузі комп’ютерної графіки та розглянуто перспективи проведення досліджень, особливо з урахуванням розвитку AR та VR технологій та впровадження автоматизації з використанням штучного інтелекту, після чого розглянуто категорії та різновиди комп’ютерної графіки, виділено область статичної та динамічної, стилізованої та реалістичної тривимірної комп’ютерної графіки для проведення дослідження. Після цього проаналізовано програмне забезпечення для тривимірної графіки, щоб обрати найоптимальніший варіант для проведення дослідження, яким взято Blender, в якому основними вимогами фактично є можливість розробки власного доповнення і доступ до коду програми, а також реалізовано одну з традиційних систем накладання матеріалів – систему вузлів. Вкінці досліджено системи накладання матеріалів у кожному з перелічених програмних продуктах та сформовано висновки щодо області проведення та програмного забезпечення Blender.

Переходячи до другого розділу вже починається більш глибоке дослідження з теоретичної точки зору, де все починалося з розписування традиційних систем створення та накладання матеріалів, а також розглянуто гібридну систему як новий підхід до створення матеріалів. До цього як доповнення досліджено переваги та недоліки кожної з систем, особливо акцентовано увагу на комфорті та на гнучкості рішень, і що система на основі вузлів доволі гнучка, але має набагато важчий для розуміння інтерфейс, коли як у системи з шарами або рівнями ситуація навпаки. Вкінці подано порівняльний опис підходів до реалізації гібридного методу створення матеріалу через використання доповнень, написаних розробником або командою, та через користувацький вузол.

В розділі з обчислювальним експериментом спочатку описано та розроблено доповнення для програми Blender, яке додає до програми гібридну систему з базовою реалізацією та функціоналом, де також розглянуто труднощі розробки доповнення та можливі способи покращення та оптимізації імплементації. Після цього проведено декілька експериментів і отримано дані ефективності, на основі яких застосовано метод аналізу ієрархій для порівняння між собою на практиці традиційних та гібридної систем. Так наприклад при більш складніших реалізаціях все почало впиратися в сиру реалізацію доповнення, коли як для простіших доповнення показало себе одним з найшвидших. Вкінці проаналізовано результати проведеного дослідження щодо ефективності гібридного підходу, який показав себе перспективним.

В останньому розділі проаналізовано поняття електробезпеки та різні аспекти дії електричного струму на живу тканину, включаючи термічну, електролітичну та біологічну дію, а також види електротравм і їх наслідки. Також описано функціонування СУОП та підкреслено важливість правильного розташування компонентів робочого місця, таких як екран, клавіатура та принтер, для забезпечення комфорту та запобігання захворювань.

ПЕРЕЛІК ДЖЕРЕЛ

1 Озіранець, Віталій Степан Володимирович. "Розробка дизайну та реалізація 3D моделей для трейлеру комп'ютерної гри "Echo of Sunset" засобами Blender." (2022).

2 Eck, David J. "Introduction to Computer Graphics." (2021).

3 Озіранець В. С. Технологічні форми віртуальної реальності / В. С. Озіранець // Збірник тез II Міжнародної наукової конференції молодих учених та студентів "Філософські виміри техніки", 4-5 грудня 2019 року. — Т. : ТНТУ, 2019. — С. 105. — (Науково-технічний прогрес: проблеми та перспективи).

4 Matplotlib documentation – Matplotlib 3.8.4 documentation. *Matplotlib – Visualization with Python*. URL: https://matplotlib.org/stable/index.html (дата звернення: 04.03.2024).

5 Blender 4.1 Manual. *Blender Documentation - blender.org*. URL: https://docs.blender.org/manual/en/latest/index.html (дата звернення: 05.03.2024).

6 Clinton-Lisell, Virginia. "Listening ears or reading eyes: A meta-analysis of reading and listening comprehension comparisons." Review of Educational Research 92.4 (2022): 543-582.

7 Husain, Akhlaq, et al. "Fractals: An eclectic survey, part II." Fractal and Fractional 6.7 (2022): 379.

8 Viro, Vi. "Problems in turning concept art into 3D objects: concept art to 3D object pipeline." (2022).

9 Озіранець В. С. Аналіз методів моделювання в Blender / Озіранець В. С. // VI Міжнародна студентська науково-технічна конференція "Природничі та гуманітарні науки. Актуальні питання", 27-28 квітня 2023. — Т. : ТНТУ, 2023. — С. 161–162. — (Інформаційні технології).

10 Blender: Texture Painting – Simply Explained. *All3DP*. URL: https://all3dp.com/2/blender-texture-painting-simply-explained/ (дата звернення: 08.03.2024).

11 Physically-Based Rendering, And You Can Too!. *Marmoset*. URL: https://marmoset.co/posts/physically-based-rendering-and-you-can-too/ (дата звернення: 11.03.2024).

12 PBR Texture Conversion | Marmoset. *Marmoset*. URL: https://marmoset.co/posts/pbr-texture-conversion/ (дата звернення: 11.03.2024).

13 3DS Help Manual. *Product Documentation | Autodesk Help*. URL: https://help.autodesk.com/view/3DSMAX/2023/ENU/ (дата звернення: 14.03.2024).

14 Maya Help Manual. *Product Documentation | Autodesk Help*. URL: https://help.autodesk.com/view/MAYAUL/2025/ENU/ (дата звернення: 15.03.2024).

15 Документація до Cinema4D. *Maxon Online Documentation*. URL: https://help.maxon.net/c4d/en-us/ (дата звернення: 18.03.2024).

16 Документація до ZBrush. URL: https://docs.pixologic.com/user-guide/ (дата звернення: 19.03.2024).

17 The complete beginners guide to Blender nodes, Eevee, Cycles and PBR - Artisticrender.com. *Artisticrender.com*. URL: https://artisticrender.com/the-complete-beginners-guide-to-blender-nodes-eevee-cycles-and-pbr/ (дата звернення: 21.03.2024).

18 Differences between Displacement, Bump and Normal Maps. URL: https://www.pluralsight.com/blog/film-games/bump-normal-and-displacement-maps (дата звернення: 21.03.2024).

