НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
імені ІГОРЯ СІКОРСЬКОГО»

Факультет електроніки
(повна назва інституту/факультету)

Кафедра акустичних та мультимедійних електронних систем
(повна назва кафедри)

«До захисту допущено»
Завідувач кафедри
С.А. Найда
(ціали, прізвище)

“14 ” січня 2024р.

Магістерська дисертація

зі спеціальності 171 Електроніка
(код і назва)

на тему: «3Д моделювання з використанням штучного інтелекту»

Виконав: студент II курсу, групи ДВ-21мп
(шифр групи)

Біденко Антон Юрійович
(прізвище, ім’я, по батькові) (підпис)

Керівник Професор., д.т.н., професор кафедри АМЕС
Власюк Г.Г..
(посада, науковий ступінь, вчене звання,прізвище та ініціали) (підпис)

Консультант
(назва розділу) (посада, вчене звання, науковий ступінь,прізвище, ініціали) (підпис)

Рецензент Професор кафедри електронних пристроїв та систем
Д.т.н., професор Мельник І.В.
(посада,науковий ступінь,вчене звання,науковий ступінь,прізвище,ініціали) (підпис)

Засвідчую, що у цій магістерській дисертації немає запозичень з праць інших авторів без відповідних посилань.

Студент /
(підпис)

Київ – 2024 року

Національний технічний університет України
«Київський політехнічний інститут
імені Ігоря Сікорського»

Інститут/факультет Факультет Електроніки
(повна назва)

Кафедра акустичних та мультимедійних електронних систем
(повна назва)

Рівень вищої освіти – другий (магістерський) за освітньо-професійною програмою
Спеціальність (спеціалізація) Електронні системи мультимедіа та засоби Інтернету речей

(код і назва)

ЗАТВЕРДЖУЮ
Завідувач кафедри
Сергій НАЙДА
(ініціали, прізвище)

«14» січня 2024 р.

ЗАВДАННЯ
на магістерську дисертацію студенту

Біденку Антону Юрійовичу
- (прізвище, ім'я, по батькові)

1. Тема дисертації 3Д моделювання з використанням штучного інтелекту
науковий керівник дисертації Власюк Ганна Григорівна, д.т.н., професор
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)

затверджені наказом щодо затвердження теми №5218-с від 9.11.2023р.

2. Строк подання студентом дисертації 07 січня 2024 р.

3. Об'єкт дослідження Методи створення 3D об'єкту з мінімумом втручання людини

4. Предмет дослідження Дослідження створення 3D графіки з використанням OpenAI’s Point-E. Архітектура моделювання об’єкту. Технології 3D графіки.

5. Перелік завдань, які потрібно розробити: Аналіз сучасних технологій для роботи з 3D графікою. Огляд рушіїв. Розглянути принцип роботи OpenAI’s Point-E. Дослідити на практиці OpenAI’s Point-E.

6. Перелік графічного (ілюстративного) матеріалу 15 слайдів в презентації,в основному наповнення складається з мети, проблематики, наявних технологіях,огляд бібліотек і їх переваг. _____

7. Орієнтовний перелік публікацій _____

_____

8. Консультанти розділів дисертації

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

9. Дата видачі завдання 01.09.2023 р.

Календарний план

№ з/п	Назва етапів виконання магістерської дисертації	Строк виконання етапів магістерської дисертації	Примітка
1	Аналіз технічного завдання. Вибір та обґрунтування технологій для розробки	07.09.2023 – 16.09.2023	Виконано
2	Вибір та обґрунтування технологій для розробки	16.09.2023 – 8.10.2023	Виконано
3	Створення проекту	8.10.2023 – 17.10.2023	Виконано
4	Підготовка матеріалів до друку та оформлення пояснювальної записки	17.10.2023 – 25.10.2023	Виконано

5	Підготовка та оформлення презентації для доповіді	06.12.2023 – 28.12.2023	Виконано

Студент _____ А. Ю.Біденко

(підпис) (ініціали, прізвище)

Науковий керівник дисертації _____ Ганна ВЛАСЮК

(підпис) (ініціали, прізв)

УДК 681.3.07

РЕФЕРАТ

Біденко А.Ю. Дослідження 3Д моделювання з використанням штучного інтелекту дис.: 171 Електроніка / Біденко Антон Юрійович. - Київ, 2023. - 74.

Магістерська дисертація: 74 с., 40 рис., 10 джерел.

Ключові слова: 3D графіка, OpenAI’s Point-E і Google Colab.

Актуальність дослідження. Сьогодні створення візуального контенту з використанням ШІ набуло дуже широкого поширення. Ми можемо часто зустріти відео, картинки, ігри, чи навіть метавсесвіти, які були цілком, чи частково створені з його використанням. Використання Шi дозволило людям без відповідних навичків створювати контент в кілька натискань і економити значну частку ресурсів і часу.

Мета дослідження: полягає в дослідженні роботи OpenAI’s Point-E і імпорті готового результату разом із запіченою текстурою в Unreal Engine.

Завдання для досягнення мети: дослідити процес створення 3D об’єкту з тексту та зображення, розглянути особливості використання штучного інтелекту. Дослідити OpenAI’s Point-E. На основі розглянутого матеріалу створити повноцінний проект з використанням вище зазначених технологій.

Об’єкт дослідження: програмне забезпечення – технологія OpenAI’s Point-E.

Предмет дослідження: компоненти, ноди, UV розгортка і текстуринг.

Методи дослідження: алгоритми та методи для опрацювання технології в середі розробки Google Colab, для створення нодової схеми на основі Python мови програмування.

Наукова новизна отриманих результатів: досліджено новітніх технологій та методи їх взаємодії, розроблено проект з використанням цих технологій, досліджено переваги використання Google Colab при 3д моделюванні.

Практичне значення одержаних результатів: результати виконаного проекту можуть бути використані при масовому створенні 3д текстур для заповнення пустих локацій відео, ігрового, чи мета простору.

ABSTRACT

Master's thesis: 74 p., 40 figures, 10 sources.

Keywords: 3D graphics, AI depth mask and visual scraping.

Relevance of the study. Today, the creation of visual content using AI has become very widespread. We can often find videos, pictures, games, or even metaverse that were created with its help in whole or in part. The use of AI has allowed people without relevant skills to create content in a few clicks and save a significant amount of resources and time.

The purpose of the study: to investigate the work of OpenAI's Point-E and import the finished result along with the baked texture into Unreal Engine.

Tasks to achieve the goal: to study the process of creating a 3D object from text and image, to consider the features of using artificial intelligence. To study OpenAI's Point-E. Based on the material reviewed, create a full-fledged project using the above technologies.

Object of study: software - OpenAI's Point-E technology.

Subject of research: components, nodes, UV scanning and texturing.

Research methods: algorithms and methods for processing the technology in the Google Colab development environment, for creating a node scheme based on the Python programming language.

Scientific novelty of the results: the latest technologies and methods of their interaction were studied, a project using these technologies was developed, and the advantages of using Google Colab in 3D modeling were investigated.

Practical significance of the results: the results of the project can be used in the mass creation of 3D textures to fill empty locations in video, gaming, or meta-space.

Зміст

ПЕРЕЛІК СКОРОЧЕНЬ І ТЕРМІНІВ

ШІ(AI) - штучний інтелект

UE - Рушій Unreal Engine

Auto-LOD - Automatic LOD generation

GAN - Generative adversarial network

API - Application Programming Interface

VFX - Visual Effects Artist

SDF - Signed Distance Function

PLY - Polygon File Format, або Stanford Triangle Format

FBX - Filmbox

ВСТУП

В наш час штучний інтелект розвивається семимильними кроками і активно впроваджується в повсякдення. Все зводиться до того, що використання штучного інтелекту настільки спрощує життя, що його починають застосовувати в усіх сферах діяльності.

Будь-яка професія потребує від людини певну кількість ланок дій, кожній з яких необхідний конкретний об’єм часу та витрачених зусиль. Штучний інтелект дозволяє зменшити цей об’єм настільки, що від людини буде необхідно лише перевірити кінцевий результат і внести дрібні виправлення.

З часом з’являються нові технології, кожна з яких несе в собі величезний об’єм інформаці. Для опрацьовування подібного потоку необхідний потужний 3D рушій, що може її обробити і використати за призначенням. В сьогоденні для 3D немає кращого варіанту за Unreal Engine для реалізації результату і Blender для текстурингу. Вони самі по собі являють не тільки набір інструментів, а й теку з великою кількістю бібліотек і потужних технологій типу Lumen, Houdini чи Nanite. Також дужи доброзичливі для новачків з нодовою схемою текстурингу і скріптингу. Для хмарної реалізації проектів одним з найкращих варіантів слугує Google Colab. Його підтримка Github і модульність значно спрощують використання бібліотек і орієнтацію при написані коду.

1 АНАЛІЗ СУЧАСНИХ ТЕХНОЛОГІЙ

1.1 Рушії та їх можливості

Загалом поняття рушій[1] - це програма або середовище, яке надає розробникам інструменти та API, необхідні для створення відеоігор, графіки та візуалізацій. Сюди входить все - від штучного інтелекту та анімації до фізичного моделювання та звуку. Ігрові рушії забезпечують основу для створення відеоігор, надаючи розробникам свободу зосередити свою увагу на ігровому контенті, а не на більш детальних технологічних деталях.

Ігрові рушії централізують багато ключових аспектів створення ігор, чи 3D о’бєктів під однією парасолькою. Вони скорочують процес створення нових ресурсів щоразу, коли ви починаєте новий проект.

Надаючи розробникам єдину екосистему для створення проектів, рушії забезпечують рівень узгодженості, сумісності та модульності, необхідний для створення унікального та оптимізованого проекту.

Середньостатистичний ігровий рушій повинен надавати вам можливості для спрощення таких критично важливих завдань, як:

Фізика - повинен бути ідеальний баланс між якістю симуляції та обмеженнями обчислювальних потужностей для кінцевого користувача.

Введення - це надзвичайно поширена проблема в крос-платформній розробці. Зазвичай надзвичайно важко оптимізувати проект під різні платформи.

Обробка візуальних ресурсів - освітлення, затінення, відображення текстур і глибина різкості вимагають менше зусиль для програмування при використанні ігрових рушіїв.

Простіше кажучи, обраний вами рушій повинен давати вам можливість виконувати вищезгадані завдання з меншими зусиллями при кодуванні. Це допомагає значно скоротити час розробки і дозволяє командам зосередитися на

створенні унікального та особливого користувацького досвіду.

1.2 Проблематика використання 3D рушіїв

Використання 3D-рушія[2] має як переваги, так і недоліки. З одного боку, ці рушії дозволяють швидше і простіше розробляти ігри порівняно з програмуванням з нуля. Вони часто постачаються з різноманітними інструментами для прискорення процесу створення гри, такими як потужний штучний інтелект, 3D-моделювання та інструменти анімації. Рушії 3D-ігор також надають широкий спектр можливостей для програмування, що дозволяє створювати більш креативні та візуально привабливі проекти. Крім того, 3D-рушії забезпечують ефективний код для фізики, освітлення, виявлення зіткнень та інших аспектів.

З іншого боку, використання 3D-рушія вимагає певного рівня знань та досвіду в розробці ігор. Оволодіння всіма необхідними навичками для успішної розробки 3D-рушія може бути складним і трудомістким завданням. Крім того, 3D-рушії можуть коштувати дорожче, ніж традиційні інструменти для розробки ігор. Крім того, код, згенерований ігровими рушіями, може бути складнішим у налагодженні та підтримці через підвищену складність. Крім того, часто виникають проблеми з продуктивністю під час запуску високополігональних 3D-об’єктів на комп'ютерах.

Зрештою, рішення про вибір рушія для 3D-проекту залежить від конкретної ситуації та потреб. Якщо вам потрібна інтенсивна фізика, анімація, 3D-моделі, звук та інші функції, може бути вигідно вибрати 3D-рушій, тоді як стандартний рушій може бути кращим для простіших завдань.

1.3 Unreal Engine

Unreal Engine[3], також відомий як UE - це інструмент для розробки відеоігор від компанії Epic Games, яка займається розробкою відеоігор та програмного забезпечення. За допомогою цього інструменту розробники мають можливість створювати симуляції, редагувати відео та звук, а також рендерити анімацію. Розробники використовували його для створення деяких з найпопулярніших ігор на сучасному ринку.

Різноманітні групи розробників, що працюють на комп'ютерах високого класу, використовують Unreal Engine для створення ігор.

Однією з головних переваг є те, що він безкоштовний і зручний для початківців. Epic Games прагне популяризувати своє програмне забезпечення по всьому світу за допомогою правильних стратегічних союзників, щоб більше людей могли спробувати себе в розробці. Як компанія з більш ніж 25-річним досвідом, Epic Games розробила програмне забезпечення, яке вважається стандартом AAA для розробки відеоігор як частини найбільш швидкозростаючого ринку у світі. AAA - неформальний термін, що позначає клас високобюджетних комп'ютерних ігор.

Написаний на C++, рушій Unreal Engine відрізняється високим ступенем портативності, і користувачі мають можливість використовувати його на різних платформах, включаючи iOS, Android, Windows, PlayStation і Xbox.

Хоча існують і більш ранні ітерації Unreal Engine, остання версія цього рушія стала важливим досягненням, оскільки дозволяє розробникам створювати і включати в свої ігри кастомні мапи. Unreal Engine також був одним з перших рушіїв, який дозволив модифікаторам легко створювати власний контент, щоб інші могли бачити і грати.

Unreal Engine використовує C++ як мову програмування для розробки ігор у цьому інструменті. C++ забезпечує загальну стабільність та відмінний розподіл пам'яті, саме тому розробники Unreal Engine використовують її.

Однак C++ також вважається досить складною мовою для вивчення, незважаючи на те, що це найпоширеніша мова програмування у світі.

C++ має різні проблеми, такі як кодування .h та .cpp для кожного класу, і є дуже вимогливою. Ця мова, безумовно, для професійних розробників через її високу криву навчання, хоча вона дає розробникам досить багато можливостей для виконання своєї роботи.

1.4 Графічні переваги Unreal Engine 5

Система рендерингу в Unreal Engine використовує конвеєри DirectX 11 і DirectX 12, які включають відкладене затінення, глобальне освітлення, освітлену прозорість і постобробку, а також симуляцію частинок на GPU з використанням векторних полів.

Unreal Engine - це потужний ігровий рушій, який відомий своєю високоякісною графікою та розширеними можливостями рендерингу. Є кілька причин, чому Unreal Engine має гарну графіку:

Передова технологія рендерингу: Unreal Engine використовує конвеєр рендерингу на фізичній основі (PBR)[4], який є методом створення реалістичних зображень шляхом імітації взаємодії світла з різними матеріалами. Це дозволяє отримати більш реалістичне освітлення, віддзеркалення та тіні у грі.

Високоякісні текстури та матеріали: Unreal Engine містить велику бібліотеку високоякісних текстур і матеріалів, які можна використовувати для створення реалістичних поверхонь і об'єктів у грі.

Динамічне освітлення: Unreal Engine використовує динамічне освітлення, яке дозволяє в реальному часі змінювати освітлення в грі на основі положення та інтенсивності джерел світла. Це створює більш захоплюючі та реалістичні світлові ефекти. З появою Lumen, Unreal Engine 5 підняв графіку на абсолютно новий рівень.

Розширені ефекти частинок: Unreal Engine включає вдосконалені ефекти частинок, які дозволяють створювати реалістичні дим, вогонь та інші спецефекти в грі.

Підтримка графіки високої роздільної здатності: Unreal Engine здатен рендерити графіку у високій роздільній здатності, що забезпечує деталізовану та реалістичну графіку навіть на дисплеях з високою роздільною здатністю.

Важка пост-обробка: Unreal Engine використовує занадто багато постобробки, щоб зробити графіку більш привабливою.

1.5 Генеративні змагальні мережі (GAN) або автокодувальники.

Генеративні змагальні мережі (Generative Adversarial Networks, або скорочено GAN) - це підхід до генеративного моделювання з використанням методів глибокого навчання, таких як згорткові нейронні мережі.

Генеративне моделювання[5] - це неконтрольоване завдання машинного навчання, яке полягає в автоматичному виявленні та вивченні закономірностей або шаблонів у вхідних даних таким чином, щоб модель можна було використовувати для генерації або виведення нових прикладів, які, ймовірно, могли бути отримані з вихідного набору даних.

