НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ ІМ. СІКОРСЬКОГО»

Факультет інформатики та обчислювальної техніки
(повне найменування інституту, факультету)

кафедра обчислювальної техніки
(повна назва кафедри)

«До захисту допущено»
Завідувач кафедри
_____ Стіренко С.Г.
(підпис)
“___”_____2022 р.

Дипломний проєкт

на здобуття ступеня бакалавра
за освітньо-професійною програмою «Комп'ютерні системи та мережі»
спеціальності 123 «Комп'ютерна інженерія»

на тему Інтерактивна система візуалізації 3д об'єктів
_____

Виконав: студент 4 курсу, групи ІВ-81
Денисюк Ярослав Вячеславович _____
(прізвище, ім’я, по батькові) (підпис)

Керівник доцент, к.т.н Ткаченко В. В. _____
(посада, вчене звання, науковий ступінь, прізвище та ініціали) (підпис)

Консультант основна частина доцент, к.т.н Ткаченко В. В. _____
(назва розділу) (вчені ступінь та звання, прізвище, ініціали) (підпис)

Рецензент _____ _____
(посада, вчене звання, науковий ступінь, прізвище та ініціали) (підпис)

Засвідчую, що у цьому дипломному проекті немає запозичень з праць інших авторів без відповідних посилань.
Студент _____
(підпис)

Київ - 2022 р.

НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ»
ФАКУЛЬТЕТ ІНФОРМАТИКИ ТА ОБЧИСЛЮВАЛЬНОЇ ТЕХНІКИ
КАФЕДРА ОБЧИСЛЮВАЛЬНОЇ ТЕХНІКИ

Освітньо-кваліфікаційний рівень спеціаліст
Напрям підготовки **123 – « Комп’ютерна інженерія»**
Спеціальність **« Комп'ютерні системи та мережі»**

ЗАТВЕРДЖУЮ
Завідувач кафедри

Сергій Стіренко
"___" _____ 20__ року

ЗАВДАННЯ
НА ДИПЛОМНУ РОБОТУ СТУДЕНТА

Денисюка Ярослава Вячеславовича
(прізвище, ім'я, по батькові)

1. Тема проекту (роботи) Інтерактивна система візуалізації 3д об'єктів
керівник проекту (роботи) доцент, к.т.н Ткаченко В. В.
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)
затверджені наказом вищого навчального закладу від "___" _____ 20__ року ___

2. Строк подання студентом проекту (роботи)

3. Вихідні дані до проекту (роботи) технічне завдання, пояснювальна записка, схема взаємодії з сервером, схема роботи програми, структурна схема бази даних

4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити)
Огляд існуючих рішень реалізації систем візуалізації 3D об’єктів, пошук технологій розробки веб-додатку та моделей та опис розробленого проекту та його виконання

5. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень)
Авторизація користувача блок схема схема роботи програми, структурна схема бази даних

6. Консультанти розділів проекту (роботи)

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
1	доцент, к.т.н Ткаченко В. В.		

7. Дата видачі завдання _____

КАЛЕНДАРНИЙ ПЛАН

N з/п	Назва етапів дипломного проекту (роботи)	Строк виконання етапів проекту (роботи)	Примітка
1	Затвердження теми роботи		
2	Вивчення та аналіз завдання		
3	Проектування програмного забезпечення		
4	Програмна реалізація продукту		
5	Оформлення пояснювальної записки		
6	Захист програмного продукту		
7	Передзахист		
8	Захист		

Студент _____ Денисюк Я.В.
(підпис) (прізвище та ініціали)

Керівник проекту (роботи) _____ Ткаченко В.В.
(підпис) (прізвище та ініціали)

Анотація

В даній дипломній роботі реалізовано інтерактивну систему візуалізації 3D об’єктів за допомогою якої користувачі зможуть створювати прості сцени з тривимірними моделями, додавати різне освітлення, ефекти пострендерингу.

Також користувач зможе завантажити власну модель на сцену, що зробить сервіс відмінним від інших, та дасть змогу користувачам дизайнерам, або ж користувачам розробникам тестувати свої моделі з різним освітленням, ефектами безпосередньо у браузері. Це є великим плюсом, оскільки часто використовуючи моделі, освітлення з таких програм, як Blender, то у браузері вони дещо відрізняються від того що було у редакторі.

Ключові слова: 3D, WebGL, Three.js, React, 3D моделювання, 3D-редактор.

Annotation

In this work an interactive system for visualization of 3D objects is implemented, through which users will be able to create simple scenes with three-dimensional models, add different lighting, postrendering effects.

Also, the user will be able to upload their own model to the scene, which will make the service different from others, and will allow users to designers or users to test their models with different lighting, effects directly in the browser. This is a big plus, because often using models lit from programs like Blender, they are somewhat different in the browser than they were in the editor.

Keywords: 3D, WebGL, Three.js, React, 3D modeling, 3D editor.

ВІДОМІСТЬ ДИПЛОМНОГО ПРОЄКТУ

№ з/п	Формат	Позначення	Найменування	Кількість листів	Примітка
1	А4		Завдання на дипломний проєкт		
2	А4	ІАЛЦ.467800.002 ТЗ	Технічне завдання	5	
3	А4	ІАЛЦ.467800.003 ПЗ	Пояснювальна записка	63	
4	А4	ІАЛЦ.467800.004 Д1	Авторизація користувача блок схема	1	
5	А4	ІАЛЦ.467800.005 Д2	Схема роботи програми	1	
6	А4	ІАЛЦ.467800.006 Д3	Структурна схема бази даних	1	
7					
8					
9					
10					
11					

					ІАЛЦ.467800.001 ВП	
Змн	Арк.	№ докум.	Підпис	Дата		
Розроб.		Денисюк Я.В			Система візуалізації 3д об'єктів Опис альбому	Літ. / Арк. 1 / Акрушів 1
Перевір.		Ткаченко В.В				НТУУ «КПІ імені Ігоря Сікорського», ФІОТ Група ІВ-81
Н. Контр.		Симоненко В.П				
Затверд.						

Технічне завдання

до дипломного проєкту

на тему: «Інтерактивна система візуалізації 3д об'єктів»

Київ – 2022 р

Технічне завдання
Зміст

1. НАЙМЕНУВАННЯ ТА ОБЛАСТЬ ЗАСТОСУВАННЯ

Найменування: “Інтерактивна система візуалізації 3д об'єктів” Дане технічне завдання поширюється для використання будь якими користувачами, для тестування своїх 3D об’єктів, або ж створення сцен з 3D об’єктами, зі списку.

2. ПІДСТАВИ ДЛЯ РОЗРОБКИ

Підставою для розробки є завдання на виконання бакалаврського дипломного проекту, затверджене кафедрою обчислювальної техніки Національного технічного університету України «Київський політехнічний інститут ім. Ігоря Сікорського»

3. МЕТА І ПРИЗНАЧЕННЯ РОЗРОБКИ

Метою даного проекту є розробка інтерактивної системи візуалізації 3д об'єктів

4. ДЖЕРЕЛА РОЗРОБКИ

Джерелом розробки є науково-технічна література та публікації в спеціалізовані публікації в мережі Інтернет по даній темі.

5. ТЕХНІЧНІ ВИМОГИ

5.1. Вимоги до розроблюваного продукту

- Реєструвати нових користувачів, заходити на сторінки до існуючих користувачів;
- Додавати та видозмінювати різного роду 3D об’єкти, освітлення, ефекти пострендерингу.

5.2. Вимоги до програмного забезпечення

- Мінімальний обсяг оперативної пам'яті - 1 ГБ;
- Мінімальна частота процесора – 1,1 ГГц;
- Наявність у пристрої підключення до Інтернету;
- Наявність встановленого на пристрої браузера;

5.3. Вимоги до апаратно частити

- Комп'ютер на базі процесора Intel Pentium та вище
- Оперативної пам'яті не менше 512 Мбайт
- Вільний простір жорсткого диску не менше 100 Мбайт

ЕТАПИ РОЗРОБКИ

	Дата
Вивчення літератури	19.02.2022
Складання та узгодження технічного завдання	06.03.2022
Проектування програмного забезпечення	19.03.2022
Програмна реалізація продукту	09.04.2022
Тестування програмного забезпечення	07.05.2022
Налагодження та виправлення помилок	10.05.2022
Оформлення документації дипломної роботи	16.05.2022

Пояснювальна записка
до дипломного проєкту

на тему: «Інтерактивна система візуалізації 3д об'єктів»

Київ – 2022 р

ЗМІСТ

Вступ

Кожен із нас під час роботи за комп'ютером, ноутбуком, переглядом інформації на веб ресурсах, розвагами у комп’ютерних іграх стикається з тривимірною графікою. Зазвичай ми не звертаємо на це уваги оскільки різного роду елементи оформлення, анімовані зображення чи 3D-моделі – стали невід’ємними частинами реклами, інтерфейсів, додатків, веб сторінок. Перш ніж створити щось на кшталт будинку або дизайну інтер'єру, необхідно створити проект. Раніше це було важким випробуванням і кропіткою справою, сьогодні ж завдяки 3D-графіці це роблять швидше та якісніше. 3D моделювання прийшло у сучасне життя давно і вже активно застосовується у рекламі, архітектурі, індустрії. Після того як з’явився 3D-друк перед користувачами відкрились нові горизонти і воно стало ще популярнішим, тривимірне моделювання ніби перейшло на новий рівень. Сьогодні за допомогою тривимірної графіки створюють точні копії об'єктів, що згодом втілюються у життя нереальними дизайнерськими задумами. 3D технології приносять користь не тільки людству та великий прибуток знавцям цієї сфери, після того як 3D-друк почали використовувати для протезування людей, багато друзів наших менших отримали шанс на нове повноцінне життя, оскільки почали створювати протези і для них.

3D давно стало важливою частиною нашого світу, тому немає нічого дивного, що можливості цієї технології перевершують будь-які інші способи донесення інформації, як от аудіо, текст чи 2D. За допомогою цієї нової технології навчання стало в рази цікавішим, сприйняття і донесення інформації стало зовсім іншим. Взяти до прикладу ту ж анатомію людини, раніше люди могли собі тільки уявити як виглядає той чи інший орган. Все чим керувалися це двомірні зображення, які не давали повного усвідомлення яким саме є цей орган. Зараз же, за допомогою 3D ми можемо розглядати їх під різними кутами, розуміти яким є їх об’єм. А надрукувавши їх на 3D принтері ще й доторкнутися до них. Це просто не вкладається в голові. Та й більшість 3D об’єктів є цілими

комплексами з інших тривимірних об’єктів. А якщо ще застосувати 3D анімації до об’єктів це дозволяє ще краще показати взаємодію між ними в комплексі. Дозволяє створювати будь-що. У фільмах вже давно використовуються такого роду технології і їх поєднання, оскільки це значно спрощує створення складних сцен, відтворення яких у реальному житті обійшлося б в десятки тисяч доларів, а також десятки якщо не сотні годин роботи персоналу. Також можна згадати такі новітні технології як віртуальна реальність, надалі - VR(virtual reality) та доповнена реальність надалі - AR(augmented reality), які набувають надзвичайної популярності і використовують 3D, для візуалізації даних. Важко знайти сферу діяльності в якій хоча б частково, але не використовувалась 3D візуалізація.

З цього можна зробити висновок що дана технологія є дуже популярною і востребуваною у наш час. А оскільки основними джерелами які задовільняють інформаційні потреби людей зараз це веб ресурси, то більшість браузерів почали вводити новий стандарт WebGL. Який дозволяє розробникам без застосування додаткових плагінів вбудовувати в вебоглядачі повноцінну 3D графіку.

Актуальність теми

Популярність та актуальність 3D у наш час переоцінити неможливо, оскільки це інструмент, що дає змогу найкраще і найбільш наочно зобразити будь який об’єкт. А у поєднанні з веб ресурсами робить цю технологію ще й в рази доступнішою. Тому зараз з’являється велика кількість сервісів які надають послуги візуалізації 3D об’єктів. Не зважаючи на те що поєднання цих технологій є відносно новим, та не має великого різноманіття.

Мета і задачі дослідження

Метою є розробка сайту, який надавав би змогу візуалізувати свої 3D об’єкти, 3D сцени та перевірити як вони відображаються у браузері, редагувати недоліки якщо такі є, додати готові анімації або ж шейдери(візуальні ефекти). Переглянувши «ринок» я не знайшов сервісів які б задовільняли ці потреби

сповна, мали простий і зрозумілий дизайн та були б безкоштовними. В результаті повинні отримати вікно розподілене на 2 частини. Одна із них це поле візуалізації/редагування 3D об’єктів, а інша це панель на якій відображатиметься інформації про певний об’єкт, або ж сцену до якої можна буде застосувати візуальні ефекти.

РОЗДІЛ 1.
ОГЛЯД ІСНУЮЧИХ РІШЕНЬ РЕАЛІЗАЦІЇ СИСТЕМ ВІЗУАЛІЗАЦІЇ 3D ОБ'ЄКТІВ

1.1 Поява, перші рішення, розвиток 3D.

1.1.1 3D від перших згадок до сьогодення

Історія 3D-моделювання пройшла через безліч етапів розвитку на шляху до того, щоб стати частиною нашого повсякденного життя. Ці розробки варіюються від туманних теорій до покращення користувальницького досвіду у програмному забезпеченні, але кожна з них внесла значний внесок у розвиток технології, який ми бачимо її сьогодні. Знайомство з ними дозволить краще зрозуміти цінність 3D-моделювання, чого воно вже досягло і що чекає на нього в майбутньому. Адже легко зрозуміти, чому 3D-моделювання та рендеринг сьогодні захоплюють світ. Його вплив величезний - можливо, ви просто не знаєте, що 3D-моделювання формує ваше професійне та повсякденне життя. Отже, давайте розглянемо моменти часу, які започаткували цілу індустрію та покращили наші робочі процеси та рекламу.

Хоч як парадоксально, але історія 3D-моделювання почалася задовго до появи першого персонального комп'ютера. Все почалося з математичних ідей, які лежать в основі 3D візуалізації. Насправді деякі з основних ідей прийшли від Евкліда, якого іноді називають "засновником геометрії", що жив у 3 столітті до нашої ери. Потім Рене Декарт у 1600-х роках подарував світові аналітичну геометрію, також відому як координатну геометрію, яка дозволяла точно відслідковувати відстані та місцезнаходження. Пізніше, в середині 18 століття, англійський математик Джеймс Джозеф Сільвестр винайшов матричну математику, яка сьогодні використовується у кожному комп'ютерному зображенні, де можна побачити відображення чи спотворення світла.

У 1950-х роках було розроблено комп'ютери, які використовувалися в багатьох математичних цілях - в основному у військових та наукових, але не в багатьох інших. На щастя, це призвело до того дня в історії, коли комусь спало на думку застосувати комп'ютер для реалістичного моделювання продуктів і конструкцій.

Перші досягнення в історії 3D-моделювання з'явилися, коли в 1960-х роках почали з'являтися перші доступні комерційно системи автоматизованого проектування (CAD) або Computer Aided Design. Найбільший прорив стався завдяки Івану Сазерленду, який у 1963 році представив програму Sketchpad, також відому як "Робот-кресляр", з її революційним інтерфейсом. Sketchpad показав, що комп'ютери можуть використовуватися не тільки для інженерних або повторюваних креслень, але і в інтерактивному режимі дизайнерами та потенційно художниками. У тому ж році партнерство General Motors та IBM створило DAC-1, Design Augmented by Computer, який був публічно представлений у 1964 році та використовувався General Motors до кінця десятиліття для прискорення робочого процесу виробництва автомобілів. Вона продемонструвала, що комп'ютерна візуалізація дизайну може скоротити обсяг роботи, яка при використанні креслярських дощок зайняла б цілу вічність. Наприкінці десятиліття, в 1968 році, Іван Сазерленд і Девід Еванс заснували першу компанію з 3D-графіки "Evans & Sutherland". Вони розпочали компанію з виробництва апаратного забезпечення для роботи систем, що розробляються, але незабаром зайнялися і розробкою програмного забезпечення. Їхня поява та успіх на ринку надихнули інших на створення власних компаній та роботу над розвитком технології.

Просте 3D-моделювання чайника У цей момент в історії 3D-моделювання нові компанії почали пропонувати автоматизовані системи проектування та креслення. ADAM, система автоматизованого проектування, випущена 1971 року, була однією з таких систем. ADAM була розроблена для роботи на максимально можливій кількості машин, що викликало величезний сплеск

доступності САПР, який тривав у міру того, як комп'ютери та програмне забезпечення ставали дедалі складнішими. У той час як такі компанії, як MAGI, що представили твердотільне 3D-моделювання, створювали новий попит на САПР, університети старанно працювали над розвитком історії 3D-моделювання та відкриттям нових та ефективніших технологій візуалізації 3D-моделей. Гуро і Фонг відкрили в Університеті Юти техніку затінювання, яка прискорила обробку даних за рахунок спрощення початкових алгоритмів рендерингу та забезпечила найкращі візуальні результати в галузі світла, відображення та затінення. Інтерес представляє модель чайника з Юти. Вона увійшла в історію як символ тривимірної комп'ютерної графіки після того, як її використав Мартін Ньюелл для тестування своїх графічних досліджень. Він знайшов 3D-модель чайника ідеальної для тестування завдяки її структурі, різноманітності поверхонь та здатності відкидати тіні на себе. Він поділився подробицями зі своїми колегами-дослідниками, які відразу почали використовувати чайник.

Поява першого IBM PC 1981 року призвело до широкого використання САПР у аерокосмічної та автомобільної промисловості, а й у комерційних інженерних підприємствах. Цьому також сприяла поява робочих станцій UNIX, які були дешевшими, високопродуктивними і вимагали менше обслуговування. Твердотельне 3D-моделювання потім розвивалося і стало основним напрямком із розробкою такого програмного забезпечення, як UniSolids CAD від Unigraphics. Пізніше, 1983 року, була випущена 2D-система AutoCAD. Це була перша в історії значна програма CAD для тривимірного моделювання для IBM PC, оскільки пропонувала майже таку ж функціональність, як і інші CAD-програми на сьогоднішній день, але за 20% від їх вартості. Але вони недовго домінували на ринку, оскільки комерційні CAD-системи з'являлися в аерокосмічній, автомобільній та інших галузях промисловості протягом усього десятиліття. Цим галузям також допомогло впровадження IGES, нейтрального

до виробників формату файлів, що дозволяє користувачам передавати 3D проекти між різними CAD-програмами.

