Форма № 24

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Національний університет «Запорізька політехніка»

Інститут інформатики та радіоелектроніки,
Факультет радіоелектроніки та телекомунікацій
(повне найменування інституту, факультету)

Кафедра інформаційні технології електронних засобів
(повне найменування кафедри)

Пояснювальна записка

до дипломного проекту (роботи)
Бакалавр
(ступінь вищої освіти)

на тему «РОЗРОБКА ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ ДЛЯ АНІМАЦІЇ 3D-МОДЕЛЕЙ»

«SOFTWARE FOR 3D MODEL ANIMATION»

Виконав: студент(ка) _4_ курсу, групи РТ-618сп

Спеціальності 151 «Автоматизація та комп'ютерно-інтегровані технології»
(код і найменування спеціальності)

Освітня програма (спеціалізація)

Довбиш Д.О.
(прізвище та ініціали)

Керівник Шило Г. М.
(прізвище та ініціали)

Рецензент Зеленьова І.Я.
(прізвище та ініціали)

2021

Форма № 25

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Національний університет «Запорізька політехніка»
(повне найменування закладу вищої освіти)

Інститут, факультет Інституту інформатики та радіоелектроніки,
Факультет радіоелектроніки та телекомунікацій
Кафедра Інформаційних технологій електронних засобів
Ступінь вищої освіти Бакалавр
Спеціальність 151 «Автоматизація та комп'ютерно-інтегровані технології»
(код і найменування)
Освітня програма (спеціалізація) _____
(назва освітньої програми (спеціалізації)

ЗАТВЕРДЖУЮ
в.о. Завідувача кафедри кандидат технічних наук Огренич Є.В.
«31» 05 2021 року

З А В Д А Н Н Я
НА ДИПЛОМНИЙ ПРОЄКТ (РОБОТУ) СТУДЕНТА(КИ)

Довбиш Дмитро Олександрович
(прізвище, ім'я, по батькові)

1. Тема проекту (роботи) Розробка програмного забезпечення для анімації 3D-моделей
керівник проекту (роботи) Шило Галина Миколаївна, доктор технічних наук, доцент
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)
затверджені наказом закладу вищої освіти від « 26 » квітня 2021року № 159
2. Строк подання студентом проєкту (роботи) 2021
3. Вихідні дані до проєкту (роботи) програма для 3D моделювання

4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити)
1. Аналіз технічного завдання
2. Проєктування та реалізація програмного забезпечення для 3d-моделювання
3. Аналіз отриманих результатів
16 рисунків.

6. Консультанти розділів проекту (роботи)

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	прийняв виконане завдання
Розділи 1-3	Шило Г. М., доц.		
Нормоконтроль	Поспеєва І. Є., ст. викладач	02.06.21	

7. Дата видачі завдання «___» _____2021 року.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів дипломного проекту (роботи)	Строк виконання етапів проекту (роботи)	Примітка
1	Аналіз поставленої задачі.	1 тиждень	Виконано
2	Розробка програми	2-3 тиждень	Виконано
3	Пошук необхідної інформації для реалізації функцій програми.	4-8 тижні	Виконано
4	Оформлення пояснювальної записки	8-9 тижні	Виконано
5	Оформлення супровідної документації	10-11 тижні	Виконано
6	Нормоконтроль та рецензування	12-13 тижні	Виконано
7	Захист роботи	14 тиждень	Виконано

Студент _____ Довбиш Д.О
(підпис) (прізвище та ініціали)

Керівник проекту (роботи) _____ Шило Г. М.
(підпис) (прізвище та ініціали)

ЗМІСТ

Вступ

Комп'ютерна анімація сьогодні зазвичай використовується в різних областях, таких як кіноіндустрія, індустрія відеоігор і архітектура. Це призвело до збільшення попиту на інструменти 3D-моделювання, які можна використовувати для створення цих анімацій. Таким чином, мета цього проекту - вивчити, які міркування необхідно враховувати при розробці програмного забезпечення для інтерактивного 3D-моделювання.

Це було вивчено шляхом проведення досліджень в цій області, включаючи існуючі інструменти тривимірного моделювання та інших пов'язаних робіт, а також розробки і впровадження інтерактивного засобу тривимірного моделювання відповідно до досліджень, проведеними в практиці проектування. Цей інструмент був спроектований для створення типових анімації м'яких тіл. Спочатку інструмент був розроблений з використанням ітеративного процесу проектування, включаючи різні методи, від створення ідей до оцінки. Потім програмна частина була розроблена з використанням поєднання популярних гнучких методів розробки програмного забезпечення. Код був розроблений з уклоном на забезпечення оптимальної продуктивності, просування модульності програмного забезпечення та мінімізацію зовнішніх залежностей.

Кінцевим результатом є високоточний прототип програмного забезпечення, який можна використовувати для створення анімації різної складності. В результаті розробки інструменту, поряд з проведеними дослідженнями, був розроблений ряд керівних принципів, які слід враховувати при створенні аналогічних інструментів. Ці рекомендації можна поділити в такий спосіб як: зовнішні бібліотеки, який робочий процес використовувати, які функції повинні бути доступні, як проектувати для різних груп користувачів і як зробити код модульним. Незважаючи на рекомендації, що випливають з роботи цього конкретного інструменту, вони

вважаються застосовними до розробки програмного забезпечення для 3D-моделювання, яке обробляє будь-які види анімації м'яких тіл.

Комп'ютерні зображення (CGI) - це зображення, зроблене за допомогою комп'ютерної графіки. Сьогодні CGI можна знайти в усіх напрямках; у відеоіграх, фільмах або навіть в рекламі. У кіно з допомогою комп'ютерної графіки вже якийсь час вдається досягати результатів, близьких до фотореалістичних. Однак нові методи все ще розробляються з метою створення аналогічних ефектів з інтерактивною швидкістю, наприклад, для використання в відеоіграх і інших інтерактивних додатках. CGI можна розділити на одну з наступних категорій: 3D-моделювання, комп'ютерна анімація і рендер.

У 1960 році був випущений перший фільм з використанням комп'ютерної анімації у вигляді 49-секундної векторної 2D-анімації. Десять років по тому, на початку 70-х, в Університеті Юти були розроблені різні методи створення 3D-анімації. До середини 70-х був випущений перший фільм з використанням комп'ютерної анімації, що поклав початок комп'ютерної анімації в державних засобах масової інформації ЗМІ. На початку 80-х років анімація на основі фізики почала з'являтися в основних ЗМІ в таких фільмах, як «Зоряний шлях 2: Гнів Хана».

Анімація на основі фізики - це підполе в області комп'ютерної анімації, яка фокусується на досягненні візуально правдоподібних фізичних ефектів з інтерактивною швидкістю. Сюди входять такі ефекти, як гравітація і зіткнення. Загалом, є три різних типи об'єктів, які слід враховувати в анімації на основі фізики: тверді тіла, м'які тіла і рідини.

Тверде тіло - це твердий об'єкт, який не деформується, що означає, що форма об'єкта не може змінитися. Це означає, що всі точки на об'єкті постійні по відношенню один до одного, незважаючи на потенційні зовнішні сили.

М'яке тіло - це твердий об'єкт, який піддається деформації, що означає, що форма об'єкта може змінюватися. Іншими словами, всі крапки на об'єкті

діють індивідуально, і тому відносна відстань між двома точками може змінитися.

Рідина - це нетвердий об'єкт, наприклад вода, газ або плазма. У порівнянні з м'яким тілом не очікується, що рідина збереже свою форму.

Хоча моделювати тверді тіла легко, м'які тіла набагато складніше через значно збільшеного числа можливих станів об'єкта. Проте, заснована на фізиці анімація за участю м'яких тіл широко використовується в відеоіграх і фільмах для створення спецефектів. Це призводить до того, що це активна область досліджень, присвячена різним методам обчислень для їх апроксимації. Швидкий розвиток апаратного забезпечення ще більше просунули цю область; обчислення, які було важко виконувати лише кілька десятиліть тому, такі як анімація рухів персонажа, тепер вважаються тривіальними. Це розширило використання анімації на основі фізики в багатьох інших додатках, таких як системи хірургічного моделювання та програмне забезпечення для автоматизованого проектування.

У зв'язку зі зростаючою необхідністю мати правдоподібні анімації в різних типах додатків, також зростає потреба в інструментах, що полегшують їх створення. Дизайнеру анімації має бути дозволено створювати їх і управляти ними, не вимагаючи особливих технічних знань. В даний час існує безліч різних додатків, які підходять під цей опис, кожне з яких має свій набір функцій і недоліків.

Компанія Deform Dynamics розробила комплект розробника програмного забезпечення (SDK) для створення реалістичних анімацій м'яких тіл на основі решателя Vivace, розробленого Марко Фратарканджелі, Валентиною Тібальда і Фабіо Пеллачіні. Цей вирішувач створює стабільну анімацію з надзвичайно високою швидкістю, що робить його придатним для інтерактивних додатків. Таким чином, мета цього проекту - Розробка програмного забезпечення для аінмації 3D-моделей.

Загальна мета цього проекту - полегшити створення і управління правдоподібними анімаціями м'яких тіл. Метою дипломної роботи є розробка і реалізація інструменту для цього. Оскільки створення повноцінного інструменту 3D-моделювання - складне завдання, зазвичай вимагає багатьох років роботи, очікуваним результатом є робочий прототип, який може відтворювати демонстрації, створені раніше Deform Dynamics.

1 АНАЛІЗ ТЕХНІЧНОГО ЗАВДАННЯ

1.1 Дослідження через дизайн

Дослідження через дизайн - це практика розробки дизайну як засобу вирішення дослідницької мети. У порівнянні з традиційними науковими областями мета не полягає в тому, щоб спростовувати теорії, скоріше, мова йде про вироблення загальних рекомендацій, які можна використовувати для вирішення аналогічних проблем проектування. Рекомендації засновані на знаннях, отриманих в процесі розробки дизайну.

3D-анімація широко застосовується в різних сферах, не тільки при створенні фільмів або відеоігор. 3D програми і обладнання використовується в різних індустріях і сферах діяльності. У кожній сфері 3D-анімації використовується абсолютно різним чином. Цілі і підхід до її реалізації сильно відрізняються. Наука і розваги - основні споживачі 3D-анімації та візуалізації. До науки відносяться вузькоспеціалізовані проекти, мета яких візуалізація будь-яких наукових процесів. Наприклад, 3D-анімацію використовують у фізиці, медицині і криміналістиці. У кіновиробництві та ігрової індустрії 3D-анімація використовується для створення персонажів, моделювання сцен, створення футуристичних або історичних світів і спецефектів. Це ціла окрема індустрія, в якій зайняті тисячі 3D-фахівців. Кіноіндустрія - найбільша індустрія, яка використовує 3D-анімацію. Фільми можуть бути повністю зроблені в 3D, так і з 3D-візуальними ефектами. Фільми з використанням візуальних ефектів відрізняються від повністю анімованих тим, що їх знімають в павільйоні, а потім зняті кадри потрапляють в руки майстрів візуальних ефектів і вони додають необхідні ефекти і елементи. Як і в кіновиробництві, є ряд телевізійних продуктів, які повністю анімовані. Для телебачення, створення таких програм, дорогий і тривалий процес. Наприклад, американське дитяче шоу Mickey Mouse Clubhouse вироблено повністю в 3D-програмах. Телевізійники частіше використовують 3D-анімацію і візуалізацію

для доповнення передач ефектами або створення заставок. Телебачення не має такими великими бюджетами та тривалими тимчасовими рамками, як кіноіндустрія. Телевізійні програми знімають і монтують за дні або тижні, але ніяк не за роки, але 3D-анімація повинна виглядати також добре. Прибуток від виробництва відеоігор значна і вимірюється сотнями тисяч, а іноді і мільйонами доларів. За прибутковістю відеоігри можуть конкурувати хіба що з кіновиробництвом. 3D-анімація використовується для створення ігрових світів, в які потрапить гравець. Відеоігри близькі до кіноіндустрії, тільки 3D-дизайнери працюю набагато швидше. Мобільні ігри розробляються швидко, а ось виробництво відеоігри може зайняти більше часу від 2 до 3 років. Створення всіх персонажів і ігрового світу процес трудомісткий, але дуже захоплюючий. Зазвичай над виробництвом 3D-персонажів для комп'ютерних ігор працюють і ігрового оточення працюють люди, які остаточно і безповоротно закохані в свою роботу і можуть годинами безперервно займатися улюбленою справою. Це відмінна професія дуже добре оплачується, але і конкуренція тут, мабуть, найвища. Рекламна індустрія швидко сприймає нові можливості, які дозволяють їй привернути увагу аудиторії і впливати на неї. 3D-анімація дуже позитивно сприймається людським оком і її ефективність в 5 разів більше, ніж у рекламній 2D-анімації. Іноді використання 3D-анімації - єдина можливість візуалізації. 3D-анімація дає маркетологам більше можливостей показати переваги продукції. Результати роботи 3D-дизайнера, зроблені в наукових цілях, рідко видно масової аудиторії. Медицина, криміналістика, архітектура - ось та спеціалізація, яка може бути у 3D-аніматора. У медицині широко використовують 3D-анімацію: від створення візуалізації медичних подій до біологічної реакції. Наприклад, складно відтворити процес того, як хвороба блокує потік крові до серця і викликає інфаркт. 3D-анімація дозволяє наочно відтворити картину того, що відбувається і це її значну перевагу для медицини. 3D-візуалізація більше використовується в освітньому процесі та маркетингу. Також її використовують під час проведення досліджень і допомагає робити

прогнози і ставити діагнози. 3D-анімація і моделювання допомагають при реконструкції подій і в криміналістиці. Анімацію і моделі створюють для того, щоб довести, спростувати або доповнити картину злочину або нещасного випадку. 3D-моделі і анімація не є прямим доказом, але вони дозволяють слідчому, судді або прокурора відтворити подію і наочно продемонструвати хід подій. Іноді криміналістам потрібно відтворити деякі відсутні предмети або їх фрагменти, які мають відношення до злочину і в цьому їм допомагають нові технології: 3D-моделювання та 3D-принтери.

Дизайн графічного інтерфейсу призначений для користувача і зазвичай називається GUI, який буде використовуватися для забезпечення взаємодії користувача з програмним додатком. При розробці графічного інтерфейсу слід враховувати кілька цілей або показників: продукт повинен бути ефективний у використанні, без помилок, простий і приємний на погляд. Щоб відповідати цим критеріям, дизайнер повинен прагнути стимулювати потік користувачів і знижувати акцизи.