19 Van Gumster, Jason. "Blender for dummies." (2020).

20 Leiro, L. Suaya, and Marc Garrigó. "Development of a Node-Based Material Editor." (2022).

21 Blain, John M. "The complete guide to Blender graphics: computer modeling & animation." (2019).

22 Layer stack | Substance 3D Painter. *Adobe Help Center*. URL: https://helpx.adobe.com/substance-3d-painter/interface/layer-stack.html (дата звернення: 21.03.2024).

23 Nodes. *Houdini – 3D modeling, animation, VFX, look development, lighting and rendering | SideFX*. URL: https://www.sidefx.com/docs/houdini/nodes/index.html (дата звернення: 21.03.2024).

24 Tree Generator | SideFX. *Houdini – 3D modeling, animation, VFX, look development, lighting and rendering | SideFX*. URL: https://www.sidefx.com/tutorials/tree-generator/ (дата звернення: 21.03.2024).

25 How to Make Your Models Look Realistic With Procedural Textures in Blender. URL: https://www.makeuseof.com/procedural-textures-in-blender-realistic-models/ (дата звернення: 21.03.2024).

26 Welcome to the Photoshop User Guide. *Adobe Help Center*. URL: https://helpx.adobe.com/photoshop/user-guide.html (дата звернення: 20.03.2024).

27 Welcome to the Krita 5.2 Manual!. *Welcome to the Krita 5.2 Manual! – Krita Manual 5.2.0 documentation*. URL: https://docs.krita.org/en/index.html (дата звернення: 20.03.2024).

28 Baechler, Oscar, and Xury Greer. "Blender 3D By Example: A project-based guide to learning the latest Blender 3D, EEVEE rendering engine, and Grease Pencil." (2020).

29 Blender Python API. *Blender Documentation – blender.org*. URL: https://docs.blender.org/api/current/ (дата звернення: 21.03.2024).

30 Python, Why. "Python." Python releases for windows 24 (2021).

31 Козуб, Д. П. "Використання скриптів в Blender для автоматизації процесу створення 3d-моделей та анімації." (2023).

32 GitHub – joshuaKnauber/layer_painter: Repository for the blender Layer Painter addon. *GitHub*. URL: https://github.com/joshuaKnauber/layer_painter (дата звернення: 21.03.2024).

33 Ravage – Layer Based Texturing. *Blender Market*. URL: https://blendermarket.com/products/ravage (дата звернення: 21.03.2024).

34 Thom Barron. "Blender 3D - Introduction to Geometry Nodes." (2022).

35 Brecht. Layered Textures Design – Developer Blog. *Developer Blog*. URL: https://code.blender.org/2022/02/layered-textures-design/ (дата звернення: 21.03.2024).

36 Shader Nodes - Blender 4.1 Manual. *Blender Documentation - blender.org*. URL: https://docs.blender.org/manual/en/latest/render/shader_nodes/index.html (дата звернення: 21.03.2024).

37 BlenderKit | Download 24,647 FREE 3D models, textures and other Blender assets. *BlenderKit | Download 24,648 FREE 3D models, textures and other Blender assets*. URL: https://www.blenderkit.com/ (дата звернення: 21.03.2024).

38 Python версія 3.12.3 Documentation. *3.12.3 Documentation*. URL: https://docs.python.org/3/ (дата звернення: 21.03.2024).

39 Microsoft. Visual Studio Code – Code Editing. Redefined. *Visual Studio Code – Code Editing. Redefined*. URL: https://code.visualstudio.com/ (дата звернення: 21.03.2024).

40 Hollister, Brad E., and Brad E. Hollister. "Blender's Embedded Python." Core Blender Development: Understanding the Essential Source Code (2021): 111-136.

41 Acampora, Paolo. Python Scripting in Blender: Extend the power of Blender using Python to create objects, animations, and effective add-ons. Packt Publishing, 2023.

42 GitHub – nutti/fake-bpy-module: Fake Blender Python API module collection for the code completion. *GitHub*. URL: https://github.com/nutti/fake-bpy-module (дата звернення: 25.03.2024).

43 Vyacheslav Nykytyuk, Vasil Dozorskyi, Oksana Dozorska, Andrii Karnaukhov and Liubomyr Matiichuk. The Method of User Identification by Speech Signal. The 2nd International Workshop on Information Technologies: Theoretical and Applied Problems (ITTAP-2022) Ternopil, Ukraine, November 22-24, 2022. Vol-3309 urn:nbn:de:0074-3309-1. P.225-232. ISSN 1613-0073 DOI: 10.1425/jsdtl. (Scopus)

44 Kryazhych O., Itskovych V., Iushchenko K., Hrytsyshyna V., Bruvier D., Nykytyuk V., Bodnarchuk I. (2023) The use of abstract moore automaton to control

the sensors of a service-oriented alarm and emergency notification network. Scientific Journal of TNTU (Tern.), vol 109, no 1, pp. 111–120. (Фахова). ISSN 2522-4433

45 Dozorskyi, V., Dediv, I., Sverstiuk, S., Nykytyuk, V., Karnaukhov, A. The Method of Commands Identification to Voice Control of the Electric Wheelchair. The Workshop is organized by the Faculty of Applied Information Technologies and Electrical Engineering of Ternopil Ivan Puluj National Technical University. The 1st International Workshop on “Computer information technologies in Industry 4.0” (CITI-2023) will be held in Ternopil, Ukraine, from June 14 to 16, 2023. The Workshop is organized by the Faculty of Applied Information Technologies and Electrical Engineering of Ternopil Ivan Puluj National Technical University. 2023, 3468, pp. 233–240. Vol-3468 urn:nbn:de:0074-3468-8, ISSN 1613-0073 (Scopus)

46 Leal, José Eugenio. "AHP-express: A simplified version of the analytical hierarchy process method." MethodsX 7 (2020): 100748.

47 Охорона праці в галузі інформаційних технологій : навч. посіб. / В.І. Голінько, М.Ю. Іконніков, Я.Я. Лебедєв ; М-во освіти і науки України, Нац. гірн. ун-т. – Д. : НГУ, 2015. – 246 с.

48 Про затвердження Вимог щодо безпеки та захисту здоров'я працівників під час роботи з екранними пристроями : Наказ Міністерства соц. політики України від 14.02.2018 р. № 207. URL: https://zakon.rada.gov.ua/laws/show/z0508-18#Text (дата звернення: 09.04.2024).