Окремою частиною історії 3D-моделювання є винахід SLA або стереолітографії – пошарового виготовлення деталей. Цей метод був винайдений у 1984 році, але перша машина, яка його використала, з'явилася у 1992 році. У цій SLA-машині використовувався матеріал на основі акрилу, який називається фотополімером. На нього впливають ультрафіолетовим лазером, щоб перетворити його на твердий пластик. У тому ж році з'явилася ще одна машина, в якій замість рідини використовувався порошок, що отримав назву SLS – селективне спікання. На той момент в історії 3D-моделювання все це було дуже нове, і компанії тільки усвідомлювали потенціал такої технології. Після цього з'явилося безліч прототипів та розробок, особливо в галузі медицини, де стартапи виготовляли органи та протези за допомогою технології 3D-друку.[2]

Великий прорив стався в 1999 році, коли людині було імплантовано перший 3D-друкарський орган - синтетичний сечовий міхур. З розвитком Інтернету відкривалися нові можливості. У 2005 році проект 3D-друку з відкритим вихідним кодом RepRap був профінансований прихильниками в Інтернеті. Він був успішно розроблений та випущений у 2008 році. Названий "Darwin", 3D-принтер був здатний друкувати більшість власних деталей за допомогою різних ниток, які можна було легко замовити. Це викликало великий ажіотаж і ентузіазм щодо 3D-друку, оскільки люди зрозуміли, що все, що вони можуть придумати, можна надрукувати прямо в себе вдома, навіть принтер. Після цього величезного сплеску інтересу Kickstarter профінансував безліч проектів 3D-друку. Ці проекти, що фінансуються державою, призвели до гнучкості та доступності, які сьогодні має 3D-друк. Існує безліч дивовижних історій про те, що 3D-друк зробила і може зробити сьогодні. Від виробництва житла та дрібної точної електроніки до органів та протезів. Але тут неможливо охопити все, не кажучи вже про те, що розвиток 3D-моделювання

продовжуватиме прогресувати. Вся історія 3D-моделювання почалася з того, що багато математиків точно описали процес створення та відстеження геометрії. Це виявилося дуже необхідним у комп'ютерах, які, після деяких досліджень, змогли створити неймовірні візуалізації. Тепер, з розвитком цієї технології, 3D-моделювання та візуалізація стали широко доступними та недорогими, як ніколи. На фоні цього всього з’явилась потреба у ще простішому способі створення та візуалізації 3D об’єктів. Тому з’явилась така технологія як WebGL для браузерів.

1.2 Базова теорія 3D

Перш за все хотілося б згадати, що таке 3D — по суті, це представлення об’єктів у тривимірному просторі з системою координат для представлення положення у цьому просторі. Найпоширенішою системою є система координат з віссю x, y та z. Де x напрямлений вправо, у вгору, а z до користувача.[3]

Різні типи об'єктів будуються з вершин. Вершина — це точка у просторі, має власне тривимірне становище у системі координат і зазвичай деяку додаткову інформацію, що її визначає. Кожна вершина описується цими атрибутами:

- Позиція: вказує місце об’єкта у тривимірному просторі за допомогою трьох координат, наприклад x, y, z.
- Колір: може бути представлений у різних форматах, проте найчастіше використовується RGBA, де перші три букви вказують на основні кольори (червоний, зелений, синій) з яких утворюються інші як це вказано на
- рис 1.1, а A це альфа, яка вказує на відсоток прозорості кольору, лежить в межах від 0 до 1.
- Нормаль: спосіб опису напрямку вершини.
- Текстура: 2D-зображення, яке вершини використовують для заповнення поверхні, частиною якої вона є.

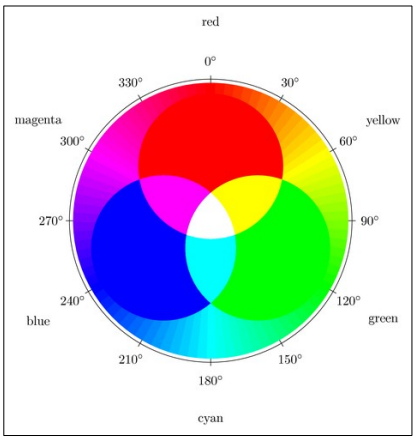


Рис. 1.1 – Палітра поєднання кольорів формату RGB.

У будь-яких редакторах можна створити якусь фігуру до прикладу куб. Розглянемо його далі він має 8 різних вершин (крапок у просторі) та 6 різних граней(грань - площина між вершинами), кожна з яких складається з 4 вершин. Нормаль визначає, в яку сторону спрямована грань. Крім того, з'єднуючи вершини, ми створюємо ребра куба. Геометрія будується з вершини та грані, а матеріал – це текстура, що використовує колір або зображення. Якщо ми поєднаємо геометрію з матеріалом, ми отримаємо сітку(mesh).

Осоновою відображення на екрані монітору є конвеєр рендерингу(Rendering pipeline) – це процес, за допомогою якого зображення готуються та виводяться на екран. Він бере 3D-об'єкти, побудовані з примітивів(трикутник, лінія, точка), описаних за допомогою вершин, застосовує обробку, обчислює фрагменти та відображає їх на 2D-екрані у вигляді пікселів(точка на екрані, що відображає колір за стандартами RGBA в залежності від положення на сітці екрану).

Першим шаром конвеєру є обробка вершин де інформація про окремі вершини поєднується для створення примітивів у встановленні їх положення у тривимірному просторі. Ця обробка складається з чотирьох етапів: перший включає розміщення об'єктів у світі і називається перетворенням моделі. Потім є перетворення виду, яке дбає про позиціонування та налаштування орієнтації камери в 3D-просторі. У камери є три параметри - місце розташування, напрямок та орієнтація - які мають бути визначені для новоствореної сцени.

Перетворення перспективи потім визначає налаштування камери. Налаштовується те, що може бачити камера - конфігурація включає видиме поле, співвідношення сторін і визначення ближніх і дальниіх площин. Останнім кроком є перетворення області перегляду, яке включає висновок всього для наступного кроку в конвеєрі рендеринга. Наступний етап це растеризація який перетворює примітиви в набір фрагментів. Після нього йде обробка фрагментів, яка фокусується на текстурах та освітленні - вона обчислює остаточні кольори на основі заданих параметрів.

Текстури - зображення, що використовуються у 3D-просторі, щоб об'єкти виглядали краще та реалістичніше. Текстури поєднуються з окремих елементів текстури, які називаються текселями, так само, як елементи зображення поєднуються з пікселів. Застосування текстур до об'єктів на етапі обробки фрагментів конвеєра рендерингу дозволяє коригувати його, обертаючи і фільтруючи при необхідності. Обтікання текстурами дозволяє повторювати 2D-зображення навколо 3D-об'єкта. Фільтрування текстур застосовується, коли вихідна роздільна здатність або зображення текстури відрізняються від фрагмента, що відображається — воно буде зменшено або збільшено відповідно.

На останньому етапі обробки виведення всі фрагменти примітивів з 3D-простору перетворюються на 2D-сітку пікселів, які потім виводяться на екран.
При об'єднанні виводу також застосовується деяка обробка для ігнорування непотрібної інформації - наприклад, не обчислюються параметри об'єктів, що знаходяться за межами екрана або за іншими об'єктами і, отже, не видно.

1.3 Розгляд 3D-технологій, стандартів

1.3.1 Direct3D

Direct3D - це пропрієтарний фреймворк інтерфейсу прикладного програмування (API) Microsoft. Він дозволяє розробникам створювати, змінювати та керувати тривимірними об'єктами, функціями та службами у програмах на базі Windows.[4]

Microsoft випустила першу версію Direct3D в 1996 частиною DirectX 2.0. Windows 95 OSR2 була першою операційною системою, що включала технологію Direct3D. З тих пір він продовжує використовуватися, а також використовується у кожному поколінні їх ігрових консолей Xbox. Починаючи з 8 версії, він також відображає двовимірну графіку, яка раніше виконувалася тільки за допомогою DirectDraw.

Direct3D надає розширені послуги рендерингу та прискорення 3D-графіки та дозволяє розробникам отримати доступ до розширених графічних функцій та можливостей, особливо для додатків вищого рівня, таких як ігри, фільми та анімація. Він полегшує доступ до розширених апаратних функцій, таких як альфа-змішування, буферизація, зіставлення та спеціальні візуальні ефекти. Direct3D звертається до всього графічного обладнання, включаючи відеокарти, графічні карти, базові процесори, оперативну пам'ять та пристрої виведення для відображення 3D-об'єктів та зображень. Він також взаємодіє з іншими серіями API DirectX та підтримує 2D-графіку, проте найчастіше використовується для роботи із 3D об’єктами.[5]

Даний фреймворк працює по принципу конвеєра, де на кожному етапі дані можуть видозмінятись, оброблятись. Перший шар займається тим що зчитує вершини з пам'яті використовуючи фіксованих операцій функції, формує геометрію та створює конвеєр робочих елементів. Автоматично генерує ідентифікатори для обробки: VertexID та InstanceID доступні для VS(Vertex Shader або ж вершинний шейдер) і надалі в конвеєрі. PrimitivelD доступні для HS(Hull Shader або ж шейдер Халла) і надалі. Ідентифікатор контрольної точки доступний тільки для HS. Далі на рівні VS обробляє вершини, виконуючи такі операції, як перетворення, скінінг та освітлення. Вершинні шейдери завжди працюють з однією вхідною вершиною та виробляють одну вихідну вершину. Далі рівень HS що має дві фази перша - фаза контрольної точки: викликається для кожної вихідної контрольної точки, кожна з яких ідентифікується ідентифікатором OutputControlPointID, з можливість зчитування всіх вхідних

контрольних точок для патча. Передає кожну контрольну точку далі коли відбувається виклик. Агрегований результат передається далі до наступної фази HS або DS(Domain shader або ж шейдера домену).

Наступна фаза патч констант: вона викликається один раз на патч, з читанням всіх вхідних та вихідних контрольних точок та констант патча, обчислених на даний момент. Вихідні дані – граничні коефіцієнти тесселяції та інші дані патчу. Код сегментований таким чином, що незалежна робота може бути розпаралелена. Далі йде TS(Tessellator) приймає числа з HS, що визначають, скільки потрібно розбивати. Генерує розташування доменів (isoLine, tri або quad) та топологію (точки, лінії або трикутники) та передає далі на DS(Domain Shader або ж шейдер доменів). Шейдер домену викликається один раз для кожної вершини, що згенерувала теселятором. Кожен виклик ідентифікується координатою на спільному домені, і роль Domain Shader полягає в тому, щоб перетворити цю координату на щось зрозуміліше (наприклад, точку в тривимірному просторі) для подальшого використання. Далі на GS(Geometry Shader або ж шейдер геометрії) запускається вказаний додатком код шейдера з вершинами як вхідними даними та можливістю генерувати вершини на виході. Вхідними даними GS є вершини для повного примітиву (дві вершини для ліній, три вершини для трикутників, одна вершина для точки або всі контрольні точки для патчу, якщо він потрапляє в GS з відключеною тесселяцією). Деякі типи примітивів можуть також включати вершини примітивів, що примикають до меж (додаткові дві вершини для лінії, додаткові три для трикутника).

В частині OM(Output Merger або ж етап поєднання вихідних даних) у ньому вершини можуть передаватися до пам’яті безпосередньо перед надходженням в растеризатор(RS - Rasterizer). Це схоже на «кран» у конвеєрі, який можна увімкнути, навіть якщо дані продовжують надходити в растеризатор. Дані, що надсилаються через SO, об'єднуються в буфер(и). Ці буфери можуть при подальших проходах повторно використовувати як вхідні

дані конвеєра. Растеризатор відповідає за відсікання, налаштування примітивів та визначення того, як викликати піксельні шейдери(Pixel Shader - PS). Багато хто розглядає це не як «етап» у конвеєрі, а скоріше як інтерфейс між етапами конвеєра, який виконує значний набір фіксованих функціональних операцій, багато з яких можуть бути скориговані розробниками програмного забезпечення. Наступним є растеризатор який виконує відсікання, поділ перспективи та застосовує масштаб/зміщення видового екрану. Останнім кроком у логічному конвеєрі є визначення видимості за допомогою трафарету чи глибини, а також запис до пам’яті.

Проте дана технолоія здебільшого підтримується та розробляється на операційній системі Windows. Що можна віднести до одного із невеликих недоліків даної технології.

1.3.2 OpenGL специфікація вбудованої прискореної 3D-графіки

Дана ж технологія OpenGL являється незалежною від мов програмування крос-платформовий інтерфейс що надає змогу створювати додатки які використовують 2D та 3D графіку. Так як і у Direct3D, дана технологія працює по принципу конвеєра. Конвеєр візуалізації відповідає за перетворення вершин у правильну систему координат, збирання вершин об’єктів, застосування кольору/текстури та відображення об’єкту на буфері кадрів за замовчуванням, тобто на екрані. Єдине що відрізняється деякими етапами.[6]

Конвеєр рендеринга OpenGL складається з шести етапів:

- Per-Vertex Operation (Повершина операція)
- Primitive Assembly (Збір примітивів)
- Primitive Processing (Обробка примітивів)
- Rasterization (Растеризація)
- Fragment Processing (Обробка фрагментів)
- Per-Fragment Operation (Пофрагментна операція)

Першим кроком до візуалізації зображення є перетворення даних геометрії з однієї системи координат в іншу що відбувається на рівні Per-Vertex

Operation. Далі вершини збираються в пари на рівні Primitive Assembly по 2, 3 тощо і збирається примітив, наприклад. Трикутник. На рівні Primitive Processing примітиви перевіряються, чи відповідають вони View-Volume. Якщо вони не проходять цей тест, вони ігноруються на наступних кроках. Цей тест називається Clipping. Растеризація в OpenGL є схожою до Direct3D де примітиви розбиваються на менші одиниці, що відповідають пікселям у буфері кадрів [7]. Кожна з цих менших одиниць називається фрагментами. Після того, як примітив був растеризований, до геометрії застосовується колір/текстура на рівні Fragment Processing. На фінальному етапі Per-Fragment Operation фрагменти проходять кілька тестів, наприклад:

- Alpha test
- Stencil test
- Depth test

Приклад роботи OpenGL наведено на рис 1.2:

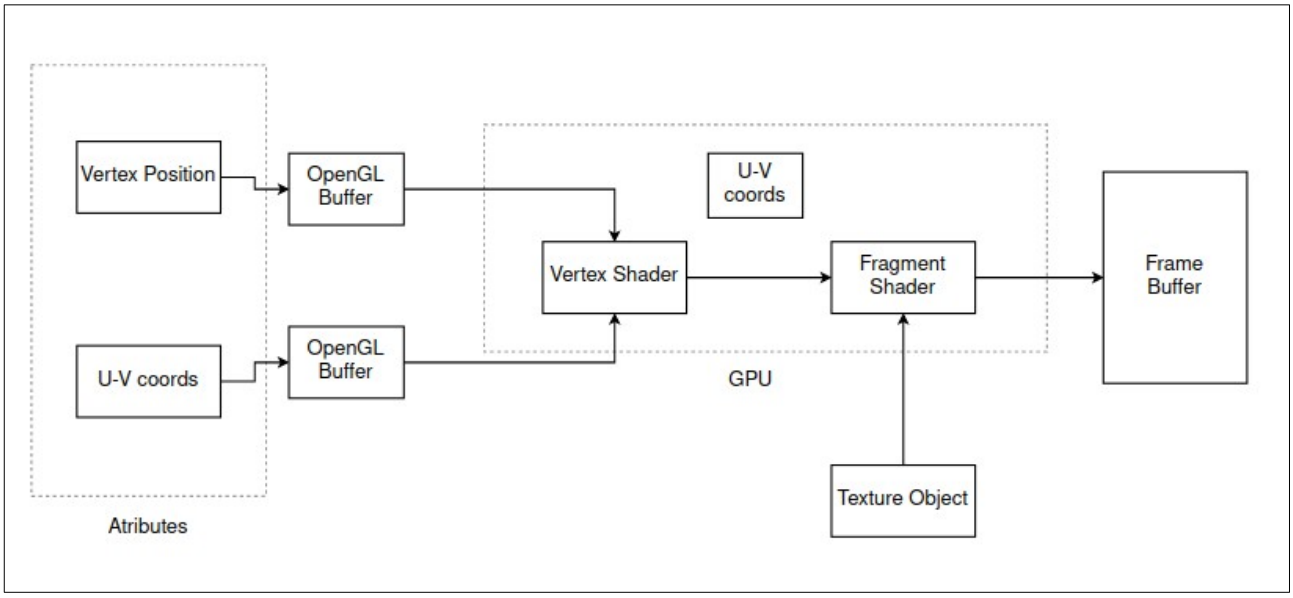


Рис. 1.2 – Приклад роботи OpenGL.

1.3.3 WebGL стандарт, що очолює розвиток 3D у веб сфері

Був час, коли стандартні веб-технології використовували лише статичний контент. Єдиною графікою, яка існувала, були інтегровані зображення. Поступово почали з'являтися нові можливості, яких розробники могли

отримати доступ через Javascript. Нарешті, виникла потреба у повністю програмованому графічному API, щоб скрипти могли створювати зображення на льоту. Елемент canvas і його API 2D-рендерінг з'явилися в WebKit і, отже, в браузерах на базі WebKit [8].

Простий canvas виглядає так:

```
<canvas id="canvas"></canvas>
```

Рис. 1.3 – Приклад простого HTML елементу canvas.

Проте без використання JavaScript та API canvas'а цей HTML елемент не виконуватиме ніяких функцій.

А от приклад з використанням вище зазначених технологій що дозволяє створити зелений квадрат, зображений на рис 1.4:

```
var canvas = document.getElementById("canvas");
var ctx = canvas.getContext("2d");

ctx.fillStyle = "green";
ctx.fillRect(10, 10, 100, 100);
```

Рис. 1.4 – Приклад використання canvas API за допомогою JavaScript коду.

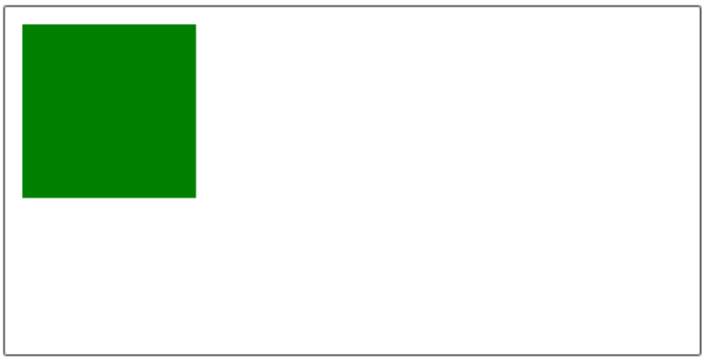


Рис. 1.5 – Результат використання canvas API за допомогою JavaScript коду.

Більшість комп'ютерних ігор, програм для дизайну почали використовувати GPU для підвищення 3D-продуктивності. Таким чином виник

попит на продуктивну 3D-графіку в браузерах. Це призвело до того, що Mozilla і Opera продемонстрували кілька перших експериментів, які розкривали контекст 3D-рендерінгу з HTML-елемента canvas. Вони були настільки захоплюючі, що спільнота вирішила зібратися, щоб стандартизувати щось, що міг би реалізувати кожен браузер.