1.2 Аналіз програмного забезпечення для 3D-моделювання

Сьогодні існують різні програмні застосування, які використовуються для створення і управління анімацією. Деякі з них є загальними, що дозволяє користувачам створювати величезну кількість різних видів анімації, а деякі більш спеціалізовані на певних її видах.

1.2.1 Autodesk Maya

Maya - це додаток для створення тривимірної графіки і анімації. Вона дуже універсальна, дозволяє користувачеві створювати практично будь-яку анімацію, яка може бути включена в безліч різних додатків.

Вона надає користувачеві велику кількість інструментів і функцій для створення цих анімацій. Однією з основних переваг Maya в порівнянні з іншими інструментами, переліченими тут, є анімація персонажів.

1.2.2 CLO3D

CLO3D - це інструмент для створення анімації, спеціально призначений для візуалізації реалістичних тривимірних предметів одягу. Головна особливість CLO3D, що відокремлює його від інших інструментів, перерахованих тут, - це можливість візуалізувати предмет одягу з клаптиків тканини, «зшиваючи» їх разом.

1.2.3 Houdini

Houdini - це програма для тривимірної анімації, яка спеціалізується на процедурної генерації. Houdini не вимагає зовнішніх завантажень для візуальних ефектів і т. д. Вважається, що новачкам важче вивчити його, ніж інші інструменти, перераховані тут, і, крім іншого, потрібні додаткові технічні знання.

1.2.4 3DS MAX

3DS MAX - це стандартне програмне забезпечення для тривимірної анімації. Воно вимагає, щоб анімація «запікалася» перед візуалізацією, в порівнянні з живим зворотним зв'язком, отриманим від деяких інших інструментів. Для 3DS MAX є безліч плагінів, що надають різні функції і ресурси. Він доступний тільки для операційних систем Windows.

1.2.5 DeformPlugin

DeformPlugin - це плагін для Unity, який надає проектам доступ до різних функцій, що надаються Dynamo SDK. Плагін спеціалізується на реалістичній анімації м'якого тіла з великою кількістю функцій для імітації тканини. Він пропонує кілька різних функцій, включаючи зшивання клаптиків тканини разом і різні види зіткнень.

1.2.6 Dynamo SDK

Dynamo SDK, на якому побудований цей інструмент, призначений для створення анімації з використанням різних типів м'яких тіл. Однак одна з його основних функцій - це можливість створювати реалістичну анімацію, засновану на фізиці, за допомогою тканини. Це означає, що результуючий інструмент буде в основному заснований на створенні і управлінні подібними анімаціями м'яких тіл. Створення внутрішніх функцій, таких як фактична анімація, колізії і т. д. Надається Dynamo SDK, тому їх не потрібно враховувати при створенні цього інструменту. Інструменти інтерактивного 3D-моделювання можуть бути неймовірно складними, надаючи користувачеві велику кількість можливих дій. Цей інструмент, зосереджений на деяких ключових функціях, не зачіпаючи інші, наприклад, зшивання латочок разом.

1.2.7 RenderMan

RenderMan - програмний продукт, з великою кількістю програм, промисловий стандарт рендеринга для 3D-анімації. Зокрема існує як стандарт опису тривимірних даних для їх подальшої візуалізації, так і окремо стоїть рендер, випущений останнім часом під тією ж назвою.

1.3 Питання, що слід враховувати при розробці та впровадженні інтерактивного інструменту тривимірного моделювання для створення та управління загальними м'якими тілами?

На питання дослідження був дано відповідь із використанням знаних, зібраних у процесі цього проекту, включаючи всі, від початкового дослідження до завершення розробки додатків. При розробці програмного забезпечення для 3D-моделювання необхідно викласти наступні основні моменти:

1) Які зовнішні бібліотеки використовувати
2) Який робочий процес використовувати
3) Які функції повинні бути доступними
4) Як розробити для різних груп користувачів
5) Як зробити код модульним

1.3.1 Які зовнішні бібліотеки використовують

Перше, що необхідно вчити, - це то, які зовнішні бібліотеки використовувати при розробці додатків. У цьому проекті пропонується більшість зовнішніх бібліотек, які розглядаються для конкретної функціональності в початкових завданнях з реалізації цієї функціональності. Кожен раз, коли функціональність могла бути представлена зовнішньою бібліотекою, початок потрібно було вирішити, слід використовувати зовнішню бібліотеку. Це було зроблено шляхом наближення компромісу між плюсами та мінусами включення бібліотек, а не кодирування функціональності з нуля.

Переваги зовнішніх бібліотек використання зовнішніх бібліотек замість створення деяких функцій з нуля - це економічна час. Звичайно це одна з основних причин включення зовнішніх бібліотек. Ще одне перевагу є в тому, що він дозволяє модульність програмного забезпечення, оскільки бібліотеки

пишуться як окремі модулі. Однак зовнішні бібліотеки можуть бути самодостатніми або залежати від додаткових залежностей, що також необхідно вчити. Останнім перевагою є те, що, як правило, тестується, це означає, що розробники можуть покластися на функціональність, надаючи модульну бібліотеку для повноцінної роботи. Це також може заощадити час, який у противному випадку було придбано при тестуванні та можливому виправленні помилок самостійно реалізованих функцій.

Недоліки зовнішніх бібліотек використання зовнішніх бібліотек є тим, що ваш проект отримав додаткову залежність. Це може допомогти тому, що приєднує багато зусиль для рефакторінгу додатків, якщо за будь-якої причини зовнішньої бібліотеки приєднується переключити на другу. Використання багатьох зовнішніх бібліотек у додатках також може викликати такі проблеми, як конфлікти залежних, які можуть бути виправлені важко. Інший недолік є в тому, що популярні бібліотеки з відкритим вихідним кодом можуть стати статтею хакерів, що вводить до проблем безпеки вашого додатка.

Рекомендації по вибору бібліотеки

Якщо вирішено, що для певної функціональності необхідна внесена бібліотека, цей набір керуючих принципів, витекаючих з цієї роботи над проектом, може бути використаний, щоб допомогти вирішити, яку бібліотеку слід використовувати:

• Використовуйте популярні бібліотеки, так як вони більш якісні.

• Використовуйте бібліотеки з відкритим вихідним кодом. Відкритий вихідний код забезпечує перевірку коду з точки зору продуктивності та безпеки та надає розробникам більшого контролю.

• Використовуйте бібліотеки, які можуть виконувати кілька функцій, необхідних вашому додатку. Дві птиці з одним каменем.

• Використовуйте бібліотеки, як можна зменшити кількість додаткових залежностей, так як це робить їх надійнішими з точки зору зручності супроводу.

Ці рекомендації не засновані на тому чи іншому конкретному для розробки інструментальних 3D-моделювань, але в більш загальній планеті доменів до будь-якого проекту розробляється програмне забезпечення. Для цього конкретного проекту з використанням зовнішніх бібліотек були реалізовані наступні функції, які враховуються особливо корисними при розробнику програмного забезпечення для 3D-моделювання:

• Візуалізація графічного інтерфейсу

• Імпорт об’єктів з файлів

• Додавання штучок трансформації

• Серіалізація даних

• Різні бібліотеки для різних математичних обчислень, у тому числі для розсилки променів.

Всі ці зовнішні бібліотеки були вибрані під час виконання проекту компанії

наступним вказаним вище рекомендаціям.

1.3.2 Який робочий процес використовувати

Гнучка розробка програмного забезпечення сьогодні стає найпопулярнішим способом роботи з розробкою програмного забезпечення. Однак для цього існує безліч різних методів, і різні аспекти проекту можуть вплинути на те, який з них буде оптимальним. Ці аспекти включають розмір команди розробників, тривалість проекту і частоту зустрічей із зацікавленими сторонами. Важливо не вибирати конкретний метод і строго його дотримуватися, а, скоріше, дозволити роботі бути гнучкою і легко адаптуватися до змін, що є основним принципом будь-якого гнучкого робочого процесу.

1.3.3 Які функції повинні бути доступні

Рішення про те, які функції повинні бути в додатку, є одним з факторів, які найбільше впливають на те, як додаток буде працювати. У цьому проекті було кілька чинників, які в значній мірі сприяли цьому: функціональність, пропонована Dynamo SDK, функції, доступні в популярних інструментах 3D-моделювання, результати користувальницького тестування, відкриття на етапі впровадження програмного забезпечення і тимчасові рамки проекту разом узяті. з розміром команди розробників програмного забезпечення. Функціональні можливості Dynamo SDK були додатково засновані на дослідженні потреб групи користувачів, виконаному Deform Dynamics до початку цього проекту. Більш того, проведені призначені для користувача тести також ґрунтувалися на найбільш типових сценаріях використання розробленого додатка. Це були дані, надані Deform Dynamics на початку проекту.

Нижче наводиться список функцій, які слід враховувати при розробці інструменту 3D-моделювання:

1) Розставляйте пріоритети у функціях, що підкреслюють унікальну функціональність вашого застосування. Наприклад, якщо ваше додаток перевершує конкурентів щодо певного типу анімації, приділіть пріоритетну увагу функціям, які допомагають створювати ці анімації.

2) Дозволити користувачеві легко додавати різні типи об'єктів.

3) Дозволити користувачеві імпортувати об'єкти і анімацію з широко використовуваних форматів файлів.

4) Дозволити користувачеві виконувати перетворення об'єктів.

5) Дозволити користувачеві скасовувати / повторювати дії.

6) Дозволити користувачеві копіювати / вставляти об'єкти.

7) Дозволити користувачеві вільно обертати і переміщати камеру по сцені за допомогою миші.

8) Дозволити користувачеві зберігати і завантажувати сцени в / з файлів проекту.

9) Уникайте розміщення функцій, чутливих до продуктивності, в тих областях додатки, де користувач проводить найбільше часу. Приклад цього включає

постійне виконання симуляції за замовчуванням; замість цього програвайте симуляцію натисканням кнопки, дозволяючи додатком працювати більш плавно протягом більшої частини часу.

1.3.4 Як проектувати для різних груп користувачів

Протягом всього проекту проводилися різні дослідження і юзабіліті-тести щодо рекомендацій по дизайну. Попередня включена теорія була націлена на програмні додатки в цілому, в той час як практичне дослідження цього проекту використовувалося для визначення того, які аспекти можуть бути конкретно застосовні до розробки програмного забезпечення для 3D-моделювання. Це практичне дослідження відноситься як до аналізу існуючого програмного забезпечення для 3D-моделювання, так і до розробки інструменту для 3D-моделювання.

Підсумкові моменти, які слід враховувати при розробці інструменту 3D-моделювання, представлені в списку нижче:

1) Використовуйте суверенну позу для додатка.

2) Дозволити пряме управління об'єктами і настроюються параметри

3) Дозволити користувачеві отримати огляд сцени зі списком всіх наявних об'єктів.

4) Надайте користувачеві негайну модальну зворотний зв'язок після виконання дії.

5) Дозволити досвідченим користувачам виконувати дії за допомогою сполучень клавіш.

6) Звести до мінімуму акцизи, щоб стимулювати потік користувачів.

7) Дозволити користувачеві виконувати одне і те ж дію різними способами, наприклад, перетворюючи об'єкт за допомогою Gizmo або задаючи параметри через графічний інтерфейс.

8) Дозволити користувачам змінювати положення панелей та інших об'єктів у програмі.

9) Додайте найбільш часто використовувані інструменти так, щоб вони були легко доступні з будь-якої точки прикладання.

10) Дозволити користувачеві перетягувати файли з робочого столу в додаток в якості альтернативного способу імпорту об'єктів.

1.3.5 Як зробити код модульним

Використання модульного підходу до програмування є важливим моментом для полегшення супроводу програмного забезпечення, поряд з ретельним вибором зовнішніх бібліотек і використанням відповідного гнучкого методу розробки програмного забезпечення [55]. При модульному підході до програмування кожна загальна функціональність програми обробляється окремим модулем. В процесі розробки інструменту 3D-моделювання в цьому проекті в ході реалізації проекту були виявлені різні функції, необхідні в додатку. В результаті вийшов наступний набір рекомендованих модулів:

1) Модуль GUI для роботи з графічним призначеним для користувача інтерфейсом.

2) Модуль рендеринга для рендеринга об'єктів в сцені.

3) Модуль шейдеров для завантаження вершинних і фрагментних шейдеров.

4) Модуль імпорту для імпорту об'єктів з файлів деяких форматів.

5) Модуль камери для обробки параметрів камери.

6) Модуль управління для обробки базових обчислень, таких як сериализация і обчислення даних вершин і індексів для об'єктів.

7) Загальний модуль, який з'єднує інші модулі і запускає додаток.

1.4 Мета та задачі

Мета роботи полягає в створенні програми для 3D моделювання. Додавання функцій які є у інших популярних програмах для моделювання наприклад: Autodesk Maya, CLO3D, Houdini, 3DS MAX.

Сама програма повинна бути розроблена на движку Python, з підтримкою альтернативних плагінів і середовищ. Також працювати на всіх популярних операційних системах: Windows, MacOSX, GNU / Linux.

2 ВИКОНАННЯ РОЗРОБКИ ПЗ ДЛЯ 3Д-МОДЕЛЮВАННЯ

У цьому розділі описана програма яку я розробив для тривимірного моделювання, анімації і рендеру для операційних систем GNU / Linux, Posix і Win32. Вона має надійну об'єктно-орієнтовану архітектуру плагінів, розроблену для масштабування відповідно до потреб професійних художників, та спроектована з нуля для створення анімацій кінематографічного якості за допомогою движків рендеру, сумісних з RenderMan.