GAN - це розумний спосіб навчання генеративної моделі, який полягає у формулюванні проблеми як задачі керованого навчання з двома підмоделями: моделлю-генератором, яку ми навчаємо генерувати нові приклади, і моделлю-дискримінатором, яка намагається класифікувати приклади як справжні (з предметної області) або фальшиві (згенеровані). Ці дві моделі тренуються разом у змагальній грі з нульовою сумою, доки дискримінаторна модель не буде обманута приблизно в половині випадків, що означає, що генераторна модель генерує правдоподібні приклади.

GAN - це захоплююча галузь, що швидко розвивається, яка виконує обіцянки генеративних моделей завдяки своїй здатності генерувати реалістичні

приклади в різних проблемних сферах, зокрема в задачах перекладу "зображення в зображення", таких як переклад фотографій з літа на зиму або з дня на ніч, а також у створенні фотореалістичних фотографій об'єктів, сцен і людей, які навіть людина не може відрізнити від фальшивих.

GAN - це архітектура для автоматичного навчання генеративної моделі, яка розглядає неконтрольовану проблему як контрольовану і використовує як генеративну, так і дискримінантну модель.

GAN забезпечують шлях до складного доповнення даних, специфічних для конкретної галузі, і вирішення проблем, які вимагають генеративного рішення, наприклад, перекладу зображення в зображення.

1.6 Прогностичне моделювання

Для цього потрібен навчальний набір даних, який використовується для навчання моделі, що складається з декількох прикладів, які називаються вибірками, кожна з яких має вхідні змінні (X) і вихідні мітки класів (y). Модель навчають, показуючи приклади вхідних даних, пропонуючи їй передбачити вихідні дані та коригуючи модель, щоб зробити вихідні дані більш схожими на очікувані.

Таку корекцію моделі зазвичай називають контрольованою формою навчання, або навчанням під наглядом (supervised learning).

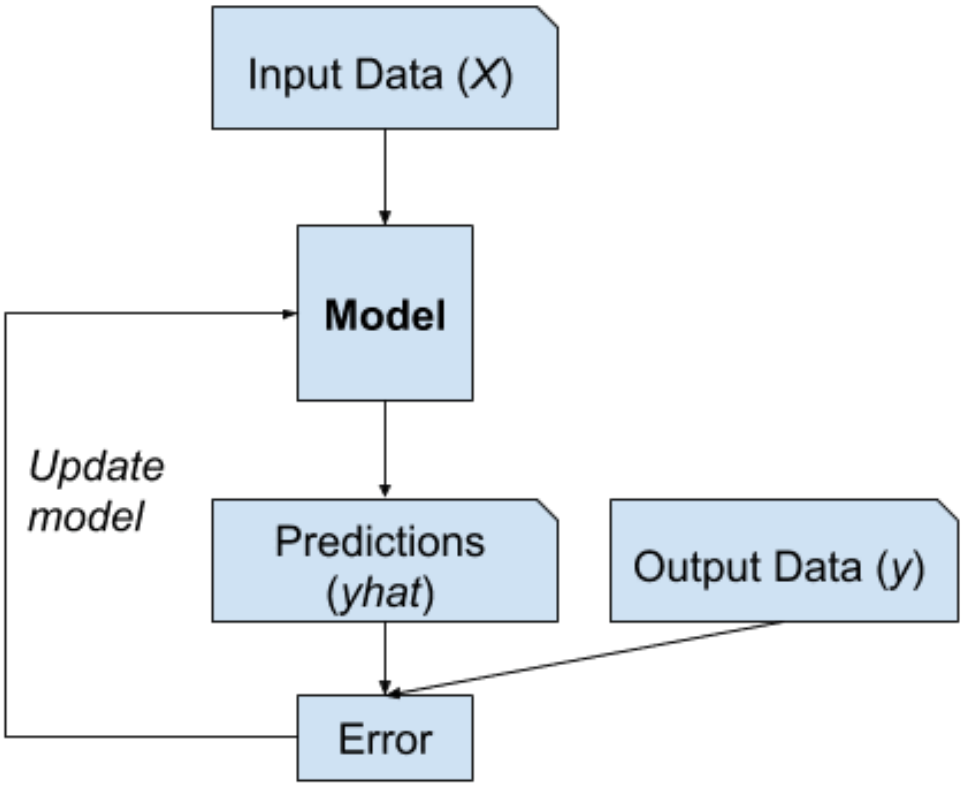

Рисунок 1.1. Приклад керованого навчання

Приклади проблем керованого навчання включають класифікацію і регресію, а приклади алгоритмів керованого навчання включають логістичну регресію і випадковий ліс.

Існує й інша парадигма навчання, коли модель має лише вхідні змінні (X) і не має вихідних змінних (y).

Модель будується шляхом вилучення або узагальнення закономірностей у вхідних даних. Модель не коригується, оскільки вона нічого не передбачає.

Другим основним типом машинного навчання є описовий або неконтрольований підхід до навчання. Тут нам дають лише вхідні дані, а мета полягає в тому, щоб знайти "цікаві закономірності" в даних. [...] Це набагато менш чітко визначена проблема, оскільки нам не кажуть, які саме закономірності шукати, і немає очевидної метрики помилок (на відміну від керованого навчання, де ми можемо порівняти наше передбачення y для заданого x зі спостережуваним значенням).

Відсутність корекції зазвичай називають неконтрольованою формою навчання, або навчанням без нагляду (unsupervised learning).

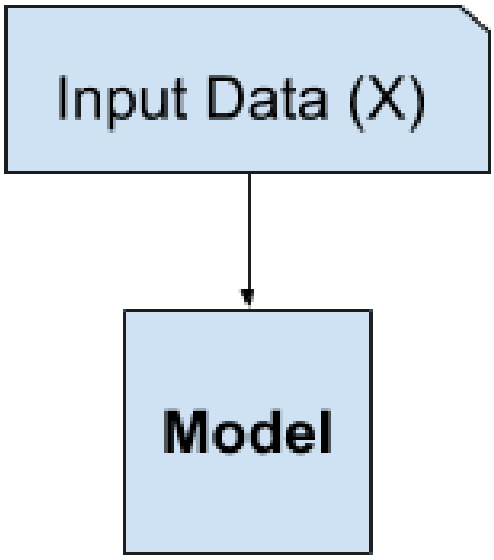

Рисунок 1.2 Приклад неконтрольованого навчання

Прикладами проблем неконтрольованого навчання є кластеризація і генеративне моделювання, а прикладами алгоритмів неконтрольованого навчання є K-середні і генеративні змагальні мережі.

1.7 Застосування машинного навчання

Машинне навчання - одна з найцікавіших технологій, з якими коли-небудь доводилося стикатися. Як видно з назви, вона дає комп'ютеру те, що робить його більш схожим на людину: здатність до навчання. Машинне навчання активно використовується сьогодні, можливо, у набагато більшій кількості сфер, ніж можна було б очікувати.

Сьогодні компанії використовують машинне навчання для покращення бізнес-рішень, підвищення продуктивності, виявлення хвороб, прогнозування погоди та багатьох інших речей. З експоненціальним зростанням технологій нам не тільки потрібні кращі інструменти для розуміння даних, які ми маємо

зараз, але й потрібно підготуватися до даних, які ми матимемо в майбутньому. Для досягнення цієї мети нам потрібно будувати інтелектуальні машини. Ми можемо написати програму для виконання простих речей. Але здебільшого вбудувати в неї інтелект складно. Найкращий спосіб зробити це - дати можливість машинам самим навчатися. Механізм навчання - якщо машина може вчитися на основі вхідних даних, то вона виконує важку роботу за нас. Саме тут вступає в дію машинне навчання. Ось деякі з найпоширеніших прикладів: розпізнавання зображень, розпізнавання мови, виявлення шахрайства, безпілотні автомобілі, медична діагностика, торгівля на фондовому ринку, віртуальна примірка і т.д.

Розпізнавання зображень є однією з причин буму, який можна було б пережити в області глибокого навчання. Завдання, яке починалося з класифікації зображень котів та собак, зараз еволюціонувало до рівня розпізнавання облич та реальних кейсів, заснованих на ньому, таких як відстеження відвідуваності співробітників.

Крім того, розпізнавання зображень допомогло зробити революцію в галузі охорони здоров'я, застосувавши інтелектуальні системи для розпізнавання та діагностики захворювань.

Розумні системи на основі розпізнавання мови, такі як Alexa і Siri, безумовно, зустрічалися і використовувалися для спілкування з ними. В основі цих систем лежать системи розпізнавання мовлення. Ці системи розроблені таким чином, що вони можуть перетворювати голосові інструкції в текст.

Ще одне застосування розпізнавання мовлення, з яким ми можемо зіткнутися в повсякденному житті, - це виконання пошуку в Google, просто розмовляючи з ним.

У сучасному світі більшість речей оцифровано, починаючи від купівлі зубних щіток і закінчуючи багатомільйонними транзакціями - все є доступним і простим у використанні. Але з цим процесом оцифрування збільшилася кількість випадків шахрайських транзакцій та шахрайської діяльності. Виявити

їх не так просто, але системи машинного навчання дуже ефективно справляються з цими завданнями.

Завдяки цим програмам лише тоді, коли система виявляє "червоні прапорці" в діяльності користувача, адміністратор отримує відповідне сповіщення, щоб ці випадки можна було належним чином відстежувати на предмет спаму чи шахрайства.

Можна було б припустити, що за кермом автомобіля перебуває якийсь привид, якби ми коли-небудь побачили машину без водія, але завдяки машинному навчанню та глибокому навчанню в сучасному світі це стало можливим, а не історією з якоїсь вигаданої книжки. Хоча алгоритми і технічний стек, що стоять за цими технологіями, є дуже досконалими, але в основі лежить машинне навчання, яке зробило ці програми можливими.

Найпоширенішим прикладом такого використання є автомобілі Tesla, які добре протестовані і перевірені для автономного водіння.

1.8 Основні можливості і переваги Google Colab

Google Colab (Colaboratory) - це безкоштовний сервіс від Google, який надає можливість використовувати хмарні обчислення за допомогою середовища Jupyter Notebook. Основні особливості і переваги Google Colab включають:

Безкоштовні ресурси GPU і TPU:

Colab надає безкоштовний доступ до графічних та тензорних процесорів (GPU і TPU), що дозволяє вам виконувати швидкі обчислення для задач машинного навчання та глибокого навчання без необхідності власного обладнання.

Легка інтеграція з Google Drive:

Ви можете легко зберігати та обмінюватися своїми проектами через Google Drive, що робить спільну роботу та збереження даних зручними.

Підтримка Jupyter Notebooks:

Colab використовує Jupyter Notebooks, що робить код більш інтерактивним та зручним для експериментів та візуалізацій.

Бібліотеки та пакети:

Більшість популярних бібліотек та пакетів Python вже встановлені, що полегшує роботу з даними та обчисленнями.

Спільна робота та обмін:

Ви можете дозволити іншим користувачам переглядати, коментувати або робити зміни у вашому ноутбуці для спільної роботи.

Легка налаштованість середовища:

Ви можете легко встановлювати власні бібліотеки та пакети, що дозволяє налаштовувати середовище для своїх потреб.

Широкий функціонал:

Google Colab має доступ до багатьох інших служб Google, таких як BigQuery, Google Sheets, Google Drive та інших, що розширює можливості аналізу даних та взаємодії з іншими сервісами.

Відсутність необхідності встановлення:

Все, що вам потрібно для роботи, - це браузер та підключення до Інтернету. Немає потреби встановлювати та конфігурувати середовище на власному комп'ютері.

Google Colab - це потужний інструмент для науковців, дослідників та розробників, які шукають зручне та доступне середовище для виконання обчислень у сферах машинного навчання та аналізу даних.

1.9 Огляд основних можливостей Blender

Blender [6] - це безкоштовний пакет для створення тривимірних зображень з відкритим вихідним кодом. З Blender ми можемо створювати 3D-візуалізації, такі як нерухомі зображення, 3D-анімації та VFX-знімки. Ви також

можете редагувати відео. Його інтерфейс використовує OpenGL для забезпечення узгодженої роботи на всіх підтримуваних апаратних засобах і платформах.

Програмне забезпечення для створення 3D, таке як Blender, має додаткову технічну складність пов'язану з технологіями, що лежать в його основі. Такі терміни, як UV-карти, матеріали, шейдери, сітки та "підрозділи" є інструментами цифрового художника, і їхнє розуміння, навіть у загальних рисах, допоможе вам використовувати Blender з максимальною користю.

Ключові особливості:

Blender - це повністю інтегрований пакет для створення 3D-контенту, що пропонує широкий спектр необхідних інструментів, включаючи моделювання, рендеринг, анімацію та монтаж, відеомонтаж, VFX, композицію, текстурування та багато типів симуляцій.

Він є крос-платформним, з графічним інтерфейсом OpenGL, який є єдиним для всіх основних платформ (і налаштовується за допомогою скриптів Python).

Він має високоякісну 3D-архітектуру, що забезпечує швидкий та ефективний робочий процес створення.

Він може похвалитися активною підтримкою спільноти. Повний список сайтів можна знайти на blender.org/community.

Має невеликий виконуваний файл, який за бажанням може бути портативним.

Висновки до розділу

Штучний інтелект можливо використовувати у всіх сферах діяльності. 3D графіка не стала виключенням. А оскільки 3D використовується у більшості прибуткових бізнесів від мерчу і закінчуючи анімацією та іграми, то використання штучного інтелекту не тільки зменшить об’єм роботи, а й прямопропорційно збільшить прибуток.

У бізнесі головними критеріями слугують швидкість та прибуток. Поки ти будеш закінчувати проект, інші компанії вже будуть ділити свою частку. Таким чином зменшення обсягу роботи призводить до збільшення можливостей у виділені ресурсів на розвиток, чи піар.

Відповідно до критеріїв та планів на розробку 3д проекту, було вирішено зупинитися на рушії Blender для зручного текстурингу і Google Colab для написання коду та генерації об’єкту.

2 ДОСЛІДЖЕННЯ ПРИНЦИПУ РОБОТИ ТЕХНОЛОГІЙ

2.1 Розробка блок-схеми 3d проекту з використанням ШІ

Головне на що треба звернути увагу на початковому етапі розробки проекту – це його структура. Якщо не створити зрозумілу структуру проекту, то з часом, коли проект буде розростатися, буде досить важко щось змінити , чи загалом розібрати код. Хоча нема єдиного стандарту як називати папки, але є приблизна структура , яку краще дотримуватися . Це передусім важливо у великих проектах та командній розробці.

На Рисунок 2.1 зображена блок-схеми веб-додатку з 3д-графікою у середовищі Google Colab:

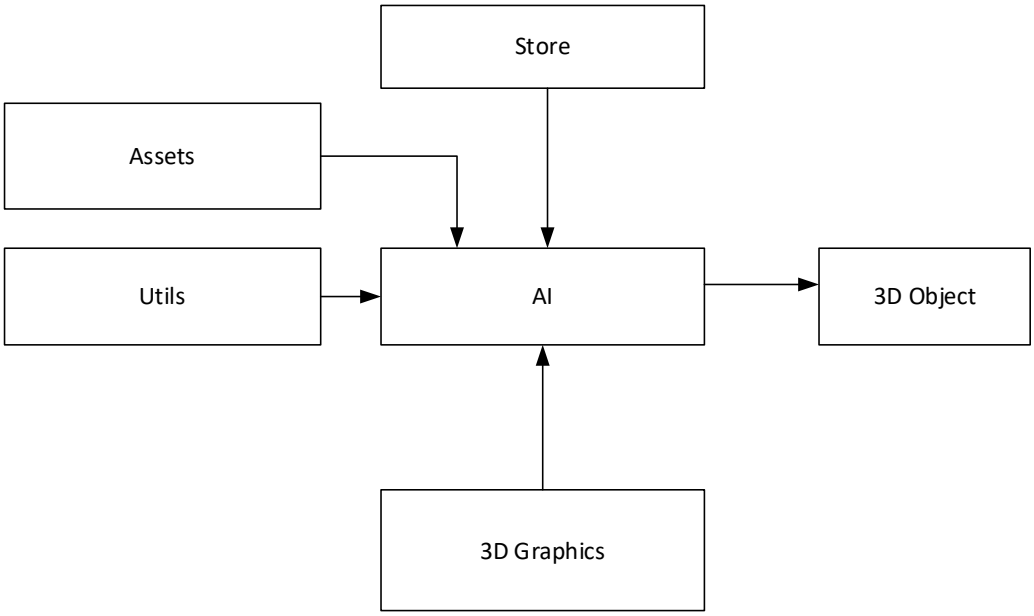

Рисунок 2.1 Блок-схема створення 3д об’єкту через ШІ

На рисунку 2.1 зображені основні блоки, та як вони взаємодіють між собою, але лише гарний приклад.