Всі браузери спільно створили WebGL, стандарт для рендерингу 3D графіки в Інтернеті. Він був заснований на OpenGL ES, кросплатформному API для графіки, призначеному для вбудованих систем. Це був, мабуть, один із найгладших процесів стандартизації в історії Інтернету, тому що, на щастя, усі браузери працювали на системах із підтримкою OpenGL. WebGL розкрив перед розробниками потужність графічних процесорів на відкритій веб-платформі, і всі основні браузери підтримують його, дозволяючи створювати ігри, 3D графіку консольної якості для Інтернету та процвітати таким спільнотам, як бібліотека three.js. З тих пір WebGL еволюціонував до WebGL 2, і всі основні браузерні двигуни взяли на себе зобов'язання щодо його підтримки.

1.4 Перспективи розвитку 3D

Як уже було сказано, 3D є дуже востребуваною технологією у сучасному світі, вона використовується майже всюди, у комп’ютерних іграх, фільмах, проектуванні медицині(для створення 3D-моделей органів, капілярних, нервових систем), дизайні і так далі. А те, що за допомогою 3D-друку можна реалізувати будь-яку найсміливішу ідею це не новина для будь-кого.

Нині у сфері 3D-моделювання відбувається багато цікавого. Майбутнє 3D-розробки виглядає ще яскравішим, а 3D-друк та віртуальна реальність займають центральне місце. Хоча віртуальна реальність все ще є молодою технологією, вона назавжди змінить, як ми взаємодіємо з 3D-моделями. Тим часом поява 3D-друку вплинула на 3D-моделювання. Це дозволило людям скоротити ресурси та час, необхідні для створення прототипів. У майбутньому люди будуть використовувати 3D-друк та 3D-моделювання для вирішення завдань, частіше ніж будь-коли раніше. Одним із найбільших недоліків 3D-

моделювання сьогодні є низька роздільна здатність. Однак існуючі технологічні тенденції припускають, що 3D-розширення, ймовірно, значно покращиться до рівня, при якому ви зможете взаємодіяти зі своєю моделлю без обмежень. Розробники зможуть візуалізувати 3D-елементи так само, як і неозброєним оком. В результаті, користувачі отримають більш реалістичний досвід 3D-моделювання. 3D-моделювання також проникне у сферу охорони здоров'я. За прогнозами, 3D-моделювання врятує життя тисяч людей у всьому світі. Деякі компанії навіть створили 3D-моделі людського серця, і, ймовірно, у найближчому майбутньому з'явиться ще багато моделей. Більшість трансплантатів людських органів будуть відлиті та надруковані у 3D, щоб підвищити ефективність хірургічних процедур.

Також розвиток текстурування підвищить попит на 3D, оскільки в цьому випадку можна буде говрити про фотореалізм моделей. А фотореалізм не може бути досягнутий без грамотного використання текстур, але вам потрібно щось, до чого можна застосувати ці текстури. У цьому аспекті детальна 3D-модель може позбавити художників за текстурами від безлічі зайвого клопоту, пов'язаних із підробкою деталей. Інноваційний крок щодо розміщення 3D-моделей у просторі, відмінному від наших екранів, значно змінив те, як ми взаємодіємо з цими нематеріальними, але гіперреальними об'єктами. Віртуальна реальність (VR) повністю переносить користувача в інше середовище, а це означає, що 3D-об'єкти у віртуальній реальності доступні через спеціальне обладнання. Робота з 3D-об'єктами у доповненій реальності (AR) означає просто доступ до цих об'єктів через контекстно-залежну програму. І, нарешті, змішана реальність – це поєднання віртуальної та доповненої реальності. Зараз з’являються програми в яких за допомогою спеціального обладнання у віртуальній, або ж навіть змішаній реальності можна творити будь-які 3D-об’єкти. Проте все впирається у обчислювальні потужності, які при такій роботі витрачаються дуже швидко.

Висновок до розділу 1

В цьому розділі описано історію створення, та розвитку 3D-технології, що покращують наші життя. Також розглянули базову теорію 3D-моделювання, деякі стандарти, технології що дозволяють користувачам працювати з 3D-моделюванням. Дослідили перспективи розвитку даної технології і можна з упевненістю сказати, що з кожним роком вона буде набувати все більшої популярності та підкорюватиме все нові вершини.

РОЗДІЛ 2
ПОШУК ТЕХНОЛОГІЙ РОЗРОБКИ ВЕБ-ДОДАТКУ, ТА 3D МОДЕЛЕЙ

2.1 Програмне забезпечення для 3D моделювання

Для того щоб якісно та швидко створювати 3D-моделі знадобиться зручне та не складне в освоєнні програмне забезпечення для 3D-моделювання, щоб зайві функції не відволікали, а інтерфейс програми був зрозумілим. Також якщо мати навички у 2D дизайні, або ж досвід роботи з 2D-редакторами це значно полегшить роботу з 3D. Але якщо ні, це не проблема оскільки програмне забезпечення для 3D-моделювання тепер є більш доступним, ніж будь-коли, і для того, щоб освоїти стандартні в галузі інструменти, не потрібно витрачати багато час. Безкоштовне програмне забезпечення, таке як Blender, можна знайти у дедалі більшій кількості професійних конвеєрів у всій галузі. Готові ассети можуть стати дуже ефективним способом підвищення продуктивності в стислі терміни та покращення роботи із комп'ютерною графікою.

2.1.1 Autodesk Maya

Autodesk Maya займе перше місце у рейтингу більшості творців, як найкраще програмне забезпечення для 3D-моделювання. Будучи галузевим стандартом для багатьох дисциплін комп'ютерної графіки, Maya пропонує неперевершений набір функцій та інструментів, привабливий інтерфейс,який можна побачити на рис 2.1. Maya - це екосистема, і завдяки цьому, розробники моделей, можуть користуватися інструментами різних дисциплін, таких як анімація. Maya має великий вибір деформаторів, які можна знайти на вкладці анімації, які ви можете використовувати з інструментами моделювання. Ви можете згинати, ліпити, скручувати та виконувати безліч інших неруйнівних маніпуляцій з вашою сіткою.[9]

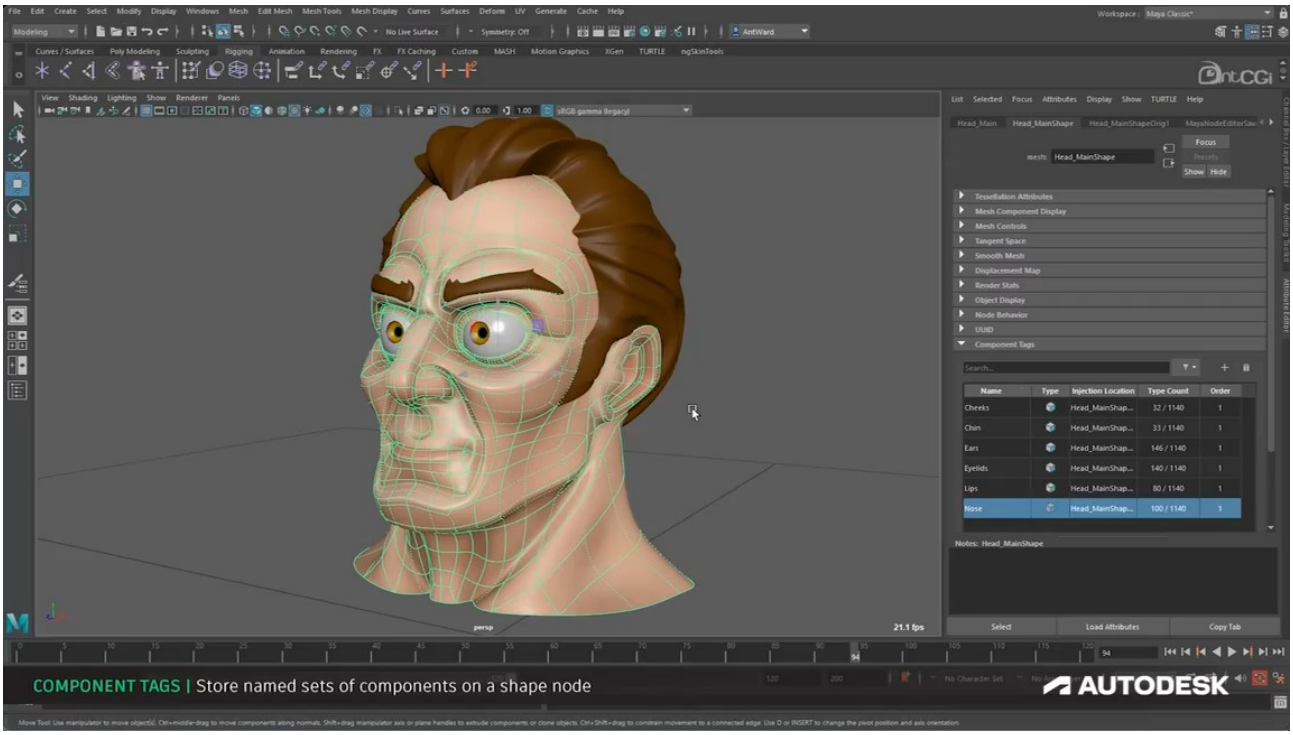


Рис. 2.1 – Зразок інтерфейсу Autodesk Maya.

Ця потужна програма розрахована здебільшого на користувачів які мають досвід у розробці 3D-моделей, її набір інструментів складний і вимагає часу для освоєння. Autodesk Maya ідеально підходить для моделювання, текстурування, освітлення та рендерингу – його великі можливості включають інструменти для частинок, волосся, фізики твердого тіла, тканини, симуляції рідини та анімації персонажів. Однак варто звернути увагу на те що велика функціональність Maya, може тільки заважати, бо для ваших потреб вони не потрібні, але відволікатимуть. Проте для тих, хто має час та терпіння, щоб освоїти дане програмне забезпечення, Autodesk Maya пропонує одні з кращих 3D-інструментів, хоч і є доволі дорогою.

2.1.2 Zbrush

ZBrush — найпопулярніша програма для 3D-скульптингу. Що відрізняє його від інших 3D-інструментів, то це те, що ZBrush імітує традиційні методи скульптури, які виконуються в цифровому вигляді на комп'ютері. Скульптура в ZBrush схожа на роботу з цифровою кулею з глини, де ви надаєте форму для неї майже так само як і в реальному житті. Інструменти скульптингу від ZBrush

надають широку свободу творчості. Мало того, що художники можуть створювати більш органічні та деталізовані моделі за допомогою ZBrush, вони часто можуть отримати готовий продукт набагато швидше ніж з іншими програмами, такими як Maya або 3ds Max. На сьогоднішній день воно визнане передовим програмним забезпеченням для 3D-скульптингу на ринку. Завдяки великому набору інструментів і різноманітності ассетів легко зрозуміти, чому ZBrush став таким успішним. Всередині ZBrush художники можуть витягувати, вдавлювати, розчавлювати, дряпати чи іншим чином маніпулювати цифровою глиною так само, як вони б працювали з реальною. Маніпулюючи віртуальною поверхнею, 3D-художник може створювати деталізовані елементи, такі як луска дракона або зморшки на обличчі людини. Дрібні деталі такого роду дуже складно відтворити без інструментів, як є у ZBrush. Користувачі навіть можуть створювати макети своїх персонажів прямо в ZBrush, заощаджуючи дорогоцінний час розробки.

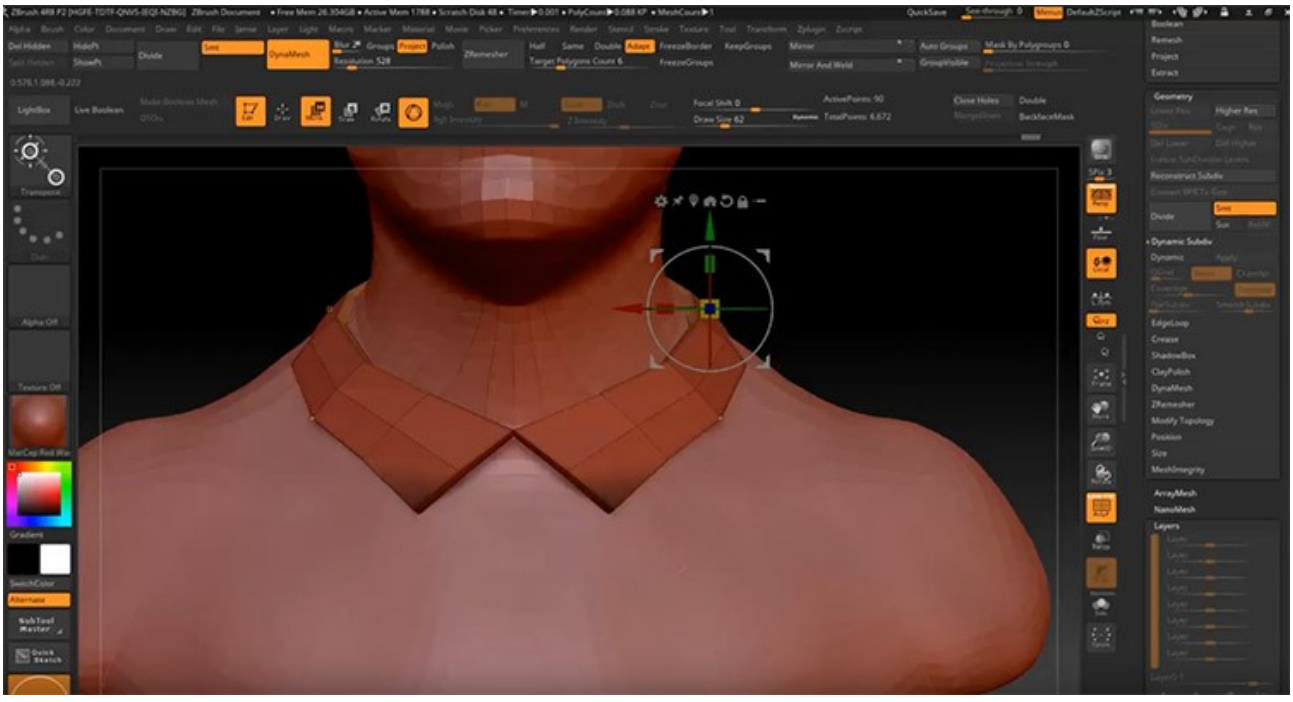


Рис. 2.2 – Зразок інтерфейсу Zbrush

На відміну від традиційного програмного забезпечення для моделювання, ZBrush немає необхідності маніпулювати окремими полігонами, оскільки він

використовує зовсім інший підхід. ZBrush зробив революцію в конвеєрі 3D-анімації. У минулому 3D-аніматорам часто доводилося створювати версії своїх персонажів із низькою роздільною здатністю в іншому 3D-пакеті наприклад Autodesk Maya. Такі моделі з низькою роздільною здатністю відомі як «базова сітка». Цю базову модель використовують далі для анімації через апаратні обмеження. Але ZBrush змінив це, дозволивши художникам створювати свої дизайни персонажів з повною міццю 3D-скульптури від початку до кінця. Тепер художники можуть ліпити свого ідеального персонажа в одній програмі . Кілька років тому скульптинг в основному використовувався для тонкого налаштування моделі з низькою роздільною здатністю. ZBrush змінив цю тенденцію, і тепер художники можуть від початку експериментувати з своїми моделями використовуючи високий рівнь деталізації.

Хотілося б ще зазначити, що ця програма є платною, має не стандартний інтерфейс зображений на рис 2.2, та меню. Але хотів би відзначити хороший набір інструментів для скульптингу та хорошу оптимізацію, що дозволяє з легкістю обробляти мільйони полігонів.

2.1.3 Blender

Попереднє програмне забезпечення являється платним, тому давайте розглянемо Blender, програму для 3D-моделювання з відкритим програмним кодом.[10]

Blender має багато переваг як навчальний інструмент для початківців у порівнянні з іншими програмами, включаючи деякі з наведених нижче:

- Його програмне забезпечення з відкритим вихідним кодом.
- Для використання не потрібна реєстрація або ліцензійний ключ.
- Можете змінити вихідний код, як забажаєте.
- Blender завжди має більше 1 способу виконати завдання.
- Можна імпортувати та експортувати безліч різних форматів файлів
- Можна анімувати не лише у 3D, а й у 2D.

• Практично будь-яке творче завдання можна виконати за допомогою однієї програми.

Існує багато переваг використання Blender 3D в порівнянні з аналогічними додатками, але, можливо, найважливішим з усіх є той факт, що Blender є програмним забезпеченням. Щоб отримати Blender, все, що вам потрібно зробити, це перейти на домашню сторінку Blender, натиснути синю кнопку вгорі, щоб перейти на сторінку завантаження, а потім вибрати іншу синю кнопку, щоб почати процес завантаження. Це також підкреслює пару інших важливих переваг, які має Blender. По-перше, це не той випадок, коли існує безкоштовна полегшена версія Blender, а потім платна версія з усіма ключовими функціями. Будь-яка версія Blender, яку ви бажаєте завантажити, буде доступна безкоштовно з веб-сайту Blender. По-друге, Blender — це програмне забезпечення з відкритим вихідним кодом. Коли ви завантажуєте Blender, вхід до системи не потрібний, і вам не потрібно реєструвати будь-які дані.

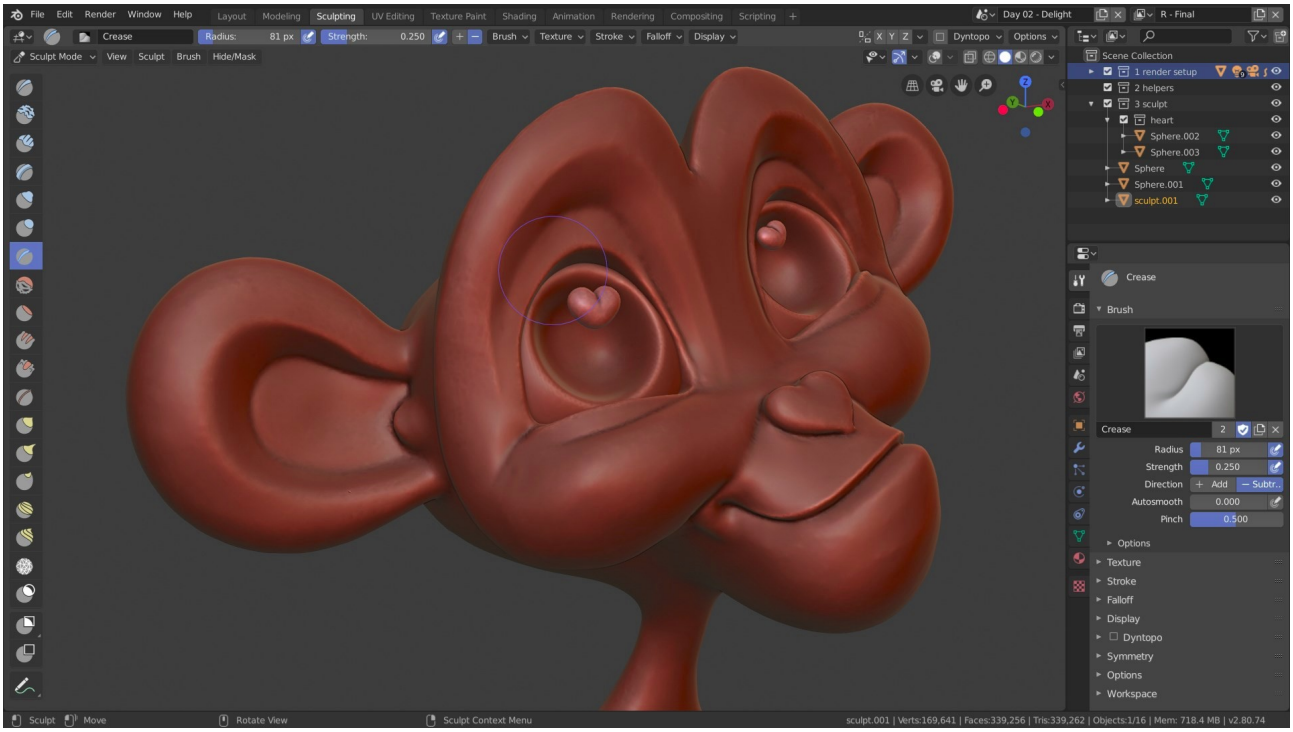


Рис. 2.3 – Зразок інтерфейсу Blender

Багато програм створюються для виконання завдань певного типу або невеликого набору завдань, схожих один на одного. З Blender 3D доступ відкритий майже до всіх інструментів, які можуть знадобитися для створення того, що ви хочете, в одному додатку.