2.1 Інтерфейс програми

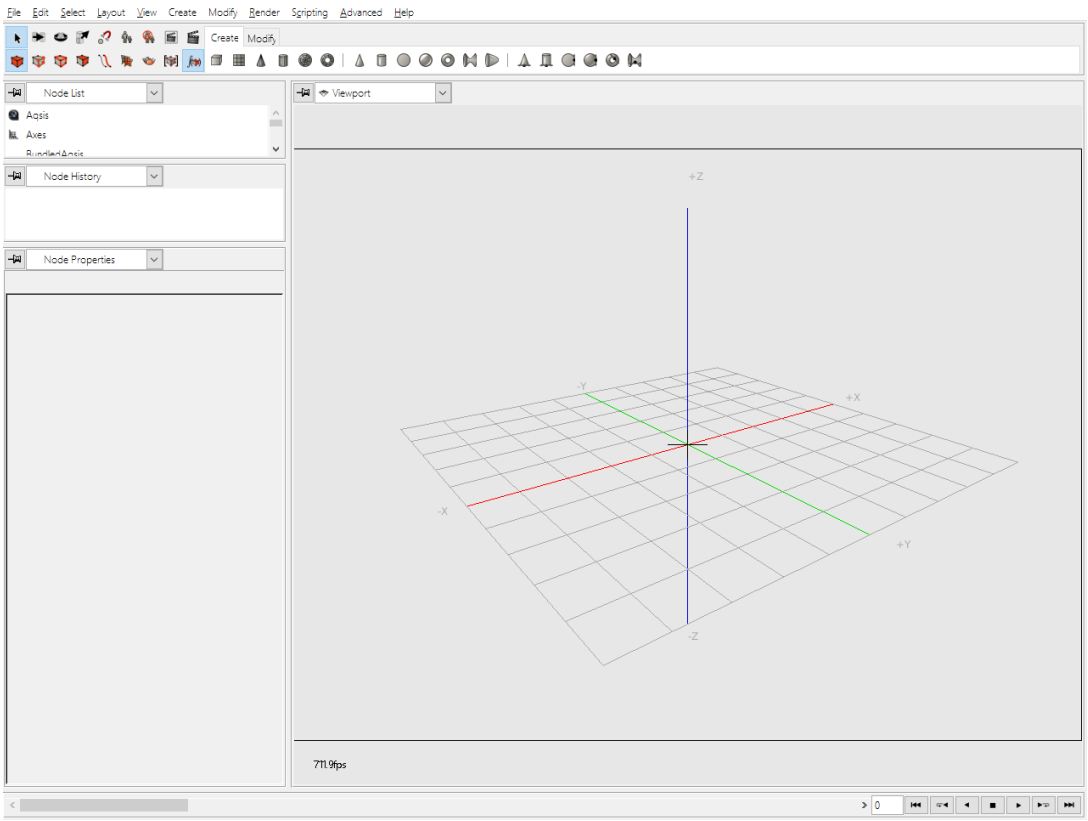

Рисунок 2.1 – Робоча площа програми

На рис. 2.1 видно інтерфейс програми у верхній частині вкладки: file, edit, select, layout, view, create, modify, render, scripting, advanced, help. Нижче є панель для формування моделі з примітивів квадратів конусів і т. д. У центрі

робоча площа де видно 3D модель, зліва можливо додавати параметри для моделі.

2.2 Функціонал програми

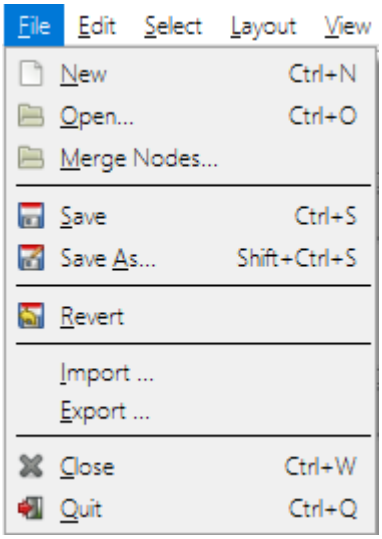

Рисунок 2.2 – вкладка file

На цій вкладці можливо створити нову модель, відкрити вже існуючу, зберегти, імпортувати або експортувати 3d модель у інші програми а також закрити програму.

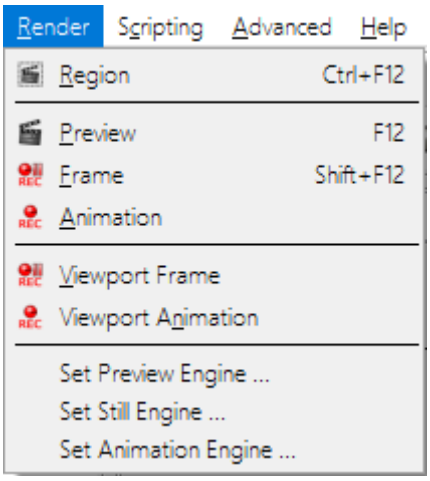

Рисунок 2.3 – вкладка render

Також програма має такий функціонал як: дублікат моделі, створення екземпляру, видалення і зміна призначення клавіш.

Також можливо крутити модель у різні сторони, віддзеркалювати, рендер сцени, процедурне моделювання та анімація, інтерактивні уроки, необмежене ієрархічне скасування / повтор.

Функції камери: панорамування / нахил, масштабування, візок, моделювання та режими штатива.

Подання: детальний вибір видимих функцій, приховати / показати геометрію.

Моделювання. Потужні графічні сцени, процедурне моделювання з повною історією. Виділення: об'єкти, сітки, межі, кромки, виправлення, криві, групи точок, точки. Типи геометрії: Polygon, NURBS, subdivision, blobby. 3D примітиви: конус, коло, подушка, циліндр, диск, сітка, параболоїд, багатогранник, сфера, тор. Операції логічного моделювання (з використанням бібліотеки GTS). Текст: підтримка FreeType2. Створення примірника: створення дублікатів без додавання гемометра до сцени.

Анімації. Visualization Pipeline допускає довільний потік даних - будь-якої властивості об'єкта може бути пов'язано з іншим сумісними властивостями. Анімація будь-якого значення. Необмежена кількість анімаційних каналів. Канали кривої Безьє. Анімація операцій моделювання. Текстури з 16-бітної глибиною кольору з плаваючою комою. Процедурні шейдери RenderMan. Повна підтримка RenderMan: Aqsis, Pixie, BMRT, PRman, 3Delight, Render Dot C. можливістю розширення підтримка альтернативних движків і моделей рендеру: Yafray. Візуалізація превью OpenGL на диск.

Движок Python (кращий движок для розробки нових скриптів). Движок K3DScript (мінімально скриптовий движок для посібників / макросів). Підтримка альтернативних плагінів і середовищ скриптового движка. Базовий 2D-композітінг. Bitdepth 16-бітове з плаваючою комою на канал.

Формати геометрії: Wavefront OBJ, GTS і необроблений формат повністю підтримуються, інші не є обов'язковими і залежать від установки PLIB. Експериментальні плагіни включають формати OpenFX, OFF, RIB і X.

Формати зображень:

1) JPEG (всі платформи).
2) PNG (всі платформи).
3) TIFF (всі платформи).
4) OpenEXR (потрібен додатковий плагін OpenEXR).
5) BMP (потрібен додатковий плагін ImageMagick).
6) SUN (потрібен додатковий плагін ImageMagick).

Платформи на яких програма працює:

7) GNU / Linux (стабильный).
8) Win32 (стабильный).
9) MacOSX (нестабильный).
10) BSD (не проверено).
11) Солярис (не проверено).

2.3 Планування

Проект буде розроблений з використанням гнучкого процесу. Таким чином, кожен з наступних кроків буде виконуватися у кілька ітерацій.

2.3.1 Фаза дослідження

Першим кроком буде фаза дослідження. На цьому етапі потрібно буде розглянути кілька речей: спочатку буде досліджено предметну область і контекст цієї роботи, включаючи дослідження пов'язаних робіт і літератури. Другим кроком буде вивчення Dynamo SDK, яке включає вивчення різних функцій, пропонованих SDK, і того, як він працює. Наступним кроком буде дослідження різних фреймворків графічного інтерфейсу з відкритим вихідним

кодом і вибір мови програмування. Обидві ці речі будуть виконуватися паралельно, оскільки вони залежать один від одного. Хороший фреймворк графічного інтерфейсу може мотивувати певну мову програмування, а вона насамперед може мотивувати певний фреймворк графічного інтерфейсу. Нарешті, буде досліджений сервер OpenGL. Це включає в себе не тільки вивчення самого API, але також вивчення різних необхідних розширень, таких як завантажувач OpenGL і творець контексту, які будуть використовуватися для включення функцій в додаток.

2.3.2 Етап проектування

Другим кроком буде етап проектування. Тут дизайн створюється з урахуванням потреб користувача, а також функцій, пропонованих Dynamo SDK і іншим використовуваним програмним забезпеченням. Етапи проектування включатимуть останні чотири етапи дизайнерського мислення, а саме: визначення, створення ідей, прототипування і тестування. Стадія емпатії виходить за рамки цього проекту; необхідні дані про домен і потребах користувачів надає компанія.

На етапі створення ідей буде розроблений ряд різних початкових дизайнів графічного інтерфейсу користувача з використанням ескізів. Потім вони будуть зрівняні і, можливо, об'єднані в остаточний ескіз. На цьому етапі також буде проведено аналіз конкурентів. На етапі створення прототипу буде створений прототип на основі поточного дизайну. Тип прототипу буде залежати від стадії процесу. Спочатку це будуть прототипи і каркаси з неякісного паперу. В середині будуть макети середньої точності. В кінці процесу це будуть макети високої якості і робочі прототипи програмного забезпечення. На етапі тестування існуючий в даний час прототип буде протестований і оцінений. Для цього будуть використовуватися методи когнітивного покрокового аналізу і протоколу думок вголос. Серед тестувальників будуть як професійні дизайнери анімації, так і аматори.

Зворотній зв'язок, отриманий на етапі тестування, буде використовуватися для поліпшення поточного дизайну перед тим, як повернутися до етапу визначення.

2.3.3 Етап впровадження

Третім кроком буде етап реалізації. Саме тут інструмент і реалізується. Реалізація буде здійснюватися поступово, для окремих функцій, з використанням комбінації методів Scrum і Feature Driven Development, спеціально адаптованих для групи розробників. Архітектура коду буде заснована на модульному програмуванні. Шаблон програмування на основі інтерфейсу і модуля будуть використовуватися для досягнення ремонтопридатності, можливості повторного використання і розширюваності коду.

2.4 Процес та виконання

У цьому розділі розповідається, як велося виконання цього проекту. Кожен з наступних розділів описує один з етапів проекту, а саме етап дослідження, етап проектування і етап розробки програмного забезпечення.

2.4.1 Фаза дослідження

У цьому розділі пояснюється, як проводилася дослідницька фаза проекту. Етап дослідження розділений на існуюче програмне забезпечення для 3D-моделювання, відповідну літературу та інструменти розробки програмного забезпечення.

2.4.2 Дослідження програмного забезпечення для 3D-моделювання

Першим кроком цього проекту було дослідження існуючих в даний час інструментів 3D-моделювання. Були вивчені інструменти Autodesk Maya, CLO3D, Houdini, 3DS MAX і DeformPlugin. На даному етапі суворого аналізу інструментів не проводилося. Замість цього первинне дослідження інструментів було зосереджено на їх загальному зовнішньому вигляді. Вибрані інструменти були засновані на їхній репутації в галузі.

2.4.3 Вивчення відповідної літератури

Це дослідження відноситься до будь-якої літератури, використаної в ході цього проекту. Це включає літературу про комп'ютерну графіку, 3D-моделювання, інтерактивний дизайн і розробку програмного забезпечення.

2.4.4 Дослідження інструментів розробки програмного забезпечення

Першим кроком при дослідженні інструментів розробки програмного забезпечення було знайомство з комплектом засобів розробки програмного забезпечення SDK Dynamo. Цей процес полягав у читанні документації по SDK, перевірці різних презентацій. Це дало цінну інформацію, таку як ключові особливості, можливості, обмеження, а також те, як це можна було б використовувати для створення анімації. Це в основному послужило основою для бачення всього проекту.

Також були досліджені різні інструменти розробки програмного забезпечення, такі як мова програмування і зовнішні залежності. Включаючи дослідження зовнішніх залежностей фреймворків і бібліотеки графічного інтерфейсу для імпорту об'єктів.

Що стосується мов програмування, то двома основними з них були C ++ і Python. Переваги C ++ полягали в тому, що він набагато швидше Python, так як це компільована мова. Однак те, що C ++ є компільованою мовою, також означає, що кожна зміна в коді призведе до його перекомпіляції. Python,

будучи інтерпретується мовою, може повністю уникнути часу компіляції. Python також є більш звичною мовою, що дозволяє швидше писати код. Більшість інших інструментів розробки програмного забезпечення, які були досліджені, перебували в аналогічній ситуації з офіційною підтримкою C ++, в той час як підтримка Python варіювалася від офіційної, неофіційної, експериментальної і так далі. Однак Dynamo SDK спочатку підтримувався тільки C ++; для того, щоб він працював з Python, повинні бути зроблені прив'язки. У підсумку мовою програмування був обраний C ++. Це сталося через те, що було вирішено, що більш швидке написання коду і відсутність часу компіляції, що забезпечується Python, не переважують офіційну підтримку досліджуваних фреймворків і більш важливо отримати підвищену швидкість і продуктивність, що забезпечується C ++. Оскільки 3D-анімація, яка надається SDK, яка повинна бути створена додатком, заснована на швидкості, ефективності, продуктивність при запуску програми. Була визначена як найбільш важлива річ в очах потенційних користувачів.

Були досліджені два основних GUI-фреймворка: Dear ImGui і Qt. Основна відмінність між ними полягала в тому, що Dear ImGui легший і простий в управлінні, в той час як Qt пропонує більше функціональних можливостей. Що стосується сумісності, обидві бібліотеки офіційно підтримувалися C ++ з неофіційними прив'язками, що дозволяють їм обом працювати з Python. Оскільки Dear ImGui, здавалося, забезпечує всі функціональні можливості, які повинні бути в додатку, він був обраний через його легкі характеристик.

Вибраний інструмент створення контексту OpenGL - GLFW. Це було пов'язано з тим, що він легкий, але при цьому надає все необхідне.

Що стосується бібліотеки завантаження OpenGL, для додатка був обраний gl3W. Це було просто тому, що він завантажує тільки основні функції, що робить його швидше, ніж альтернативні додатки. Оскільки ніяких функцій розширення не було потрібно, це було найкращим вибором.

При виборі бібліотеки для імпорту об'єктів в додаток була обрана бібліотека імпорту ресурсів Assimp. Це відбулося головним чином через величезної кількості рекомендацій при вивченні варіантів, а також через те, що існувало безліч посібників для початку роботи з ним в OpenGL.