49 Конспект лекцій з курсу «Охорона праці в галузі» / Укладачі: Яскілка В.Я., Олійник М.З. – Тернопіль: Вид-во ТНТУ імені Івана Пулюя, 2016. – 56 с.

50 Охорона праці в галузі [текст] : навчальний посібник / П. С. Атаманчук, В. В. Мендерецький, О. П. Панчук, Р. М. Білий - К. : «Центр учбової літератури», 2017. - 322 с.

ДОДАТКИ

Додаток А

Теза на конференцію “Філософські виміри техніки”

Міністерство освіти і науки України

ЗБІРНИК ТЕЗ

II Міжнародної науково-теоретичної конференції

молодих учених та студентів

Abstracts collection

of the II International scientific-theoretical conference of young scientists and students

ФІЛОСОФСЬКІ ВИМІРИ ТЕХНІКИ

PHILOSOPHY TECHNOLOGY ASPECTS

4-5 грудня, 2019 р.

Тернопіль, Україна

Тернопільський національний технічний університет імені Івана Пулюя (Україна)

Тернопільський національний педагогічний університет імені Володимира Гнатюка (Україна)

Національний університет водного господарства та природокористування (м. Рівне, Україна)

Державний університет «Люблінська Політехніка» (м. Люблін, Польща)

Державна Вища Технічно-економічна школа ім. кс. Броніслава Маркевича (м. Ярослав, Польща)

УДК 004.05
В. С. Озіранець
Тернопільський національний технічний університет ім. І. Пулюя (Україна)

ТЕХНОЛОГІЧНІ ФОРМИ ВІРТУАЛЬНОЇ РЕАЛЬНОСТІ

V. C. Oziranets

TECHNOLOGICAL FORMS OF VIRTUAL REALITY

“Наука не стоїть на місці” - ця фраза добре описує всі сучасні тенденції розвитку світу та кожної людини в цілому. З кожним днем технології все більше впливають на наше життя. І теперішнє існування не можливо уявити без комп’ютерних та інших девайсів. Однією з новітніх віток розвитку науки є 3Д-дизайн та подальші його форми, такі, як доповнена та віртуальна реальності.

Розглянемо вище перелічені форми. Augmented reality (з англ. – доповнена реальність) - це доповнення фізичного світу за допомогою цифрових даних, яке забезпечується комп'ютерними пристроями (смартфонами, планшетами та окулярами AR) в режимі реального часу. Якщо добре оцінити ситуацію, то ця технологія тільки приносить користь. Наприклад, коли людина вирішила зробити ремонт і не знає, чи поміститься їй той чи інший об’єкт у кімнаті. За допомогою доповненої реальності вона зможе просто вибрати його 3Д модель і розмістити у тому місці, де потрібно. Навіть зараз багато компаній, які продають меблі, створюють програмні застосунки, щоб кожен зміг оцінити габарити об’єкта і визначитися, чи він йому підходить, що економить час. Проблема лиш у тому, що доповнена реальність не дає змогу спробувати відчути себе у певних умовах. Тож, як додаток до попередньої виникає віртуальна реальність – одна з неоднозначних технологій сьогодення.

Virtual reality (з англ. – віртуальна реальність) - ілюзія дійсності, створювана за допомогою комп'ютерних систем, які забезпечують зорові, звукові та інші відчуття. З одного боку, вона дозволить покращити життя людей, спростити навчання та багато інших функцій життєдіяльності. Та з іншої сторони, кожен з нас може перестати відчувати різницю між реальним та віртуальним світами, а то й гірше – стати залежними від другого. Крім того, через неї людина стане необачною, що різко може збільшити смертність серед населення. Хоча, якщо перебувати у віртуальній реальності в міру, то, можливо, це не буде шкодити і навпаки – допоможе покращити багато сфер життєдіяльності людини, таких як медицина (навчання лікарів-хірургів не на живих істотах, а у віртуальному світі).

Отже, 3Д-технології разом із комп’ютерними, загалом покращують життя людини. Головне, як і з будь-якою технікою дотримуватися правила: “Використовувати в міру”.

Додаток Б

Теза на конференцію “Природничі та гуманітарні науки. Актуальні питання”

Міністерство освіти і науки України,
Тернопільський національний технічний університет
імені Івана Пулюя
Маріборський університет (Словенія)
Технічний університет в Кошице (Словаччина)
Каунаський технологічний університет (Литва)
Львівський національний університет
імені Івана Франка,
Гірничо-металургійна академія ім. Станіслава Сташиця (Польща)
Луцький національний технічний університет,
Чернівецький національний університет
імені Юрія Федьковича,
Вроцлавський економічний університет (Польща)
Університет технологій та економіки
імені Хелени Ходковської (Польща)
Донбаська державна машинобудівна академія

Студентське наукове товариство

VI МІЖНАРОДНА

студентська науково - технічна конференція

"ПРИРОДНИЧІ ТА ГУМАНІТАРНІ НАУКИ.

АКТУАЛЬНІ ПИТАННЯ"

27-28 квітня 2023 р.

(збірник тез конференції)

Тернопіль 2023

УДК 004.928

Озіранець В. С. – ст. гр. СНнм-51
Тернопільський національний технічний університет імені Івана Пулюя

АНАЛІЗ МЕТОДІВ МОДЕЛЮВАННЯ В BLENDER

Науковий керівник: старший викладач Шимчук Г.

Oziranets V. S.
Ternopil Ivan Puluj National Technical University

MODELING METHODS ANALYSIS IN BLENDER

Supervisor: Senior Lecturer Shymchuk G.

Ключові слова: 3D, blender, моделювання, скульптинг, метабол, kit bash.
Key words: 3D, blender, modeling, sculpting, metaball, kit bash.

У 3D графіці для створення сцен з анімаціями чи просто для подальшої візуалізації, використовується програмне забезпечення та алгоритм дій, необхідний для реалізації об'єктів. Для створення об'єктів розроблено декілька методів моделювання, які розглянуто на основі програмного забезпечення Blender. Для початку, поняття моделювання – це процес створення об'єкту в тривимірному просторі та надання йому певної форми. Прикладом моделювання можна назвати як створення куба, так і скульптинг персонажа, чи редагування цього куба, перетворюючи його в паралелепіпед тощо [1].