AI – це набір даних, що відповідає за аналіз та виконання поставленої задачі.

Store – папка, яка відповідає за збереження даних та їх обробку, представляє собою простір, з якого можна брати дані, та в який їх можна зберігати.

Assets – папка для завантажених картинок, текстур та моделей, що будуть використовуватися як приклад.

Utils – це місце, де ви можете розміщувати невеликі фрагменти, які можна використовувати в програмі. Маленькі функції для створення великих речей.

3D Object – це кінцевий продукт, що ми отримуємо після обробки AI.

3D Graphics – файли , які охоплюють увесь код 3D-об’єктів та їх методи додавання та інтеграції з основним кодом.

Відповідно до створеної структури файлів та принципу взаємодії цих файлів та папок, буде розроблений проект, який поєднає в собі всі основні концепсії та принципи вказані у розробленій блок-схемі.

2.2 Огляд основних можливостей Unreal Engine 5

Unreal Engine 5 – це потужний рушій, що має майже необмежені можливості для розробника в сфері 3D, ігор, чи анімацій . Але перед початком розробки проекту , необхідно розібрати основні можливості та їх принцип роботи в програмі, задля кращого розуміння та швидшої розробки проекту.

Тож, для початку необхідно створити проект, тому одразу обираємо заготовлений розробниками пресет, мову C++, чи Blueprint, платформу на якій буде використовуватися продукт і пресет якості. Після цього відкриється проект по обраному пресету і можно приступати до виконання.

Усі асети будуть братися через quixel bridge в папці контент. Для простого візуального програмування використовується мова blueprint з нодами, а для більш складних потреб C++. Blueprint відкривається автоматично при створені проекту.

2.3 Огляд основних можливостей Auto-LOD

Auto-LOD, також відомий як автоматичний рівень деталізації, є функцією в Unreal Engine, яка спрямована на спрощення та оптимізацію управління рівнем деталізації (LOD) для 3D-моделей. LOD - це техніка, яка використовується в 3D-комп'ютерній графіці з метою підвищення продуктивності рендерингу шляхом зменшення складності 3D-моделі з ростом відстані від камери. Unreal Engine надає інструменти для ручного створення LOD для ваших моделей, але також пропонує автоматичну генерацію LOD. Нижче наведено кроки використання функції Auto-LOD в Unreal Engine:

Імпортуйте модель: Спершу імпортуйте вашу 3D-модель в Unreal Engine. Для цього можна використовувати різні програми для 3D-моделювання для створення початкової моделі.

Увімкніть Auto-LOD: В панелі вмісту виберіть ваш актив 3D-моделі. У розділі "Деталі" ви знайдете параметр "Статична сітка". Тут ви можете увімкнути опцію "Автоматично обчислювати відстань LOD".

Встановіть групу LOD: У тому ж розділі "Статична сітка" ви можете визначити групу LOD. Це визначає, які налаштування якості та профілі налаштувань будуть використовуватися для генерації LOD. Unreal Engine має попередньо визначені групи LOD, такі як "WorldStatic", "WorldDynamic", "Weapon" та інші.

Налаштуйте параметри LOD: Ви можете подальше налаштувати процес генерації LOD, визначивши параметри, такі як кількість рівнів LOD, параметри зменшення та розмір екрану. Це дозволяє контролювати ступінь агресивності автоматичної генерації LOD.

Створіть LOD: Натисніть кнопку "Генерувати проксі-меш". Unreal Engine проаналізує вашу 3D-модель і створить її менш деталізовані версії для різних рівнів LOD.

Готові рівні LOD, будуть автоматично використовуватися в системі рівнів деталізації, коли камера віддалятиметься від об'єкта.

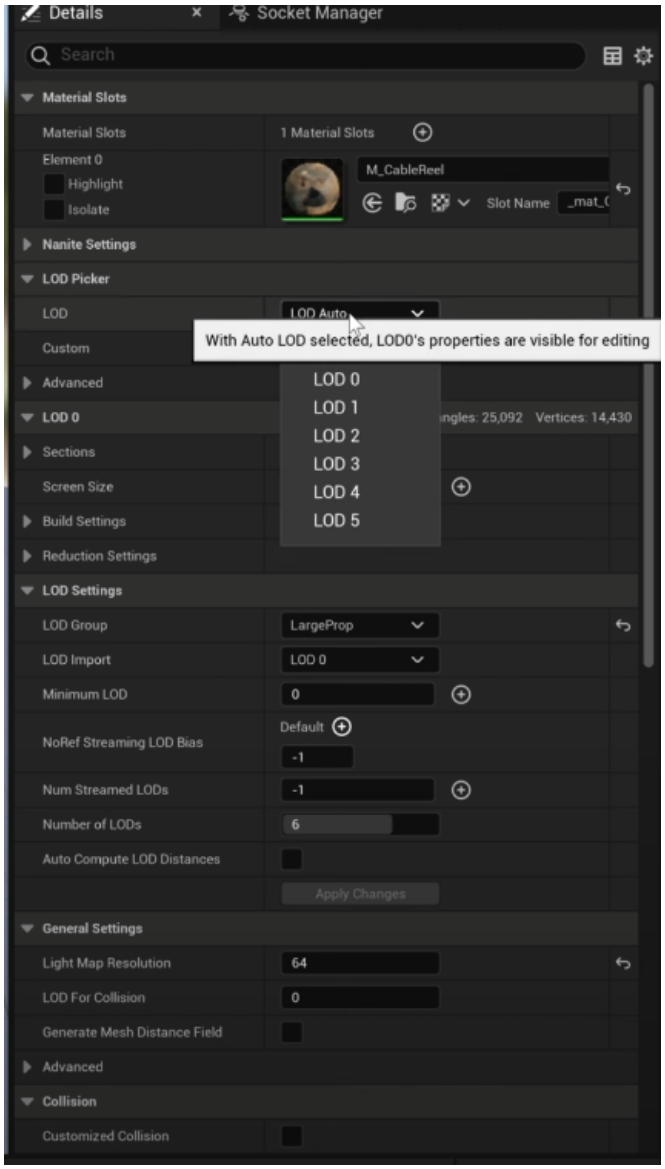

Рис.2.2 Auto-LOD Unreal Engine

Авто-LOD спрощує процес генерації LOD, проте не завжди гарантує найоптимальніші результати. Для критичних об'єктів може бути необхідно налаштувати LOD вручну або подальше налаштувати автоматично згенеровані рівні LOD, щоб забезпечити гарну продуктивність гри без жертвування якістю зображення.

2.4 Blueprints

Система візуальних скриптів Blueprint [7] в Unreal Engine - це повноцінна система написання ігрових скриптів, заснована на концепції використання вузлового інтерфейсу для створення елементів ігрового процесу в Unreal Editor. Як і багато інших поширених мов сценаріїв, вона використовується для визначення об'єктно-орієнтованих класів або об'єктів у рушії. Використовуючи UE, ви часто побачите, що об'єкти, визначені за допомогою Blueprint, в розмовній мові називають просто "Blueprints".

Ця система є надзвичайно гнучкою та потужною, оскільки надає дизайнерам можливість використовувати практично весь спектр концепцій та інструментів, які зазвичай доступні лише програмістам. Крім того, специфічна для Blueprint розмітка, доступна в реалізації Unreal Engine на C++, дозволяє програмістам створювати базові системи, які можуть бути розширені дизайнерами.

2.5 OpenAI’s Point-E

Хоча нещодавні роботи з генерації 3D-об'єктів на основі тексту показали багатообіцяючі результати, сучасні методи зазвичай вимагають декількох графічних годин для створення одного зразка. Це різко контрастує з сучасними генеративними моделями зображень, які створюють зразки за кілька секунд або хвилин[9]. Ми досліджуємо альтернативний метод генерації 3D-об'єктів, який дозволяє створювати 3D-моделі всього за 1-2 хвилини на одному графічному процесорі. Наш метод спочатку генерує одне синтетичне зображення за допомогою моделі дифузії текст-зображення, а потім створює 3D хмару точок за допомогою другої моделі дифузії, яка залежить від згенерованого зображення. Хоча наш метод все ще не дотягує до найсучасніших з точки зору

якості вибірки, він на один-два порядки швидше робить вибірку, пропонуючи практичний компроміс для деяких випадків використання.

2.6 PointE: система для генерації 3D хмарних точок на основі складних підказок(prompt)

Завдяки нещодавньому вибуху генеративних моделей[11] перетворення тексту в зображення, з'явилася можливість генерувати та модифікувати високоякісні зображення з описів на природній мові за кілька секунд. У цій роботі ми зосереджуємося саме на проблемі перетворення тексту в 3D-зображення, яка має значний потенціал при створенні 3D-контенту для широкого спектру застосувань.

Сучасні методи синтезу текст-тривимірних зображень, як правило, відносяться до однієї з двох категорій:

Методи, які навчають генеративні моделі безпосередньо на сумісних текстових і 3D або немаркованих 3D-даних. Хоча ці методи можуть використовувати існуючі генеративні підходи до моделювання і ефективного виготовлення зразків, їх важко масштабувати до різноманітного та складного текстового запиту через відсутність великомасштабних наборів 3D-даних.

Методи, які використовують попередньо навчені моделі текст-зображення для оптимізації диференційованих 3D-зображень. Ці методи часто здатні обробляти складні та різноманітні текстові підказки, але вимагають дорогих процесів оптимізації для створення кожного зразка. Крім того, через відсутності сильного 3D-пріоритету, ці методи можуть потрапляти в локальні мінімуми, які не відповідають значущим або когерентним 3D-об'єктам.

Ми прагнемо поєднати переваги обох категорій, об’єднавши модель тексту в зображення з моделлю зображення в 3D[8]. Наша модель «текст-зображення» використовує великий набір пар (текст, зображення), що дозволяє виконувати різноманітні та складні підказки, тоді як наша модель «зображення-

3D» навчається на меншому наборі даних пар (зображення, 3D). Щоб створити 3D-об’єкт із текстової підказки, ми спочатку беремо вибірку зображення за допомогою моделі перетворення тексту в зображення, а потім беремо вибірку 3D-об’єкта, створеного на основі зразкового зображення. Обидва ці кроки можна виконати за кілька секунд і не вимагають дорогих процедур оптимізації. По-перше, текстова підказка подається в модель GLIDE для створення синтетичного рендерингу. Далі хмара точок дифузії обробляє це зображення для створення 3D хмари точок RGB.

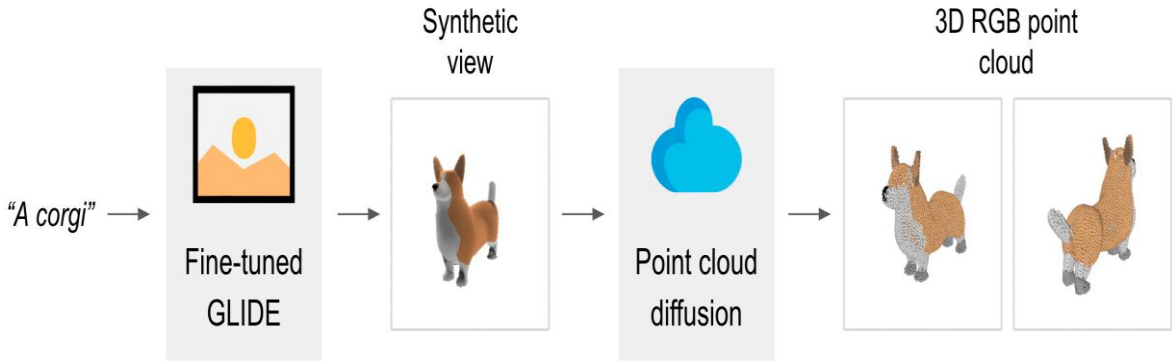

Рисунок 2.3 Загальний огляд нашого перетворення

GLIDE - модель OpenAI генерації зображення за його описом. Дифузійна модель, що забезпечує продуктивність, конкурентоспроможну з DALL-E, при використанні менше однієї третини її параметрів (3.5 млрд. параметрів проти 12 млрд.). Нещодавні дослідження показали, що дифузійні моделі мають здатність генерувати високоякісні синтетичні зображення.

2.7 Метод PointE

Замість того, щоб навчати одну генеративну модель безпосередньо створювати хмари точок на основі тексту, ми розбиваємо процес генерації на три кроки. Спочатку ми генеруємо синтетичне зображення на основі текстового підпису. [10]Далі ми створюємо грубу хмару точок (1,024 точки) на основі на основі синтетичного вигляду. І, нарешті, ми створюємо хмару точок (4096 точок) на основі хмари точок з низькою роздільною здатністю хмари точок низької роздільної здатності та синтетичного вигляду. На практиці ми припускаємо, що зображення містить релевантну інформацію з тексту, і не прив'язуємо хмари точок до тексту явно.

Для генерації умовних синтетичних зображень з текстом ми використовуємо модель GLIDE з 3 мільярдами параметрів, налаштовану на відрендерених 3D-моделях з набору даних. Для генерації хмар точок з низькою роздільною здатністю ми використовуємо умовну, інваріантну до перестановок модель дифузії. Для повторної дискретизації цих хмар точок з низькою роздільною здатністю ми використовуємо подібну (але меншу) модель дифузії, яка додатково залежить від хмари точок з низькою роздільною здатністю.

Модель навчена на наборі даних з декількох мільйонів 3D-моделей та пов'язаних з ними метаданих. Вона обробляє набір даних у візуалізовані зображення, текстові описи та хмари 3D точок PointE: Система генерації хмар 3D точок з комплексних підказок з відповідними кольорами RGB для кожної точки.

<u>Point-E</u> використовує методи дифузії, зазначені у роботі Гауса та ін. Він ставиnm за мету взяти зразок із певного розподілу q(x0), використовуючи наближення нейронної мережі p(x0):

$$q(x_t|x_{t-1}) := \mathrm{N}(x_t;\sqrt{(1-\beta_t)}x_{t-1},\beta_t\mathbf{I})$$

Під Гаусовою дифузією ми визначаємо процес шуму для цілих часових кроків t 2 [0; T]. Інтуїтивно цей процес поступово додає Гаусівський шум до сигналу з величиною шуму, доданого на кожному кроці часу, визначеному шумомим графіком t. Ми використовуємо графік шуму таким чином, щоб, на остаточний часовий крок t = T, зразок xT майже не містить інформації (тобто виглядає як шум Гауса). Зауважимо, що можна безпосередньо перейти до заданого часового кроку шумового процесу без запуску всього ланцюгу:

$$x_t=\sqrt{(\bar{a}_t)}x_0+\sqrt{(1-\bar{a}_t)}\ Є,$$

$$\text{Де } Є \sim N\ (0, I) \text{ та } \bar{a}_t := \pi^t_{s=0}1-\beta_t$$

Під гаусовою дифузією ми визначаємо процес шуму для всіх цілих часових кроків t у діапазоні [0; T]. Інтуїтивно цей процес поступово додає гаусівський шум до сигналу з величиною шуму, яка додається на кожному кроці часу і визначається певним шумовим графіком t. Ми використовуємо графік шуму так, щоб на остаточний часовий крок t = T зразок xT майже не містив інформації (іншими словами, це нагадувало б шум Гауса).

2.8 PointE набір даних

Моделі навчені на кількох мільйонах 3D-моделей[10]. З них помітно, що формати та якість даних дуже різняться залежно від набору даних, через що було розроблено різні етапи постобробки, щоб забезпечити вищу якість даних.