2.5 Етап проектування

Коли всі рішення були прийняті, наступним етапом був початок роботи над дизайном. Був розроблений ряд різних ескізів, які представляють початковий дизайн графічного інтерфейсу користувача. Основна увага в цих начерках приділялася розміщенню елементів і структурі навігації графічного інтерфейсу. Елементи, включені в ескізи, були засновані на найбільш типових

сценаріях використання, а також функціональні можливості, що надаються Dynamo SDK. Ці ескізи потім порівнювалися, перероблялися і об'єднувалися кілька разів до тих пір, поки не з'явився єдиний ескіз, що складається з кращих рис кожного незалежного ескізу. Цей первісний начерк був зроблений до проведення аналізу конкурентів, щоб звести до мінімуму їх вплив на дизайн. Потім ескіз був оцінений за допомогою когнітивного покрокового керівництва, щоб виявити деякі ранні недоліки дизайну і інші потенційні проблеми. Результати когнітивного покрокового керівництва були використані для поліпшення дизайну і усунення виявлених проблем.

Потім був проведений ретельний аналіз конкурентів для існуючих інструментів 3D-моделювання, згаданих в попередніх розділах. Це дозволило зосередитись на аналізі певних частин їх графічних інтерфейсів, які були представлені різними налаштованими параметрами і спроектовані як різні елементи графічного інтерфейсу розміщені на екрані. Це дало уявлення про те, які інструменти на панелі зазвичай використовуються і як вони представлені користувачеві. Ці результати були використані для поліпшення поточного дизайну інструменту тривимірного моделювання, розробленого в цьому проекті.

Вдосконалений дизайн був втілений в паперовому прототипі. Цей паперовий прототип використовувався для оцінки дизайну з двома фактичними користувачами початкового рівня, які використовують протокол «думки вголос». Фактичні користувачі початкового рівня відносяться до осіб в цільовій групі, які практично не мають досвіду роботи з аналогічним програмним забезпеченням. Перед користувачами були поставлені такі завдання, як додавання об'єкта в сцену, виконання перетворень, імпорт об'єктів, запуск моделювання і збереження проекту. В ході цих тестів були помічені деякі цікаві речі, наприклад, що різні користувачі іноді використовували два абсолютно різних підходи до однієї і тієї ж задачі. Спосіб виконання перетворень також можуть відрізнятися при підключенні однієї і тієї ж людини; користувач міг спробувати перемістити об'єкт, перетягнувши його мишею, в той час як той же користувач міг пізніше спробувати повернути той самий об'єкт, змінивши значення повороту через графічний інтерфейс. Крім того, було відмічена, що відсутність пояснювальних написів на деяких елементах графічного інтерфейсу збиває з пантелику новачків.

Відгуки, отримані в ході сеансів «подумай вголос», були використані для створення нових ескізів поліпшеного дизайну. Після порівняння і об'єднання нових ескізів був зроблений остаточний ескіз, який потім перетворився в новий паперовий прототип. Цей паперовий прототип був використаний для повторної оцінки дизайну. Це було зроблено аналогічно першим випробуванням; протокол «думай вголос» з двома різними користувачами. Однак на цей раз ці користувачі були користувачами середнього рівня, тобто у них був певний досвід роботи з програмним забезпеченням для 3D-моделювання. Список завдань був тим же самим, як і в попередньому юзабіліті-тесті, щоб отримати зворотній зв'язок з тих самих питань. Під час другої ітерації призначеної для користувача тестове завдання було виконано набагато швидше, а користувачі в цілому менше заплуталися, що свідчить про поліпшення в порівнянні з попереднім дизайном. І знову цей зворотний зв'язок привів до нових поліпшень поточного дизайну.

Покращений дизайн був перетворений в цифровий каркас за допомогою веб-інструменту Figma. Figma була обрана для цієї мети з кількох причин: по-перше, це веб-інтерфейс. Це означає, що ніякого програмного забезпечення не вимагається, а каркаси будуть зберігатися в Інтернеті. Зберігання його в мережі означає, що він захищений від втрати на комп'ютері і з ним можна працювати звідки завгодно. Друга причина полягала в тому, що Figma - це інструмент, який я використовував раніше, а це значить, що немає необхідності вивчати новий інструмент тільки для цієї мети. Потім він був обговорений і оцінений, і в результаті був отриманий великий зворотній зв'язок з точки зору поточних проблем, можливих виправлень і речей, які можна додати. Цей зворотний зв'язок включав такі речі, як наявність сполучень клавіш і візуальних підказок, що пояснюють їх, посилення модального зворотного зв'язку при виконанні різних дій, поведінку камери, вибір слів для різних елементів графічного інтерфейсу і розмір елементів, таких як кнопки і значки.

Останній запланований етап проектування, який оцінював реалізований прототип з реальними користувачами, повинен був бути виконаний після етапу розробки програмного забезпечення в кінці проекту. Однак через пандемію COVID-19 це було скасовано.

2.6 Розробка програмного забезпечення

Оскільки дизайн був досить далеким, було вирішено, що прийшов час налаштувати початковий проект для програми. Це включало настройку проекту, завантаження всіх обраних залежностей і їх налаштування для роботи з додатком. Щоб отримати графічний інтерфейс, наданий бібліотекою Dear ImGui для рендеру, був розроблений модуль графічного інтерфейсу користувача за допомогою наданого ними коду прикладу. Після успішного запуску вікна з OpenGL і Dear ImGui настав час для фактичної розробки

програмного забезпечення. Основна увага приділялася функціональності, оскільки на неї не вплинули якісь потенційні зміни в графічному інтерфейсі.

Перш ніж почати програмування, був складений список завдань, необхідних для завершення програми, тобто технічне завдання. Ці завдання були засновані на функціях, які було вирішено обрати для готового інструменту, таких як додавання об'єктів на сцену і фактичний рендер існуючих об'єктів. Також визначалося, чи залежать одні завдання від інших завдань. На початку кожного тижня вибиралася завдання з беклога продукту. Обрані завдання будуть ґрунтуватися на необхідності (наприклад, рендер об'єктів життєво важливий для більшості інших функцій, отже, повинен бути виконаний якнайскоріше), складності (більші завдання краще виконувати на початку спринту, менші завдання пізніше в спринті) і особисті переваги.

2.6.1 Спринт 1 - Реалізація рендеру

Метою першого спринту було реалізувати рендер м'яких тіл, що завантажуються в сцену за допомогою Dynamo SDK. Було вирішено, що це буде зроблено з нуля, а не з використанням бібліотеки рендеру, таких як Magnum Engine. Головною мотивацією цього вибору був той факт, що компанія хотіла мінімізувати зовнішні залежності. Інша причина полягала в тому, що процес налаштування Magnum Engine для роботи з додатком не здавалося тривіальним. Третя причина полягала в тому, що без цього було б простіше повторно використовувати шейдери і інший пов'язаний з рендером код, який компанія розробляла для своїх демонстрацій. Підводячи підсумок, можна сказати, що це не вважалося досить великим засобом економії часу для проекту, оскільки переважували недоліки використання зовнішньої залежності. Маючи на увазі модульний підхід до програмування, був реалізований як модуль рендеру.

Оскільки були всі дані вершин для м'яких тіл в завантаженої сцені, ця початкова робота над модулем рендеру в основному полягала в створенні

буферів OpenGL і передачі їм покажчиків на дані вершин. Фрагментальні і вершинні шейдери, що забезпечують базову модель блискавки і однорідні кольори об'єктів, також були розроблені, щоб мати можливість візуалізувати сцену. Крім того, був розроблений шейдерний модуль для ефективного завантаження шейдерного коду в додаток. Використовуючи цей модуль, завантажити шейдер з файлу буде так само просто, як передати шлях до файлу при створенні нового об'єкта шейдеру. Це спрощує використання безлічі різних шейдерів в одному додатку.

Оскільки рендер об'єктів з м'яким тілом працюватиме до завершення спринту, також я працюватиму над функціональністю по рендеру різних об'єктів коллайдера. Оскільки колайдери, що зберігаються в Dynamo SDK, не були пов'язані з будь-якими даними вершин або індексів, їх доводилося візуалізувати вручну. Щоб полегшити цю частину рендеру, була додана бібліотека GLU. Ця бібліотека надає функцію для рендеру сфери в даній точці, gluSphere, яка використовувалася для коллайдеров сфер, а також обох кінців коллайдеров капсул. Причина того, що бібліотека GLU була включена, незважаючи на використання застарілих функцій, полягала в основному в тому, що вона була включена в пакет OpenGL, тому функції можна було використовувати, просто включивши файл заголовка GLU.h. Хоча спочатку планувалося, що це тимчасове рішення, ніколи не було першочерговим завданням замінити його налаштування функцій для рендеру сфер. У порівнянні зі сферичними коллайдерами, площинна і прямокутні колайдери були візуалізовано шляхом ручної побудови даних вершин з їх позиції, які були надані SDK. До кінця першого спринту сферичні і прямокутні колайдери були намальовані успішно.

В якості останнього завдання спринту також була додана проста панель графічного інтерфейсу з кнопками, що дозволяють створювати патчі і колайдери одним клацанням миші. Основна причина додавання панелі полягала в тому, щоб протестувати взаємодія між Dear ImGui і Dynamo SDK.

У цій версії всі об'єкти, створені за допомогою кнопок, без можливості їх зміни з точки зору користувача.

2.6.2 Спринт 2 - Імпорт об'єктів

Метою другого спринту було завершити рендеру інших коллайдеров і встигнути імпортувати об'єкти з файлів .obj. Візуалізація плоских коллайдеров були виконані швидко, але рендер капсул мав деякі проблеми. Колайдери капсул були візуалізовані, але їх орієнтація і положення не відповідали їх фактичній орієнтації і положенні з причин, невідомих в той час. Пошук помилок, не привів до якогось рішення, функціонал відклали. Замість цього пріоритет був відданий роботі над модулем імпортера для м'яких тіл в сцену.

Функція імпорту була розділена на дві задачі: вибірка даних вершин з файлу, що містить об'єкт, за допомогою Assimp для ручного рендеру і завантаження даних вершин в сцену за допомогою Dynamo SDK. Перша завдача, хоча і досить складна, пройшла досить гладко завдяки різним посібникам і документаціям, існуючим для роботи з Assimp. Друге завдання, яке спочатку здавалася легким, виявилася складніше, ніж очікувалося, так як результатом були б погано промальовані об'єкти з деякими файлами, що викликало збій. Це призвело до інтенсивного сеансу налагодження.

З'ясувати, що викликає помилки, було непросто. Дані вершин, надані розробленим імпортером, здавалися правильними, оскільки об'єкти Рендер вручну, і їх завантаження в сцену проводився відповідно до документації для SDK. Таким чином, все виявилося в порядку. На щастя, була документація Deform Dynamics, в якій вони імпортували файл .obj, він включав вихідний код. Тут було виявлено, що вони використовували іншу зовнішню бібліотеку під назвою tiny-objloader замість Assimp. Після дослідження і тестування відмінностей між двома бібліотеками було виявлено кілька цікавих моментів. Найважливішим відкриттям було те, що при використанні бібліотеки tiny-objloader кількість вершин для імпортованого об'єкта були набагато менше,

ніж кількість вершин для того ж об'єкта, завантаженого Assimp. Припускаючи, що причиною є інша обробка даних вершин, був розроблений інший модуль імпорту, який використовує бібліотеку tiny-objloader. При перемиканні старого модуля імпортера на новий, завантаження імпортованих об'єктів в сцену успішно спрацювала. Перемикання між двома різними імпортерами додатково показало модульність вихідного коду і то, наскільки добре він працював на практиці до цих пір.

Останньою завданням цього спринту було додати робочий діалог з файлом, який спливає при натисканні кнопки імпорту в графічному інтерфейсі. Оскільки спочатку не підтримувалась бібліотекою Dear ImGui, необхідно було вивчити альтернативні способи реалізації. Перша спроба полягала у використанні неофіційного розширення з відкритим вихідним кодом. Однак це рішення не дало задовільних результатів, так як діалогове вікно з файлом виглядало неякісно з естетичної точки зору. Зрештою за допомогою Windows 10 SDK був створений правильний діалог з файлом, що забезпечує роботу з файлом Windows за замовчуванням.

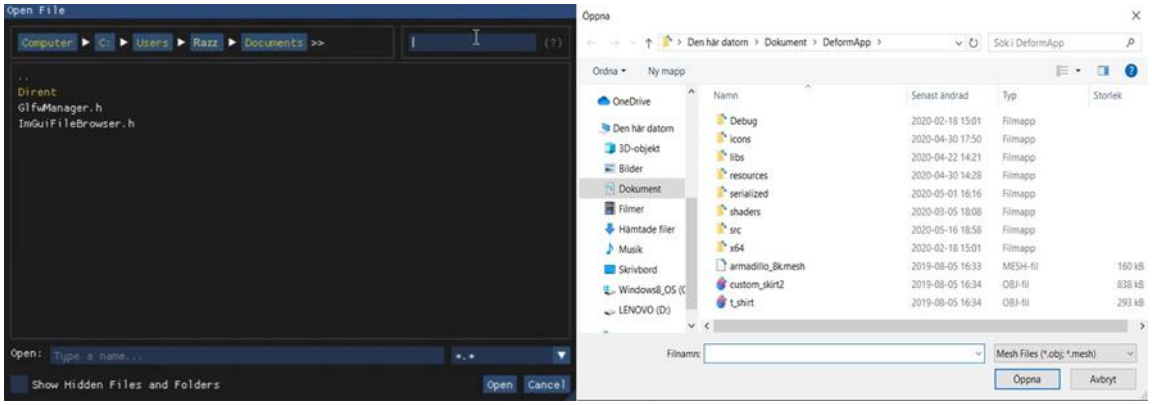

Рисунок 2.5 - На цьому рисунку порівнюються дві версії файлового діалогового вікна. Ліва частина старе зображення, а права частина – нове

2.6.3 Спринт 3 - Функція вибору

Основна мета третього спринту полягала в реалізації функції вибору. Ця функція дозволяє користувачеві перетягувати або розтягувати шматок

тканини за допомогою курсору миші. Оскільки SDK надає функції для зміни маси зазначеної частки, основний функціональністю, яка повинна бути реалізована для вибору роботи, був метод пошуку конкретної частки за допомогою курсору миші. Це було зроблено з використанням класичного підходу з методом розподілу променів, коли було виконано кілька тестів на перетин променів і трикутників, щоб визначити, чи потрапила частинка. Ці тести проводилися між променем, перетягуються курсором миші на сцену зі всіма трикутниками. Щоб полегшити цей процес, в проект було додано бібліотека VCG. Ця бібліотека дозволяла легко створювати промені, а також використовувати функцію для ефективного тестування перетину променів і трикутників.