За допомогою жирного олівця ви можете створити 2D-концепт майбутньої 3D-моделі. Потім можна приступити до фактичного етапу моделювання проекту у вікні 3D-перегляду, маючи можливість створювати свою модель різними способами. Ви можете використовувати методи моделювання в Blender для створення базової форми, а потім використовувати набір інструментів для скульптингу, щоб створити дрібніші деталі вашої моделі. Потім ви можете перейти до наступного етапу проекту та додати матеріали та текстури до свого персонажа та одягу персонажа за допомогою інструментів UV-картографування блендерів та чудової системи вузлів для застосування матеріалів. Потім ви можете перейти до стадії оснастки процесу, де ви можете підготувати свого персонажа до рухів, природних для людей, і підготувати його до анімації, яка стає наступним кроком у процесі, коли ви можете створювати анімовані рухи для моделі персонажа. І все це ви можете робити всередині Blender без необхідності будь-якого іншого програмного забезпечення. Якщо ви використовуєте чистий робочий процес анімації, весь ваш проект можна легко виконати в Blender без необхідності будь-якого іншого програмного забезпечення.

Насправді ви навіть можете редагувати візуалізовану анімацію, яку ви створили, використовуючи набір програм для редагування відео. І ви можете додавати ефекти, фільтри та накладання до своїх анімацій, використовуючи компонувальник вузлів Blenders. Є лише кілька програм, які дозволять створювати власні 3D-анімації, а потім візуалізувати їх усередині цієї програми. Є також кілька прикладів, де можна створювати 2D-анімації за допомогою певних додатків. Крім того, є програмне забезпечення, яке дозволяє редагувати відеоконтент, обрізати кліпи, додавати фільтри тощо. Але є дуже мало програм,

які дозволяють вам робити все це в одному додатку і Blender добре вписується в цю дуже маленьку категорію. У Blender можна пройти весь процес анімації, не вимагаючи використання будь-яких інших додатків, будь то етап концепції, етап розробки або навіть етап рендерингу. І навіть після того, як ви візуалізували свою 3D- або 2D-анімацію, у вас все ще є можливість редагувати рендеринг у Blender за допомогою редактора та компонувальника. Хоча Blender здатний робити багато різних речей, він не може робити все, і зрештою те, що ви створюєте в Blender, ви захочете використати в іншому місці, наприклад, для рендерингу зображень.

2.2 Монорепозиторій

Монорепозиторій — це код репозиторію з контролем версії, в якому зберігається безліч проектів. Хоча ці проекти можуть бути пов’язані, вони часто логічно незалежні та виконуються різними командами [12]. Монорепозиторії іноді називають монолітними репозиторіями, але їх не слід путати з монолітною архітектурою, яка є практичною розробкою програмного забезпечення для опису автономних додатків. Протилежністю монорепозиторія є мультирепозиторій, де кожен проект зберігається в абсолютно окремому репозиторії з контрольованою версією. Універсального результату для проектів немає, оскільки кожен має свої переваги та недоліки. Перевагами монорепозиторію є:

- Спрощене керування залежностями . Немає необхідності вказувати номери версій для залежностей, оскільки для всіх проектів існує тільки одна універсальна версія.
- Більш простое розгортування проекту. І інтерфейсний код, і внутрішній код зберігаються в одному місці, і, оскільки їх можна розвернути одночасно за допомогою одного запиту на включення, розгортання займає менше часу.
- Повторне використання коду . Кожен в команді може бачити весь проект у будь-який час, тому вони можуть легко ідентифікувати загальні компоненти та,

таким чином, використовувати загальні функції в мікросервісах монорепозиторію.

- Більш легкий рефакторинг . Реструктуризація також спрощується, коли весь код знаходиться в одному місці.
- Більш швидка перевірка коду. Легко переглядати код і відстежувати зміни, коли все зберігається в одному репозиторії.
- Краще тестування . Оскільки розробники можуть запускати всю платформу локально, вони можуть краще зрозуміти, як усі сервіси працюють разом, і, таким чином, знайти більше помилок. Крім того, якщо вся кодова база використовує одину мову програмування, для перевірки всієї системи також буде достатньо одного запуску тестів.

Проте існують випадки коли монорепозиторії є недоречними та краще використати мультирепозиторій. Один із часто згадуваних аргументів проти використання єдиного репозиторію полягає в тому, що він погано масштабується. Продуктивність Git знижується зі збільшенням кількості коммітів і поглиблення історії. Однак це вірно тільки для гігантських додатків і команд з сотнями розробників, які щодня працюють в одному репозиторії. Як бачимо, насправді існує дуже обмежена кількість ситуацій, у яких вам слід відмовитися від використання монорепозитория. Наприклад з міркувань безпеки або з інших причин ви не хочете, щоб усі члени команди могли переглядати весь код. Неможливо обмежити доступ до будь-якої частини монорепозиторію. Є багато проектів, і всі вони виконуються різними командами на різних технологіях. Якщо це так, командам не потрібно зберігати всю кодову базу локально. Є частина коду, яка часто змінюється, та інша частина, яка майже ніколи не змінюється. Наприклад, інтерфейсна частина вашого продукту змінюється щомісяця, а внутрішня частина залишається практично незмінною. Кожне розгортання - це ризик щось зламати (особливо якщо немає автотестів), і ви, ймовірно, не хочете регулярно ризикувати пошкодженням серверної системи, що ідеально функціонує.

2.2.1 Lerna – менеджер монорепозиторіїв для проектів JavaScript.

Lerna допоможе вам взяти велику кодову базу і розділити її на пакети, що незалежно розгортаються. Lerna обробляє кожен крок у процесі релізу — від завантаження залежностей та зв'язування пакетів до тестування та публікації оновлених пакетів у реєстрі NPM. Працюючи поверх традиційних інструментів, таких як npm та Yarn, Lerna може зрозуміти, як взаємопов'язані пакети в репозиторії. Зайве говорити, що це полегшує спільне використання коду в репозиторії.

Припустимо, потрібно створити веб додаток з декількома репозиторіями, що містять різні кодові бази. Один містить фронт-енд базу коду, інший містить службу API. Тому зручно було б використовувати один і той же інструментарій у всіх репозиторіях. Під інструментами мається на увазі використання таких інструментів як eslint, prettier, babel і т. д. Замість того, щоб мати окремі конфігурації для кожної бази, ми можемо мати спільні конфігурації, які можуть бути спільними для всіх проектів, у цьому нам допомагає lerna. Тоді структура репозиторію мала б вигляд зображений на рис 2.4. Тут кореневий файл package.json містить загальні залежності для всіх проектів (front-end, back-end). Кореневий node_modules містить усі потрібні залежності. Проекти перелічені у каталозі пакетів. Каталог node_modules у back-end та front-end проектах містить залежності back-end та front-end відповідно.

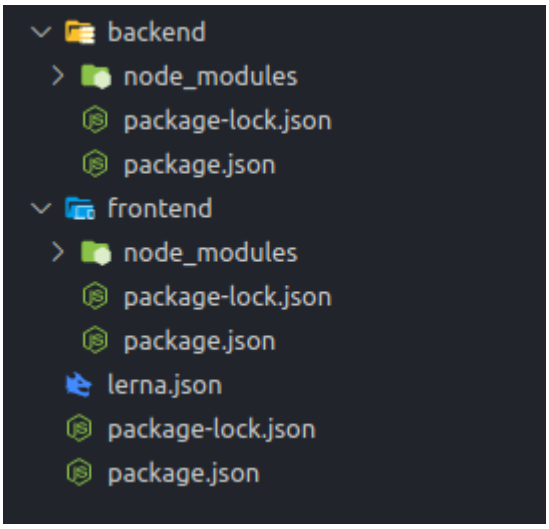


Рис. 2.4 – Структура репозиторію з Lerna.

Lerna завантажує всі пакети у основний репозиторій. Іншими словами, він встановлює всі залежності та пов'язує будь-які перехресні залежності, якщо вони є. Він пов'язує загальні залежності, створюючи символічні посилання. Якщо будь-яке оновлення виконується в загальній залежності, воно негайно набирає чинності у коді, що його використовує. Lerna дозволяє перевірити, які пакети були змінені з моменту останнього випуску. Ви також можете перевірити різницю для всіх пакетів або окремого пакету з моменту останнього випуску.

2.3 Вибір веб-технологій

Веб-браузер — це програма за допомогою якої Інтернет-користувач може відвідувати і переглядати веб-сайти на комп’ютерах та гаджетах призначених для цього.

Основними протоколами обміну інформації між інтернет-ресурсами є HTTP та HTTPS протоколи. HTTP — це протокол передачі даних між учасниками комп’ютерних мереж. Являється скороченням від HyperText Transfer Protocol. HTTPS синтаксично схожий до HTTP проте відрізняється додатковим шаром шифрування.

2.3.1 Основні технології для front-end частини

Основними технологіями для створення веб сторінок є три мови програмування — це JavaScrip, HTML, CSS.

HTML — це мова програмування за допомогою якої створюється розмітка веб-сторінок. Розмітка здійснюється за допомогою тегів і оприділяє структуру, вміст наповнення сторінки. Прикладом такого тега може бути HTML елемент текстового абзаца <p>, приклад якого зображений нижче на рис 2.5 :

```
<p class="text-paragraph">A simple text paragraph</p>
```

Рис. 2.5 – Приклад простого HTML елементу p.

JavaScrip — об’єктно прототипна мова програмування, використовується для доступу та змін структури веб-сторінки, зазвичай використовується для додавання інтерактивності, анімацій, динамічної зміни стилів. Також важливою частиною є робота з сервером — отримання інформації, відправлення отриманої від користувача. І звичайно ж створення 3D графіки після того як браузери додали стандарт WebGL . Приклад виклику діалогового вікна в браузері за допомогою JavaScrip:

```
<script>
  alert( "Hello world" );
</script>
```

Рис. 2.6 – Приклад виклику діалогового вікна.

CSS (Cascading Style Sheets) — мова стилізації веб-сторінок, що дозволяє змінювати зовнішній вигляд як окремих елементів сторінки, так і її вцілому.

За допомогою цієї стилізації можна змінювати шрифти, колір тексту, заливку фону(рис 2.7), прозорість елементів або ж цілих блоків, анімації елементів

сторінки. Дозволяє зробити сайти більш різноманітними та візуально приємнішими, зрозумілішими для користувача.

Рис. 2.7 – Приклад CSS коду.

TypeScript - це мова програмування з відкритим вихідним кодом, розроблена Microsoft, яка компілюється в JavaScript. З моменту випуску в 2012 році мова активно розвивається і з кожним роком продовжує набирати популярності. TypeScript У 2019 році зайняв 7-е місце за популярністю і 5-е місце серед найшвидших мов на GitHub з більш ніж 5 мільйонами щотижневих завантажень його компілятора.

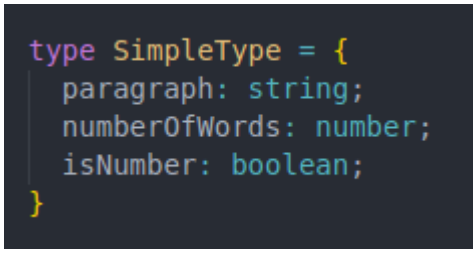


Рис. 2.8 – Приклад типу реалізованого у Typescript.

Найважливіша різниця між TypeScript і JavaScript полягає в тому, що це дві окремі мови програмування, хоча TypeScript значною мірою заснований на JavaScript. Насправді TypeScript - це надмножина JavaScript, що означає, що весь допустимий код JavaScript також є допустимим кодом TypeScript. Все, що можна написати на JavaScript, можна написати і на TypeScript:

- фронтенд з будь-яким фреймворком
- бекенд з Node.js
- кросплатформні мобільні програми в React Native.

Ключове доповнення, яке TypeScript привносить у JavaScript, - це система типів. Тип - це визначення, яке говорить вам про передбачуване використання даних (наприклад, у нас є числові, логічні або рядкові типи, і це лише деякі з них). Традиційно JavaScript має динамічну типізацію. Це означає, що одна змінна може містити текст, число або навіть цілу сутність бази даних, залежно від випадку, і це нормально. TypeScript суворо визначає, що може містити ця змінна. Припустимо, один із розробників пише функцію compareString і використовує типи для визначення того, що функція може приймати лише два рядки. У цьому випадку TypeScript поверне помилку, якщо інший розробник спробує надати число функції, тоді як у JavaScript така операція була б цілком прийнятною.

2.3.2 Технології для роботи з 3D-об’єктами front-end частини

ThreeJS — це бібліотека Javascript, яка дозволяє вам керувати 3D-об'єктами прямо в браузері. , Вона через Javascript дозволяє використовувати WebGL на полотні(canvas) HTML5. Таким чином, ми маємо бібліотеку ThreeJS, яка маніпулює API Javascript для виконання графічного рендерингу WebGL. Вона максимально спрощує та скорочує код, необхідний для того, щоб робити все, що ви хочете з 3D. ThreeJS зробить усю складну частину. Нам просто потрібно використати наявні функції бібліотеки. Отже, якщо у вас є прості знання Javascript, ThreeJS дає можливість робити неймовірні речі в 3D.

Щоб зрозуміти, як ThreeJS працює на високому рівні, потрібно поставити себе на місце кінорежисера. Оскільки тут теж є схожі терміни, процеси, які нагадують зйомку фільму. Давайте розглянемо їх детальніше:

- Сцена – це по суті як тривимірний світ, де ми збираємось працювати. Ми можемо мати будь-яку кількість об'єктів в цій сцені за допомогою мешів(mesh), задавати їм будь-яке положення, орієнтацію в просторі.
- Меші – це просто об'єкти, які будуть присутні у сцені. Нам потрібно буде освітити ці об'єкти, щоб їх побачити, оскільки початково тривимірний світ не має освітлення і ми не побачимо об’єкти додані в нього. Також щоб побачити їх,

вам доведеться додати камеру, яка буде аналогом елементу візуального світосприйняття.

• Камера працює аналогічним чином, як і в реальному житті, переміщуючи камеру, переміщатимуться і об’єкти в полі зору даної камери, або ж поза нього. Те, що потрапляє в поле зору камери далі відправиться на процес рендерингу.

• Процес рендерингу приймає сцену та камеру як параметри. При цьому він відображає все на канвасі(canvas) HTML5, про який йшла мова раніше. Рендеринг буде створювати зображення щоразу, коли екран оновлюється. Зазвичай це 60 кадрів за секунду. Це те, що дає змогу створювати анімації додаати ефекти пост рендерингу і тому подібні речі.

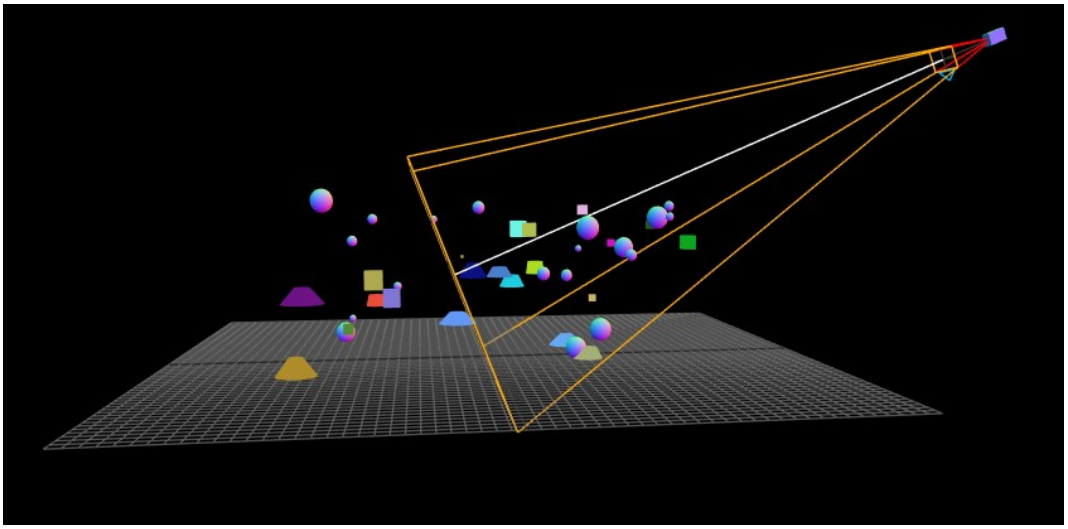


Рис. 2.9 – Сцена з мешами та камерою.

Зважаючи на те, що у front-end частині ми використовуватимемо React переваги якого будуть описані нижче, доцільним буде використання бібліотеки react-three-fiber який надасть всі переваги реактивності, про яку піде мова далі, для обробки 3D сцен. Нам потрібно просто створювати React компоненти як і раніше а react-three-fiber сам перетворить їх на залежності, функції які використовує ThreeJS. Саме це дозволяє нам дозволяє залишатись парадигмі реактивності React. Також хотілося б згадати про такі поняття як геометрія, та матеріал. Геометрія це те, як ми призначаємо різні форми сітці. Вона дозволяє нам створювати сфери, куби, піраміди тощо. Матеріал – це компонент, який

надає сітці властивості матеріалу. Ефекти на рівні поверхні (такі як блиск, кольори тощо) призначаються за допомогою матеріалу.