Щоб перетворити всі дані в один загальний формат, було відтворено кожну 3D-модель із 20 випадкових ракурсів камери як зображення RGBAD за допомогою Blender, який підтримує різноманітні 3D-формати та постачається з оптимізованою системою відтворення. Для кожної моделі скрипт Blender нормалізує модель до обмежувального куба, налаштовує стандартні налаштування освітлення та, нарешті, експортує зображення RGBAD за допомогою вбудованого механізму візуалізації Blender у реальному часі. Потім було перетворено кожен об’єкт на кольорову хмару точок, використовуючи його відтворення. Зокрема, спочатку побудували щільну хмару точок для кожного об’єкта, обчисливши точки для кожного пікселя в кожному зображенні RGBAD. Ці хмари точок зазвичай містять сотні тисяч нерівномірно розташованих точок, тому додатково була використана вибірка найдальших точок, щоб створити рівномірні хмари точок 4K. Створюючи хмари точок безпосередньо з візуалізацій, змогли уникнути різноманітних проблем, які могли виникнути під час спроби вибірки точок безпосередньо з 3D-сіток, наприклад точок вибірки, які містяться в моделі, або роботи з 3D-моделями, які зберігаються у незвичних форматах файлів. Нарешті, були застосовані різні евристики, щоб зменшити частоту низькоякісних моделей у наборі даних. По-перше, усунули плоскі об’єкти, обчисливши SVD кожної хмари точок і зберігши лише ті, де найменше сингулярне значення перевищувало певний поріг. Далі кластеризували набір даних за функціями CLIP (для кожного об’єкта усереднені функції для всіх рендерів). Деякі кластери містили багато низькоякісних категорій моделей, тоді як інші кластери виглядали більш різноманітними або придатними для інтерпретації. Ці кластери було об'єднано

в кілька сегментів різної якості, і зважена суміш отриманих сегментів була використана як остаточний набір даних.

2.9 модель синтезу GLIDE

Моделі хмар[10] точок базуються на візуалізованих представленнях з набору даних, які були згенеровані за допомогою того ж самого рендерера та налаштувань освітлення. З метою переконатися, що ці моделі правильно обробляють створені синтетичні види, ми маємо намір явно генерувати 3D-візуалізації, що відповідають розподілу нашого набору даних. Для цього ми точно налаштовуємо GLIDE, використовуючи суміш його оригінального набору даних і набору даних 3D-візуалізацій. Оскільки набір 3D-даних є невеликим порівняно з оригінальним навчанням GLIDE, архітектура нашої дифузійної моделі хмар точок враховує це. Зображення подаються через заморожену, попередньо навчену модель CLIP, а вихідна сітка використовується як жетони в трансформаторі. Кожен часовий крок t та зашумлений вхідний сигнал xt також подаються як маркери. Вихідні маркери, що відповідають xt, використовуються для прогнозування $\sum$.

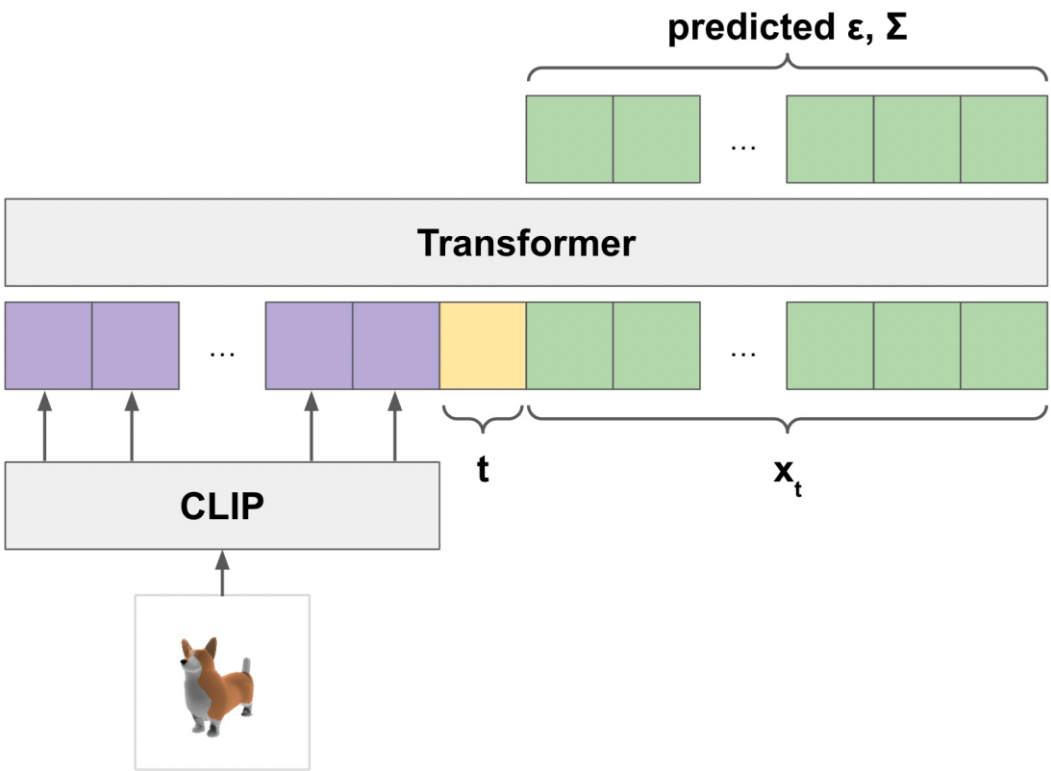

Рисунок 2.4 Архітектура дифузійної моделі хмари точок

Здійснюється вибіркове використання зображень із набору 3D-даних у 5% випадків, використовуючи оригінальний набір даних для решти 95%. Виконується тонке налаштування протягом 100 тисяч ітерацій, що відповідає кільком епохам у наборі 3D-даних (з умовою, що кожна точка зору візуалізації не повторюється точно двічі). З метою забезпечення постійного вибору зразків рендерингу в розподілі (а не лише у 5% випадків), додається спеціальний маркер до кожного текстового підказки 3D-рендеру, який вказує, що це 3D-рендер. Під час тестування перевіряється наявність цього маркера.

Висновки до розділу

Перше з чим треба розібратися на початку розробки – це приблизна блок-схема нашого проекту.

Після підготовки основних компонентів, плагінів та пресетів, перед нами встало завдання зробити огляд основних можливостей Blender, описаних основних переваг анрілу та новітній підхід з використанням компонентів та Auto-LOD . Компоненти в свою чергу дають можливість перевикористовувати код. Auto-LOD же дає можливість спрощення та оптимізації моделей для зручної роботи з рушієм. Далі підготувати папку з файлами для навчання, чи використання штучним інтелектом та обрати мову написання його поведінки. Через Blueprint можливості дуже обмежені, але прості в реалізації, тому використовуватиметься Blueprint.

3 РОЗРОБКА ПРОЕКТУ

3.1 Розгортання проекту

Після вирішення етапів і середовищ де будемо виконувати проект ми переходимо в Google Colab і підванажуємо репозиторій. Необхідно завантажити його у свій блокнот за посиланням на Github і обрати text2pointcloud.ipynb. Також не забуваємо змінити тип середовища виконання на GPU, якщо його ще не встановлено, і підключити.

Google Colab являє собою виділену віртуальну машину, що дозволяє виконувати певні операції в браузері. Тому, щоб процес обробки не займав більше 20 хвилин ми встановлюємо обробку GPU ядрами. Це прискорить усі процеси до хвилини.

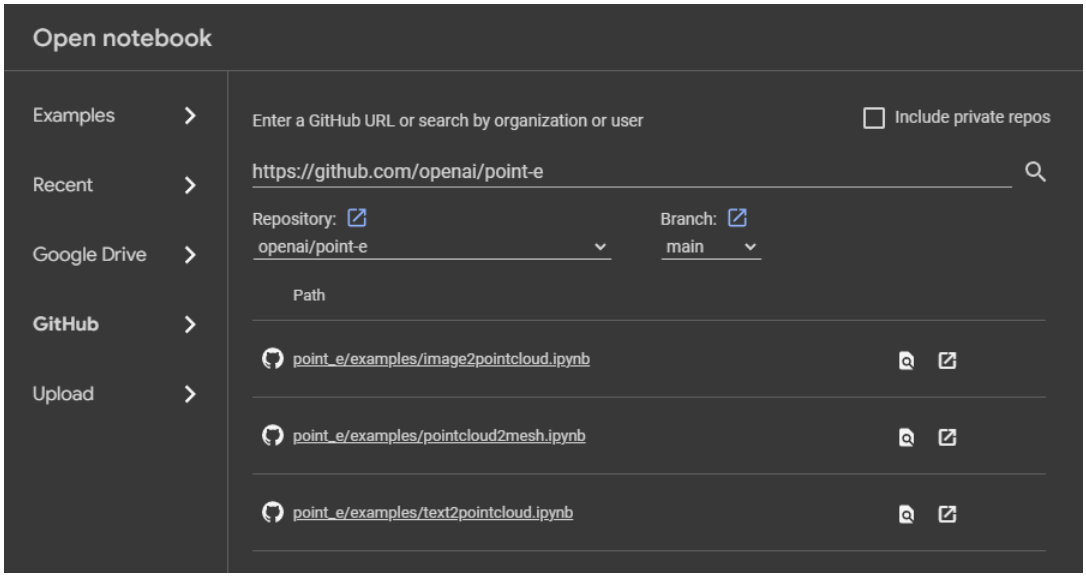

Рисунок 3.1 - Стартовий пресет

Перш ніж запустити комірки, нам знадобляться файли бібліотеки з оригінального репозиторію, а також потрібно встановити деякі залежності pip, тому ми зробимо це спочатку. Ми можемо зробити це за допомогою наступного коду в окремій комірці

```
!git clone https://github.com/openai/point-e.git
%cd ./point-e
!pip install -e .
```

Рисунок 3.2 - Завантаження бібліотек з оригінального репозиторію

Воно має успішно встановити потрібні нам залежності. Тепер ми можемо просто запустити наступні 3 клітинки від імпорту всіх бібліотек, завантаження базових моделей до ініціалізації зразка точок хмари.

3.2 Аналіз комірок

Після завантаження бібліотек ми маємо спочатку імпортувати ряд класів та функцій з різних модулів великого проекту Point-E з бібліотеки PyTorch, які пов'язані із генерацією тривимірних точкових хмар (point clouds). Давайте докладніше розглянемо імпорти:

Import torch. Це імпорт PyTorch, відкритої бібліотеки для роботи з нейронними мережами та тензорами.

from tqdm.auto import tqdm. Цей імпорт використовує бібліотеку tqdm для створення красивих інтерактивних смуг прогресу при виконанні циклів.

from point_e.diffusion.configs import DIFFUSION_CONFIGS, diffusion_from_config. Цей імпорт стосується конфігурацій і деяких функцій, пов'язаних із дифузією (diffusion) в моделі. Вірогідно, що це пов'язано з алгоритмами дифузії у генеративних моделях, але конкретний вміст залежить від внутрішньої структури проекту Point-E.

from point_e.diffusion.sampler import PointCloudSampler. Цей імпорт включає клас PointCloudSampler, який, ймовірно, відповідає за вибірку (sampling) точкових хмар з моделі.

from point_e.models.download import load_checkpoint. Тут імпортується функція load_checkpoint, яка ймовірно використовується для завантаження попередньо навчених контрольних точок моделі.

from point_e.models.configs import MODEL_CONFIGS, model_from_config. Цей імпорт стосується конфігурацій та функцій, пов'язаних зі створенням моделей. Model_from_config використовується для створення моделі з вказаною конфігурацією.

from point_e.util.plotting import plot_point_cloud. Цей імпорт включає функцію plot_point_cloud, яка, призначена для візуалізації точкових хмар.

```
[2] import torch
    from tqdm.auto import tqdm

    from point_e.diffusion.configs import DIFFUSION_CONFIGS, diffusion_from_config
    from point_e.diffusion.sampler import PointCloudSampler
    from point_e.models.download import load_checkpoint
    from point_e.models.configs import MODEL_CONFIGS, model_from_config
    from point_e.util.plotting import plot_point_cloud
```

Рисунок 3.3 - Імпорт класів та функцій Point-E з бібліотеки PyTorch

В наступній комірці відбувається ініціалізація та завантаження двох моделей: базової моделі (base_model) і моделі для узорського збільшення (upsampling) (upsampler_model). Давайте розглянемо це докладніше:

Визначення пристрою (device). Ця частина коду визначає пристрій, на якому будуть виконуватися обчислення. Використовується GPU (cuda), якщо він доступний, в іншому випадку використовується CPU.

```
device = torch.device('cuda' if torch.cuda.is_available() else 'cpu')
```

Рисунок 3.4 - Визначення пристрою (device)

Наступним етапом йде створення базової моделі:

base_name вказує на конфігурацію базової моделі.

model_from_config створює модель за вказаною конфігурацією.

base_model.eval() переводить модель у режим оцінки (evaluation), що може впливати на поведінку деяких шарів, таких як dropout.

diffusion_from_config створює об'єкт дифузії для базової моделі.

```
base_name = 'base40M-textvec'
base_model = model_from_config(MODEL_CONFIGS[base_name], device)
base_model.eval()
base_diffusion = diffusion_from_config(DIFFUSION_CONFIGS[base_name])
```

Рисунок 3.5 - Створення базової моделі.

Далі команда Аналогічно базовій моделі, але для узорського збільшення.

```
upsampler_model = model_from_config(MODEL_CONFIGS['upsample'], device)
upsampler_model.eval()
upsampler_diffusion = diffusion_from_config(DIFFUSION_CONFIGS['upsample'])
```

Рисунок 3.6 - Створення моделі для узорського збільшення

І наприкінці завантажується шар з контрольної точки для базової моделі і моделі узорського збільшення.

```
print('downloading base checkpoint...')
base_model.load_state_dict(load_checkpoint(base_name, device))

print('downloading upsampler checkpoint...')
upsampler_model.load_state_dict(load_checkpoint('upsample', device))
```

Рисунок 3.7 - Шар з контрольної точки для базової моделі і моделі узорського збільшення

У четвертій комірці створюється об'єкт класу PointCloudSampler, який, використовується для вибірки (sampling) точкових хмар з моделей base_model та upsampler_model. Давайте розглянемо параметри цього об'єкта:

device=device. Параметр, який вказує пристрій, на якому буде виконуватися вибірка точок. Це значення передається змінній device, яку ми визначили раніше (cuda або cpu).

models=[base_model, upsampler_model]. Список моделей, які будуть використовуватися для вибірки. В даному випадку, це базова модель (base_model) і модель для узорського збільшення (upsampler_model).

diffusions=[base_diffusion, upsampler_diffusion]. Список об'єктів дифузії, які використовуються для вибірки точок. Це важливі параметри для процесу генерації точок.

num_points=[1024, 4096 - 1024]. Кількість точок, які потрібно вибрати для кожної моделі. В даному випадку, для базової моделі обирається 1024 точки, а для моделі узорського збільшення решта (4096 - 1024) точок.

aux_channels=['R', 'G', 'B']. Список додаткових каналів. Воно пов'язано з кольоровою інформацією (канали Red, Green, Blue), яка може використовуватися під час генерації точок.

guidance_scale=[3.0, 0.0]. Параметр, який, ймовірно, визначає важливість вказаних направляючих величин. В даному випадку, для текстової інформації ('texts'), яка використовується як направляючий сигнал, задано значення 3.0, а для інших направляючих величин (вказаних пустим рядком), значення 0.0.

model_kwargs_key_filter=('texts', ''): Фільтр ключів для аргументів моделі. У мене використовується фільтр для ключа 'texts', тобто я враховую тільки аргументи з цим ключем.

```
[4] sampler = PointCloudSampler(
        device=device,
        models=[base_model, upsampler_model],
        diffusions=[base_diffusion, upsampler_diffusion],
        num_points=[1024, 4096 - 1024],
        aux_channels=['R', 'G', 'B'],
        guidance_scale=[3.0, 0.0],
        model_kwargs_key_filter=('texts', ''), # Do not condition the upsampler at all
    )
```

Рисунок 3.8 - Вибірка (sampling) тривимірних точкових хмар з моделей генерації

Отже, PointCloudSampler використовується для зручного та ефективного вибору точок із заданими моделями та параметрами.

У п’ятій комірці ми встановлюємо умову (або запит) для моделі, а потім використовуєте цей запит для генерації прикладу за допомогою самої моделі. Змінна prompt використовується як умова або запит для генеруючої моделі, а samples вже генерує приклад, заснований на вашому запиті.

```
[5] # Set a prompt to condition on.
    prompt = 'a red motorcycle'

    # Produce a sample from the model.
    samples = None
    for x in tqdm(sampler.sample_batch_progressive(batch_size=1, model_kwargs=dict(texts=[prompt]))):
        samples = x

    130/? [01:04<00:00, 1.16it/s]
```

Рисунок 3.9 - Встановлення умови для моделі та генерації прикладу

Наступною командою ми використовуємо результати, отримані від моделі, щоб створити та відобразити точковий хмаринку у просторі.