Ще одна функція, реалізована під час цього спринту, - це можливість перетягувати об'єкти за допомогою миші. У поточному стані він був обмежений осями x і y. Щоб перетягнути об'єкт, його спочатку потрібно було вибрати в ієрархічному дереві графічного інтерфейсу користувача, оскільки вибір об'єктів в сцені за допомогою перетворення променів ще не був реалізований. Ця функція також виявила помилку з рендером при трансляції коллайдеров, яка була негайно виправлена.

У цьому спринті була побудована логічна модель для об'єктів, що обробляються додатком. Це було зроблено шляхом визначення класу SceneObjects з підкласами, що представляють різні типи об'єктів в сцені, такі як колайдери і деформовані об'єкти. Потім їм були надані різні атрибути, такі як ідентифікатор, ім'я, положення, масштаб і тип. Основна причина створення цієї моделі полягала в тому, щоб представити об'єкти сцени користувачеві на панелі ієрархії. Тут також була створенна панель ієрархії. З метою тестування також було додано перегляд «Деталі об'єкта», що дозволяє змінювати основні параметри, такі як положення і масштаб для обраного об'єкта, через графічний інтерфейс.

Заключна робота цього спринту полягала в реалізації базового руху камери. Рух був реалізований простим переміщенням по осях x і z, а також

можливість обертати камеру навколо себе за допомогою миші. Щоб цей фрагмент коду був слабо пов'язаний з іншою частиною програми, був розроблений модуль камери, який об'єднує всі функції камери в один файл заголовка. Камера була реалізована на цьому етапі в основному для цілей налагодження, таким чином, спосіб виконання переміщення був заснований на тому, як воно буде сприяти налагодженню, а не на тому, як це буде краще для кінцевого користувача. Реалізація перекладу за допомогою миші також породила питання про дизайн, що стосується того, як обробляти різні функції на основі миші. «Чи слід переміщати камеру тільки тоді, коли обраний інструмент камери?», «Чи повинна камера переміщатися тільки тоді, коли об'єкт не обраний?» і "чи слід при натисканні на порожню частину екрану знімати виділення об'єкта і автоматично активувати інструмент камери?" були деякі з питань, що виникли в результаті цієї роботи. Відповідь на ці питання були дані не відразу; замість цього вони були внесені в список завдань, які потрібно виконати. Можливість змінювати значення камери через графічний інтерфейс також була додана в кінці спринту.

2.6.4 Спринт 4 - Функція малювання

Головною особливістю четвертого спринту була реалізація функції малювання. Ця функція дозволить користувачеві малювати частки курсором за допомогою різних інструментів. Різні речі, які повинні були малюватися користувачем було на сам перед: тертя, маска зіткнення і фіксовані вершини. Ця функція була розділена на наступні під задачі: зміна значення частки, ураження променем, рендер «фарби» на порушених частинках, щоб дати користувачеві візуальну зворотний зв'язок, і дозвіл змінювати розмір кисті. Щоб ударяти по більшій поверхні з кожним клацанням. Оскільки функція перетворення променів вже існувала з моменту реалізації функції вибору, реалізація першої підзадачі була очевидною.

Перша спроба візуалізації намальованих частинок полягала в тому, щоб спочатку зберегти їх індекс в списку, а потім використовувати функції, що надаються Dynamo SDK, для отримання їх позиції і візуалізації. Потім сфера буде візуалізована в цьому місці за допомогою функції gluSphere, що надається бібліотекою GLUT. Хоча технічно це призвело до пристойного результату, були деякі побоювання. Одна проблема полягала в тому, що візуальні ефекти були досить потворними. Інша проблема, яка була найважливішою, полягала в тому, що один об'єкт з усіма намальованими частинками міг викликати серйозні проблеми з продуктивністю. Це призвело до розробки інших підходів. У наступній і останній спробі було використано аналогічне, хоча і набагато більш ефективне рішення. У цій версії вибрані індекси частинок зберігалися в списку, як і в першій версії. На цей раз, однак, покажчики даних вершин були створені вручну на основі позицій частинок зі списку. Потім покажчики були відправлені в засіб візуалізації, який малював їх як GL POINTS. У цій версії не було серйозних проблем з продуктивністю.

Останньою під задачею було змінити розмір кисті. Це було зроблено за допомогою функції, наданої SDK для пошуку частинок в межах певного радіусу від даної частки.

Опції для перемикання між різними інструментами малювання і зміни різних параметрів.

Елементи для кожного інструменту також були додані в графічний інтерфейс. Ці параметри включали установку величини тертя, активної маски зіткнення і радіуса пензля. Крім того, в графічний інтерфейс були додані кнопки для застосування обраної фарби до всіх частинок і видалення їх з обраних об'єктів. Різні інструменти малювання вибираються натисканням кнопок зі значками на панелі інструментів. Щоб мати зображення для кнопок зі значками, в імпортер була додана можливість імпортувати звичайні файли зображень і пов'язувати їх з текстурою.

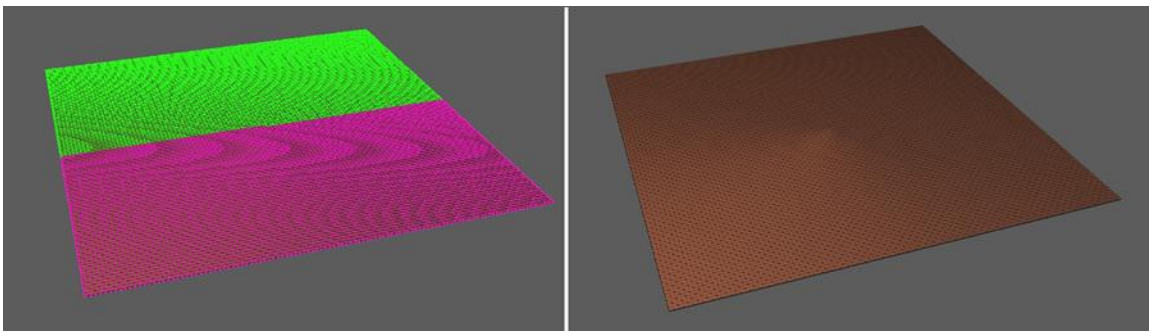

Рисунок 2.6 - Зліва на цьому рисунку показана деформована ділянка, де одна половина пофарбована однією фарбою, а інша половина - другою. У правій частині показана таж ділянка без будь-якої фарби

2.6.5 Спринт 5 - Реалізація поведінки камери

Першим завданням, яке потрібно було виконати для п'ятого спринту, була можливість вибирати об'єкти в сцені за допомогою світла променів. Це також було зроблено за допомогою функції SDK для пошуку найближчих частинок, оскільки одним з вихідних параметрів був покажчик, що містить всі ідентифікатори об'єктів. Таким чином, використання невеликого радіусу гарантує, що ідентифікатор вихідного об'єкта буде належати йому. Спочатку здавалося, що ця функція працює правильно, але пізніше було виявлено, що вона не повертає ідентифікатори об'єктів коллайдера. Отже, він дозволяв виділяти в сцені тільки об'єкти з м'яким тілом.

Друга, більша функція цього спринту полягала в тому, щоб реалізувати камеру таким чином, щоб вона вела себе так, як вважав за краще б користувач, а не так, щоб розробник міг швидко виконувати налагодження. Щоб отримати уявлення про те, яким повинен бути кінцевий результат, було проведено аналіз конкурентів. Houdini, Maya3D і 3DS MAX - все їх камери були ретельно вивчені. Як виявилося, всі вони мали однаковий тип рухів камери, хоча і з дещо різними способами доступу до них. Таким чином, було прийнято рішення реалізувати такий ж рух камери в додатку. Бажане положення камери було реалізовано як у конкурентів. Основна відмінність полягала в засобах управління. Таким чином, елементи управління були оновлені, щоб привести

їх у відповідність, оскільки не було нічого, що вказувало б на відхилення від галузевого стандарту. Найбільшою зміною, внесеною в поведінку камери, став поворот камери. Первісна поведінка, при якій камера оберталася навколо свого власного становища, було замінено поведінкою камери, при якому камера обертається навколо центральної точки свого поля зору. Останньою задачею п'ятого спринту було почати роботу над включенням коллайдеров у додаток. Довелось розділити задачу на дві під задачі: додавання коллайдеров сітки в сцену і їхній рендер. Додавання коллайдеров сітки було тривіальним завданням після збереження даних вершин для імпортованих об'єктів. Візуалізація була реалізована, хоча і з величезною помилкою; ніякі позиційні дані не впливали на рендер, і тому візуалізовані колайдери завжди були статичними. Ця проблема є частиною використовуваної версії Dynamo SDK. Таким чином, виправлення поки було не можливо.

2.6.6 Спринт 6 - Додавання зручних функцій

В ході шостого спринту основна увага приділялася розробці та реалізації деяких важливих дій користувача, таких як функції скасування / повернення, функції копіювання / вставки і перетворення відразу декількох обраних об'єктів. Ці функції повинні правильно себе вести і бути доступними для користувача.

Таким чином, перше завдання полягала в розробці поведінки. Щоб змусити їх вести себе так, як очікував би користувач, був проведений конкурентний аналіз того, як ці функції були реалізовані в існуючому програмному забезпеченні для 3D-моделювання. Ще раз, аналіз показав, що, по-видимому, існує безсумнівний стандарт того, які функції повині бути в таких інструментах, що призводить до той ж поведінки, яка імітується в додатку.

Друге завдання полягала в тому, щоб фактично реалізувати функцію копіювання / вставки. Оскільки всі об'єкти в сцені зберігалися в списку,

причому кожен об'єкт містив всю інформацію, необхідну для його відтворення, копіювання об'єкта було таким же простим, як створення нового об'єкта з тим же типом і параметрами та розміщення його в списку, що представляє типовий буфер обміну. Функція вставки спочатку додає об'єкт з буфера обміну в список об'єктів, а потім додає об'єкт відповідного типу в сцену.

Останнім завданням було реалізувати функцію скасування / повтору. Це було розділене на дві підзадачі: рішення, як це повинно бути реалізовано, і фактична реалізація. Вирішуючи, як його реалізувати, було розглянуто два варіанти. Першим, про що було задумано, зберігати кожну виконану дію в стеці. Причина створення стека полягала в тому, що вони працюють за принципом «першим приходить - останній», що відповідає типовому поводженню функцій скасування / повтору в програмному забезпеченні. Оскільки дії в додатках часто прив'язані до одного або декількох об'єктів, список порушених об'єктів повинен бути збережений разом з дією. Якщо дія буде додавати об'єкт “x”, то при виконанні дії скасування це буде просто видалення об'єкта “x”. Після цього видалення об'єкта “x” буде поміщено в окремий стек, з якого буде вилучатись функція повтору. Цей підхід спочатку здавався ідеальним, але мав один великий недолік: функція видалення, відсутня в Dynamo SDK.

Альтернативне рішення, прийшло від функції яку я робив раніше скасування / повтору в Adobe Photoshop, воно працює так коли виконується дія зберігається стан всієї програми. Це дозволить скасувати незворотні дії, але було б набагато дорожче в обчислювальному відношенні. Це також змусить сцену ініціалізуватися кожен раз, коли користувач виконує скасування / повтор, що може привести до порушення призначеного для потоку.

В ході зустрічі настав величезний поворотний момент. Після дослідження проблеми, з якою додаток зіткнулося з Dynamo SDK, компанія знайшла джерело проблеми. Як виявилося, проблема полягала в тому, що SDK не був призначений для виконання таких дій, як додавання об'єктів, виконання

перетворень об'єктів і т. Д Після того, як сцена вже була инициализирована. Однак для додавання і рендеринга об'єктів сцена повинна бути инициализирована. Рішенням цієї проблеми було переробити більшу частину програми, щоб воно не додавало об'єкти в SDK і не ініціалізувати сцену кожен раз, коли користувач виконував дію. Замість цього його довелося б переробити в більш типовий інструмент тривимірного моделювання з редактором, що містить статичні об'єкти, і симуляцією, що запускається після натискання кнопки відтворення / запуску. Це дозволить додавати об'єкти в сцену, оброблювану SDK, тільки при натисканні кнопки відтворення, тому кожен об'єкт може бути доданий і перетворений перед ініціалізацією сцени. Це також означало, що отрисовка об'єктів і більшість інших дій користувача до цього моменту повинні були виконуватися повністю без функцій, що надаються SDK.

Хоча це означало, що обсяг роботи, який необхідно виконати, значно збільшиться, це дійсно вирішило дилему, що стосується функції скасування / повтору, про яку говорилося раніше. Оскільки додавання об'єкта більше не буде використовувати функції SDK, відсутність функції видалення більше не було проблемою. Таким чином, тепер оптимальним вибором став метод зберігання дій в двох стеках (один для скасування, інший для повтору).

2.6.7 Спринт 7 - Рефакторинг додатків

Основне завдання першого з цих чотирьох спринтів полягала в тому, щоб змінити додаток для ручного рендеру статичних наповнювачів для всіх об'єктів в редакторі, а потім переключитися на використання Dynamo SDK для імітації анімації після натискання кнопки відтворення / запуску. Іншою частиною тієї ж завдання був рефакторинг інших функцій в додатку, щоб змусити його працювати з цим новим підходом. Оскільки передбачалося, що це не займе весь спринт, решту тижня буде присвячений початку роботи над серіалізацією даних для збереження і відкриття файлів проекту.

Зміни в додатку не вплинули на все, що стосується об'єктів-коллайдеров, оскільки вони не покладалися на функції SDK і, отже, не вимагали подальшої роботи.

Для більшості доступних типів об'єктів рендеру статичних наповнювачів представляв собою не що інше, як ручне зв'язування буферів з відповідними вершинами, і index-data перед виконанням виклику відтворювала для кожного об'єкта в списку об'єкти сцени. Для багатьох з цих об'єктів дані вже були доступні через імпортер. Однак це не відносилося до деформації ділянок і об'ємних сіток, оскільки Dynamo SDK раніше розраховував для них дані. На щастя, був вихідний код, який використовується SDK для генерації даних, що призвело до ефективного рефакторингу рендеру об'єктів.