2.3.3 Найпопулярніші фреймворки та бібліотеки front-end частини

Фреймворки користувацької частини використовуються для розробки зовнішнього інтерфейсу або інтерфейсу користувача веб-додатка. Фронтенд-фреймворки в основному засновані на мовах програмування, як JavaScript, HTML і CSS. Відповідальність переднього плану полягає у розробці дизайну UX/UI та вирішенні інших важливих аспектів програми, таких як SEO-оптимізація, фрагменти коду, шаблони, керування взаємодією з користувачем та багато іншого. В Україні та й по всьому світу, за останній рік лідирував фреймворк React, після нього Vue.js, а потім Angular. Давайте розглянемо перші два поближче.

2.3.3.1 React

React це в основнові своїй бібліотека JS, представлена Facebook (наразі Meta). React працює як веб-фреймворк і переважно використовується для розробки односторінкових інтерфейсних додатків. Крім того, він підтримує розробку мобільних програм, що зазвичай не підтримується більшістю інших фреймворків. React легко інтегрується з кількома іншими бібліотеками для успішного виконання операцій маршрутизації, керування станом та взаємодії з API.

React в основному дозволяє розробникам використовувати окремі частини свого додатка як на стороні клієнта, так і на стороні сервера, що зрештою підвищує швидкість процесу розробки. Говорячи простою мовою, різні розробники можуть писати окремі частини та всі внесені зміни не вплинуть на логіку програми. У порівнянні з іншими інтерфейсними фреймворками код React легше підтримувати і він є гнучким завдяки своїй модульній структурі. Ця гнучкість, у свою чергу, заощаджує величезну кількість часу та коштів для бізнесу. React було розроблено для забезпечення високої продуктивності. Ядро фреймворку пропонує віртуальну програму DOM та рендеринг на стороні

сервера, завдяки чому складні програми працюють дуже швидко. Однією з основних переваг використання React є можливість повторного використання компонентів. Це заощаджує час розробників, оскільки їм не потрібно писати різні коди для тих самих функцій. Крім того, якщо будь-які зміни будуть внесені до будь-якої конкретної частини, це не вплине на інші частини додатка. React поставляється з JSX, необов'язкове розширення синтаксису, яке дозволяє писати ваші власні компоненти. Ці компоненти переважно приймають цитування HTML, а також роблять рендеринг всіх підкомпонентів чудовим досвідом для розробників. Хоча було багато дебатів з приводу JSX, але він вже використовується для написання компонентів користувача, створення додатків великого об'єму і перетворення макетів HTML на дерева ReactElement.

Переваги:

- Він складається із віртуального DOM, який дозволяє швидко маніпулювати документами.
- Легко інтегрується з іншими бібліотеками.
- Він підтримує розробку мобільних програм.
- React надає багато інструментів розробки та залежностей, щоб допомогти розробникам React. Серед них React Developer Tools та Redux Developer Tools — розширення Chrome, які допомагають виявляти помилки в коді та виправляти помилки. Це не вимагає жодних додаткових навичок: ви просто встановлюєте їх та запускаєте натисканням кнопки. Крім того, ми вже згадували про різноманітні бібліотеки компонентів React, які можна вільно використовувати у різних проектах

Недоліки:

- React ніколи не стоїть на одному місці. Бібліотека постійно розвивається, але це може призвести до певних негативних наслідків. Оскільки зміни продовжуються, середовище постійно змінюється, і розробники можуть не встигати за оновленнями. З іншого боку, така швидка еволюція покращує продукт та полегшує роботу.

• Цей недолік багато в чому випливає із попереднього. Оскільки темпи розробки React високі, своєчасно оновлювати документацію стає складно. Розробникам складно шукати навчальні матеріали про останні оновлення. У деяких випадках розробники створюють власні інструкції, але вони можуть бути погано написані.

2.3.3.2 Vue.js

Vue.js - один з найпопулярніших прогресивних фреймворків, що використовуються для створення інтерфейсів і односторінкових додатків. Фреймворк Vue.js призначений для інкрементальності та масштабованості. Його також дуже легко інтегрувати з іншими бібліотеками та функціями, орієнтованими на рівень вистави. Як і інші фреймворки JavaScript, Vue викликав фурор у світі розробки веб-застосунків і зарекомендував себе як революційна технологія, що має безліч варіантів використання. Крім того, це фреймворк, який можна використовувати для створення як настільних, так і мобільних додатків. І розширення HTML, що поставляються з базою JavaScript, швидко зростають. В результаті багато технологічних компаній, таких як Adobe, тепер віддають перевагу цьому інтерфейсному інструменту.

Vue.js був здебільшого розроблений з ідеєю досягнення позитивних результатів з мінімальними зусиллями. Це дозволяє розробникам створювати програми або розробляти інтерфейс користувача з дуже невеликою кількістю рядків коду. Vue.js є не дуже складним фреймворком особливо для початківців, він потребує лише базових знань HTML, CSS та JavaScript. Також він протиставляється Angular або React, оскільки ці технології вимагають знання додаткових мов програмування, які вимагають просунутих навичок кодування. Крім того, Vue.js постачається з плагіном для браузера Chrome та Firefox, що полегшує початок роботи з цією технологією. Коли ми говоримо про переваги Vue.js, то можна згадати про досить таки високий рівень продуктивності. І головна причина цього в тому, що Vue.js не тільки працює з Virtual DOM (об'єктна модель документа), але й дуже добре справляється із усуненням

недоліків Virtual DOM. Він також може керувати високою частотою кадрів. Все це призводить до кращої продуктивності, ніж React. Завдяки простій структурі Vue.js дозволяє розробникам легко додавати даний фреймворк у свої веб-проекти. Цей JavaScript-фреймворк має деякі чудові функції, такі як спостерігачі та директиви, які полегшують експертам створення сучасного веб-додатку. Це також дозволяє розробникам використовувати інші гнучкі мови програмування під час роботи з цією структурою. Оскільки Vue.js є одним з найпопулярніших фреймворків серед розробників веб-застосунків, він спрощує інтеграцію з існуючими програмами. І все це можливо, тому що Vue базується на JavaScript. Він має функції, що дозволяють розробникам легко інтегрувати цю технологію до інших програм, побудованих на JavaScript.

З моменту появи Vue.js він отримав надійний набір інструментів, які будь-яка команда розробників може використовувати під час створення програм. Крім того, Vue.js має власний менеджер станів, інструменти налагодження браузера та серверний рендерер. Ось деякі недоліки Vue.js. Як ми вже знаємо, Vue поставляється з досить широкою екосистемою та різними інструментами, які можуть знадобитися розробнику, коли він починає працювати над програмою. Тим не менш, Vue не такий великий, як Angular та React. Наприклад, якщо ви порівняєте кількість плагінів, доступних у Vue.js та React, різниця обчислюється тисячами. Крім того, існуючі плагіни, які розробники хотіли б використовувати з іншими фреймворками, часто погано підтримуються. Складність реактивності Vue – це технологія двосторонньої прив'язки даних, яка керує оновленнями віртуальної моделі DOM. І хоча це зручний інструмент, який синхронізує всі необхідні компоненти, є деякі проблеми, які можуть стосуватися роботи системи реактивності. Веб-програма Vue.js складається з безлічі інтерактивних компонентів. Кожен компонент поставляється зі спостерігачем, який перемальовує дані щоразу, коли користувач запускає компонент. І тоді система реактивності перемальовує лише фрагменти даних, що їх запустив користувач. Але проблема в тому, що він може

робити помилки у процесі читання даних. І саме тому він вимагає, щоб дані були злагоджені.

Переваги:

- Він легкий у вивченні.
- Структура сприяє швидкому розвитку.
- Легко інтегрується зі сторонніми програмами.

Недоліки:

- Він не підтримується провідними IT-підприємствами, такими як Google, Microsoft тощо.
- Доступні інструменти не дуже стабільні.

2.3.4 Технології back-end частини

Бекенд-фреймворки використовуються для розробки серверної або прихованої частини програми, яка відповідає за функціонування веб-програми. Вони засновані на таких мовах програмування, як Python, .NET, Ruby, Java, PHP, NodeJs. Ці фреймворки рацюють, маючи справу з функціонуванням сервера та бази даних, протоколами маршрутизації, логікою та архітектурою рішення, параметрами авторизації, безпекою даних та багатьом іншим.

2.3.4.1 NodeJs

Веб-застосунки, що працюють на Node.js, отримують величезні переваги завдяки своїй здатності до багатозадачності. На відміну від інших платформ, його однопотокова, подійно-орієнтована архітектура ефективно обробляє безліч одночасних запитів, не засмічуючи оперативну пам'ять. Більш того, його цикли подій та незалежні операції введення-виводу дозволяють виконувати код у хорошому темпі, що суттєво впливає на загальну продуктивність проекту.

Надздібність Node.js полягає в тому, що він побудований на движку V8 Google Chrome і написаний на C++. Він розбиває функції Javascript на машинні коди з високою ефективністю та швидкістю. Це важливо тому, що Google вкладає значні кошти у свій двигун для підвищення продуктивності, безпеки та забезпечення чудової підтримки сучасних функцій JavaScript. Це безпосередньо

впливає на продуктивність V8 і, зрештою, на Node.js. Ця передова технологія забезпечена багатьма функціями, такими як кластерний модуль. Він забезпечує розподіл навантаження на кілька ядер процесора, що дозволяє легко досягати бажаних результатів за допомогою невеликих модулів без перевантаження оперативної пам'яті. Крім того, в Node.js використовується неблокуючий механізм event-loop, який забезпечує високу масштабованість та дозволяє серверу безперешкодно обробляти запити. З точки зору розробки, Node.js дозволяє використовувати мікросервіси, які дозволяють розділити вашу програму на більш дрібні частини. Таким чином, можна легше визначати завдання та ефективно розподіляти їх між різними командами для прискорення розробки, розгортання та обслуговування кожного розділу вашої програми.

2.3.4.2 Express

Це веб-система, яка дозволяє структурувати веб-додаток для роботи з різноманітними запитами HTTP за певним URL. Express - це мінімальна, з відкритим вихідним кодом система веб-додатків Node.js, призначена для спрощення створення сайтів, веб-додатків і API. Вона дозволяє відповідати на запити за допомогою маршрутів, що дозволяє записувати відповіді на певні URL-адреси. Щоб просто генерувати HTML, Express.js підтримує безліч шаблонів. Ключовим елементом Express.js є те, що він є дуже простим і має відкритий вихідний код. Крім того, фреймворк використовується для розробки програм реального часу, таких як чати та програми для інформаційних панелей, де стає просто інтегрувати WebSocket у фреймворк. Express.js обробляє частину процесу маршрутизації та проміжного програмного забезпечення, дозволяючи зосередитись на життєво важливій бізнес-логіці цих функцій у реальному часі. Express.js дуже простий у використанні і гнучкий, і він швидше, ніж будь-який інший фреймворк Node.js. Цей мінімалістичний фреймворк пропонує швидку розробку додатків і полегшує освоєння безлічі різних частин більшого фреймворку. Він також надає багаті можливості, такі як відмінна система маршрутизації, проміжне програмне забезпечення та узгодження контенту

прямо з коробки. Express.js довів свою хорошу масштабованість протягом багатьох років завдяки тому, що багато великих компаній щодня використовують цей фреймворк на своїх серверах. Він ефективно обробляє запити та відповіді користувачів і практично не вимагає додаткового налаштування при розробці великомасштабної веб-програми [13]. Він має відмінні модулі, пакети та додаткові ресурси, що допомагає розробникам створювати надійні та масштабовані веб-програми.

2.3.4.3 FeathersJS

Feathers - це відомий, легкий фреймворк для програм реального часу та REST API, що використовує JavaScript або TypeScript. Це мікросервісний фреймворк Node JS, створений для спрощення процедури розробки веб-додатків та модульних програм. З його допомогою можна створювати неймовірні програми за менший час. Він може легко взаємодіяти з будь-якою технологією бекенду та фронтенду, такий як Angular, Vue JS, React тощо. Створення прототипів та готових до виробництва додатків за лічені хвилини є його унікальною перевагою. Він призначений для створення складних програм. Це ідеальне поєднання інструментів та архітектури для створення веб-додатків у реальному часі з самого початку та масштабованих REST API. Є безліч варіантів підтримки плагінів, які можуть легко внести аутентифікацію [14].

Переваги:

- Підтримуються CLI та плагіни за замовчуванням для авторизації
- Автоматизовані RESTful API
- Проста інтеграція з фреймворком на стороні клієнта
- Незалежність від сховища даних

Висновок до розділу 2

У цьому розділі було розглянуто технології які можуть допомогти у створенні веб додатку для візуалізації 3D-об’єктів, їх переваги та недоліки. Дослідили моно та мульти репозиторії, у висновку хочу сказати що для даного проекту краще підійде монорепозиторій, що полегшить роботу.

Також було проаналізовано технології для створення серверної та клієнтської частини, у результаті маємо такий стек технологій:

- Для серверної частини було обрано FeathersJS що значно спростить роботу з запитами та аутентифікацією.
- Для клієнтської частини обрано мову JavaScript та її фреймворк React з бібліотекою Three.js для 3D-графіки.

Розділ 3.

Опис розробленого проекту, та його виконання

3.1 Структура проекту

Оскільки ми використовуємо монорепозиторій який допомагає краще структурувати роботу з кожною частиною проекту то його структура матиме наступний вигляд:

- Весь проект зберігається в папці 3d-editor у якій зберігаються налаштування для lerna, .gitignore, package.json в якому зазначені основні скрипти для запуску проекту, та версії допоміжних бібліотек, також у ній зноходиться папка node_modules, яка зберігає всі використані бібліотеки для front-end та back-end чатини, оскільки у нас монорепозиторій.

- Є папка packages яка зберігає у собі основні частини проекту, такі як back-end, front-end, common. Остання створена і налаштована таким чином, що доступ до неї наявний як у back-end частини так і у front-end. Це дозволяє продуктивніше і краще писати якісь спільні типи, утиліти, чи допоміжні функції що використовуватимуться у різних частинах проекту.

3.1.1 Структура common частини

Common має 3 основні папки це dist, node_modules, src. Dist Це папка що зберігає білд common частини і саме до неї звертатимуться back-end та front-end частини. Node_modules має бібліотеки які не потрібні для інших частин проекту, що локалізує їх і збільшує продуктивність.

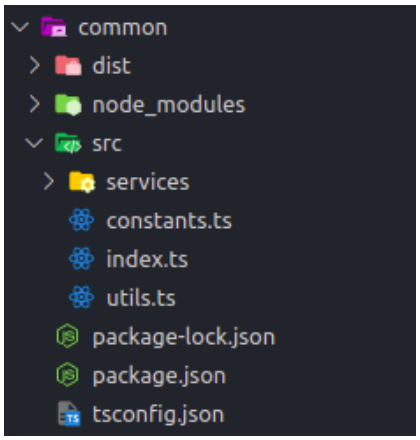


Рис. 3.1 – Структура common.

У папці src зберігаються основні дані зокрема файли з константами, утилітами, папки з типами для сервісів та основний index.ts файл, що експортує попередні. Таокж common має package.json для контролю використовуваних модулів, скрипти що дозволяють запускати дану частину проекту.

3.1.2 Структура back-end частини

Back-end частина має значно складнішу структуру що зберігається у папці backend, проте вона дещо схожа, ми також маємо папку src у якій зберігається основа.

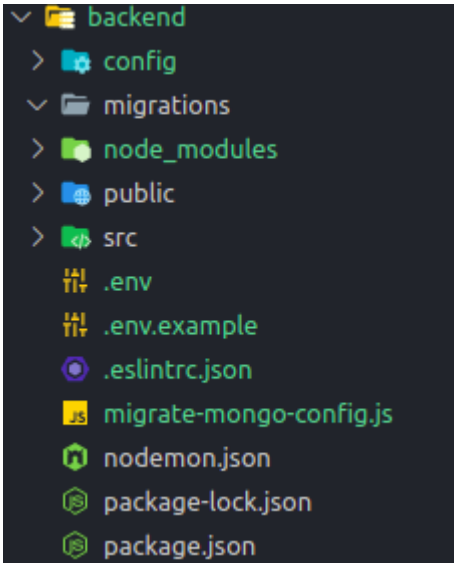


Рис. 3.2 – Структура backend.

Package.json для контролю використовуваних модулів, скрипти що дозволяють запускати дану частину проекту. Та Node_modules що має бібліотеки які не потрібні для інших частин проекту.

В даній частині проекту з’являється додатковий модуль Nodemon. Коли ми розробляємо програму серверу та вносимо деякі зміни, щоб побачити ці зміни, нам потрібно було перезапускати сервер. Але коли ми запускаємо серверну частину node.js за допомогою Nodemon, вона відстежує будь-які зміни та автоматично перезапускає сервер, підвищуючи нашу продуктивність.

Також тут з’являються .env файли, у яких зберігаються змінні навколишнього середовища, наприклад url front-end частини, що дозволяє розробляти серверну частину без статичних параметрів, а використовувати динамічні, які можуть відрізнятись для проекту який запускається не локально.

Окрім папок які вже були згадані, тут є ще config,migrations та public. Папка config має JSON файли, що зберігають конфігурації для FeathersJS, за замовчуванням вважається що їх є 3 default.json для локального запуску, production.json для конфігурації FeathersJS безпосередньо на сервері та test.json для налаштувань при тестуванні бекенду.

Папка migrations призначена для зберігання інтеграцій, які дозволяють видозмінювати базу даних, після того як схему потрібно змінити. Простіше кажучи, міграція бази даних – це серія сценаріїв або кроків, які змінюють схему бази даних відповідно до наших потреб. Оскільки дуже рідко схема ідеальна з самого початку проекту, ні, в процесі роботи її доводиться змінювати. Таким чином, міграція бази даних є загальноприйнятою передовою практикою для безпечного та простого оновлення схем бази даних.

Public містить статичні файли, які використовуватимуться FeathersJS безпосередньо в майбутньому. Зазвичай це favicon та index.html який відображатиметься при переході безпосередньо на URL сервера.

Повернемося до папки src та розглянемо її детальніше. Вона має в собі дві папки middleware та services, у middleware зберігаються функції проміжкового програмного забезпечення, що дозволяють обробляти дані до того як вони потраплять до наступного проміжкового по, або основної частини безпосередньо. Services зберігає набір сервісів що реалізують певні методи і допомагають нам взаємодіяти з іншими API, виконувати читання/запис до бази даних або ж взаємодії з файловою системою. В даній папці ми створюємо інші папки для кожного з сервісів, де описуємо кожен з них за допомогою 3 файлів це <назва сервісу>.class.ts, <назва сервісу>.hooks.ts, <назва сервісу>.service.ts.

Також src має такі файли як app.js, app.hooks.js, authentication.js, channels.js, index.js що використовуються для налаштування серверної частини, аутентифікації і тд.