Розглянемо види моделювання, реалізовані в Blender версії 3.5. До методів 3D моделювання відносяться: моделювання по сітці мешів або ж полігональне, моделювання з використанням кривих, метабол моделювання, kit bash моделювання, скульптинг, geometry nodes та моделювання через модифікатори.

Моделювання по сітці меша вимагає зміщення кожної вершини та додавання нових вершин для надання форми об'єкта, а також використання примітивів з подальшим їх редагуванням. Часто поділяється на два підвиди, а саме полігональне моделювання та box моделювання, різниця між якими швидше у підході розподілення об'єкта. Наприклад, годинник можна зробити як єдиний меш або як сукупність мешів. Цей вид моделювання є класичним і основним, застосовується в комбінації з іншими для ефективнішого за часом створення моделі [1].

Моделювання з використанням кривих корисне для кабелів, волосся, шлангів та інших витих об'єктів, з можливістю конвертації в класичний меш. Даний метод оснований на кривих Без'є, Нюрб кривих тощо, якими можна задати певну форму, використовуючи точки і дуги, які ці точки утворюють [2].

Тип hard-surface моделювання – це моделювання, сфокусоване на створенні механізмів та пов'язаних з ними об'єктів. Даний тип застосовує метод через сітку та через модифікатори за основу, так як вимагає створення гладких нерухомих поверхонь, та методи kit bash та metaball для додавання деталей або створення концепту, на основі якого будуватиметься остаточний меш. Він не відноситься до методів моделювання, але задає набір інструкцій для створення меша і знадобиться для розуміння методів kit bash та metaball [1].

Моделювання через метабол корисне як для рідких об'єктів, так і навпаки для hard-surface моделювання. Даний вид складається з об'єктів званих metaball, суть яких полягає в можливості взаємовідносин один між одним у вигляді можливості об'єднуватися в один, або розтягувати точки контакту за необхідності. Наприклад, коли

дві метабол сфери наближаються один до одного, то вони починають поступово з'єднуватися, утворюючи спочатку міст, а після того утворюючи одну єдину сферу. Також конвертуються в звичайний меш, як і метод через криві.

Метод kit bash моделювання, пов'язаний з hard-surface моделюванням, полягає в створенні кількох колекцій мешів та основного об'єкту, які потім формують остаточний об'єкт. Для початку створюється основний об'єкт [1], суть якого вказувати на базову форму і розміщення менших деталей. У свою чергу, ці деталі формуються в колекції, які потім "кидаються (bash)" в основну модель, формуючи складний механізм або концепт. Для накладання застосовується як звичайне розташування об'єктів у просторі, так і система часточок або geometry nodes [3], про які мова піде далі.

Спосіб створення комплексних об'єктів з використанням geometry nodes [3] з'явився з версією Blender 2.92 і встиг змінитися в подальших оновленнях. Метод полягає у використанні деревоподібної системи, яка складається з листків та зв'язків між ними, кожен з яких впливає на остаточний об'єкт. Спочатку даний метод базувався на системі часточок, які би взаємодіяли через умовні листки, але розробники розширили функціонал, і тепер це є повноцінний алгоритм, розписаний у листках. Дана конструкція частково замінила систему часточок в Blender в плані розробки масивних об'єктів, хоч і не змогла повністю витіснити у фізичній 3D симуляції.

Ще один класичний метод моделювання – скульптинг, суть якого полягає в використанні умовного "пластиліну" та кистей, які змінюють форму цього "пластиліну" [2]. Даний метод вимагає пост обробки, так як скульпт, що є результатом ліпки, є не оптимізованим мешем. З іншого боку, скульптинг дозволяє робити дуже деталізовані об'єкти і потім перевести деталі з нього на оптимізований меш у вигляді карти нормалей – карти текстур, яка складається з трьох кольорів, кожен з яких задає розташування "уявної" точки просторі, створюючи ілюзію форми.

Останнім, та не менш важливим є метод моделювання через модифікатори, який можна також розділити на два типи – булеве моделювання та параметричне моделювання. Параметричне моделювання розуміє під собою повний контроль над формою об'єкта використовуючи параметри модифікатора. Наприклад, модифікатор subdivision surface дає більше полігонів за рахунок поділу вже існуючих у n-разів. До нього також можна віднести використання панелі властивостей об'єкту, яка з'являється при створенні базового об'єкту, такого як куб чи сфера. У свою чергу, булеве моделювання назване в честь модифікатора Boolean, який дозволяє вирізати/об'єднувати/обрізати спільну форму між двома об'єктами.

Загалом, кожен з методів застосовується в комбінації як мінімум з одним іншим для реалізації задуму 3D художника або для виконання замовлення клієнта тощо. В остаточному вигляді все одно буду меш, створений для виконання ролі у майбутньому проекті, будь-то об'єкт на фоні фільму, головний персонаж у грі чи зображення на робочому столі.

Література:

1. J.M. Blain. The Complete Guide to Blender Graphics: Computer Modeling & Animation. / Blain J.M. – Taylor & Francis Group, LLC (4th edition), 2018. – 697 p.
2. J. Chronister. Blender Basics: A Classroom Tutorial Book / Chronister J. – cdschools.org (5th Edtion, 2017; 4th Edition, 2011); eBook (Creative Commons Licensed), 2017. – 266 p.
3. J.M. Blain. The Complete Guide to Blender Graphics: Computer Modeling & Animation. / Blain J.M. – CRC Press (7th edition), 2022. – 664 p. ISBN 9781003226420

Додаток В

Тези на сьому міжнародну конференцію “Природничі та гуманітарні науки. Актуальні питання ”

Міністерство освіти і науки України
Тернопільський національний технічний університет
імені Івана Пулюя
Маріборський університет (Словенія)
Технічний університет в Кошице (Словаччина)
Каунаський технологічний університет (Литва)
Львівський національний університет
імені Івана Франка,
Гірничо-металургійна академія ім. Станіслава Сташиця (Польща)
Луцький національний технічний університет,
Чернівецький національний університет
імені Юрія Федьковича,
Вроцлавський економічний університет (Польща)
Університет технологій та економіки
імені Хелени Ходковської (Польща)
Донбаська державна машинобудівна академія

Студентське наукове товариство

VII МІЖНАРОДНА

студентська науково - технічна конференція

"ПРИРОДНИЧІ ТА ГУМАНІТАРНІ НАУКИ.