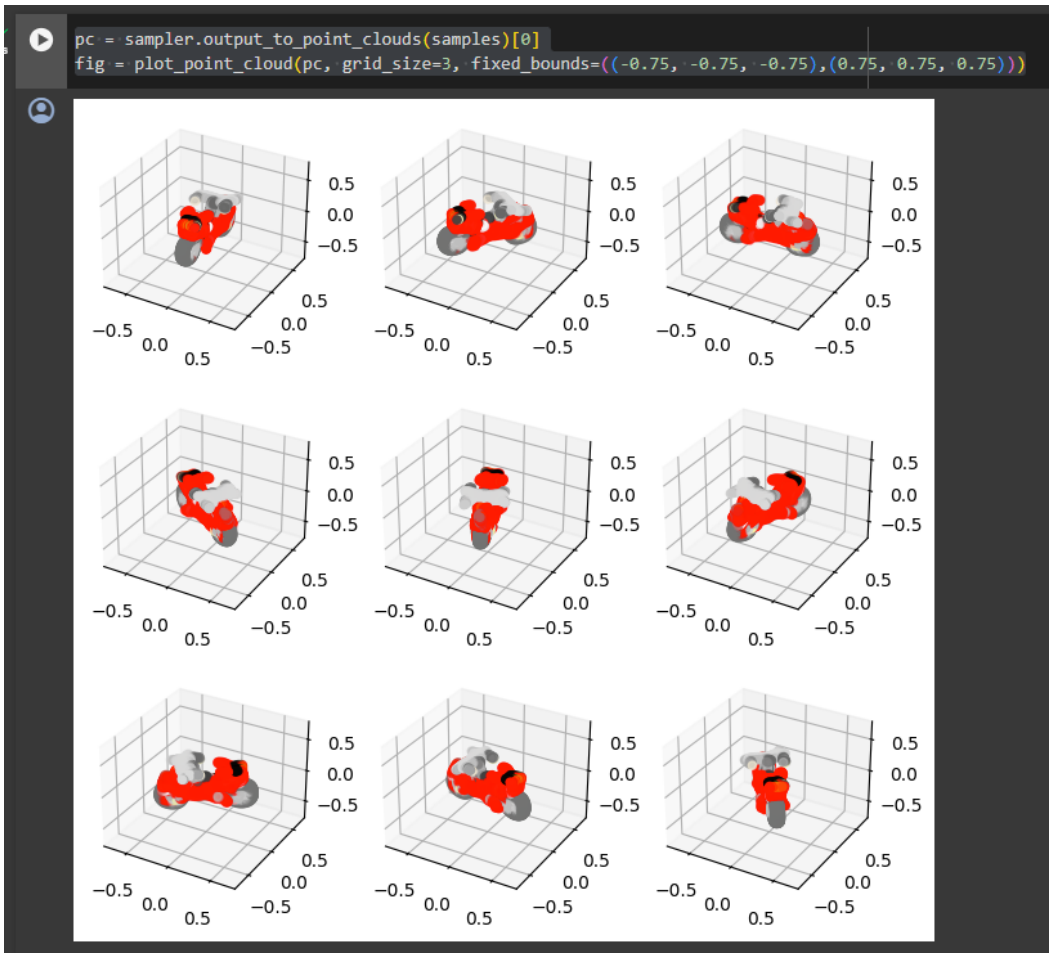

Рисунок 3.10 - Відображення результату

3.3 Покращення коду та вивід результату

3.3.1 Виділена машина для роботи

Для початку хотілося б дізнатись параметри виділеного нам пристрою. В цьому нам допоможе команда !nvidia-smi. Ця утиліта дозволяє адміністраторам запитувати стан пристрою графічного процесора та, маючи відповідні привілеї, дозволяє змінювати стан пристрою графічного процесора. З отриманої інформації ми бачимо, що нам було виділено машину з ядрами на базі Tesla T4 і 16 гігабайтами оперативної пам’яті. Цього більш ніж достатньо виконання поставлених завдань.

```
[1] !nvidia-smi

Tue Jan  2 15:23:06 2024
+-----------------------------------------------------------------------------------------+
| NVIDIA-SMI 535.104.05             Driver Version: 535.104.05   CUDA Version: 12.2     |
|-----------------------------------------+----------------------+----------------------+
| GPU  Name                 Persistence-M | Bus-Id        Disp.A | Volatile Uncorr. ECC |
| Fan  Temp   Perf          Pwr:Usage/Cap |         Memory-Usage | GPU-Util  Compute M. |
|                                         |                      |               MIG M. |
|=========================================+======================+======================|
|   0  Tesla T4                       Off | 00000000:00:04.0 Off |                    0 |
| N/A   71C    P8              12W /  70W |      0MiB / 15360MiB |      0%      Default |
|                                         |                      |                  N/A |
+-----------------------------------------+----------------------+----------------------+

+-----------------------------------------------------------------------------------------+
| Processes:                                                                              |
|  GPU   GI   CI        PID   Type   Process name                            GPU Memory |
|        ID   ID                                                             Usage      |
|=========================================================================================|
|  No running processes found                                                             |
+-----------------------------------------------------------------------------------------+
```

Рисунок 3.11 - Виділений пристрій

3.3.2 Перетворення зображення в 3D через модель кварків

На цей раз ми будемо перетворювати не текст в 3D, а саме зображення. Для цього необхідно файл завантажити в середовище за допомогою команди img = Image.open(‘Назва файлу’).

```
# Load an image to condition on.
img = Image.open('tree-1716991_1280.webp')

# Produce a sample from the model.
samples = None
for x in tqdm(sampler.sample_batch_progressive(batch_size=1, model_kwargs=dict(images=[img]))):
    samples = x

130/? [01:48<00:00, 1.46s/it]
```

Рисунок 3.12 - Перетворення зображення в 3D

Зараз було використано модель кварків якими ми фактично надсилаємо зображення, тому сама модель містить фактичний файл, що ми завантажили і одразу створить зразки (samples). Тому ми можемо візуалізувати вихідне зображення через команду img.

Рисунок 3.13- Візуалізацію зображення

Потім ми за допомогою семплера можемо конвертувати його в хмару точок і це те, що ми в коді називаємо pc. Таким чином при бажані можемо візуалізувати хмару точок у вигляді 2D.

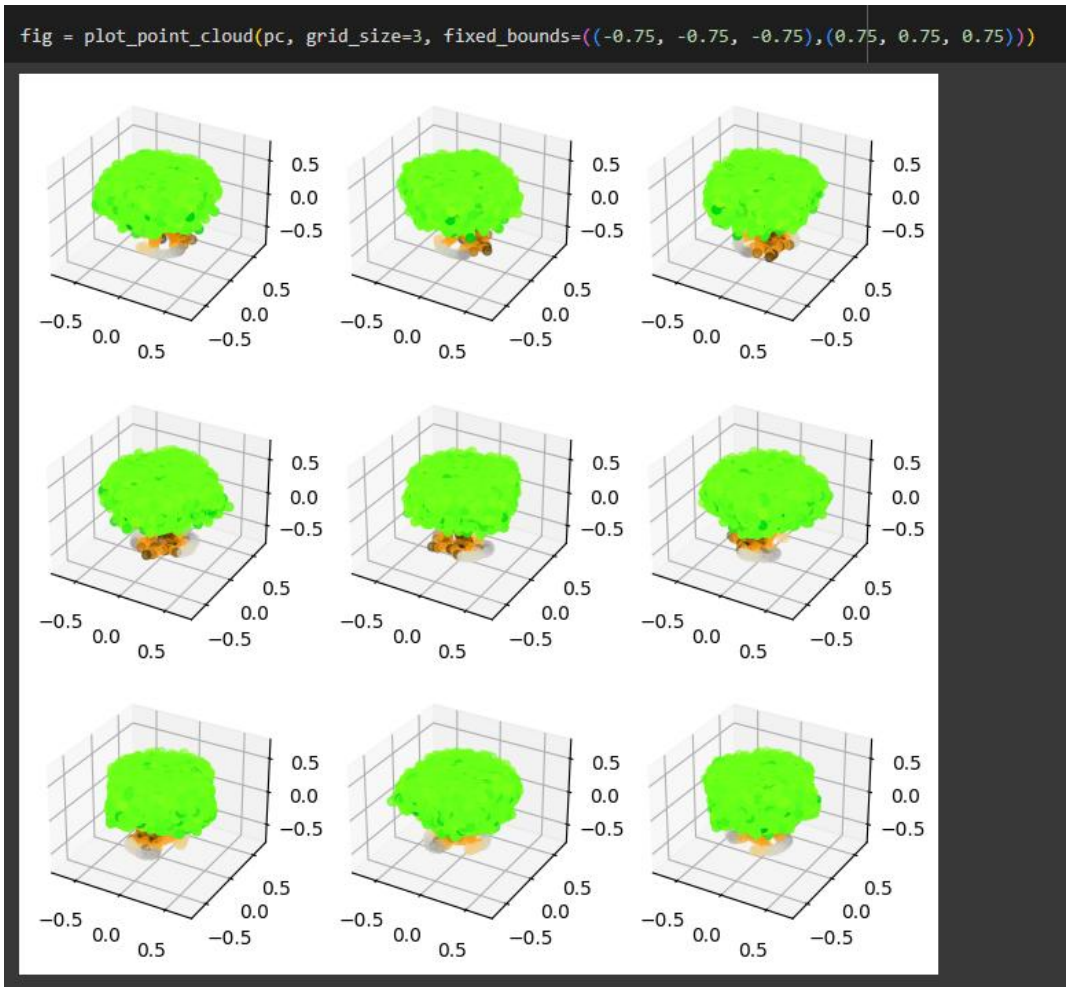

Рисунок 3.14 - Візуалізація хмари точок у 2D

На базі хмари точок ми можемо згенерувати 3D зображення. Для цього необхідно виконати 3 дії, а саме:

Створити об’єкт Figure, додати тривимірний розсіювальний графік (scatter plot). Координати точок задаються через x=pc.coords[:,0], y=pc.coords[:,1], z=pc.coords[:,2]. Кожна точка представлена як маркер, розмір та кольори якої задаються через параметр marker. Також сконфігурувати макет графіку, встановлюючи, що осі x, y та z мають бути приховані (visible=False). Це допомагає створити графік без видимих осей, щоб акцентувати увагу на точковій хмарі. Отже, після виконання цього коду об'єкт fig_plotly буде готовий для відображення графіка. Використаємо fig_plotly.show(), щоб побачити результат на екрані.

```
[14] import plotly.graph_objects as go

fig_plotly = go.Figure(
        data=[
            go.Scatter3d(
                x=pc.coords[:,0], y=pc.coords[:,1], z=pc.coords[:,2],
                mode='markers',
                marker=dict(
                  size=2,
                  color=['rgb({},{},{})'.format(r,g,b) for r,g,b in zip(pc.channels["R"], pc.channels["G"], pc.channels["B"])],
              )
            )
        ],
        layout=dict(
            scene=dict(
                xaxis=dict(visible=False),
                yaxis=dict(visible=False),
                zaxis=dict(visible=False)
            )
        ),
    )
```

Рисунок 3.15 - Створення і конфігурація графіку

```
fig_plotly.show(renderer="colab")
```

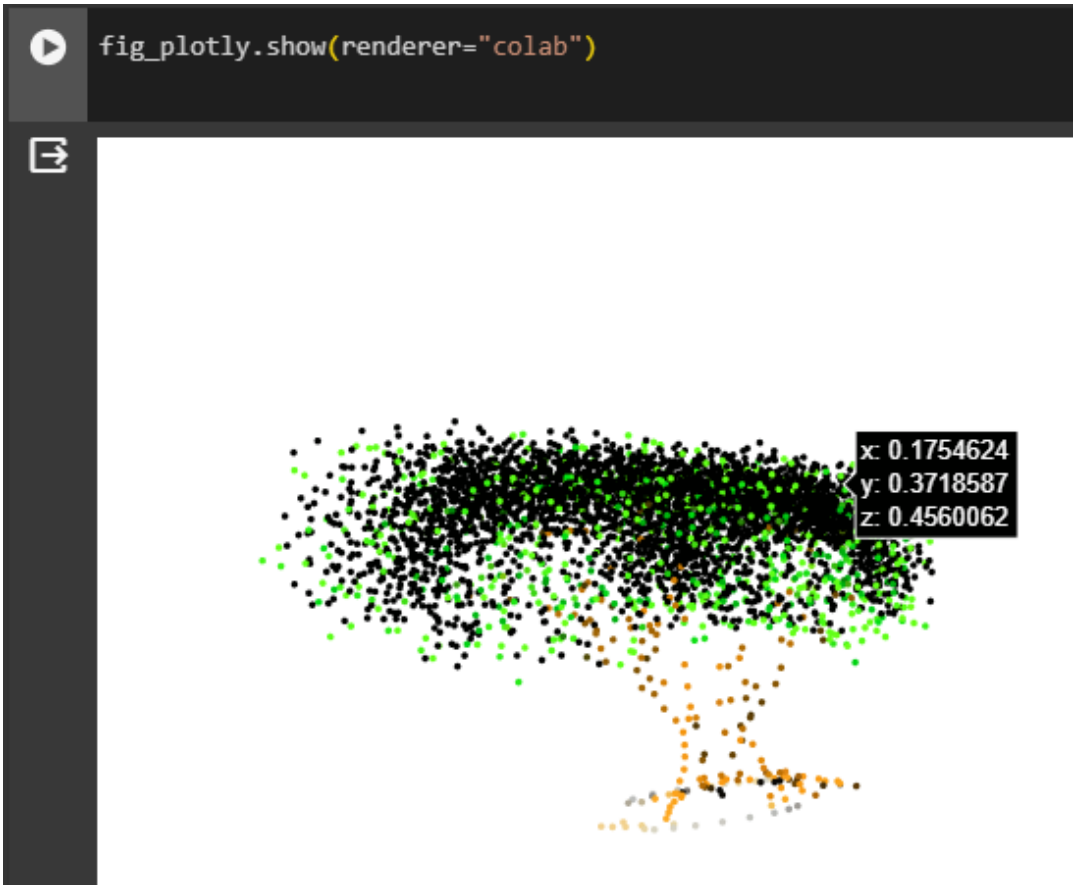

Рисунок 3.16 - Результат зі хмарових точок

Ми отримали 3D зображення, що було засноване на хмарі точок, що ми створили. Як можна помітити, результат виглядає не надто ідеально, але він може зайняти буквально кілька хвилин написання коду на мові Python і з сильним графічним процесором ми одразу отримуємо готову модель, що

можемо в подальшому використовувати в таких програмах як blender, unreal engine, cinema 4d і т.д.

3.3.3 Експорт

Для початку необхідно імпортувати помічну функцію з точки е. Тому ми з загостреної за допомогою pc в меш(сітка) імпортуємо меш куба, як сітку для підкреслення. Меш – це сукупність вершин, ребер та полігонів, які становлять один 3D об'єкт.

```
[17] from point_e.util.pc_to_mesh import marching_cubes_mesh
```

Рисунок 3.17- Створення мешу об’єкту

Далі ми маємо створити модель, яка визначена конфігурацією 'sdf'. Ця модель завантажується з використанням збережених контрольних точок, які були попередньо треновані.

```
[18] device = torch.device('cuda' if torch.cuda.is_available() else 'cpu')

     print('creating SDF model...')
     name = 'sdf'
     model = model_from_config(MODEL_CONFIGS[name], device)
     model.eval()

     print('loading SDF model...')
     model.load_state_dict(load_checkpoint(name, device))
```

Рисунок 3.18 -Створення та завантаження моделі SDF

Дуже важливим етапом є імпорт точкового простору зображення SDF в якості міри, що завадить виникненню помилки із залежностями.

```
[19] import skimage.measure as measure
```

Рисунок 3.19 - Імпорт точкового простору зображення SDF

Наступним кроком буде використання функції marching_cubes_mesh для створення мешу (триангульована поверхня) на основі точкової хмари pc та моделі SDF. Триангульована поверхня – це мережа трикутників, яка покриває дану поверхню частково чи повністю або процедура побудови точок і трикутників цієї мережі. Роздільність сітки визначається параметром grid_size, а розмір пакета та відображення прогресу також вказуються.

```
[21] # Produce a mesh (with vertex colors)
     mesh = marching_cubes_mesh(
         pc=pc,
         model=model,
         batch_size=4096,
         grid_size=32, # increase to 128 for resolution used in evals
         progress=True,
     )
```

Рисунок 3.20 -Використання алгоритму Marching Cubes для створення мешу

Для кінцевого експорту ми скористаємось службовою командою write_ply для створення сітки файлу, що ми будемо виводити в форматі ply, який зможемо в подальшому імпортувати в інше середовище.

```
# Write the mesh to a PLY file to import into some other program.
with open('chair.ply', 'wb') as f:
    mesh.write_ply(f)
```

Рисунок 3.21 Експорт у форматі ply

3.4 Імпорт в Blender зі створення запечених текстур

3.4.1 Імпорт та

Настав час імпортувати файл в робоче середовище blender. Для цього при імпорті необхідно обрати формат ply і можна починати працювати з ним.