3.1.3 Структура front-end частини

У фронт-енд частині, папці frontend є такі ж файли/папки про які було згадано раніше, та мають аналогічне призначення, проте значно відрізняється src. Тож давайте її розгянемо, всередині вона має такий набір папок: components, containers, customHooks, pages, redux, utils.

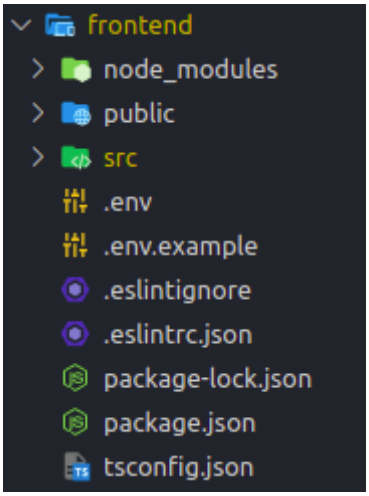


Рис. 3.2 – Структура frontend.

Components зберігає компоненти з яких складатиметься інтерфейс користувача, щось типу блоків з яких потім можна конструювати що завгодно. Containers це набір зібраних певним чином компонентів, можна провести аналогію зі стіною зібраною із блоків(компонентів), яку можна використовувати для створення інших об’єктів. CustomHooks набір хуків для React що розширює можливості React. Pages це папка з основними сторінками нашого користувацького інтерфейсу, в яких ми з контейнерів, та компонентів складаємо сторінку. У папці redux зберігаються налаштування сховища, та його частин, хуки. Utils зберігає утиліти для допомоги в обробці даних. Також в src є такі файли як App.tsx для зручності винесений компонент що рендериться в index.tsx, constants.ts для констант, feathersClient.ts для налаштування клієнтської частини FeathersJS, Icons.tsx для іконок, index.css з стилями що обнуляють стандартні стилі браузера, і задає деякі свої, та routes.ts з типами для роутів.

3.2 Створення користувацької частини

Розглнемо створення користувацької частини на прикладі створення вікна реєстрації/авторизації.

```
import { FC } from 'react';  5.1k (gzipped: 2.2k)

export interface IProps {
  signType: 'signIn'|'signUp';
}

export const Sign: FC<IProps> = (props) => {

  return (
    <div>
      <button>Sign up</button>
      <button>Sign in</button>
      <div>
        <input type="email" />
        <input type="password" />
      </div>
    </div>
  );
};
```

Рис. 3.3 – Створення компонента Sign

Дане вікно складатиметься з двох кнопок, що дозволять обрати тип входу на сайт, полів для введення пошти, пароля, та кнопки підтвердження. Для цього потрібно створити функціональний компонент Sign, що прийматиме в якості об’єкту props дані необхідні для створення цього компоненту, а повертатиме JSX (розширений синтакс JavaScript для React, що дозволяє легко створювати React-елементи) з потрібним нам вікном. Також зважаючи на те, що ми використовуємо TypeScript, необхідно задати вхідні параметри для нашого компоненту. Вони вказані в інтерфейсі IProps на рис 3.3.

```tsx
import { FC, useEffect, useState } from 'react';  5.2k (gzipped: 2.3k)
import { useNavigate, useLocation } from 'react-router-dom';  2.2k (gzipped: 1.1k)
import { route } from '../../routes';
import * as redux from '../../redux';
import { Inputs, IProps } from './Sign';


export const Sign:FC = () => {
  const authenticated = redux.hooks.useAppSelector(redux.storeParts.user.getAuthenticated);
  const navigate = useNavigate();
  const location = useLocation();
  const [signType, setSignType] = useState<IProps['signType']>('signUp');

  useEffect(() => {
    if (!authenticated || window.location.pathname !== '/sign-in') {
      return;
    }

    navigate(route('/scene'));
  }, [authenticated, navigate, location]);

  if (authenticated) return null;

  const handleSetSignUp = () => {
    setSignType('signUp');
  };
  const handleSetSignIn = () => {
    setSignType('signIn');
  };

  const isSignUp = signType === 'signUp';

  return (
    <div>
      <h2>Sign {isSignUp ? 'up' : 'in'}</h2>
      <button onClick={handleSetSignUp}>Sign Up</button>
      <button onClick={handleSetSignIn}>Sign In</button>
      <Inputs signType={signType}/>
    </div>
  );
};
```

Рис. 3.4 – Вдосконалення компонента Sign

Після цього поля для введення пошти та паролю краще винести в окремий компонент, який в залежності від типу входу матиме різну логіку авторизації. Проте щоб «прокинути» тип входу в компонент потрібно мати якусь змінну яка б його зберігала. Але після зміни звичайної змінної компонент Inputs не зміниться, оскільки React не знатиме що компонент треба перемалювати. В цьому випадку використовується state(стан) компонента, який відслідковуєтья React'ом і у випадку зміни компонент оновлюється. Для цього імпортуємо хук useState, який приймає початкове значення стану, а повертає вказівник на нього, та функцію яка цей стан змінює. Проте нам варто врахувати той факт, що користувач вже може бути авторизованим, для цього ми звертаємось в редакс, який зберігає дані клієнтської частини проекту щоб отримати статус авторизації. Далі перевіряємо чи авторизований користувач і чи правильний у нього роут, якщо користувач авторизований переадресовуємо його за роутом „/scene“ який є основною сторінкою сайту, якщо ж ні то залишаємось на сторінці авторизації. Проте цю дію задля уникнення помилок потрібно виконувати при кожній зміні статусу авторизації та змінних що використовуються, для цього варто застосувати useEffect хук, який першим аргументом приймає функцію, яку потрібно виконати, а другим масив від якого даний хук залежить. Якщо не задати другий параметр, useEffect виконуватиметься при кожній зміні стейту. Все що описано вище можна побачити на рис 3.4.

```
import { useEffect } from 'react';  5.2k (gzipped: 2.3k)
import {
  BrowserRouter as Router,
  Routes,
  Route,
} from 'react-router-dom';  8k (gzipped: 3.3k)
import { Signup } from './containers/Signup';
```

Рис. 3.5 – Імпортування компонентів для роутингу

Далі потрібно налаштувати роутинг на сайті. Це можна зробити за допомогою бібліотеки react-router-dom, а саме за допомогою компонентів BrowserRouter, Routes, Route. Для цього потрібно просто обгорнути Route в Routes та BrowserRouter. Приклад можна побачити на рис 3.6

```
const innerContent = authenticated === null
  ? null
  : (
    <Routes>
      <Route path={route('/sign-in')} element={<Signup />} />
      <Route path={route('/scene')} element={<pages.ScenePage />} />
    </Routes>
  );

return (
  <Router>
    {innerContent}
  </Router>
);
```

Рис. 3.6 – Реалізація роутингу на сайті

Даний роутинг вже реалізований у компоненті App, тому нам залишилося додати App до створеного додатку. Але так як ми використовуємо redux сховище, то до функції рендерингу React потрібно додати Provider з бібліотеки react-redux, який дозволить нам використовувати контекст сховища у всіх компонентах App.

```
import React from 'react';  7.3k (gzipped: 3k)
import ReactDOM from 'react-dom';  124k (gzipped: 40.4k)
import { Provider } from 'react-redux';  5.2k (gzipped: 2k)

import App from './App';
import store from './redux/store';


ReactDOM.render(
  <Provider store={store}>
    <App />
  </Provider>,
  document.getElementById('root'),
);
```

Рис. 3.7 – Використання компоненту App в додатку

3.3 Стилізація компонентів

Styled-components дозволяє використовувати стилі на рівні компонентів у додатках. Стилізовані компоненти використовують суміш JavaScript та CSS з використанням техніки, що називається CSS-in-JS. Styled-components засновані на літералах, що означає, що фактичний код CSS записується між зворотними лапками при стилізації компонентів. Іншими словами, коли ми визначаємо стилі, фактично створюємо компонент, до якого прив'язуємо стилі, використовуючи літерали шаблонів. При використанні styled-components немає потреби зіставляти створені компоненти із зовнішніми стилями CSS, єдине на що варто звернути увагу, це пріоритети під час компіляції. Даний тип стилізації є одним із найлегших, та найменш енергозатратним, оскільки не потребує ніяких специфічних знань, для створення стилів. Все що потрібно, це імпортувати styled компонент з бібліотеки styled-components далі ввести «styled.» за яким слідує допустимий компонент React або ім'я тега, наприклад «div». Або ж як показано на рис 3.8 для input та button

```
import styled from 'styled-components';

export const Input = styled.input`
  font-size: 17px;
  background: #1c1c1c;
  border-radius: 8px;
  padding: 16px 18px 16px 51px;
  color: rgb(220,220,220);
  outline: none;
  border: none;
`;

export const Button = styled.button`
  background: #d81a82;
  padding: 14px 52px;
  border-radius: 12px;
  color: #260618;
  font-weight: 700;
  font-size: 21px;
  cursor: pointer;
  outline: none;
  border: none;
  margin-top: 60px;
`;
```

Рис. 3.8 – Приклад створення стилізованих компонентів

Далі потрібно просто імпортувати дані стилізовані компоненти до компоненту який їх поєднає і все, стилі задані.

```
import * as st from './style';

export const Sign:FC = () => {
  return (
    <div>
      <st.Input/>
      <st.Button>Sign up</st.Button>
    </div>

  );
};
```

Рис. 3.9 – Імпорт і використання стилізованих компонентів

3.4 Створення сцени з 3D об’єктом

Почати варто з того, що імпортувати компонент <Canvas /> з бібліотеки «@react-three/fiber» і помістити його в наш компонент. Після цього потрібно створити <mesh/> який є основним елементом кожного об’єкта. Далі додамо до меша BoxGeometry та MeshStandardMaterial, які автоматично прикріплюються до батьківського елемента, тобто до <mesh/>. Повинен вийти компонент схожий до того що зображений на рис 3.10.

```
import { Canvas } from '@react-three/fiber';
import { FC } from 'react';  5.1k (gzipped: 2

export const Scene:FC = () => {
  return (
    <div id="scene-Wrapper">
      <Canvas>
        <mesh>
          <boxGeometry />
          <meshStandardMaterial />
        </mesh>
      </Canvas>
    </div>
  );
};
```

Рис. 3.10 – Реалізація сцени з об’єктом

Проте після використання такого компонента ми нічого не побачимо. Це все через те що нам потрібно освітити нашу сцену, для цього ми можемо використати <ambientLight intensity={0.1} />.

Детальніше можна дізнатись тут [1]. В результаті отримаємо компонент:

```
import { Canvas } from '@react-three/fiber';  116.6k (gzipped
import { FC } from 'react';  5.1k (gzipped: 2.2k)

export const Scene:FC = () => {
  return (
    <div id="scene-Wrapper">
      <Canvas>
        <ambientLight intensity={0.1} />
        <directionalLight color="red" position={[0, 0, 5]} />
        <mesh>
          <boxGeometry />
          <meshStandardMaterial />
        </mesh>
      </Canvas>
    </div>
  );
};
```

Рис. 3.11 – Повноцінна сцена з освітленням та 3D об’єктом

3.5 Використання FeathersJS

Для використання сайту користувач повинен зареєструватись, або аутентифікуватись, для цього використаємо всі можливості FeathersJS. Для початку потрібно налаштувати клієнтську частину FeathersJS, для цього потрібно створити файл feathersClient.ts з якого експортувати змінну feathersClient яка буде використовуватись для створення запитів на сервер.

Для цього імпортуємо feathers з «@feathersjs/feathers» і призначаємо цю функцію для feathersClient, для того щоб вказати шлях до серверу імпортуємо rest з «@feathersjs/rest-client» і для цієї функції передаємо силку до нашого сервера. Проте потрібно поєднати результат виконання функції rest з feathersClient, для цього у feathersClient існує метод configure, в який ми можемо передати цей результат, також в configure передамо функцію auth з «@feathersjs/authentication-client», що дозволить нам простіше авторизуватись.

```
import feathers from '@feathersjs/feathers';  18.8k (gzippec
import rest from '@feathersjs/rest-client';  30.5k (gzipped:
import auth from '@feathersjs/authentication-client';  20k (

import { BACKEND_URL } from './constants';


export const feathersClient = feathers();

const restClient = rest(BACKEND_URL);

// Set up Rest client
feathersClient.configure(restClient.fetch(window.fetch));
feathersClient.configure(auth());
```

Рис. 3.12 – Реалізація feathersClient

На сервері потрібно створити файл в якому ми налаштуємо можливі способи авторизації, для цього імпортуємо AuthenticationService з «@feathersjs/authentication» це клас який в якості аргумента приймає Express додаток, та має метод register в який можна додати різного роду стратегії авторизації, такі як авторизація з JWT токеном, за допомогою логіна та пароля, за допомогою Google, Git акаунтів.

```
import { ServiceAddons } from '@feathersjs/feathers';  18.8k (gzipped: 6.6k)
import { AuthenticationService, JWTStrategy } from '@feathersjs/authentication';
import { LocalStrategy } from '@feathersjs/authentication-local';  141.6k (gzipp
import { expressOauth } from '@feathersjs/authentication-oauth';

import { Application } from './declarations';

declare module './declarations' {
  interface ServiceTypes {
    'authentication': AuthenticationService & ServiceAddons<any>;
  }
}

export default function(app: Application): void {
  const authentication = new AuthenticationService(app);

  authentication.register('jwt', new JWTStrategy());
  authentication.register('local', new LocalStrategy());

  app.use('/authentication', authentication);
  app.configure(expressOauth());
}
```

Рис. 3.13 – Реалізація authentication на сервері

Створення нового користувача реалізуємо в сервісі users.service.ts у якому в якості вхідних параметрів буде Express додаток, який ми модифікуємо, а також додамо для нього хуки feathers. Для цього у вхідного app(Express додаток) використаємо метод use для якого як вхідні параметри задамо роут „/users“, та клас, що відповідатиме за обробку запитів. Використовуючи метод hooks у сервісу app, додамо хуки на before та after запиту, які дозволять обробляти дані до передачі безпосередньо до сервісів.

```ts
import { ServiceAddons } from '@feathersjs/feathers';
import { Application } from '../../declarations';
import { Users } from './users.class';
import hooks from './users.hooks';

// Add this service to the service type index
declare module '../../declarations' {
  interface ServiceTypes {
    'users': Users & ServiceAddons<any>;
  }
}

export default function (app: Application): void {
  const options = {
    paginate: app.get('paginate'),
  };

  app.use('/users', new Users(options, app));

  const service = app.service('users');

  service.hooks(hooks);
}
```

Рис. 3.13 – Реалізація сервісу users для запитів

Висновки до розділу 3

В даному розділі ми розглянули всі етапи створення проекту, на кожному зупинилися детально, та розглянули на прикладах. Також згадали про структуру проекту і кожної з частин проекту окремо, розписав за що відповідають основні файли, та яка їх роль в цьому проекті. Згадали про більшість фреймворків використаних у даному проекті, на скріншотах було вказано приклади виконання тих чи інших завдань.