АКТУАЛЬНІ ПИТАННЯ"

25-26 квітня 2024 р.

(збірник тез конференції)

Тернопіль 2024

УДК 004.928
Озіранець В.С.В. – ст. гр. СНнм-61, Карнаухов А.К. – асист. каф КН,
Орловська А.В. – ст. гр. СТ-31.
Тернопільський національний технічний університет імені Івана Пулюя

АНАЛІЗ ЕФЕКТИВНОСТІ СИСТЕМ НАКЛАДАННЯ МАТЕРІАЛІВ З ВИКОРИСТАННЯМ МЕТОДУ САТІ

Науковий керівник – канд. тех. наук, доц. Никитюк В. В.

Oziranets V.S.V., Karnaukhov A., Orlovska A.
Ternopil Ivan Puluj National Technical University

MATERIAL APPLIANCE SYSTEM EFFICIENCY ANALYSIS USING SAATI METHODOLOGY

Supervisor: Ph.D., Assoc. Prof. Nykytyuk V.V.

Ключові слова: Blender, тривимірне текстурування, система нодів, система рівнів.
Key words: Blender, 3d texturing, node system, layer system.

У наш час одними з найпотужніших інструментів для створення та редагування матеріалів є графічні редактори з використанням саме систем рівнів та нодів. Системи нодів складаються з вузлів, кожен з яких змінює певні атрибути об'єкта, дозволяючи математично та фізично прорахувати матеріал [1]. Прикладами таких систем є Shader Editor у Blender [1]. З іншого боку, система рівнів, що використовується для створення матеріалів у графічних програмах, включає в себе концепцію шарів, які представляють собою окремі структури, які можна налаштовувати та комбінувати для створення складних матеріалів, як наприклад у Substance Painter чи Photoshop [2].

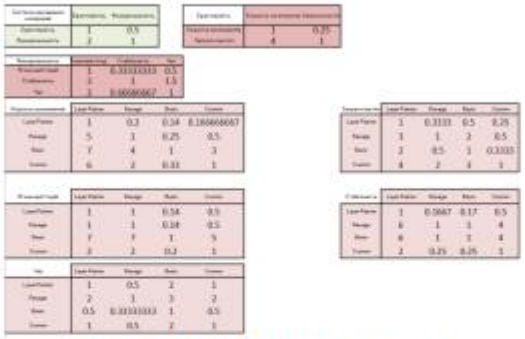

Рис. 1 – Матриці попарних порівнянь

Так чому ж не спробувати об'єднати дані системи і перевірити її ефективність? Для цього необхідною і достатньою умовою буде застосувати метод Сааті (див. рисунок 1) на основі п'яти основних критеріїв на прикладі різних систем у програмі Blender – кількість компонентів, затрати пам'яті, різноманіття, стабільність та час. Різноманіття відповідає за кількість можливих дій і представляє собою шкалу 0, 0.4,

0.7, 1 від гіршого до кращого, де 0 – неможливо нічого змінювати, 1 – можна робити будь-що. Аналогічні шкали і для стабільності, тобто перевірки на оптимальність роботи системи, і для часу, який необхідний для створення такого матеріалу.

Для аналізу ефективності систем візьмемо наступні рішення – Ravage 2 Lite(Layer), Layer Painter 2 (Layer), Blender Shader Editor (Node), Custom (Node-Layer). При проведенні тестування кожної з них за вищевказаними п'ятьма критеріями отримано наступні значення параметрів (к. – кількість компонентів, з. – затрати пам'яті, р. – різноманіття дій, с. – стабільність, ч. – час):

1. LayerPainter к. 104, з. 218.7, р. 0.4, с. 0, ч. 0.7.
2. Ravage к. 27, з. 153.0, р. 0.4, с. 1, ч. 1.
3. Basic к. 16, з. 165.7, р. 1, с. 1, ч. 0.4.
4. Custom к. 23, з. 141.3, р. 0.7, с. 0.4, ч. 0.7.

На основі цих даних можна розглянути яка ж з альтернатив найкраща саме для користувача і пояснити чому, для чого застосовуємо метод Сааті і знайшовши власні вектори матриць можна синтезуємо локальні пріоритети, щоб отримати набір загальних пріоритетів для ієрархії, або ж простими словами можна визначити ієрархію серед чотирьох альтернатив (див. рисунок 2).

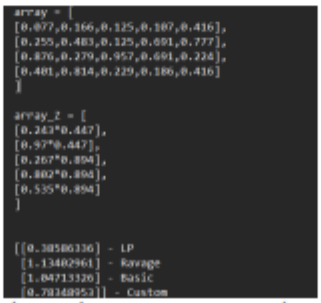

Рис. 2 – Розраховані власні вектора та отримані глобальні пріоритети

Підсумовуючи система рівнів Ravage є найкращою за вищевказаними перед розрахунками критеріями, в основному через стабільність роботи та ефективність. На другому місці іде базова версія програми Blender. На передостанньому стоїть власне розроблене рішення, яке змогло набагато швидше виконати задачу через накладання двох матеріалів у вигляді рівнів з можливістю комбінування з нодами, але проблеми зі стабільністю опустили її нижче, що повідомляє про необхідність доопрацювання системи. На останньому місці LayerPainter із його системою рівнів, яка показала себе не з найкращого боку і дуже часто видавала помилки. Відповідно робимо висновок, що ідея комбінованої системи має місце з точки зору затрат часу та комфорту праці та дозволяє автоматизувати процеси накладання кількох матеріалів.

Література

1. Озіранець В. С. В. Розробка дизайну та реалізація 3D моделей для трейлеру комп'ютерної гри "Echo of Sunset" засобами Blender: кваліфікаційна робота освітнього рівня „Бакалавр" – Тернопіль : ТНТУ, 2022. – 50 с.