Наступним завданням було рефакторинг інших функцій програми. Цими функціями були функція малювання частинок і вибір об'єктів за допомогою променів. Для перетворення променів це означало проходження всіх деформованих об'єктів в сцені і виконання тесту на перетин променів з трикутниками для кожного об'єкта. Це також означало, що всі деформовані об'єкти повинні зберігати свої вершини і індекси, чого раніше не було. Нові зміни також означають, що кожен об'єкт повинен буде зберігати свої власні намальовані частки, щоб їх можна було передати в сцену Dynamo після запуску моделювання. Після того, як були додані об'єкти сцени, рефакторинг вважався завершеним.

В останній день тижня почалася робота по серіалізації даних. Це було зроблено з використанням зовнішньої бібліотеки JSON під назвою «JSON для сучасного C ++». Основна причина зберігання файлів проекту в форматі JSON полягала в тому, що компанія вимагала, щоб серіалізовані файли були легкими для читання, що зробило JSON ідеальним кандидатом. Формат файлу JSON також дуже поширений і простий у використанні, що робить його очевидним вибором. До кінця тижня використана бібліотека, збереження і відкриття файлів повністю працювала.

2.6.8 Спринт 8 - Різні виправлення

Завданням спринту була реалізація функції скасування / повернення. Це було зроблено, як і планувалося за два тижні до цього, і передбачалося використовувати стек скасування і стек повтору, що містить виконані дії разом зі списком порушених об'єктів.

Під час першої половини спринту було виявлено витік пам'яті, через яку додаток вилітав після того, як він був активним протягом приблизно десяти хвилин. Після деякого розслідування, було виявлено, що ця проблема викликана тим, що нові буфери генеруються під час кожного кадру, але ніколи не звільняються. Це було виправлено: все буфери були згенеровані тільки один раз, а потім правильно прив'язані до кожного кадру.

Останнім завданням спринту було дозволити імпорт анімації з файлів .fbx. Оскільки бібліотека tinyobjectloader, використовувана для імпорту статичних деформуються об'єктів, підтримує тільки формат файлу .obj, Assimp знову був використаний для імпорту анімації. Крім імпорту даних вершин і індексів з файлу, необхідно було створити функцію для обчислення даних вершин і індексів в будь-який момент часу. Це також означало, що потрібно було обчислювати дані про кістки для анімованих персонажів. Це була досить складна задача, але до кінця тижня ця функція була повністю реалізована.

2.6.9 Спринт 9 - Додавання Guizmo трансформації

Першим завданням 9-го спринту було додавання пристосувань трансформації до об'єктів сцени. Для цього необхідно було провести дослідження можливих бібліотек Guizmo. Три розглянуті бібліотеки були маленькими, ImGuizmo і Im3D. Після певного тестування tiny gizmo було усунуто, оскільки для цього потрібно було б змінити спосіб відображення об’єктів у додатку. У той час як інші два здавалися досить простими для

включення в додаток, ImGuizmo здавався найпростішим. Більш того, загальним недоліком ImGuizmo, здавалося, було те, що він вимагає, щоб додаток використовував бібліотеку Dear ImGUI для графічного інтерфейсу. Це, звичайно, не було проблемою, оскільки додаток вже використало цю бібліотеку для графічного призначеного для користувача інтерфейсу. Im3D також має більш дешевшу графіку, що, поряд з іншими причинами, призвело до того, що була обрана бібліотека ImGuizmo.

Після того як я вибрав бібліотеку для перетворення Guizmo, додавання гаджетів до вибраних об'єктів виявилося простим завданням. Однак до сих пір обертання об'єктів не було частиною програми, але тепер його потрібно було додати. Це було зроблено шляхом додавання одиничного кватерніона до кожного об'єкту сцени, який представляє орієнтацію.

Щоб врахувати повороти в додатку, деякі функції, такі як приведення променів, повинні були бути оновлені з урахуванням нових змін, що стало другим завданням спринту. У цю задачу також входило дозвіл зміни орієнтації об'єкта через графічний інтерфейс. Оскільки уявлення орієнтації об'єктів в графічному інтерфейсі користувача було поданням кута Ейлера, а не поданням кватерніона, необхідно було реалізувати перетворення між ними. Той факт, що одну орієнтацію можна уявити як кілька різних послідовностей обертання, також зробив цю функцію трохи повільною, хоча і працюючою.

Третє завдання полягало в додаванні функціональності, яка дозволяє користувачеві перетягувати тимчасову шкалу на імпортовану анімацію, щоб показати кадр в будь-який момент часу. Оскільки ми вже реалізували функцію для отримання вершин і індексів анімації в будь-який момент часу, технічна реалізація була тривіальною задачею. Однак, оскільки ця функція не входила в вихідну концепцію програми, необхідно було вирішити, де розташувати цю функцію у графічному інтерфейсі. Зрештою, він був поміщений в якості повзунка на вкладку «Відомості про об'єкт» для обраної анімації.

Останнє завдання полягала в додаванні тегів в уявлення ієрархії, щоб спростити поділ різних типів об'єктів друг на друга. Ці теги представляли

собою прості тексти, що визначають тип об'єкта, укладені в квадратні дужки. Наприклад, тег для об'єкта коллайдера буде «[Collider]».

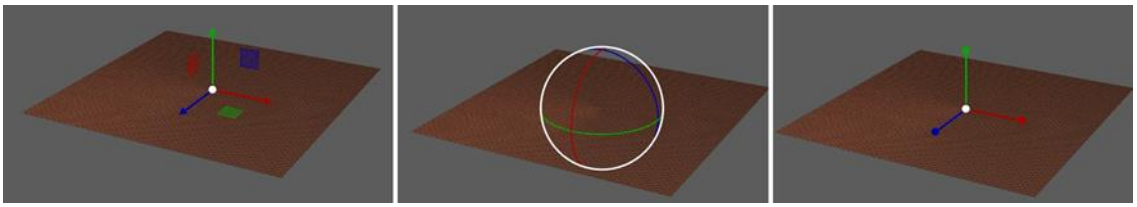

Рисунок 2.7 - Зліва на цьому рисунку показано як деформується ділянка, де одна половина пофарбована однією фарбою, а інша половина - другою. У правій частині показані ті ж ділянки без будь-якої фарби

2.6.10 Спринт 10 - Фінальний спринт

Основна увага на фінальному спринті проекту було зосереджено на тому, щоб переконатися, що додаток можна використовувати для відтворення всіх демонстрацій, які компанія розробила раніше. Друга мета полягала в тому, щоб очистити графічний інтерфейс і виправити естетику, наприклад, додати правильні значки для інструментів на панелі інструментів.

Щоб переконатися, що всі демонстрації можуть бути відтворені, найбільш логічним підходом було просто спробувати відтворити їх. При цьому були виявлені деякі відсутні функції, такі як прив'язка деформованого об'єкта до коллайдера. Інший відсутньої функцією була завантаженні файлом з розширенням .def, що містять серіалізовані дані про одяг, які повинні бути додані до імпортованих анімованих персонажів. Іншими відсутніми характеристиками були, в основному, зміна різних параметрів, таких як жорсткість об'єкта і деякі параметри розміру для різних об'єктів.

Зрештою, була реалізована функція прив'язки, так що користувач отримує доступ до інструмента прив'язки, якщо обраний відповідний об'єкт коллайдера. При використанні інструменту прив'язки користувачеві пропонується вибрати об'єкт для прив'язки. Про те чи закріплений об'єкт, додатково візуалізується через графічний інтерфейс.

Функція для завантаження файлів .def була надана компанією, а це означало, що єдине, що потрібно було реалізувати, - це можливість переглядати на комп'ютері файл, який потрібно додати до вже імпортованої анімації. Також необхідно було вирішити, як користувач повинен отримати до нього доступ через графічний інтерфейс. В кінцевому підсумку було вирішено зробити це, додавши кнопку на панель відомостей про об'єкт для обраної анімації, якщо файл ще не був завантажений.

Останнє завдання тижня полягала в тому, щоб виправити естетику графічного інтерфейсу, наприклад, правильно вирівняти все, а також успішно відтворити і зберегти більшість демонстрацій. Оскільки демонстрації були успішно відтворені і збережені, прототип інструменту тривимірного моделювання був готовим.

2.7 Виконання і процес

У цьому розділі буде обговорюватися весь процес цього проекту, пройшовши всі етапи і детально обговоривши їх.

2.7.1 Етап дослідження

На етапі дослідження було проведено багато досліджень для вивчення інших конкуруючих інструментів 3D-моделювання, а також інформації, необхідної для розробки програмного забезпечення. Це дослідження включало оптимальний мову програмування, доступні зовнішні залежності і Dynamo SDK. Цей крок дав багато ідей, які зіграли вирішальну роль в успіху проекту. Що найбільш важливо, він дав уявлення про те, як створюються інструменти 3D-моделювання та як вони зазвичай використовуються. Це також дало уявлення про те, як функції в таких програмах повинні вести себе.

Протягом всього проекту час від часу виникала потреба в нових зовнішніх бібліотеках, які раніше не досліджувалися. Це сталося або тому, що

була виявлена необхідність у новій функціональності, або тому, що раніше обрана бібліотека не відповідала тому, що від неї очікувалося. Однак той факт, що в проекті використовувався гнучкий робочий процес в поєднанні з модульностью коду, дозволяв досить легко замінити одну бібліотеку на іншу або просто включити нову в додаток. Це також, можливо, було пов'язано з вибором написання програмного забезпечення на C ++ замість Python, оскільки існує набагато більше бібліотек, написаних на C ++, і що не буде ніяких обмежень, пов'язаних з конкретною бібліотекою, яка не має прив'язок Python. Вибір використання C ++ в цілому вважався хорошим вибором, оскільки час компіляції, яке було основним аргументом проти його використання, в ретроспективі вважалося незначним.

2.7.2 Етап проектування

Етап проектування почався відразу після етапу досліджень. Рішення не проводити конкурентний аналіз існуючих інструментів 3D-моделювання та ретельно досліджувати їх перед тим, як накидати ескіз, було визнано хорошим вибором. Проведення аналізу конкурентів на більш ранньому етапі могло б прискорити початкові етапи етапу проектування, однак зменшення систематичної помилки дозволило процесу проектування створити більш унікальний дизайн.

Виконання когнітивних покрокових інструкцій на ранніх етапах фази проектування перед тестуванням з реальними користувачами було ефективним методом поліпшення дизайну. Це зробило дизайн досить хорошим, щоб можна було провести на ньому реальне призначене для користувача тестування, не втрачаючи занадто багато часу. Тестування отриманих паперових прототипів на реальних користувачів дозволило більше впливати на дизайн реальних потреб користувачів, зокрема користувачів початкового та середнього рівня, де останнім було основною метою. Дані, отримані в результаті цих користувальницьких тестів, стали ще більш

важливими, оскільки реальні користувачі були виключені з етапу оцінки через поточної пандемії, що зробило це єдиним реальним моментом.

користувача введення був отриманий від користувача, який не є експертом.

Попередньо закінчений дизайн перед початком розробки програмного забезпечення був хороший, оскільки дозволяв вибирати зовнішні залежності в залежності від того, які функції повинні бути доступні в готовому прототипі. У той же час був ряд причин, за якими дизайн не був повністю завершений до початку реалізації.

Основна причина в тому, що процес прототипування виявився хорошим джерелом ідей, що призвело до прийняття нових проектних рішень в процесі. Наприклад, вкладки «Параметри моделювання» і «Деталі об'єкта» помінялися місцями в реалізованому прототипі з того місця, де вони спочатку знаходилися в макеті проекту. Хоча спочатку це було реалізовано таким чином випадково, в кінцевому підсумку було визнано, що це поліпшить роботу додатка.

Друга причина в тому, що іноді буває корисно мати різні завдання для роботи. Наприклад, наявність обох деталей проекту, над якими потрібно працювати, а також реалізації, в подальшому дозволить перемикатися між двома завданнями, якщо щось перешкоджає негайному продовження будь-якої з них.

Остання причина полягає в тому, що компанія запросила демонстрацію першого набору функцій до певної дати, що означало б, що було б вигідно почати роботу над ними в фіксований час (а не тоді, коли фаза проектування досягла певної точки). якщо це прохання повинна бути задоволена. Це також відповідало особистим бажанням почати розробку програмного забезпечення на ранній стадії.

Найбільшим недоліком етапу проектування була відсутність оцінки користувачем готового прототипу через пандемію COVID-19. Замість цього корисність прототипу визначалася розробником разом з керівником компанії. Хоча оцінка, проведена реальними користувачами, дала б більш надійні дані,

оцінка, виконана разом з даними призначеного для користувача тестування цифрового макету, була визнана досить хорошою з урахуванням обставин.

2.7.3 Розробка програмного забезпечення

У цьому розділі буде обговорюватися етап розробки програмного забезпечення. Обговорення розділене на дві частини: обговорення використовуваних методів розробки програмного забезпечення і обговорення фактичної реалізації.

Методика - процес розробки програмного забезпечення проводився з використанням поєднання деяких популярних методів гнучкої розробки програмного забезпечення. Однак одна важлива річ відрізнялася від того, як методи використовувалися в цьому проекті в порівнянні зі звичайним використанням: цим проектом керував один чоловік. Причина, по якій це має значення, полягає в тому, що одне з основних відмінностей між деякими з популярних методів полягає в тому, як здійснюється комунікація між командою розробників. Крім того, кроки, на яких команда розробників обговорює різні питання, такі як щоденні зустрічі Scrum, також стають трохи інше, коли команда - це одна людина. Призначати завдання різним людям стало легко, так як всі завдання виконував один і той же чоловік. Поділ завдань таким чином, щоб не було конфліктів злиття, також не було проблемою, оскільки проект гарантував, що ніякі дві людини не будуть випадково перекривати один одного в роботі. Це також призвело до того, що завдання можна було починати в кінці тижня, оскільки не було б ніяких негативних наслідків, якби завдання була завершена тільки наполовину до кінця тижня.

У проекті використовувалися спринти тривалістю в один тиждень. В основному це було зроблено для того, щоб мати можливість представити компанії тижневу роботу під час щотижневих зустрічей по п'ятницях, щоб кожна зустріч функціонувала як демонстрація зацікавленим сторонам, яка

зазвичай включається в огляд спринту. Хоча в цілому це спрацювало, демонстрація роботи не була чимось то, що було потрібно від зацікавлених сторін на кожній зустрічі, і, отже, спринти могли бути, наприклад, удвічі довше без будь-яких наслідків у цьому сенсі.