Розділ 4

Тестування та візуальна складова

4.1 Сторінка реєстрації

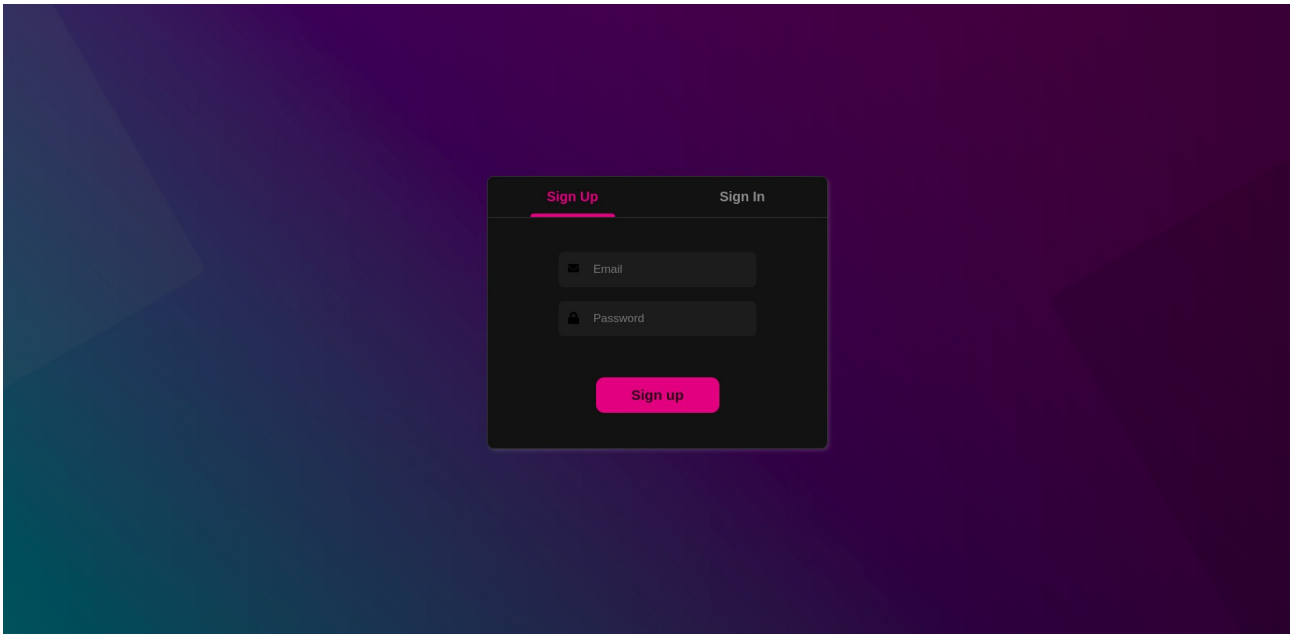


Рис. 4.1 – Сторінка реєстрації

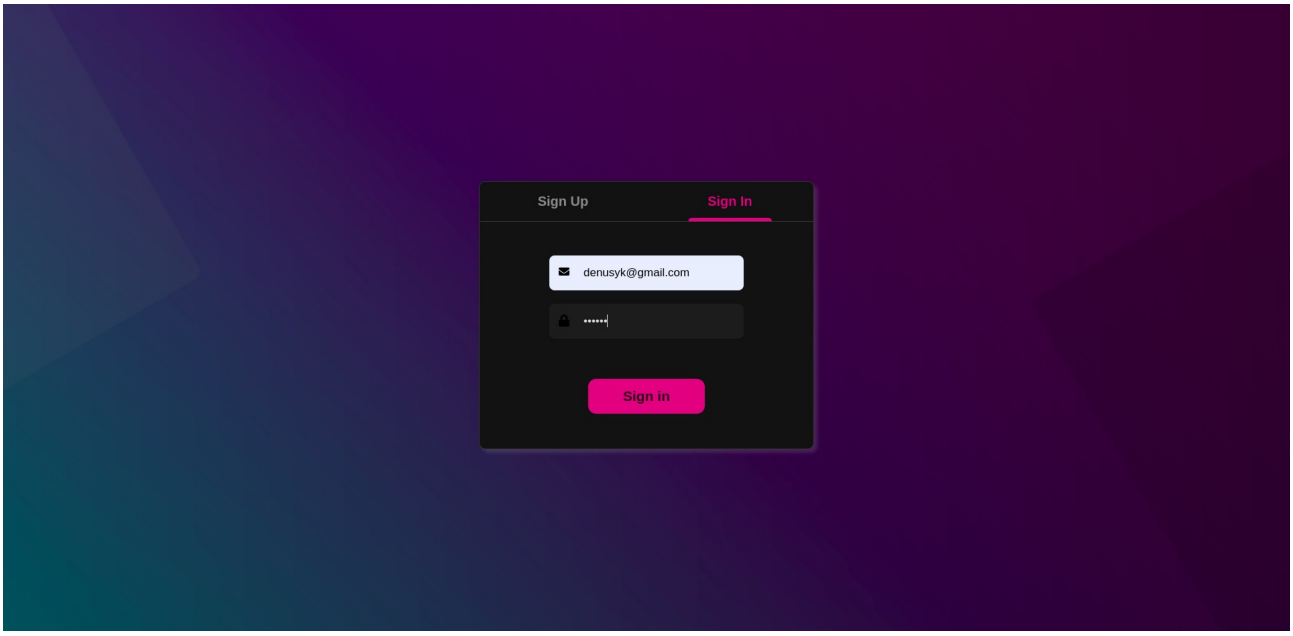


Рис. 4.1 – Сторінка авторизації з заповненими полями

4.2 Сцена із 3D об’єктами

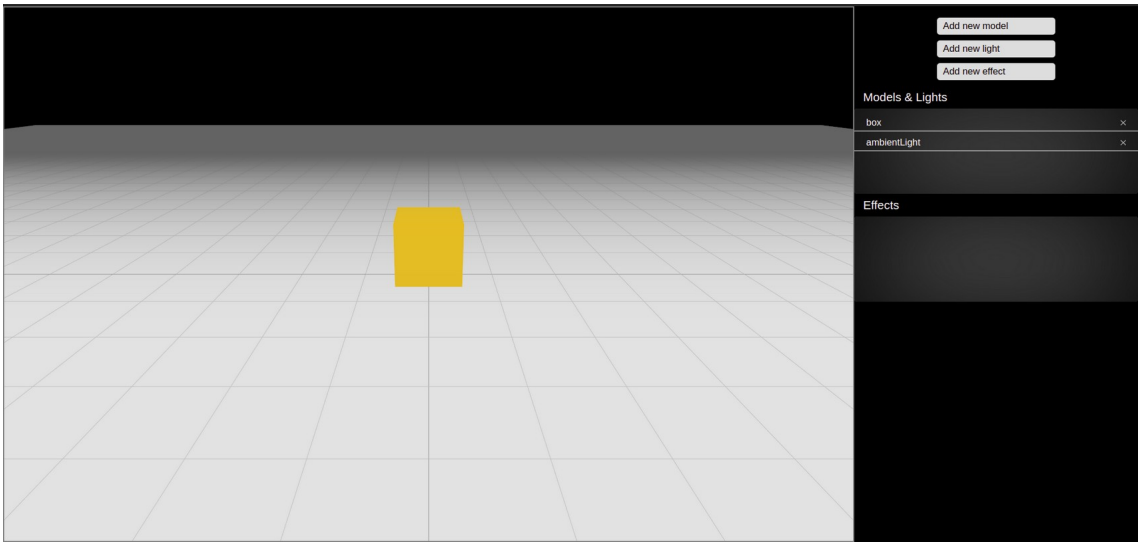


Рис. 4.2 – Початкова сторінка після реєстрації

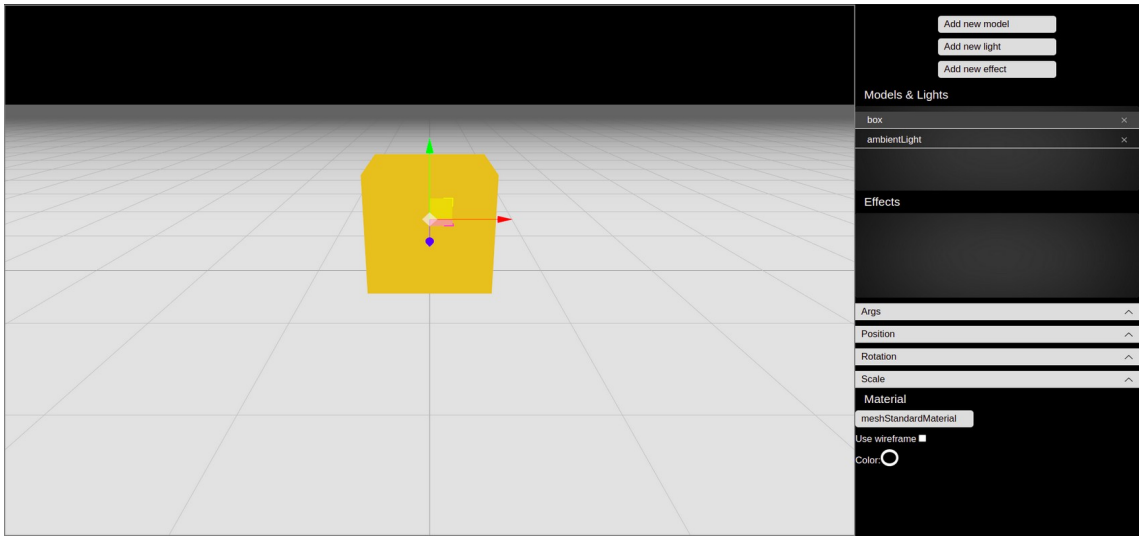


Рис. 4.3 – Вигляд сторінки після того як обрали один із об’єктів

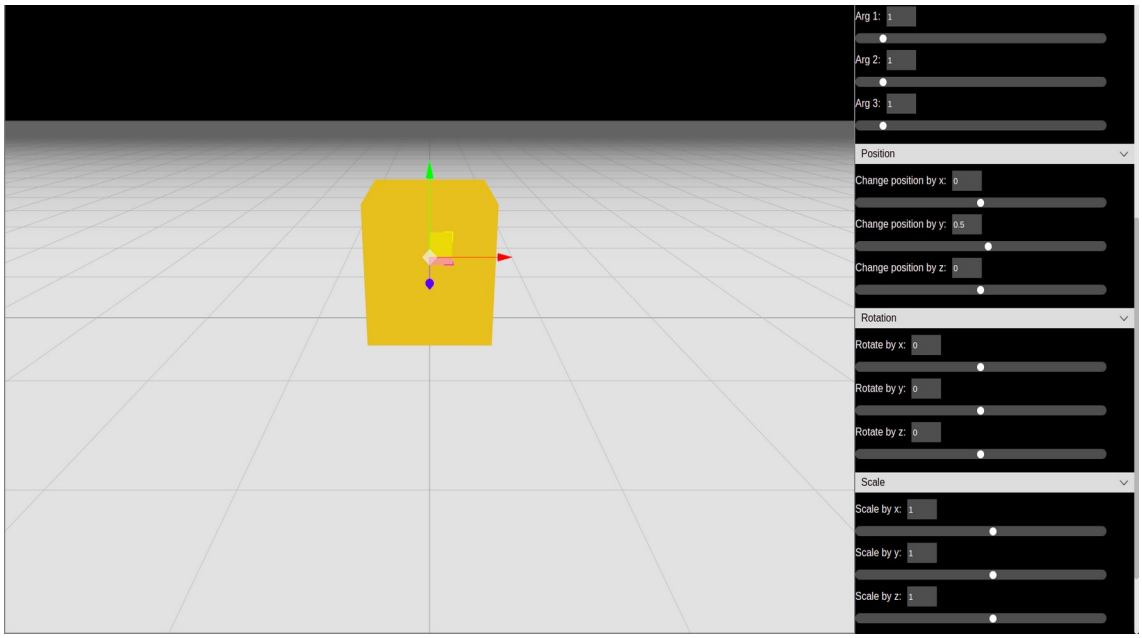


Рис. 4.4 – Вигляд сторінки з параметрами які можна змінити у об’єкта

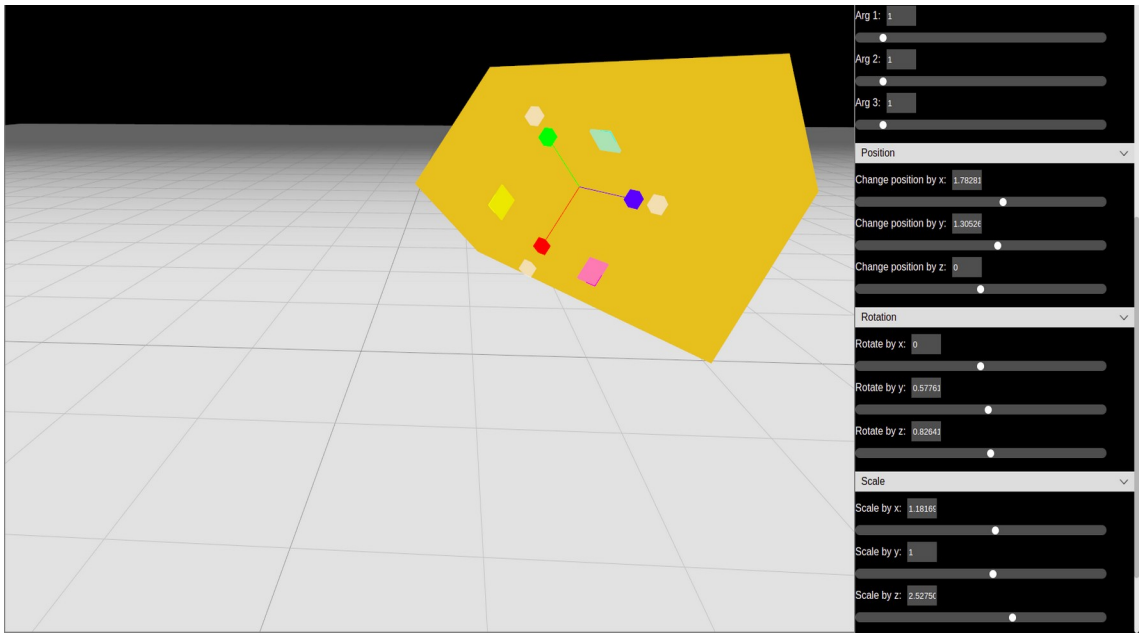


Рис. 4.5 – Вигляд сторінки з зміненими параметрами об’єка

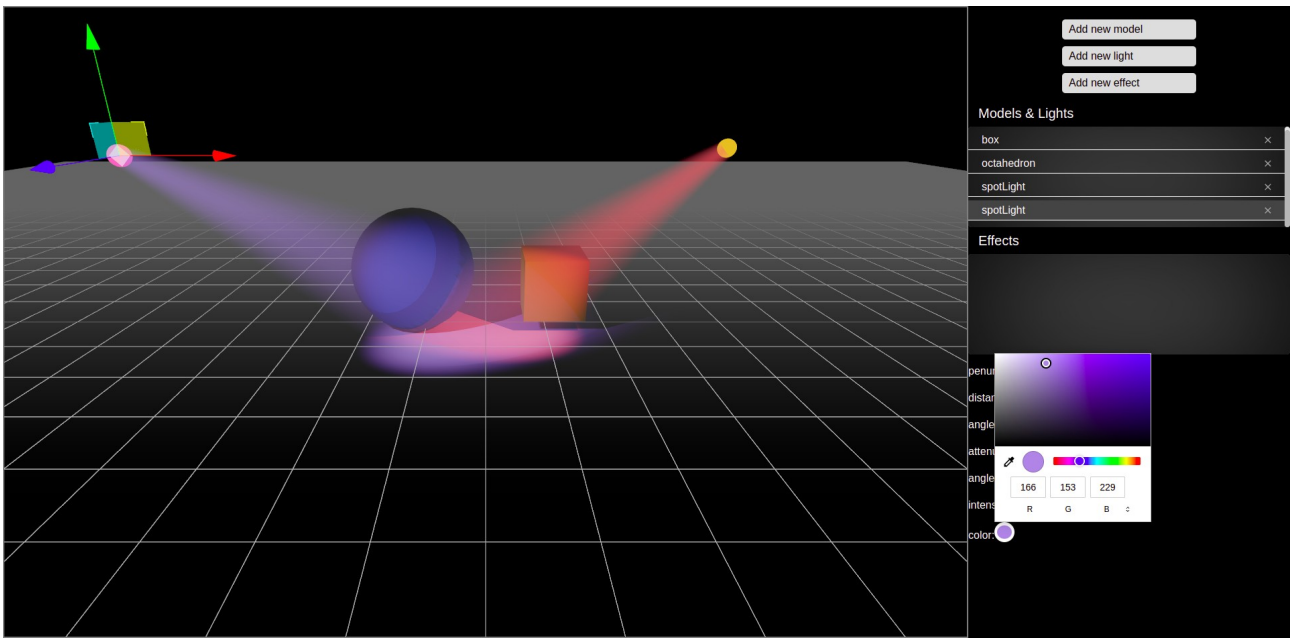


Рис. 4.6 – Вигляд сторінки без основного освітлення, а з використанням ламп

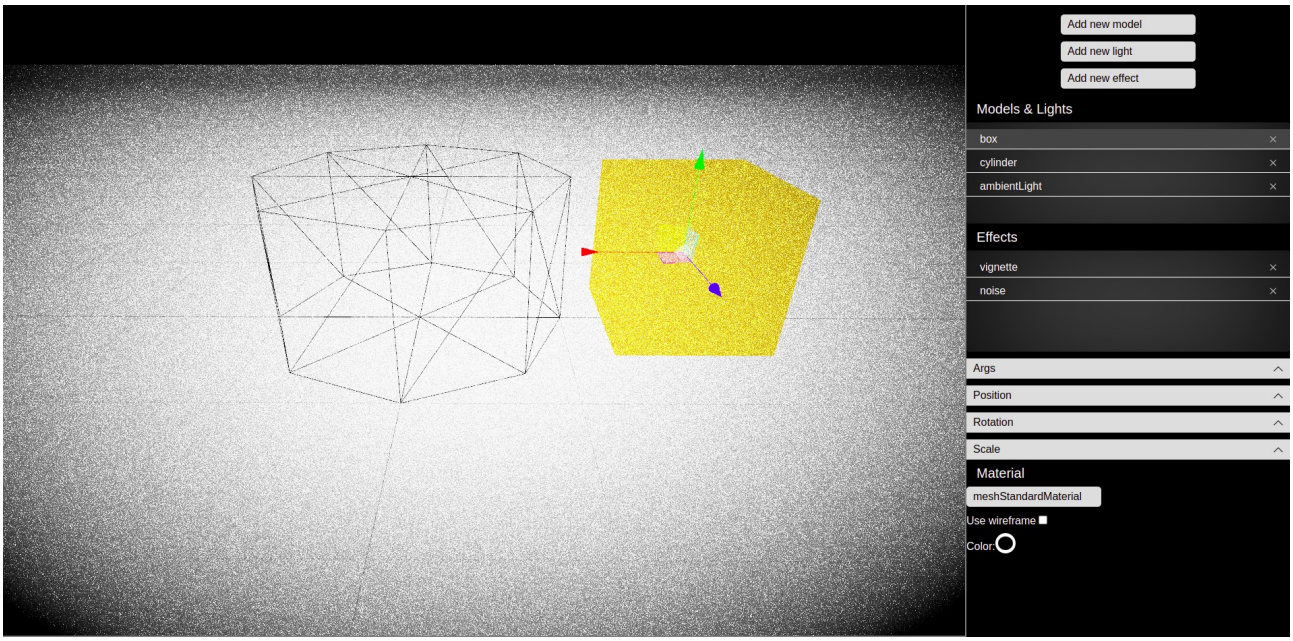


Рис. 4.7 – Вигляд сторінки з використанням ефектів Вінетка та Шум

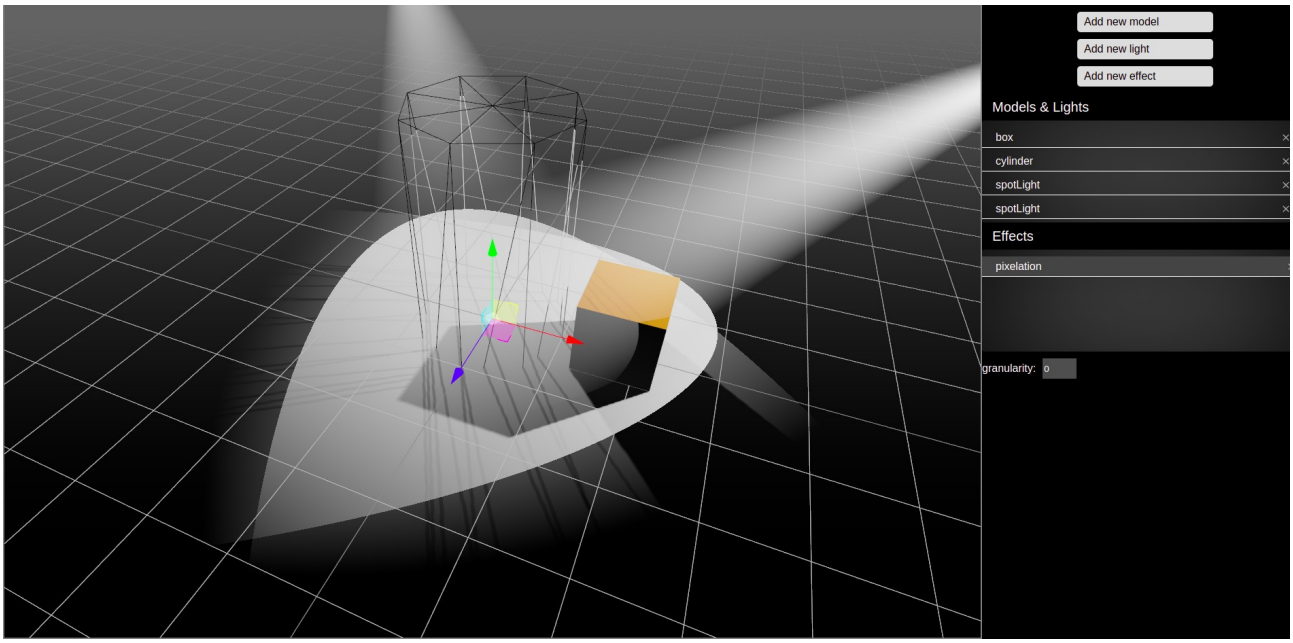


Рис. 4.8 – Сцена до використання ефекту Пікселізація

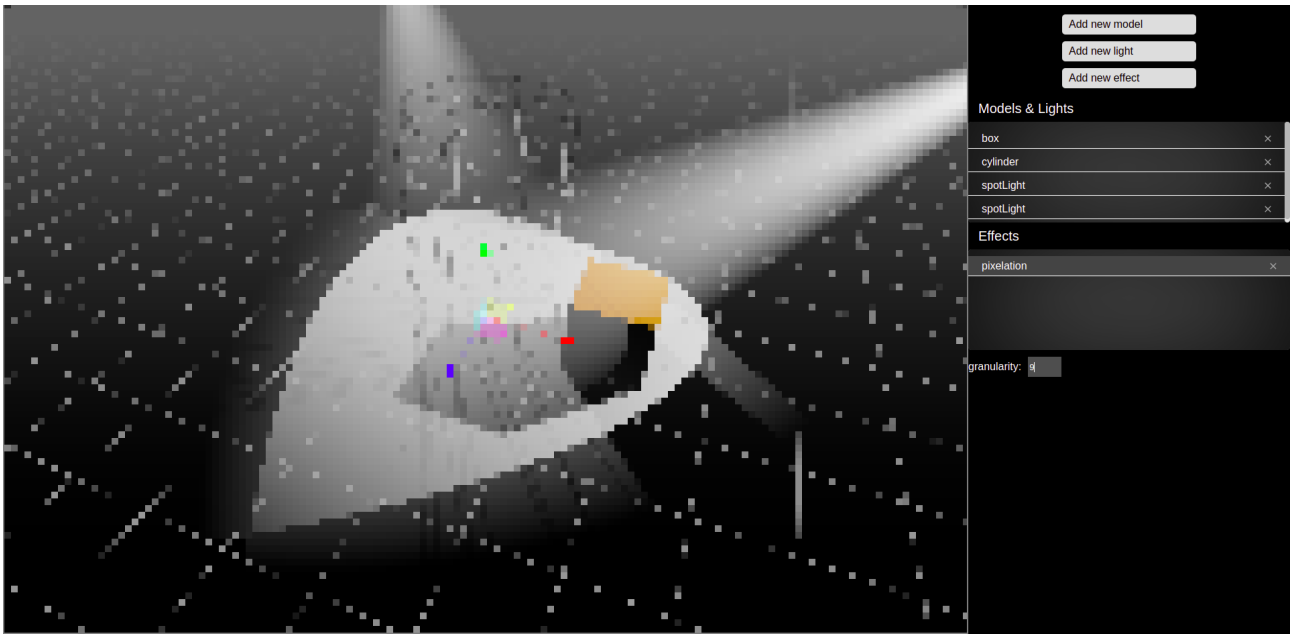


Рис. 4.9 – Сцена після використання ефекту Пікселізація

Висновок до розділу 4

В цьому розділі ми наглядних прикладах розглянули як працює застосунок, було показано скріншоти з реєстрації, входу в особистий обліковий запис. Розглянули приклади роботи додатку з різними об’єктами, з різного роду освітленням, та ефектами пострендерингу. Було показано параметри які ми можемо змінювати, та те як ці параметри впливають на об’єкти, та сцену в цілому.