2. Leiro, L. Suaya, and Marc Garrigó. Development of a Node-Based Material Editor. Centre de la Imatge i la Tecnologia Multimèdia - Universitat Politècnica de Catalunya. – Eurographics Proceedings 2022 The Eurographics Association., p. 59-63.

УДК 004.928
Озіранець В.С.В. – ст. гр. СНнм-61, Орловська А.В. – ст. гр. СТ-31.
Тернопільський національний технічний університет імені Івана Пулюя

ГІБРИДНИЙ ПІДХІД ДО СТВОРЕННЯ МАТЕРІАЛІВ У ТРИВИМІРНІЙ ГРАФІЦІ

Науковий керівник – канд. тех. наук, доц. Никитюк В. В.

Oziranets V.S.V., Orlovska A.
Ternopil Ivan Puluj National Technical University

THE HYBRID APPROACH FOR THE CREATION OF MATERIALS IN THREE-DIMENSIONAL GRAPHICS

Supervisor: Ph.D., Assoc. Prof. Nykytyuk V.V.

Ключові слова: Blender, тривимірне текстурування, система нодів, система рівнів.
Key words: Blender, 3d texturing, node system, layer system.

З активним розвитком галузі комп'ютерної графіки з'являються нові методи та засоби спрощення та автоматизації процесів. Так, все більше та більше різноманітного програмного забезпечення для комп'ютерної графіки почало перехід від рівневої системи, яка була простою для розуміння художникам, які працювало у двовимірній графіці, але менш функціональною та гнучкою, що змушувало вдаватися до розробки власних доповнень та спроб обійти обмеження систем, до системи вузлів [1], яка була трохи важчою для освоєння, зате компенсувала це можливістю реалізувати будь-який віртуальний матеріал, який міг не уступати реальному. Так у Cinema 4D з 2018 [1] року в оновленні версії R20 з'являється система вузлів, яка одразу ж привернула увагу користувачів і проявила себе ефективною для роботи з матеріалами.

Питання, яке виникає при розробці нової системи – яка її основна функція, компоненти, зв'язки та чи взагалі вона є актуальною та корисною? Тут необхідно покроково все розписати і проаналізувати. Перш за все, розбити гібридну систему створення матеріалів на рівні декомпозиції для розуміння роботи кожного з них.

Перший рівень – система вузлів та система рівнів. Тут важливо зрозуміти, що ці дві компоненти є взаємопов'язаними, так як виконують дію над одним і тим же об'єктом – матеріалом. Систем рівнів представляє собою набір послідовно накладених один на одного структур, кожна з яких додає властивість жорсткості, кольору, прозорості або іншої фізичної властивості поверх попередньої, формуючи таким чином комплексний матеріал. Дана система є проста для засвоєння та є дуже ефективною для матеріалів, що містять у собі декілька реальних фізичних матеріалів. Прикладом застосування можна виділити матеріал металу з іржею. Система вузлів у свою чергу представляє собою деревоподібну структуру, кожен лист в якій відповідає за певну фізичну характеристику. Різниця з системою рівнів полягає у додатковому маніпулюванні кожною фізичною властивістю, тобто коли як в системі рівнів робота проводиться зі стеком рівнів (переміщення верх, вниз, накладання у вигляді маски тощо), то в другій у нас є основний вузол, в який вже входять усі інші, формуючи наприклад з білого шуму карту нормалей [2] або використовуючи його як масу безпосередньо, не створюючи додаткових структур/рівнів. Важливо зазначити, що як система рівнів, так і система вузлів виконають завдання і створять необхідний матеріал,

різниця лише в гнучкості самого новоутвореного матеріалу, що відноситься до головної функції системи.

Другий рівень – шари або ж рівні для однієї системи, та вузли для другої. Шар в системі рівнів для накладання матеріалів – це один із елементів в структурі, який може містити різні текстури або параметри матеріалу, такі як кольори, блиск, прозорість та інші властивості. У свою чергу вузол – це базовий елемент графічного інтерфейсу, який використовується для створення та налаштування візуальних ефектів, матеріалів, текстур, освітлення та багато іншого. Кожен вузол представляє собою певну функцію або операцію, яка може бути об'єднана з іншими вузлами для створення складних графічних ефектів або матеріалів. Вузли зазвичай використовуються в комбінації зі змішувачами, конвертерами, генераторами та іншими інструментами для створення складних графічних сцен. Простими словами для переведення або об'єднання система достатньо кілька вузлів прирівнювати до одного рівня, або ж розбивати один рівень за властивостями, кожна з яких це вузол. Третій рівень – властивості шару та вузла відповідно, описані раніше.

Розглянемо переваги та недоліки гібридної системи на основі вище описаної структури, зв'язків та основної функції. До переваг одразу ж можна віднести все те, що належить до переваг системи на основі вузлів та декілька від системи на основі рівнів, відповідно інтерфейс, гнучкість, універсальність та організація.

Інтуїтивно зрозумілий інтерфейс, який поєднує у собі простоту системи рівнів для базового і простого налаштування матеріалу та комплексні дерева вузлів для більш досвідчених художників.

Гнучкість створення матеріалу – полягає в можливості маніпулювати майже будь-якими якщо не усіма властивостями матеріалу навіть на найпростішому рівні, включаючи роботу над процедурними матеріалами, з якими система рівнів мала складності.

Універсальність створення матеріалу, що відображає можливість використанням будь-який з традиційних методів розробки матеріалу, будь-то вузли чи рівнів, що дозволяє мати як мінімум часткову сумісність із старими вже розробленими матеріалами.

З іншого боку, не обійшлося і без мінусів, а конкретно зростання складності навчання та роботи з матеріалом з точки зору вивчення нового процесу, так як тепер треба вивчити обидві системи для більш ефективної роботи над тривимірними проєктами. Гібридний підхід також може призвести до збільшених витрат на продуктивність, особливо під час роботи зі складними мережами вузлів. Щодо сумісності, важливо враховувати, що не всі програми для тривимірної графіки підтримують однакові функції та можуть виникнути проблеми при обміні матеріалами.

Підсумовуючи, новоутворена система має місце при реалізації її як доповнення вже до однієї існуючої, так як це дозволить простіше інтегрувати її, враховуючи недоліки цього підходу.