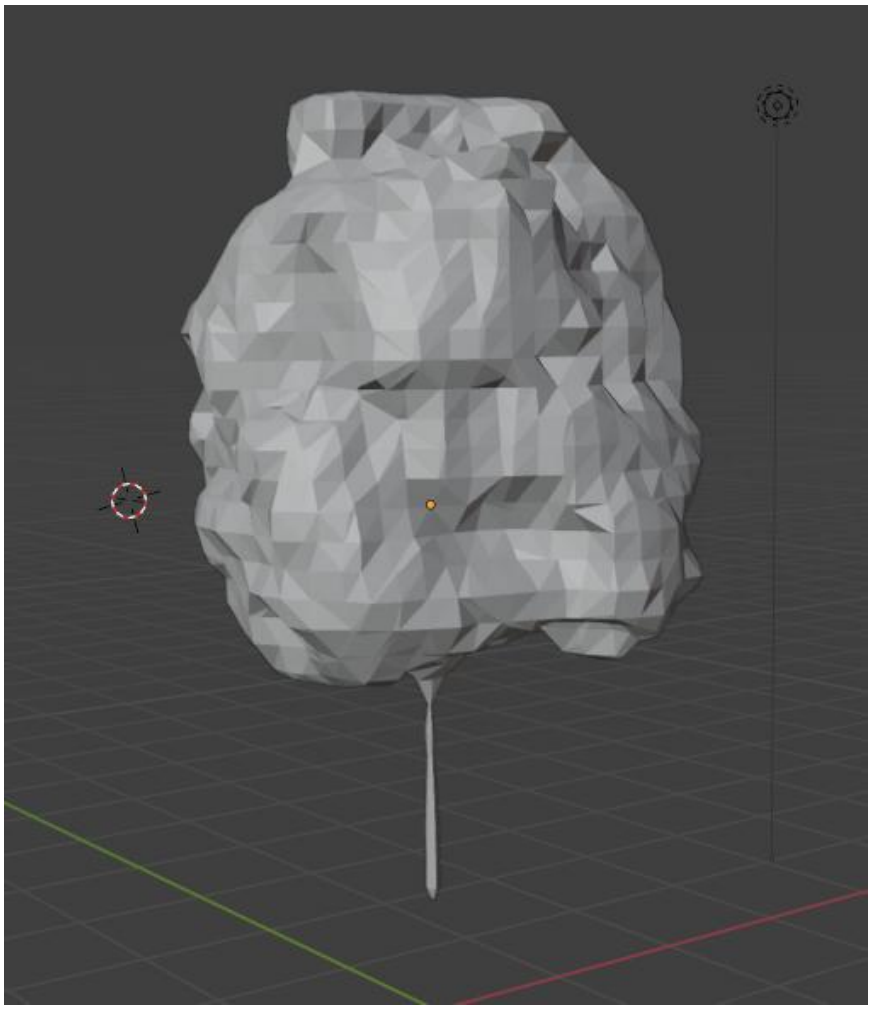

Рисунок 3.22 -Імпортований файл в середовище blender

Тепер ми можемо створити UV map для нашої сітки. Через Smart UV Project змінюємо Angle Limit на 50 градусів і Island Margin на 0,005.

Angle Limit визначає максимальний кут між суміжніми гранями, які можуть бути згруповані разом при автоматичній розгортці UV. Грані з меншим кутом вважаються суміжніми і розглядаються як частина одного UV-острова.

Цей параметр корисний для контролю розділення UV-островів на поверхні моделі. Ви можете налаштовувати його, щоб забезпечити розгортку,

яка відображає геометрію об'єкта з оптимальним використанням текстурного простору.

A Island Margin визначає простір (маржу), який буде залишено навколо кожного UV-острова при автоматичній розгортці. Це поле додає певну відстань між окремими UV-островами, щоб уникнути перекриття текстур або артефактів. Цей параметр дозволяє управляти проміжком між окремими UV-островами, забезпечуючи, що вони не будуть перетинатися в текстурному просторі.

Ці параметри корисні при розгортанні UV для текстурування 3D-моделей та гарантують ефективне використання текстур та мінімізацію артефактів на текстурах, що накладаються.

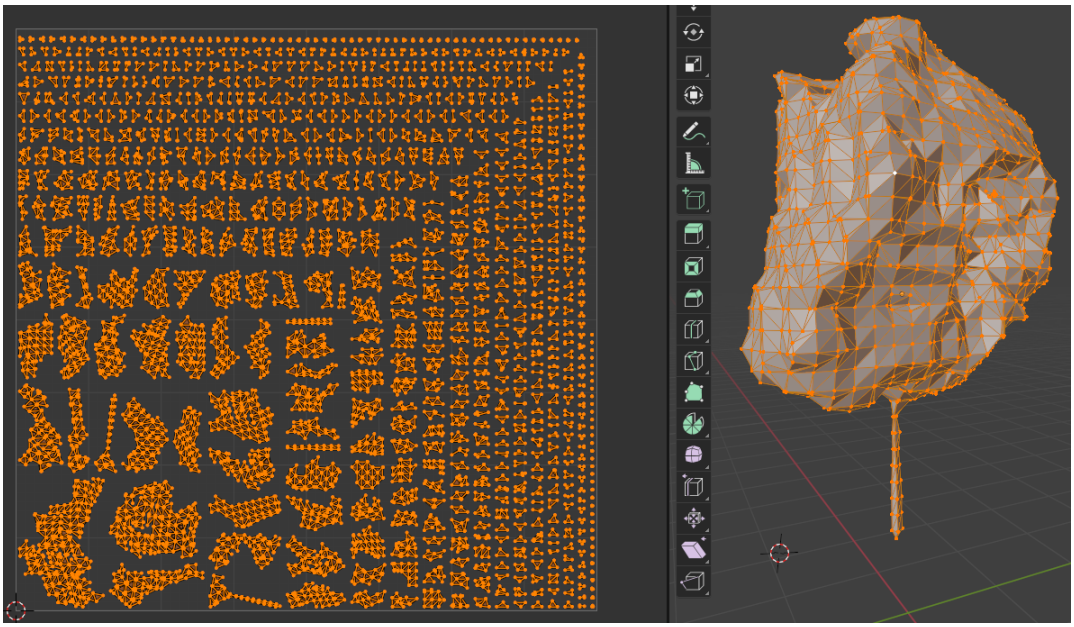

Рисунок 3.23- UV розгортка

3.4.2 Матеріал та атрибути кольорів

Тепер, коли у нас є UV-карта для нашої моделі, ми створимо матеріал і призначимо йому кольори вершин. Для цього додамо матеріал і в нодовому інтерфейсі прив’яжемо вузол кольорових атрибутів до базового кольору матеріалу. Після створюємо вузол зображення, але обов’язково вимикаємо

альфа канал. Таким чином ми визначили основний колір об’єкту і створили фонове зображення для UV розгортки.

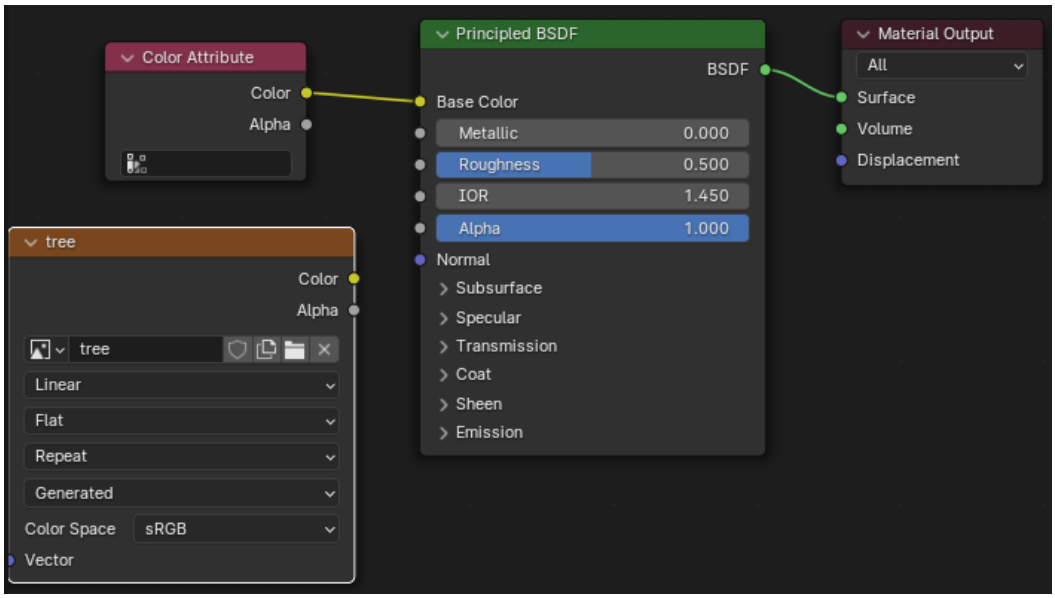

Рисунок 3.24 -Матеріал та атрибути кольорів

3.4.3 Запікання матеріалу

Настав час запекти матеріал. Для цього ми маємо встановити тип рендеру на Cycles. Цей рендер рушій чудово підійде для відтворення справжніх зображень. Через те, що він підтримує трасування променів, реалістичне освітлення з матеріалами та вузловий редактор шейдерів. Таким чином при запіканні ми отримаємо справжні кольори без втрат.

Далі ми маємо в налаштуваннях запікання встановити розсіяний тип запікання, прибрати в налаштуваннях впливу усе окрім кольору, що забезпечить запікання лише інформації про дифузний колір. Вже можна запікати результат.

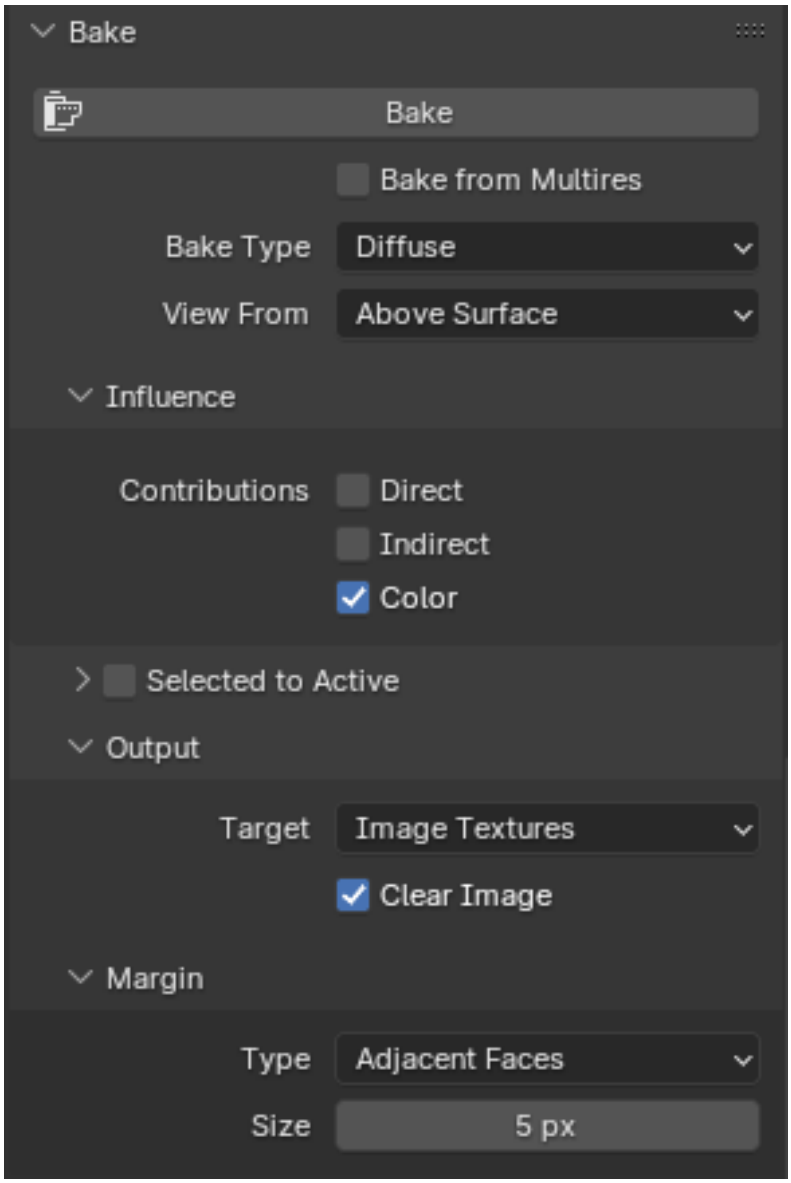

Рисунок 3.25 -Налаштування запікання

Після завершення процесу запікання ми можемо побачити UV-map з кольорами сітки в «Редакторі зображень». Цю запечену текстуру тепер можна використовувати в об’єкті FBX, дозволяючи легко імпортувати свою модель у Unreal Engine, чи будь-який інший 3D рушій із збереженням її оригінального вигляду.

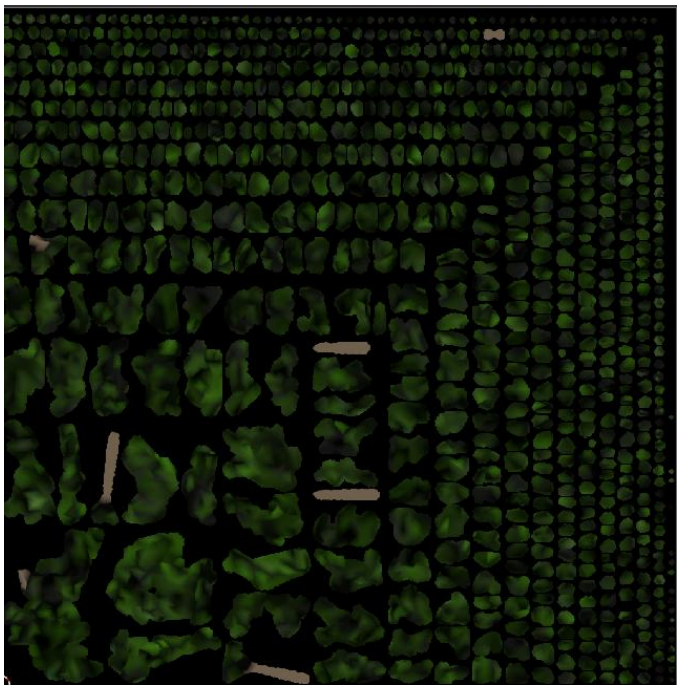

Рисунок 3.26 -UV-map після запікання

FBX (Filmbox) є форматом файлу для обміну даними між різними програмами для 3D-моделювання, анімації і візуалізації. Сам формат досить універсальний для передачі інформації між середовищами.

Рисунок 3.27 - Дерево після запікання

3.4.4 Експорт FBX об’єкту в інший рушій

Перед експортом необхідно повернутися до редактора шейдерів і від’єднайте вузол коліровий атрибут від принципового BSDF. Потім підключіть вузол текстури зображення до входу базовий колір вузла принциповий BSDF. Це забезпечить використання запеченої текстури під час імпорту моделі в Unreal Engine.