Висновки

В результаті виконання даного дипломного проєкту було створено сайт, що найбільше підійде для розробників, та творчих людей що працюють з 3D моделюванням. Він дає змогу створювати 3D об’єкти навіть не використовуючи допоміжних редакторів, у ньому можна різним чином змінювати та поєднувати 3D об’єкти, освітлення та ефекти пострендерингу.

У першому розділі було описано історію створення, та розвитку 3D-технологій. Розглянуто базову теорію 3D-моделювання, деякі стандарти, технології що дозволяють користувачам працювати з 3D-моделюванням. Оглянули перспективи розвитку даної технології.

У другому розділі було розглянуто технології які можуть допомогти у створенні веб додатку для візуалізації 3D-об’єктів, їх переваги та недоліки. У ньому ми дослідили моно та мульти репозиторії, описано основні плюс бібліотек та мов програмування, обрано стек для подальшої розробки проекту.

У третьому розділі було використано і детальніше описано фреймворки, бібліотеки які ми обрали у другому розділі. Розглянули базову структуру проекту, кожної з частин проекту та створення деяких елементів у кожній частини проекту. Було показано створення сцени з базовим 3D об’єктом, та освітленням. Налаштовано реєстрацію та авторизацію.

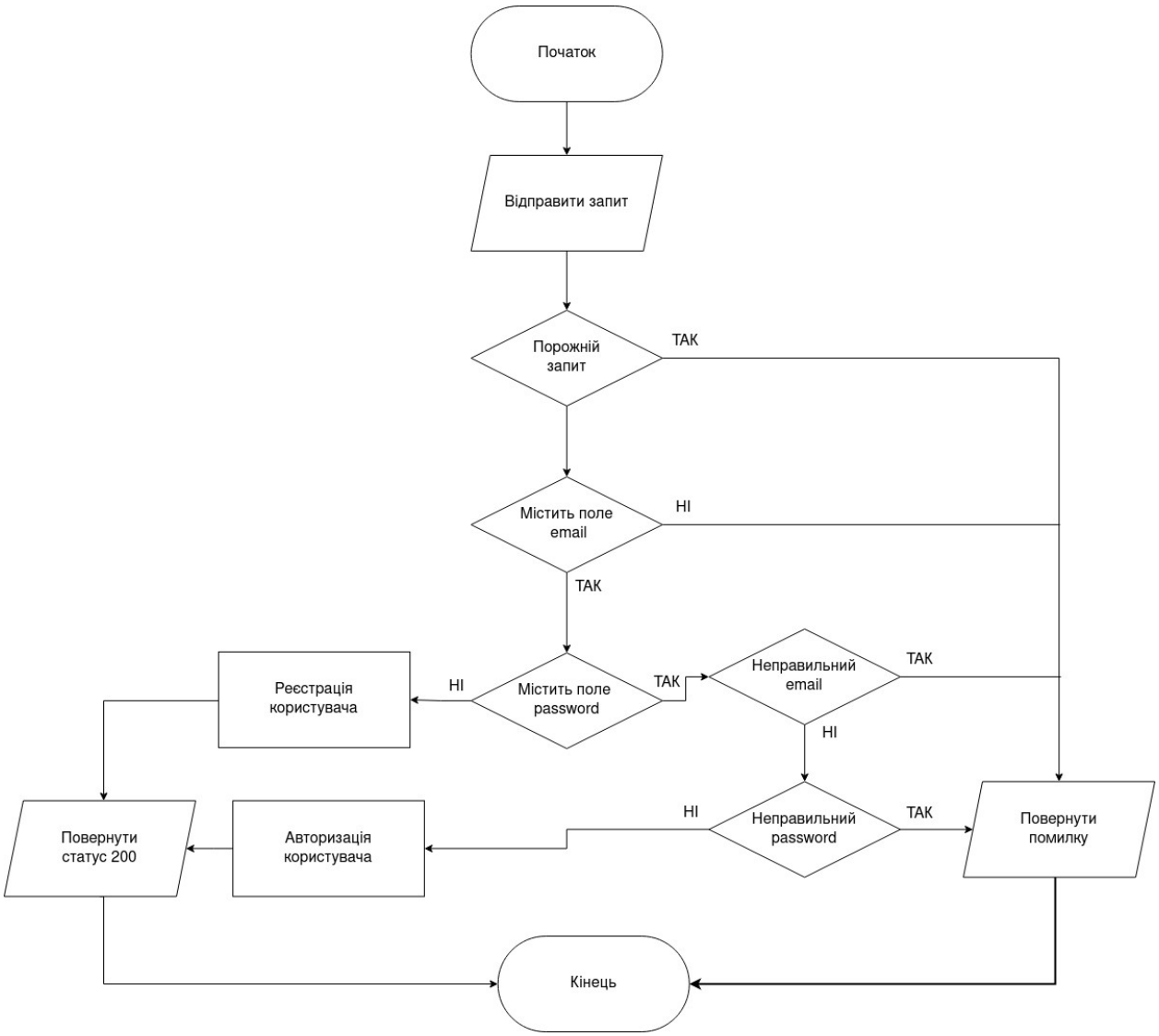
У четвертому розділі на наглядних прикладах було розглянуто як працює застосунок, було показано скріншоти з реєстрації, входу в особистий обліковий запис. Розглянуто приклади роботи додатку з різними об’єктами, різного роду освітленням, та ефектами пострендерингу.

Загалом створений проект є простим і зручним у використанні і надає користувачу великі можливості у творчості, або ж тестуванні власних моделей.

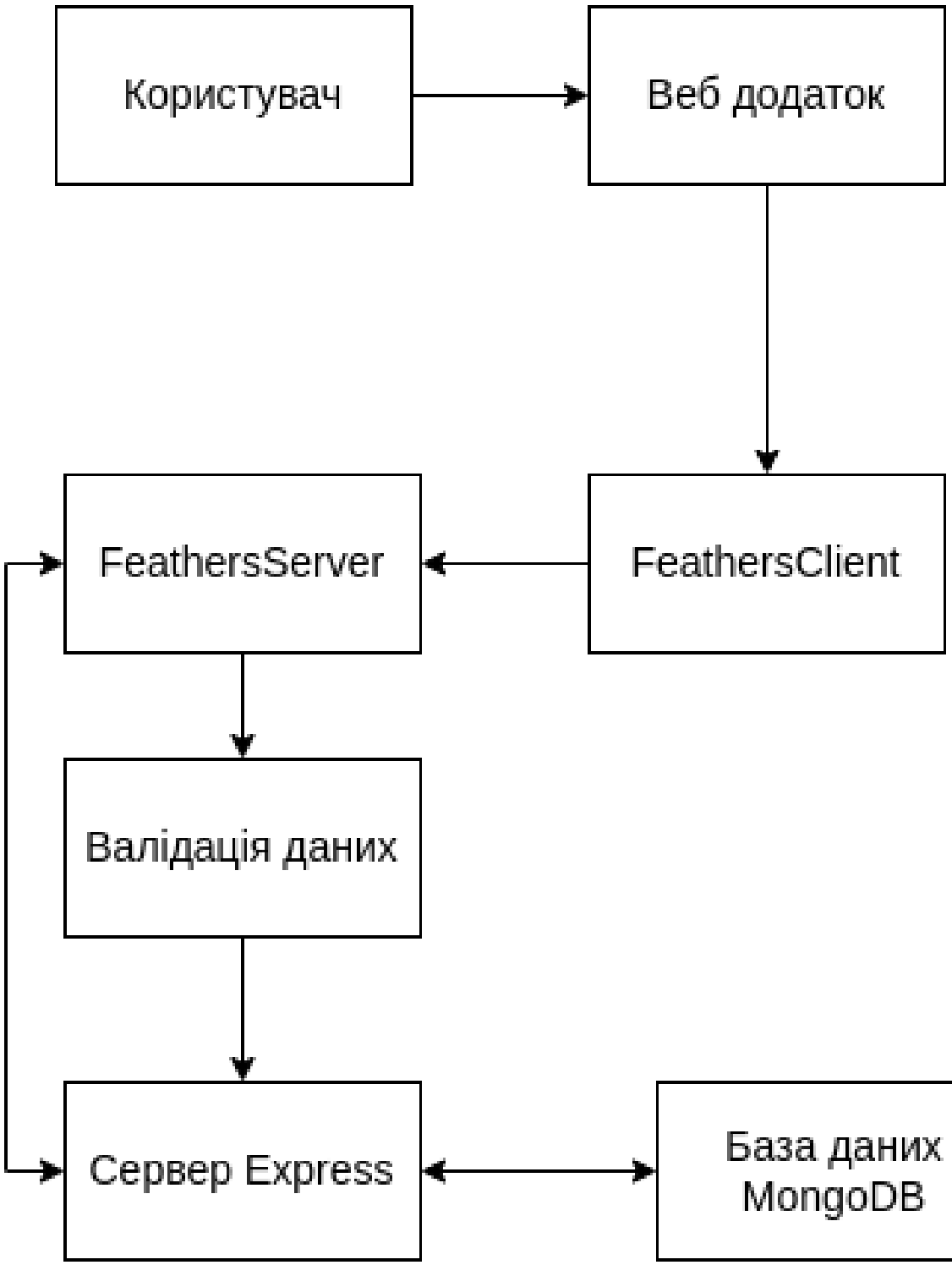
СПИСОК ВИКОРИСТАНОЇ ЛІТЕРАТУРИ

1. React-three-fiber [Електронний ресурс] – Режим доступу до ресурсу: https://docs.pmnd.rs/react-three-fiber/getting-started/introduction
2. Kathryn Brown. The Routledge Companion to Digital Humanities and Art History — 2020.
3. Theo Zizka. 3D Modeling — 2014
4. Direct3D 11.3 Functional Specification [Електронний ресурс] – Режим доступу до ресурсу: https://microsoft.github.io/DirectX-Specs/
5. DirectX Programming [Електронний ресурс] – Режим доступу до ресурсу: https://docs.microsoft.com/
6. Dan Ginsburg, Budirijanto Purnomo. OpenGL ES 3.0. Руководство разработчика — 2022. — 23с.
7. OpenGL [Електронний ресурс] – Режим доступу до ресурсу: https://www.opengl.org/
8. WebGL [Електронний ресурс] – Режим доступу до ресурсу: https://www.khronos.org/webgl/
9. Autodesk Maya [Електронний ресурс] – Режим доступу до ресурсу: https://www.autodesk.com/
10. Gordon Fisher. Blender 3D Basics Beginner's Guide — 2014.
11. Guide to Monorepos [Електронний ресурс] – Режим доступу до ресурсу: https://www.toptal.com/front-end/guide-to-monorepos
12. Vue vs React in 2022 [Електронний ресурс] – Режим доступу до ресурсу: https://www.monterail.com/blog/vue-vs-react
13. Benefits of Using Express.js for Backend Development [Електронний ресурс] https://www.techomoro.com/what-are-the-benefits-of-using-express-js-for-backend-development/
14. FeathersJS [Електронний ресурс] – Режим доступу до ресурсу: https://feathersjs.com/

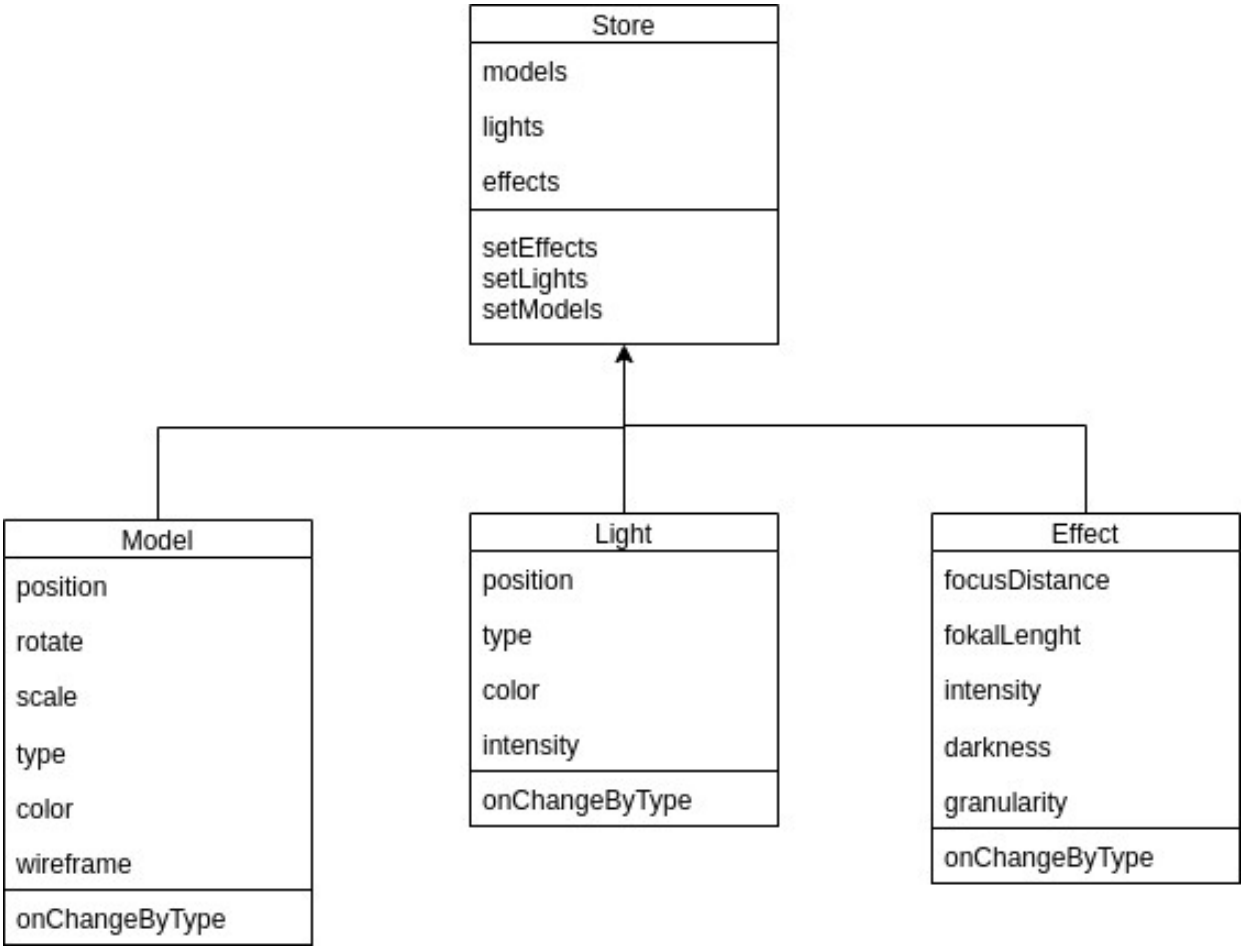
ДОДАТКИ



					ІАЛЦ.467800.004.Д1		
Зм.	*Арк.*	*№ докум.*	*Підпис*	*Дата*			
Разробив		*Денисюк Я.В.*			*Інтерактивна система візуалізації 3D об'єктів Авторизація користувача блок схема*	*Літ.*	*Аркуш* 1 / *Аркушів* 1
Керівник		*Ткаченко В.В.*					
Реценз.							
Н. Контр.							
Затв.		*Стіренко С.Г.*					



					ІАЛЦ.467800.005.Д2	
Зм.	Арк.	№ докум.	Підпис	Дата		
Разробив		Денисюк Я.В.			Інтерактивна система візуалізації 3д об'єктів Схема роботи програми	Літ. / Аркуш 1 / Аркушів 1
Керівник		Ткаченко В.В				
Реценз.						НТУУ «КПІ імені Ігоря Сікорського», ФІОТ Група ІВ-81
Н. Контр.						
Затв.		Стіренко С.Г.				



					ІАЛЦ.467800.006.Д3		
Зм.	Арк.	№ докум.	Підпис	Дата			
Разробив		Денисюк Я.В.			Інтерактивна система візуалізації 3D об'єктів Структурна схема бази даних	Літ.	Аркуш 1 / Аркушів 1
Керівник		Ткаченко В.В.					
Реценз.						НТУУ «КПІ імені Ігоря Сікорського», ФІОТ Група ІВ-81	
Н. Контр.							
Затв.		Стіренко С.Г.					