Література

1. Node-Based Materials [Електронний ресурс] // Maxon – Режим доступу до ресурсу: https://www.maxon.net/en/cinema-4d/features/node-based-materials.

2. Озіранець В. С. В. Розробка дизайну та реалізація 3D моделей для трейлеру комп'ютерної гри "Echo of Sunset" засобами Blender: кваліфікаційна робота освітнього рівня „Бакалавр" – Тернопіль : ТНТУ, 2022. – 50 с.

Додаток Д

Код доповнення для Blender версії 3.6

Основний файл ініціалізації доповнення init.py

```
# import modules
import bpy
from . import ui, properties, operations

# addon info
bl_info = {
    "name": "WiseShader",
    "author": "Vitalii Stepan WiseDeka Oziranets, Ivan Osiichuk",
    "description": "Layer based material system",
    "blender": (4, 0, 0),
    "version": (0, 2, 0),
    "location": "View3D > N-Panel > WiseShader",
    "category": "Material"
}
def register():
    """Registers classes, order is important."""
    properties.register()
    ui.register()
    operations.register()

def unregister():
    """Unregisters classes, order is important."""
    properties.unregister()
    operations.unregister()
    ui.unregister()
```

Панель в інтерфейсі draw_layer.py

```
     # import modules
import bpy

from ..tools import utils

# global variables
w_utils = utils.WiseUtils  # for main utils

class Layer_PT_Panel(bpy.types.Panel):
    """Panel for the layer system in the N-panel"""

    bl_label = "Layer System"  # label displayed
    bl_idname = "WISEUI_PT_layer_panel"  # panel id
    bl_space_type = 'VIEW_3D'  # where to show panel
    bl_region_type = 'UI'  # what type is it
    bl_category = "WiseShader"  # how category named

    @classmethod
    def poll(cls, context):
```

```
        # global class object
        global w_utils

        # check for current object and material
        return w_utils.base_poll(context)

    def draw_header(self, context):

        # draw layers header
        self.layout.label(text="Layers")

    def draw_layers(self, active_material, layout):
        """Draw all layers in the panel"""

        # List of layers for active material
                                for       i,       layer       in
enumerate(active_material.material_layers):

            # a basic container
            box = layout.box()

            # add row in the container
            row = box.row(align=True)

            # # expand layer
            row.operator("material.expand_layer", text=layer.name,
                         icon="TRIA_DOWN" if layer.expand else
"TRIA_RIGHT", emboss=False).index = i

            # visibility toggle for layer
            row.operator("material.make_visible_layer", text="",
                          icon="HIDE_OFF" if layer.visibility else
"HIDE_ON", emboss=False).index = i

            # remove layer
                     row.operator("material.remove_material_layer",
text="",
                         icon="X", emboss=False).index = i

            if layer.expand:
                # if layer is not first, it is a base layer
                if i != 0:
                    # add additional row for masking options
                    sub_row = box.row(align=True)

                sub_row = box.row(align=True)
                sub_row.label(text="Channels:")

                # Channel options
                col = box.column_flow(columns=3, align=True)

                # get each channel as a checkbox
                for j, channel in enumerate(layer.channels):

                    # call active channel operator
```

```
                                                    new_channel    =
col.operator("layer.active_channel",
                                                text=channel.name,
                                                 icon="CHECKBOX_HLT"
if channel.status else "CHECKBOX_DEHLT",
                                                    emboss=False)  #
make button look like a checkbox

                    # get active layer
                    new_channel.layer_index = i

                    # get active channel
                    new_channel.channel_index = j

                    if channel.status:
                        # draw channel params
                        self.draw_channel_params(
                                  active_material, channel, i, j,
layout)

       def  draw_channel_params(self,  active_material,  channel,
l_index, c_index, layout: bpy.types.UILayout):
        """Visualize channel parameters to twik for user"""

        # draw a row with twiks
        box = layout.box()
        row = box.row(align=False)
        row.label(text=channel.name)
        row = box.row(align=False)

        properties = None
        properties = channel.get_mix_option()
        if properties is not None:
            for property in properties:
                if property is not None:
                    row.prop(channel.nodes[0].node, property)

        properties = None
        properties = channel.show_properties()

        for i, node_properties in enumerate(properties):
            if node_properties is not None:
                for property in node_properties:
                    if property is not None:
                                  if  channel.nodes[i+1].name  !=
"ShaderNodeValToRGB":
                                row.prop(channel.nodes[i+1].node,
property)
                        else:
                            "TODO: for colorramp its complicated"
                             row.template_color_ramp(channel.nodes[
i+1].node, property, expand=True)

        out_properties = None
        out_properties = channel.show_output_data()
```

```
        for i, outputs in enumerate(out_properties):
            if outputs is not None:
                for output in outputs:
                    if output is not None:
                        row.prop(channel.nodes[i+1].node.outputs[o
utput], "default_value")

        in_properties = None
        in_properties = channel.show_input_data()

        for i, inputs in enumerate(in_properties):
            if inputs is not None:
                for input in inputs:
                    if input is not None:
                        row.prop(channel.nodes[i+1].node.inputs[in
put], "default_value")

        row = box.row(align=False)

        switch = None
             switch  =  row.operator("channel.change_rgb_to_image",
text="", icon="IMAGE")
        switch.layer_index = l_index
        switch.channel_index = c_index

    def draw(self, context):

        # global variable
        global w_utils

        # Panel's layout
        layout = self.layout

        # get active material and store it in a variable
        active_material = context.active_object.active_material

        # Add Layer button
          layout.operator("material.add_material_layer",  text="Add
Layer")

        # Draw Layers
        self.draw_layers(active_material, layout)
```