Відсутність необхідності в спілкуванні всередині команди призвело до того, що візуалізація задач стала більш гнучкою. Замість формального носія, такого як дошка Kanban, використовувався простий текстовий файл зі списком всіх різних завдань, які ще не були виконані. Це спрацювало добре, так як було спеціально розроблено.

Основна увага при впровадженні приділялася забезпеченню найкращого взаємодії з користувачем з точки зору продуктивності, просуванню модульності програмного забезпечення, щоб спростити управління і розробку інструменту і, нарешті, звести до мінімуму використання зовнішніх залежностей.

Продуктивність забезпечувалася вибором мови програмування і увагою до внутрішніх структур даних і алгоритмів, що використовуються для досягнення певної функціональності. Крім того, це було досягнуто за рахунок реалізації функції запуску моделювання одним натисканням кнопки, поміщаючи обчислення, що виконуються для візуалізації анімації, в окреме місце, де користувач виконує роботу в додатку.

Модульність коду переслідувалася за рахунок наявності окремого модуля або файлу заголовка для кожної із загальних частин програми. На ранніх етапах розробки це дотримувалося з легкістю. Перемикання модуля імпортера через кілька тижнів в проект продемонструвало модульність програмного забезпечення на практиці. Однак частина модульної конструкції шляхом загубилася. Зокрема, в готовому прототипі модуль, який відповідає за графічний інтерфейс, тепер також відповідає за функціональність, більш підходящу для якогось модуля-менеджера, як зазначено в керівних принципах модуля в розділі результатів. Ця функціональність включає створення

вершинних і індексних даних для деформуються патчів, сериализацию сцен і зручні функції, такі як копіювання / вставка і скасування / повтор.

Можна стверджувати, що причина цього небажаного результату полягає в тому, що в уже існуючий модуль було вкладено більше відповідальності, оскільки виникла потреба в нових функціях. На той час, коли стало ясно, що необхідно ввести новий модуль для розвантаження модуля графічного інтерфейсу користувача, обсяг роботи, необхідної для усунення проблеми, склав

це завдання з низьким пріоритетом. Низький пріоритет завдання привів до того, що її ніколи не вирішували, оскільки завжди було більше невідкладних питань, якими можна було зайнятися. Проте, можна стверджувати, що причина була в тому, що модуль-менеджер не виявлявся на етапі планування, що робило це скоріше помилкою планування.

Зведення до мінімуму зовнішніх залежностей було одним з рішень, яке спочатку розглядалося, тому що компанія явно просила про це. У той час як це активно переслідувалося, використання зовнішньої бібліотеки для будь-якої функціональності завжди ставало вибором між компромісом між плюсами і мінусами її використання, як згадувалося в розділі «Які зовнішні бібліотеки використовувати» в розділі про результати. Наприклад, не була включена зовнішня бібліотека для рендеру, так як це не вважалося великою економією часу. Однак бібліотеки для графічного інтерфейсу, пристрої перекладу, імпорту об'єктів, сериализации і т. Д. Були включені, оскільки відтворення цих функцій з нуля було б трудомістким завданням, що негативно впливає на загальну якість проекту. Це було б схоже на винахід колеса. В цілому, всі включені зовнішні бібліотеки були залежностями, які були оцінені як необхідні для забезпечення успіху проекту у встановлені терміни.

Іноді при ухваленні рішення про те, які функції реалізувати, акцент робився на важливості функції в готовому продукті, а не на важливості функції щодо дослідницького питання цього проекту. Хорошим прикладом таких функцій є функції скасування / повтору і функції копіювання / вставки. Хоча

вони є необхідною функціональністю в будь-якому програмному забезпеченні для 3D-моделювання, вони не потрібні для реалізації в прототипі в тому сенсі, що вони не є необхідними для створення будь-яких анімацій. Їх реалізація не вносить прямого вкладу в питання дослідження; досить просто визнати, що вони повинні бути. Мало того, що на створення цих функцій було витрачено час, але після того, як додаток мало перетворитися на інструмент, в якому ви граєте в моделювання одним натисканням кнопки, вони також були перероблені, щоб працювати з новим підходом. Озираючись назад, можна сказати, що час, витрачений на все це, було б краще витратити на інші речі, наприклад, витратити більше часу на естетику графічного інтерфейсу.

2.8 Етичні аспекти

Спрощення створення таких анімацій для всіх може позбавити від необхідності людей з певним набором навичок. Хоча це може бути добре, так як додавання анімації буде більш доступним, це також може привести до деякої втрати ринкової вартості деяких людей. Більш того, підвищення доступності доступних інструментів 3D-моделювання може зробити цей ринок більш конкурентоспроможним, що може привести до втрати цінності деяких продуктів.

3 АНАЛІЗ ОТРИМАНИХ РЕЗУЛЬТАТІВ

У цьому розділі будуть описані результати, отримані в результаті цього проекту. Спочатку буде детально представлено розроблений інструмент 3D-моделювання.

3.1 Інструмент 3D-моделювання

Інструмент, створений в ході цього проекту, є прототипом високої якості. Більшість ключових функцій, таких як додавання різних об'єктів і управління ними, повністю реалізовані і працюють. Графічний інтерфейс користувача побудований таким чином, щоб візуалізувати розміщення елементів і результуючий робочий процес користувача, однак деякі естетичні аспекти, такі як кольори, не були частиною основного уваги.

Інструмент дозволяє користувачеві створювати анімацію за допомогою додатку Vivace, використовуючого ряд різних інструментів. Зовнішні об'єкти і анімацію можна імпортувати в додаток з деяких поширених форматів файлів. Можливість додавати ділянки тканини і різні геометричні об'єкти-колайдери ці функції доступні користувачеві одним або двома кліками миші. Створені таким чином об'єкти додаються в сцену з використанням параметрів, визначених додатком, які можуть бути змінені в будь-який час через графічний інтерфейс користувача з прямим управлінням. Об'єкти можна перетворювати і налаштовувати безліччю різних способів, щоб домогтися бажаної анімації.

У наступних розділах описуються реалізовані функції прототипу, а також те, як вони використовуються звичайним користувачем для створення анімації.

3.1.1 Почати перегляд

Початкове уявлення, показане на рис. 3.1, - це те, як додаток виглядає, коли користувач відкриває його. У верхній частині програми знаходиться рядок меню, що містить кілька загальних вкладок. У лівій частині програми знаходиться панель інструментів. Панель інструментів пропонує користувачеві безліч різних інструментів, таких як зміна тертя об'єкта на основі частинок, фіксація вершин, тобто ігнорування зовнішніх сил і перетворення об'єктів. На додаток до інструментів, дві кнопки розташовані у верхній частині панелі інструментів: одна для додавання деформованого патча в сцену, інша для додавання об'єкта коллайдера в сцену.

У правому верхньому куті є панель з вкладкою «Налаштування моделювання», яка обрана за замовчуванням. Панель містить кнопку «Відтворити симуляцію» - кнопку для запуску готової симуляції, а також різні глобальні параметри, які можуть бути налаштовані користувачем. Ці настройки включають включення вітру і гравітації з різними значеннями для кожної осі, управління різними параметрами камери і включення або відключення рендеру каркасів об'єкту. Конфігурація параметрів виконується прямим маніпулюванням.

Під панеллю «Налаштування моделювання» знаходиться панель «Ієрархія». Ця панель показує огляд ієрархії всіх об'єктів в сцені, яка, порожня під час запуску програми. У центрі екрана відображається візуалізована прямокутна площина. Ця площина використовується, щоб надати користувачеві візуальний погляд на поточний стан і орієнтацію камери.

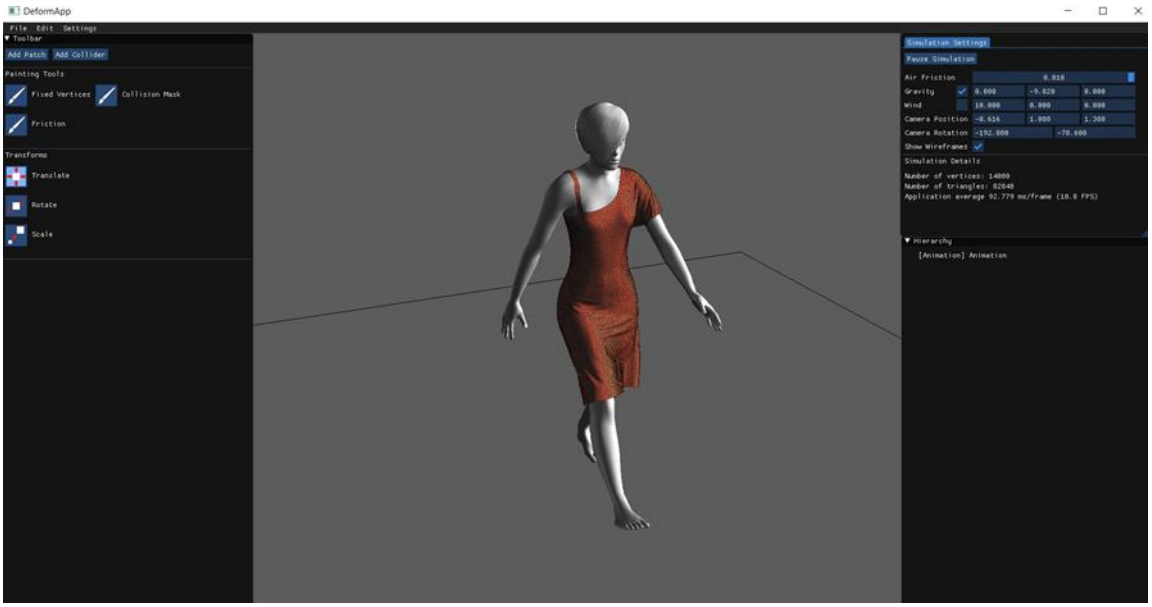

Рисунок 3.1 - Початковий вигляд програми, який показує сцену, яка містить імпортованих анімованих персонажів в сукнях

3.1.2 Перегляд відомостей про об'єкт

На рис. 3.2 показано, як виглядає додаток після того, як користувач додав, що деформується патч в сцену, натиснувши кнопку «Додати патч», а потім вибравши його, клацнувши об'єкт на екрані або клацнувши його ім'я в ієрархії. Вкладка «Відомості про об'єкт» показує деякі настроюються параметри для поточного обраного об'єкта. Приклади параметрів: розмір, положення, масштаб і поворот об'єкта. Подібно глобальним параметрам в поданні вкладки «Налаштування моделювання», конфігурація виконується шляхом прямого управління. Зміна одного з раніше згаданих параметрів призведе до негайного зміни порушеного об'єкта.

У центрі обраного об'єкта можна побачити Gizmo трансформації. Gizmo перетворення - це, зокрема, Gizmo перекладу, тому що інструмент перекладу активний в даний момент, як показано на рис. 3.2 зліва. Переключившись на один з інших інструментів перетворення, наприклад масштабування або обертання, відповідна Gizmo буде тією, яка відображається в позиції об'єкта. При перетягуванні елементів управління Gizmoі перетворення виконується

вздовж обраної осі. Ім'я об'єкта також виділяється в ієрархічному поданні справа, щоб візуалізувати, що об'єкт вибраний в даний момент.

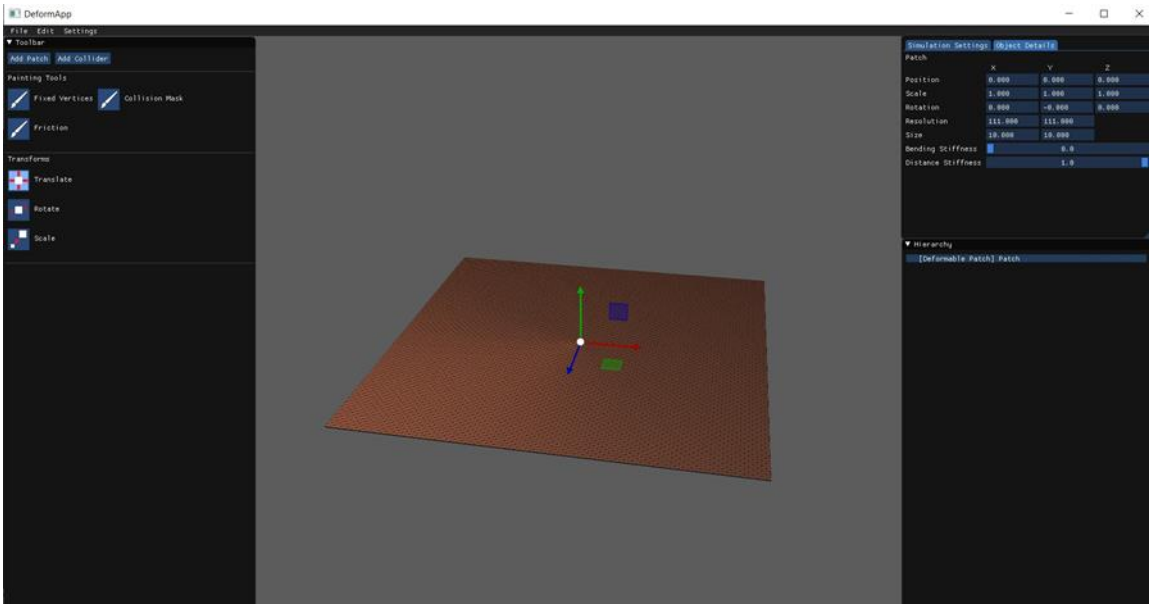

Рисунок 3.2 - Детальний вид об'єкта обраної ділянки, що деформується

3.1.3 Інструменти для малювання

Інструменти малювання - це набір інструментів, в яких користувач може застосувати конфігурацію для кожної частки до об'єкта, зафарбовуючи курсором миші. Кожна вершина, яка утворює об'єкт, являє собою єдину частку.

Коли користувач вибирає будь-який з інструментів малювання, панель, яка містить настройки для цього конкретного інструменту, з'являється під звичайною панеллю інструментів. Як видно на рисунку А.10, настройки інструменту малювання маски зіткнення включають зміна радіуса пензля і установку активної маски. Курсор миші також зміниться з курсора за умовчанням на білий кружок, що представляє кисть. Розмір курсору миші кисті змінюється в залежності від параметра радіуса пензля.