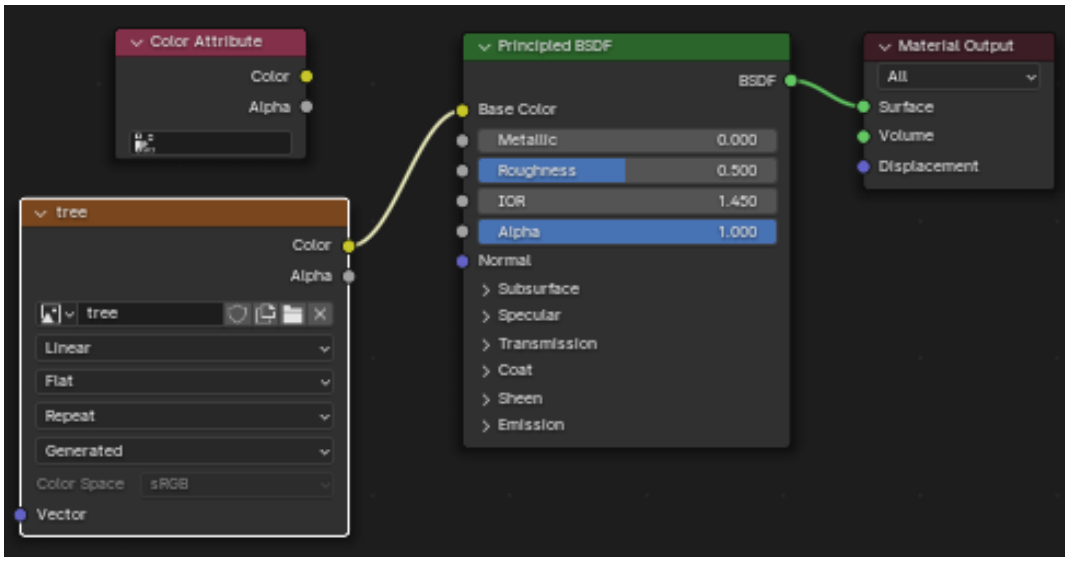

Рисунок 3.28 - Присвоєння запеченої текстури до матеріалу

Для правильного збереження об’єкту з текстурою необхідно експортувати її як FBX в режимі копії. Таким чином ми скопіюємо увесь каталог. Відтепер ми маємо змогу його використовувати у будь-якому середовищі просто імпортувавши, чи перетягнувши теку у браузер вмісту.

Рисунок 3.29 - Імпортована модель у Unreal Engine

3.4.5 Сцена із використанням згенерованого об’єкту і маски AI deapth

Unreal engine йде з часом і пропонує великий інструментарій навчений на штучному інтелекті. Для створення сцени було узято згенероване Stability AI зображення і маска глибини до нього. Для просторової сцени необхідно було створити і поєднати інтерфейс регулювання параметрами глибини, яскравості і туману до маски. Для цього ідеально підійшов інструмент blueprint, що дозволив нам побудувати скрипт на вузлах за кілька хвилин. Поєднавши із зображенням отримали матеріал, що деформує площину і створює ілюзію об’ємності.

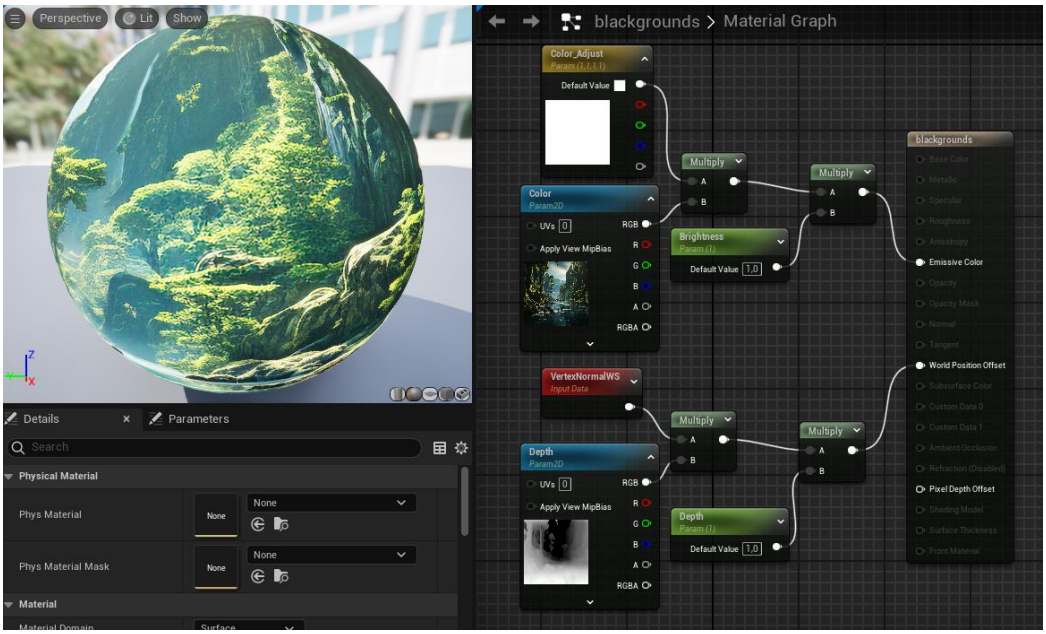

Рисунок 3.30 - Схема вузлів для маски AI Deapth

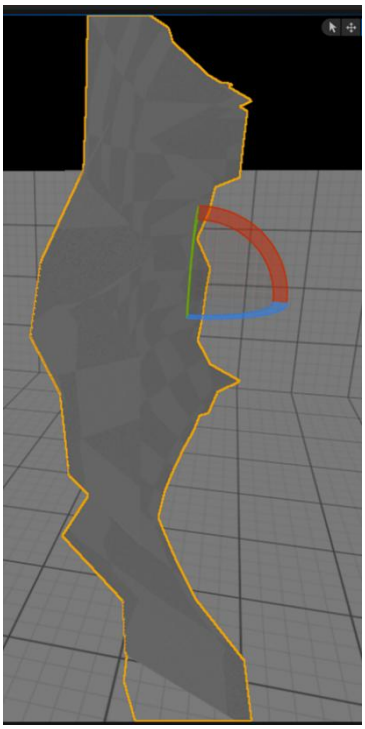

Рисунок 3.31 - AI деформація площини по масці

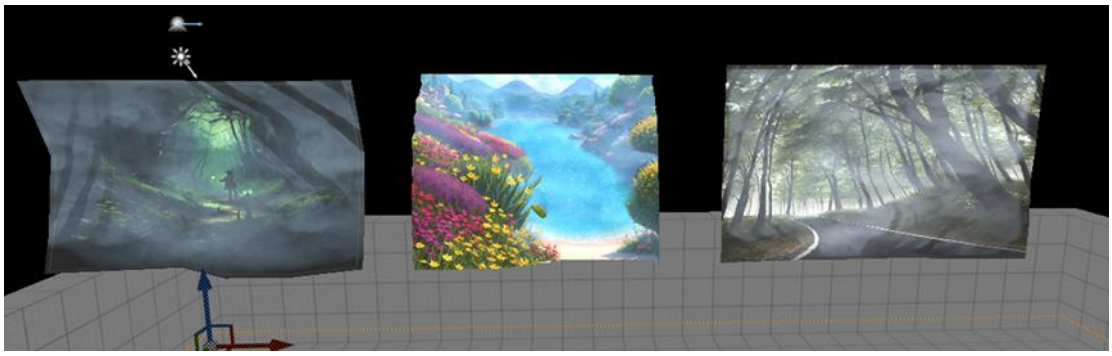

Рисунок 3.11 - Деформовані зображення спереду

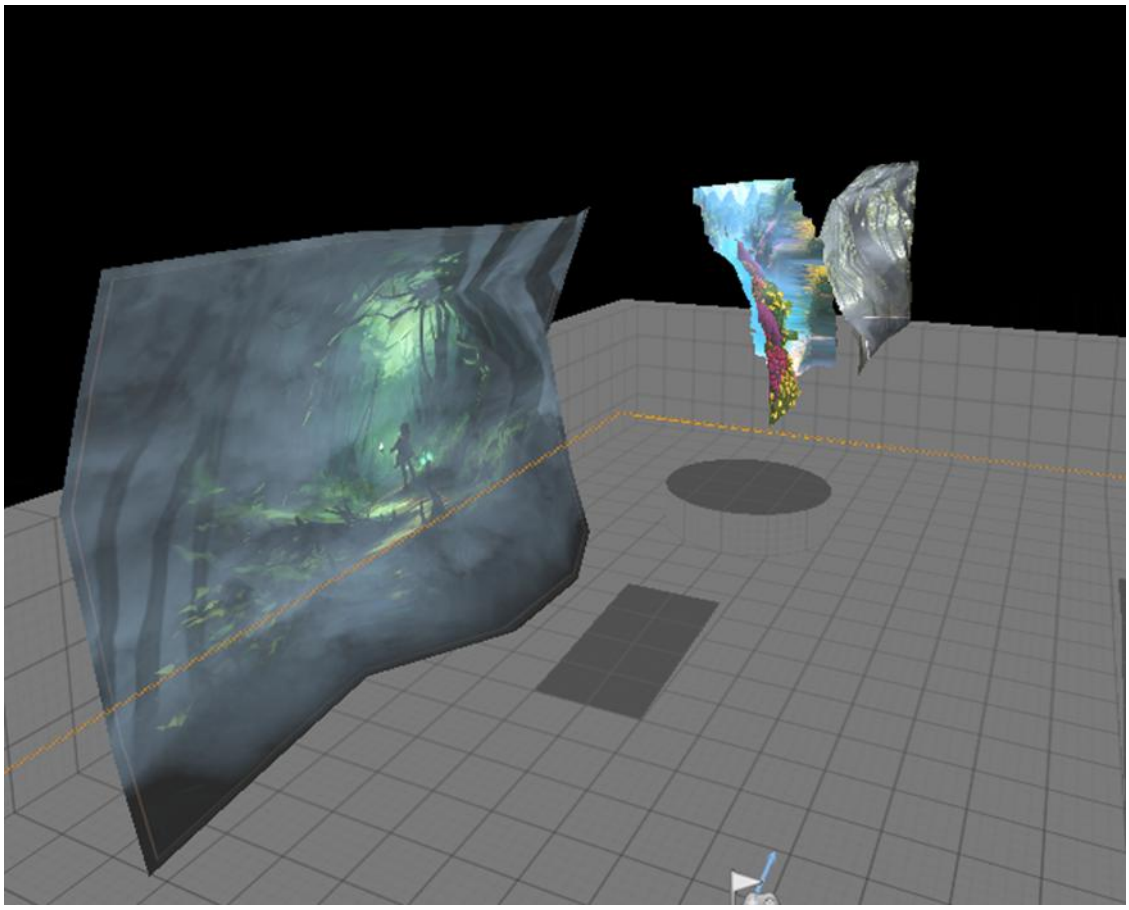

Рисунок 3.32 - Деформовані зображення збоку

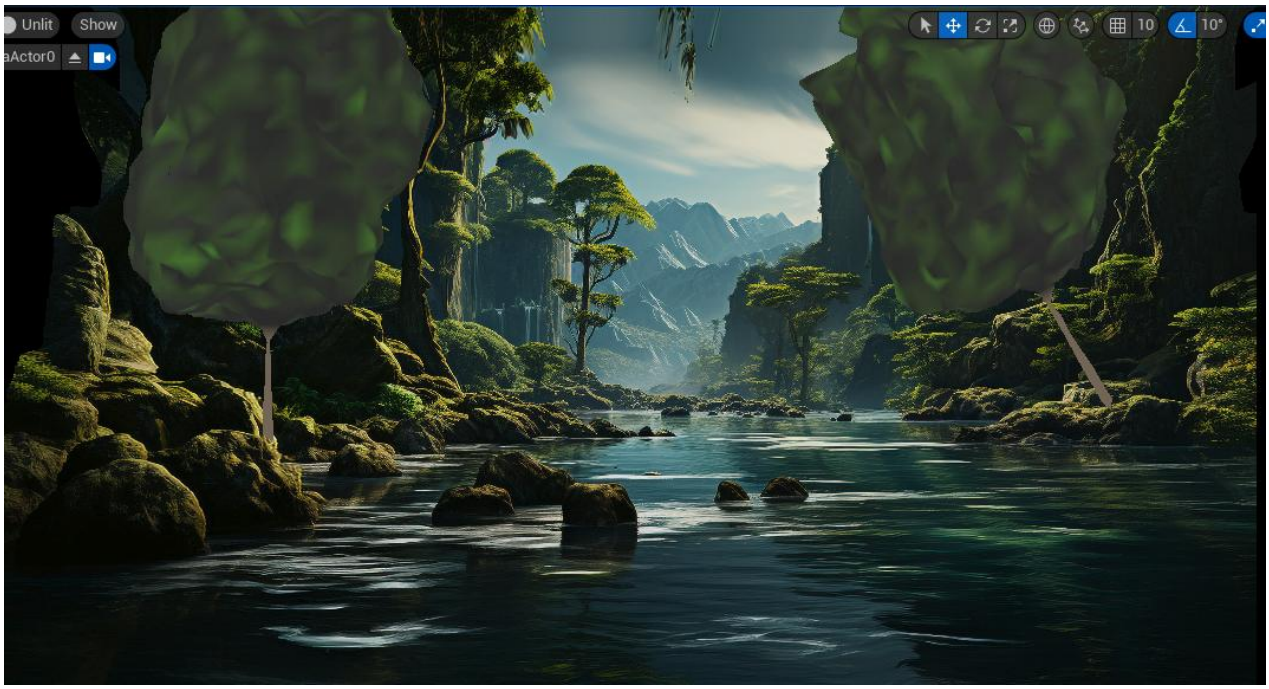

Рисунок 3.33 - Приклад використання згенерованого об’єкту

3.4.6 Аналог Point-E

Існує кілька доступних способів згенерувати 3D. Найефективнішим буде використовувати розширення на рушій і генерувати об’єкт на власному пристрої. Нажаль більшість з них платні, вимагають високих системних вимог, потребують надзвичайно багато часу на генерацію і мають багато проблем при встановленні. Наприклад плагін DreamCraft3D вимагає від користувача відеокартку лінійки NVIDIA з пам’яттю мінімум аж на 20 гігабайт. Коли середня у сучасних користувачів йде на 8. Оскільки перший спосіб нам не підходить, то перейдемо до другого.

Існують різні веб додатки, що дозволяють згенерувати потрібний об’єкт на віртуальній машині. Воно має свої переваги і недоліки. Головним недоліком буде обмеженість впливу на процес з ціллю удосконалення. Нам як правило надають інструментарій з мінімальною кількістю налаштувань, що дозволяє створити об’єкт, але не отримати бажаний результат. Серед плюсів звісно йде простота і відсутність до системних вимог.

Для прикладу я узяв сайт huggingface і закинув на нього наше зображення дерева, що було використане в роботі. Генерація тривала близько 20 хвилин, що мінімум у 20 разів більше за Point-E, але натомість ми отримали дерево непоганої якості у форматі glb. Особливістю цього пакету є представлення 3D-моделей у двійковому форматі файлів, збережених у форматі передачі GL (glTF), а також відсутність підтримки у більшості 3D рушіїв. Тому для використання його в програмах типу blender, чи Unreal Engine потребують додаткової конвертації через сторонній софт. На щастя Windows дозволяє відкрити його у власному 3D редакторі Paint.