Механізм створення основних вузлів default_nodes.py

```
# import modules
import bpy

from         ..properties.nodes.node_material_output         import
Node_MaterialOutput
from         ..properties.nodes.node_principled_bsdf         import
Node_PrincipledBSDF
from ..properties.nodes.node_bump import Node_Bump
from ..properties.nodes.node_normalmap import Node_NormalMap
```

```
class DefaultNodes:
       """Check and add main PrincipledBSDF, MaterialOutput,
NormalMap, \n
    Bump nodes to start working with layers and channels."""

             def     add_missing_nodes(self,     active_material:
bpy.types.Material):
         """Add missing PrincipledBSDF, MaterialOutput, NormalGLMap
and Bump nodes."""

        defnode_objects = None

                                              defnode_objects         =
[Node_MaterialOutput(material=active_material),
                             Node_PrincipledBSDF(material=active_mat
erial),
                            Node_Bump(material=active_material),
                             Node_NormalMap(material=active_material
)]

        node_collection = []

        for obj in defnode_objects:
            obj.add_in_Blender()

        for obj in defnode_objects:
            node_collection.append(obj.node)

        # get nodes from collection
        out = node_collection[0]
        out.is_active_output = True

        princ = node_collection[1]
        princ.location = (out.location[0]-300, out.location[1])

        bump = node_collection[2]
           bump.location  =  (princ.location[0],  princ.location[1]-
35*19)

        normal = node_collection[3]
          normal.location  =  (princ.location[0],  princ.location[1]-
35*23)

        # procced to link those nodes
          return  self.add_nodes_links(active_material,  out,  princ,
bump, normal)

    def add_nodes_links(self, active_material: bpy.types.Material,
                        out: bpy.types.Node,
                        princ: bpy.types.Node,
                        bump: bpy.types.Node,
                        normal: bpy.types.Node):
        """Add missing links for nodes."""
```

```
        # principled to output link
        if not princ.outputs[0].is_linked:
            active_material.node_tree.links.new(
                princ.outputs[0], out.inputs[0])

        # bump to principled link
        if not bump.outputs[0].is_linked:
            active_material.node_tree.links.new(
                bump.outputs[0], princ.inputs[22])

        # normal to bump link
        if not normal.outputs[0].is_linked:
            active_material.node_tree.links.new(
                normal.outputs[0], bump.inputs[3])

        # return all nodes
        return out, princ, bump, normal
```

Утиліти utils.py

```
# import modules
import bpy

class WiseUtils:
    """Most used utils in different operators and panels."""

    @staticmethod
    def base_start_poll(context: bpy.types.Context):
        """Basic poll function"""

        # check for current object
                            return     context.active_object     and
context.active_object.type == "MESH"

    @staticmethod
    def base_poll(context: bpy.types.Context):
        """Start panel poll function"""

        # check for current object and material
                            return     context.active_object     and
context.active_object.type          ==          "MESH"          and
context.active_object.active_material
```

Ініціалізація вузлів nodes init.py

```
# import modules
import bpy

from .layers import Layer
from typing import List

def register():
      """Registers  classes  and  additional  properties,  order  is
important."""
    bpy.types.Material.material_layers: List[Layer] = []
```

```
def unregister():
     """Unregisters classes and additional properties, order is
important."""
    del bpy.types.Material.material_layers
```

Структура каналу channel_base.py

```
import bpy

from .channel_color import Channel_Color

from ..nodes.node_mix_color import Node_Mix_Color
from ..nodes.node_rgb import Node_Rgb
from ..nodes.node_colorramp import Node_ColorRamp

class Channel_Base(Channel_Color):
    """Define a basic channel as a class in python. \n
    Example: Roughness, Metallic"""

    def add(self):
        """Adds nodes MixColor, ColorRamp, RGB/ImageTexture to the
channel."""

        self.nodes = None

                                                self.nodes          =
[Node_Mix_Color(material=self.active_material,
                                      layer=self.layer,
                                      channel=self.name),
                       Node_ColorRamp(material=self.active_material
,
                                      layer=self.layer,
                                      channel=self.name),
                      Node_Rgb(material=self.active_material,
                               layer=self.layer,
                               channel=self.name)
                      ]

        for node in self.nodes:
            node.add_in_Blender()

        self.check_inside_links()
        self.align_nodes()

    def check_inside_links(self):

        if self.nodes is not None:
            if not self.nodes[1].node.outputs[0].is_linked:
                self.active_material.node_tree.links.new(
                                       self.nodes[0].node.inputs[6],
self.nodes[1].node.outputs[0])
            if not self.nodes[2].node.outputs[0].is_linked:
                self.active_material.node_tree.links.new(
```

```
                                        self.nodes[1].node.inputs[0],
self.nodes[2].node.outputs[0])

    def __init__(self, name: str,
                 active_material: bpy.types.Material,
                 layer,
                 nodes: list = None,
                 status: bool = False,
                 expand: bool = True):
        """Values: \n
        - channel's name
        - channel's material \n
        - channel's layer \n
        - channel's node objects \n
        - channel's status \n
        - channel's expand value
        """
        super().__init__(name=name,
                         active_material=active_material,
                         layer=layer,
                         nodes=nodes,
                         status=status,
                         expand=expand)
```

Оператор перевірки активного каналу active_channel.py

```
# import modules
import bpy

from ..properties.nodes.node import *

from ..tools import default_nodes, utils

# global variables
w_utils = utils.WiseUtils
w_main_nodes = default_nodes.DefaultNodes()

class ActiveChannel_OT_Operator(bpy.types.Operator):
    """Operator to activate/deactivate channel"""

     bl_idname = "layer.active_channel"  # unique identifier for
operator
    bl_label = "Active Layer Channel"  # operator label
    bl_description = "make channel active"  # operator description

    @classmethod
    def poll(cls, context):

        # global class object
        global w_utils

        # check for current object and material
        return w_utils.base_poll(context)
```

```
    layer_index: bpy.props.IntProperty()  # get active layer index
     channel_index: bpy.props.IntProperty()  # get active channel
index

    def change_status_channel(self, active_material):
        """Change channels status and add or remove nodes"""

        # global class objects
        global w_main_nodes

        # get active layer
                                              active_layer          =
active_material.material_layers[self.layer_index]

        # get active channel
        active_channel = active_layer.channels[self.channel_index]

         # setting channel and getting 4 main nodes for the first
layer
        out, princ, bump, normal = w_main_nodes.add_missing_nodes(
            active_material)  # getting 4 main nodes

        # change status
        active_channel.change_status()
         active_layer.connect_to_bsdf(active_material, princ, bump,
normal)

    def execute(self, context: bpy.types.Context):

        # store active material
        active_material = context.active_object.active_material

        # activate or remove channel
        self.change_status_channel(active_material)

        return {'FINISHED'}
```