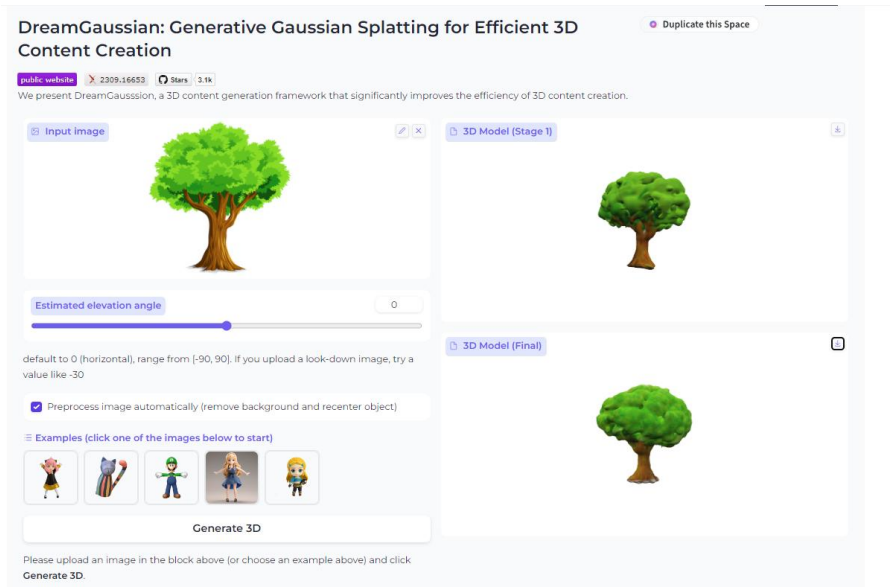

Рисунок 3.32 - Генерація 3D через huggingface.co

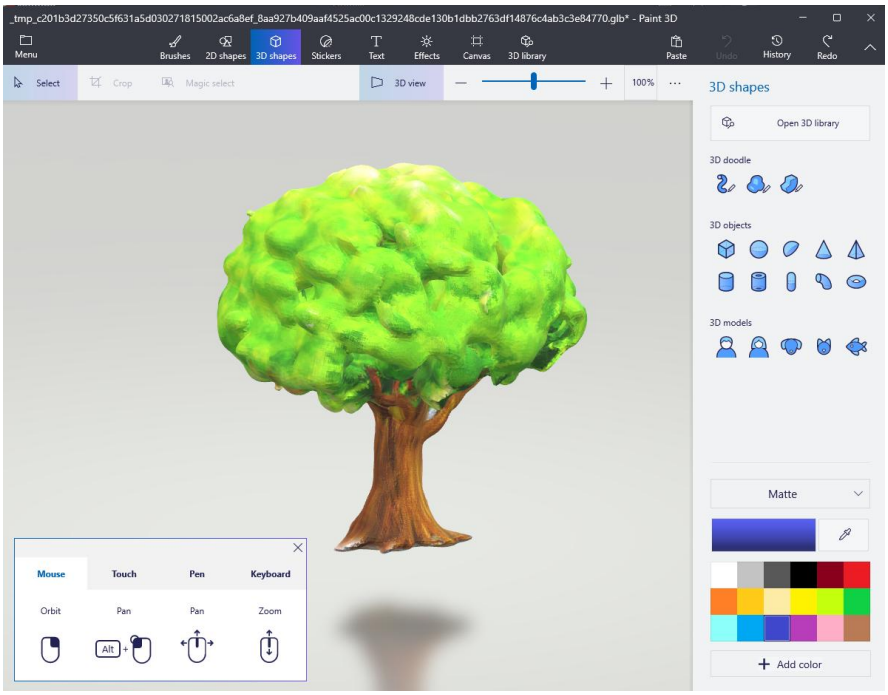

Рисунок 3.33 - Результат в Paint 3D

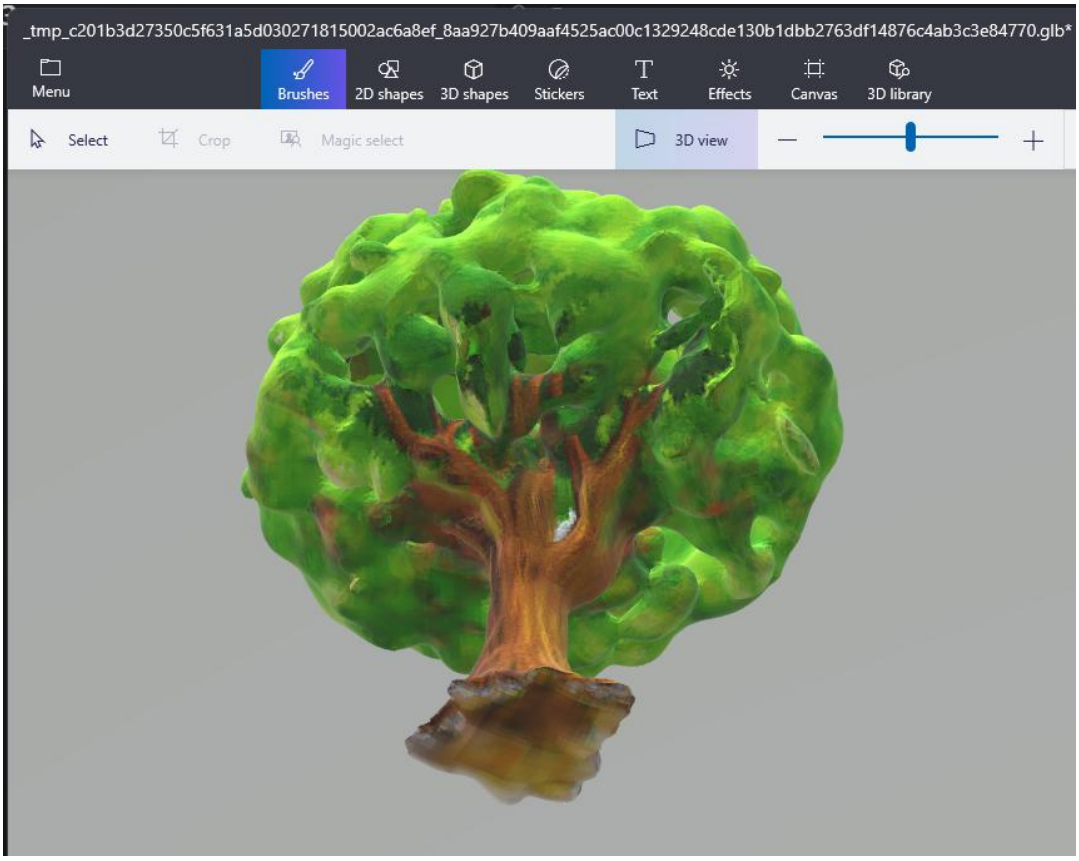

Рисунок 3.34 - Результат в Paint 3D під кутом

Висновки до розділу

Перше з чого ми почали – це розгортання проекту. Завантажили усі необхідні бібліотеки у середовище Google Colab. Далі проаналізували готові комірки коду і створили 3D модель. Це дозволило побачити недоліки і зрозуміти принцип дії.

Наступним кроком ми покращили результат і перейшли на модель кварків. Це дозволило працювати із зображеннями і робити на їх основі семли. Також вивели інформацію про пристрій на якому працюємо

Далі конвертували зображення в хмару точок, створили графік і візуалізували їх в його в середовищі. Після перетворили хмари точок в об’єкт надавши їм сітку і експортували в 3D рушій для подальшої обробки .

Кінцевим етапом створили і запікли текстури і у форматі FBX експортували в Unreal Engine.

Як приклад використання, була створена сцена з маскою AI Depth Mask, а також надано аналог. Можна побачити, що наш проект працює швидше більше ніж в 20 разів порівняні. Це доводить ефективність нашого методу, а його гнучкість в написані дозволить в подальшому вдосконалити і перевершити за якістю аналог.

4 СТАРТАП

4 Опис ідеї проекту

Доведення до автоматизації створення 60-80% низькополігональних 3д текстур для ігор, чи віртуальних кімнат метавсесвіту. Це дозволить значно прискорити створення проекту, що дасть змогу перевести значну часку зусиль на більш важливі та складні завдання.

4.2 Опис існуючих рішень та конкуренти

Існують різні типи програм, що генерувати 3D об’єкти з тексту, чи зображення, але усі вони мають спільні проблеми у вигляді високих ресурсно та часо витрат. Наш метод нехай і створює моделі поганої якості, але він на це витрачає усього хвилину і не потребує від вас високоефективної машини. Це значна домінація серед таких конкурентів як Stability Ai, Midjourney, Leonardo AI і т.д.

4.3 Розв’язок проблеми

Створенні 3д зображення чудово вирішують проблему із заповненням пустих локацій. Найбільш ефект помітно при дуже високій, чи малій швидкості картинці. Що здебільшого притаманно гоночним симуляторам, чи коротким відео вставкам. Також це чудовий ресурс для заповнення відкритих світів метавсесвіту з можливістю покращення результату.

4.4 Сфера використання продукту

3D об’экти можна продавати за криптовалюту в різні сфери діяльності. Через якість ціна буде занизька, але вона компенсується кількістю і часом. Чи можна просто надавати послуги із генерації великої кількості об’єктів на конкретний проект, чи тематику. Потенційні споживачі –ігрова індустрія та метавсесвіт.

4.5 Бізнес модель Канвас

Ключевими споживачами будуть дизайнери рівнів та локацій. Але окрім них можуть бути і звичайні користувачі соціальних мереж. Багато людей готово віддати певну суму, аби похизуватися дивним витвором ШІ.

Основною цінністю для споживача є безперечно швидкість створення і універсальність використання.

Каналами збиту для запропонованої технології в першу чергу буде співпраця з відомими ігровими та мета компаніями.

Зазвичай взаємодія з клієнтом буде проводитися через телеграм бота, а якщо буде необхідність у консультації, то буде доступним номер телефону з підв’язкою до Zoom/Skype/Discord . Головною зручністю для клієнта буде дизайн сайту компанії, що допоможе в навігації.

Збір ресурсів для майбутнього розвитку проекту в основному спрямовується на фінансування компаній, які тісно пов’язані з продуктом. На етапі реалізації проекту основні надходження коштів передбачається отримувати від продажу NFT.

4.6 Сильні та слабкі сторони продукту

Сильними сторонами продукту є його універсальність, швидкість, простота використання і низька енергозатратність в застосуванні.

Слабкими сторонами є якість результату, що змушує в подальшому самостійно виправляти недоліки. Так як результат не завжди виходить бажаним.

Хоча слабкі сторони і є досить значними, але вирішити проблему можна досить просто, якщо навчити штучний інтелект виправляти конкретну проблему, яку вказав користувач і збільшити кількість операцій на генерацію.

4.7 Прогнозування розвитку стартапу

Початковою датою для старту розробки метавсесвіту буде другий квартал 2023 року. За перший квартал 2023 року планується долучити до розробки спеціалістів по штучному інтелекту і просування своїх послуг через власний сайт і рекламні пости в соціальних мережах. За наступні 3 роки планується набрати базу клієнтів, що будуть слугувати додадковою помічю в просуванні продукту. Спочатку команда буде складатися з 5 людей, а саме керівника, C++ розробника, ще одного розробника фронтенд, одного маркетолога і одного дизайнера.

4.8 Ціна пропозиції

На етапі формування команди потрібно мати мінімальний резерв коштів на рік, і це оцінюється приблизно в 3 мільйони гривень, з яких майже вся сума йде на зарплатню. Починаючи розсилку реклами та використання платних методів реклами, додатково потрібно виділити ще 100 тисяч гривень.

Після отримання достатньої бази клієнтів затрати на зарплату будуть вирішуватися за рахунок прибутку компанії.

4.9 План реалізації стартапу

Основним призначенням нашого продукту є кардинальне скорочення витраченого часу і швидке заповнення метавсесвітів згенерованими об’єктами з готовими текстурами.

План :

-створення команди розробників

- автоматизація перших тестових проектів командою та реклама своїх послуг

- проведення заходів та кваліфікаційних іспитів направлених на ріст та прогресс в плані навичків написання коду

- співпраця з відомими компаніями з розробки AI.

- розширення команди спеціалістів, створення повноцінної компанії та переїзду в офіс

Висновки до розділу

У даному розділі розглядається створення стартап-проекту та ключові кроки, які слід вживати під час його створення. Висвітлено концепцію проекту, проведено аналіз конкурентоспроможності на ринку та оглянуто існуючі рішення для запуску стартапу. Розглянуті проблеми та шляхи їх вирішення. Після цього розглянута область застосування продукту, його основні користувачі та типи компаній, які можуть виявити інтерес до подібної послуги. Далі розглянуті всі ключові етапи по бізнес-моделі Канвас. Проведено аналіз переваг і недоліків стартапу та складений прогноз розвитку наступні три роки. Розрахована цінність року розробки продукту та створений план впровадження стартапу.

ВИСНОВКИ

Магістерська дисертація присвячена дослідженню використання технології Point-E. Розібрано проблематику створення 3D-графіки, після чого з’ясовано , що необхідно використовувати штучний інтелект для швидкого і більш точного виконання поставлених завдань, а також модульність для спрощення роботи. Розібрали технології, що автоматично застосовуються до 3D об’єкту в цілях оптимізації. Ми пролили світло на значну роль, яку AI відіграє в моделюванні. Досліджено можливості рушія Unreal Engine, Blender і сервісу Google Colab які надає багато переваг 3D розробникам, монтажерам, чи навіть художникам.

Було також розглянуто принцип навчання штучного інтелекту. Також було створено блок-схему взаємодії ШІ з 3д-графікою і обрано Google Colab для роботи із проектом.

Після засвоєння базової інформації обраних технологій , ми почали розробляти проект. Було вирішено робити 3D об’єкт в Google Colab на мові Python через зручність і безкоштовну віртуальну машину. Спочатку було розгорнуто проект , після чого ми розглянули компоненти і завантажили усі необхідні теки з бібліотеками. Розглянули вже готові комірки с кодом для створення тестового проекту і доповнили їх. Таким чином візуалізували вивід інформації та покращили результат на виході. Наступним кроком експортували об’єкт в blender та запікли текстури. Отриманий результат було перенесено на рушій Unreal Engine і створено сцену з використання проекту і маски AI Deapth. Сама сцена була реалізована через технологію вузлів Blueprints. Це дозволіло візуально побудувати схему, що можна буде використовувати і в інших проектах за кілька хвилин. Також було наведено порівняння з конкурентом і виявлено, що наш проект більше ніж в 20 разів ефективніший та з можливістю покращення в подальшій розробці. Дослідження у даній дипломній роботі сприяє спрощенню процесу 3D моделювання і значно підвищую сам процес, що

дозволяє зосередитися на більш актуальних проблемах і зменшити кількість необхідної робочої сили в процесі. Сама технологія значно прискорила процес моделювання, текстурингу та UV розгортки. Від люди треба лише зробити незначні виправлення і можна використовувати роботу в проектах.

В розділі присвяченому стартапу ми оглянули саму ідею стартапу, розглянули які проблеми розв’язує нам стартап, розглянули сфери використання продукту , обдумали слабкі сторони нашого проекту , та припустили варіанти їх вирішення. Створили бізнес модель Канвас та зробили прогноз розвитку нашого стартапу на найближчі 3 роки.

Перелік джерел посилань

1. The Best Gaming Engines You Should Consider for 2023 [Електронний ресурс]: https://www.incredibuild.com/blog/top-gaming-engines-you-should-consider

2. What are the advantages and disadvantages of using a 3D game engine? Are they safe? [Електронний ресурс]: https://www.quora.com/What-are-the-advantages-and-disadvantages-of-using-a-3D-game-engine-Are-they-safe

3. What Is Unreal Engine? [Електронний ресурс]: https://www.bairesdev.com/blog/what-is-unreal-engine/

4. Why does Unreal Engine have good graphics? [Електронний ресурс]: https://www.quora.com/Why-does-Unreal-Engine-have-good-graphics

5. A Gentle Introduction to Generative Adversarial Networks (GANs) [Електронний ресурс]: https://machinelearningmastery.com/what-are-generative-adversarial-networks-gans/

6. About Blender Introduction [Електронний ресурс]: https://docs.blender.org/manual/en/latest/getting_started/about/introduction.html

7. Introduction to Blueprints [Електронний ресурс]: https://docs.unrealengine.com/4.27/en-US/ProgrammingAndScripting/Blueprints/GettingStarted/

8. Point-E: A System for Generating 3D Point Clouds from Complex Prompts [Електронний ресурс]: https://arxiv.org/abs/2212.08751

9. GLIDE: Openair model for generating images by text [Електронний ресурс]: https://neurohive.io/en/state-of-the-art/glide-openair-model-for-generating-images-by-text/

10. Point·E:ASystemforGenerating3DPointCloudsfromComplexPrompts [Електронний ресурс]: https://arxiv.org/pdf/2212.08751.pdf

Додаток А

SUMMARY

3D graphics are ubiquitous in various fields: movies, posters, games, metaverse, construction, etc. The creation process itself takes too much time, which can lead to a delay in time to market, postponements to the next quarter, and, consequently, low profits. Therefore, widespread familiarization with the process can not only increase productivity but also solve the most pressing problems. And these are not the only advantages, it is a generation of new technologies that are actively used by the world's leading companies. And it is clear that for such tasks, it is necessary to use the technology that is best suited to fulfill the goal and can make the application development process interesting and simple, as well as significantly simplify the process.

In accordance with the criteria and plans for developing 3D applications, it was decided to focus on the Python programming language, which is one of the most popular languages in the browser and its language is directly integrated into the engine. To simplify development and use new approaches, I chose OpenAI's Point-E. It's a great technology that has a very low entry threshold, supports Google Colab notebooks, which allows a beginner to get started and create projects from the very beginning.

The first thing we started with was the creation, namely the deployment of the project environment. For development, we decided to use OpenAI's Point-E technology, which is extremely fast and allows us to get interesting results without significant energy consumption.

Next, we considered the possibilities of modifying the code. After that, we added the ability to make clearer results through the quark model. In the process, we explained each code snippet, library, or just an operation that was being performed. Then we added the ability to output the result in PLY format for further baking and working in other 3D environments. Comparing it to a competitor we found out that our project is 20 times faster and bendable enough for further modifications.