Кожна маска представлена кольором, який відображається на об'єкті, коли користувач зафарбовує частки / вершини. Пофарбовані частки для

кожного з інструментів малювання видно тільки тоді, коли цей конкретний інструмент в даний час активний. Кнопки «Застосувати до всіх» і «Очистити все» застосовують або очищають поточну обрану фарбу відповідно до всіх частинкам поточного обраного об'єкта.

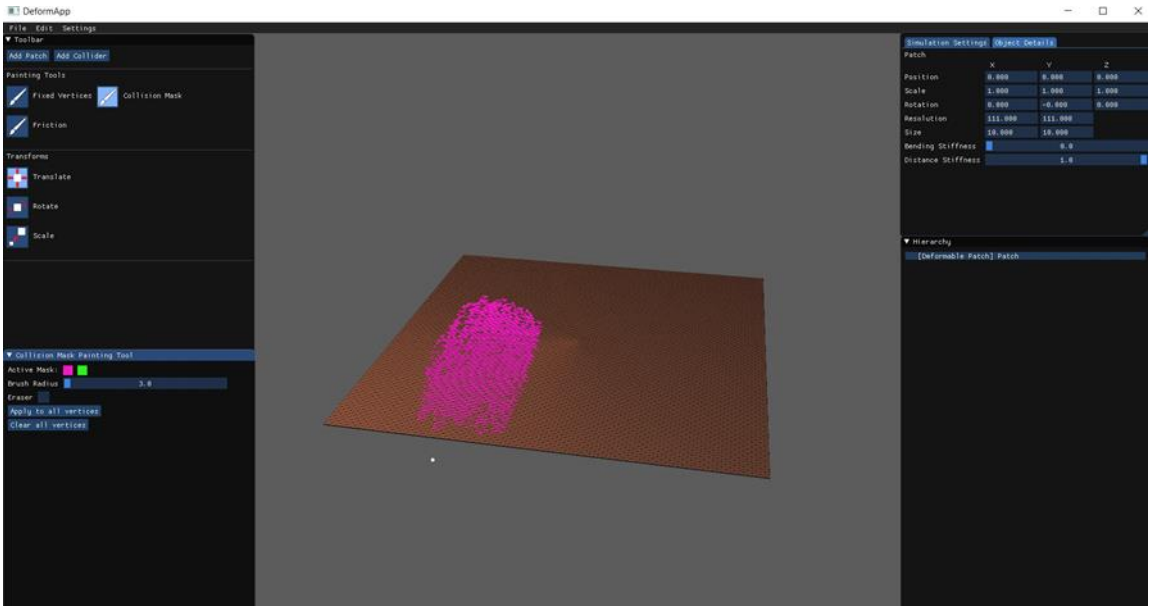

Рисунок 3.3 - Налаштування інструментів малювання маски зіткнення, показана в нижньому лівому кутку. Тут невелика частина патча була зафарбована рожевою фарбою

3.1.4 Запуск моделювання

Після настройки сцени моделювання можна запустити, натиснувши кнопку «Відтворити моделювання», розташовану вгорі в поданні «Параметри моделювання». Коли додаток знаходиться в режимі відтворення, ніякі інструменти не можна використовувати для застосування будь-яких змін до об'єкту в сцені. Якщо користувач бажає внести будь-які зміни в об'єкт, він повинен спочатку призупинити моделювання, натиснувши кнопку «Призупинити моделювання», розташовану в тому ж місці, де раніше була кнопка відтворення, щоб повернутися в режим редактора. . Глобальні

налаштування, такі як зміна значень сили тяжіння і вітру, можна змінювати під час моделювання.

У режимі відтворення камеру можна обертати, утримуючи ліву кнопку миші і перетягуючи курсор. Переміщення камери по осях x і y поточного поля зору можна виконати, утримуючи кнопку коліщатка миші і перетягуючи курсор. Клацання правою кнопкою миші при наведенні курсору на деформується об'єкт дозволить користувачеві перетягнути обрану частку об'єкта, як показано на рис. 3.4.

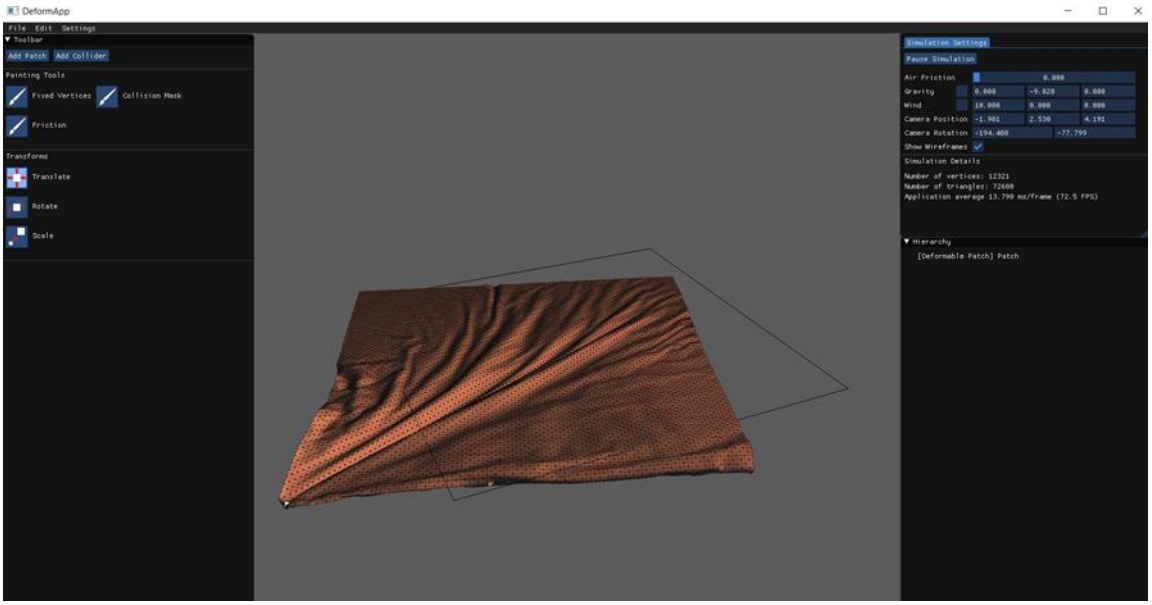

Рисунок 3.4 - На цьому рисунку показана деформована ділянка в процесі моделювання, коли користувач використовує пікіровку, щоб перетягнути її на вершиу вліво. Положення миші користувача відображається білою крапкою

3.1.5 Додавання об'єктів коллайдера

Якщо користувач натискає кнопку «Додати коллайдер» в правому верхньому кутку панелі інструментів, в місці, показаному на рис. 3.5, з'явиться невелике спливаюче вікно. Вікно запропонує користувачеві вибрати тип додається об'єкта-коллайдера, надавши окрему позначену кнопку для

кожного типу. Можна вибрати різні типи коллайдеров: обмежує прямокутник з вирівнюванням по осі, об'єктно-орієнтована обмежує прямокутник, сфера, капсула і площину. Крім того, якщо обраний деформується об'єкт, також існує можливість додати коллайдер сітки. При натисканні будь-якої з кнопок спливаюче вікно зникає, і обраний об'єкт коллайдера додається в центр сцени.

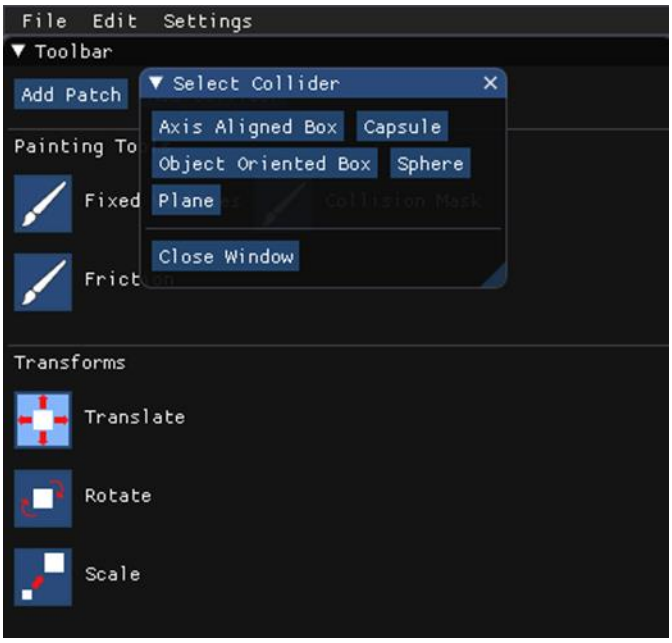

Рисунок 3.5 - На цьому рисунку показано спливаюче вікно для вибору типу коллайдера

3.1.6 Прив'язка об'єкта до коллайдера

До деяких об'єктах коллайдера може бути прикріплений деформується об'єкт, тобто вершини деформованого об'єкта, що лежать всередині коллайдера, будуть прикріплені до цього коллайдера. Ці колайдери матимуть текст, в якому перераховані всі прив'язані до них об'єкти під деталями об'єкта. Якщо до коллайдера не прив'язаний ні один об'єкт, прив'язаний об'єкт буде вказано як «NONE».

Щоб прив'язати деформується об'єкт до коллайдера, користувач повинен спочатку розмістити об'єкти таким чином, щоб коллайдер і деформується

об'єкт перекривали один одного. Потім користувач повинен вибрати коллайдер. Якщо коллайдер підтримує прив'язку, на панелі інструментів буде видно інструмент «Об'єкт прив'язки». Клацання по інструменту запропонує користувачеві вибрати об'єкт для прив'язки. Після вибору деформованого об'єкта зі сцени, клацнувши його безпосередньо або клацнувши його ім'я в ієрархії, інструмент буде активовано, і обраний об'єкт тепер буде вказано в докладному поданні для коллайдера, як показано на малюнку ffd.

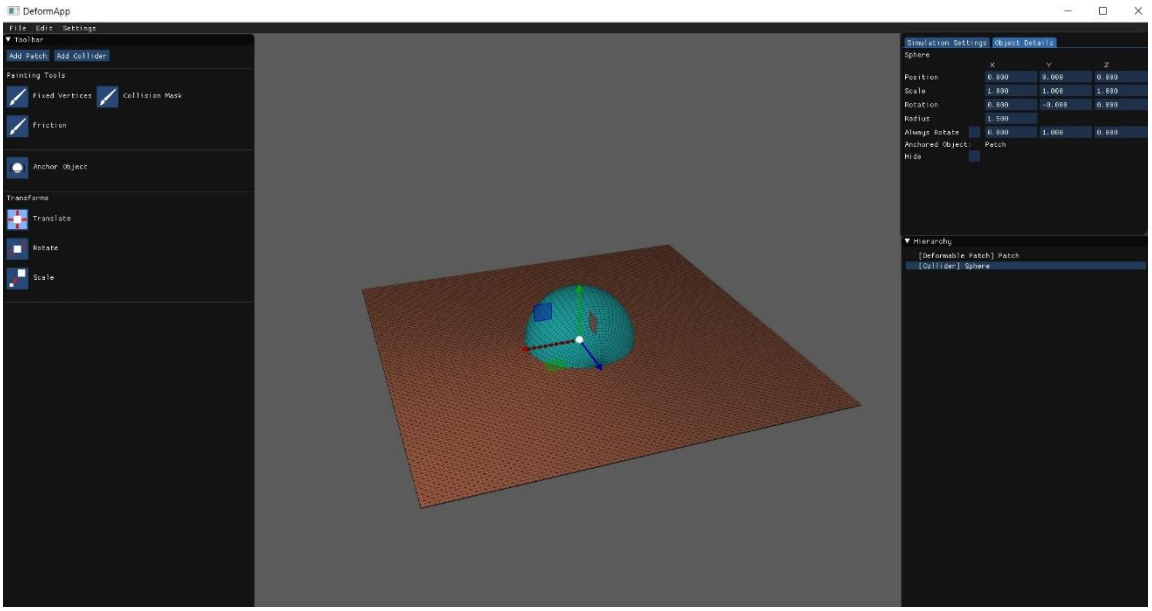

Рисунок 3.6 - Детальний вид обраного коллайдера, який показує закріплений об'єкт. Інструмент «Якірний об'єкт» також відображається на панелі інструментів

3.1.7 Файл

Елемент рядка меню «Файл» - це перший елемент, розташований в рядку меню у верхньому лівому кутку програми. Меню, що випадає «Файл» містить параметри для запуску нового проекту, відкриття раніше збереженого проекту, збереження поточного проекту, імпорту різних типів геометрії, експорту моделювання і виходу з програми, як показано в лівій

частині рис. 3.7. Швидкий доступ для кожного пункту меню затемнений поруч з його звичайним текстом.

При натисканні на два пункти меню відкривається підміню. Це поведінку можна визначити по стрілці в дальньому правому куті пункту меню, що вказує в тому ж напрямку. Для пункту «Відкрити недавні» в цьому підменю будуть перераховані раніше відкриті файли проекту. Для пункту «Імпорт» в підміню будуть перераховані різні типи речей, які користувач може імпортувати в додаток. Як видно на правій стороні рис. 3.7, це деформуються об'єкти, анімації, об'ємні сітки і сцени.

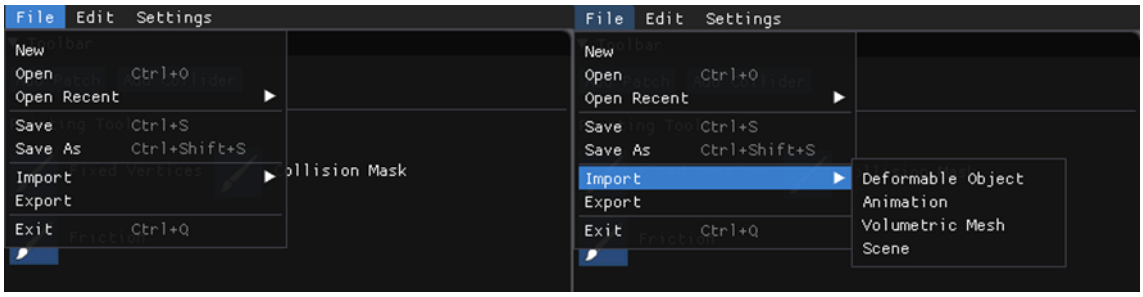

Рисунок 3.7 - У лівій частині рисунку показано спадне меню файлу. У правій частині показано підміню, яке відкривається при натисканні на пункт меню «Імпорт»

3.1.8 Імпорт анімації

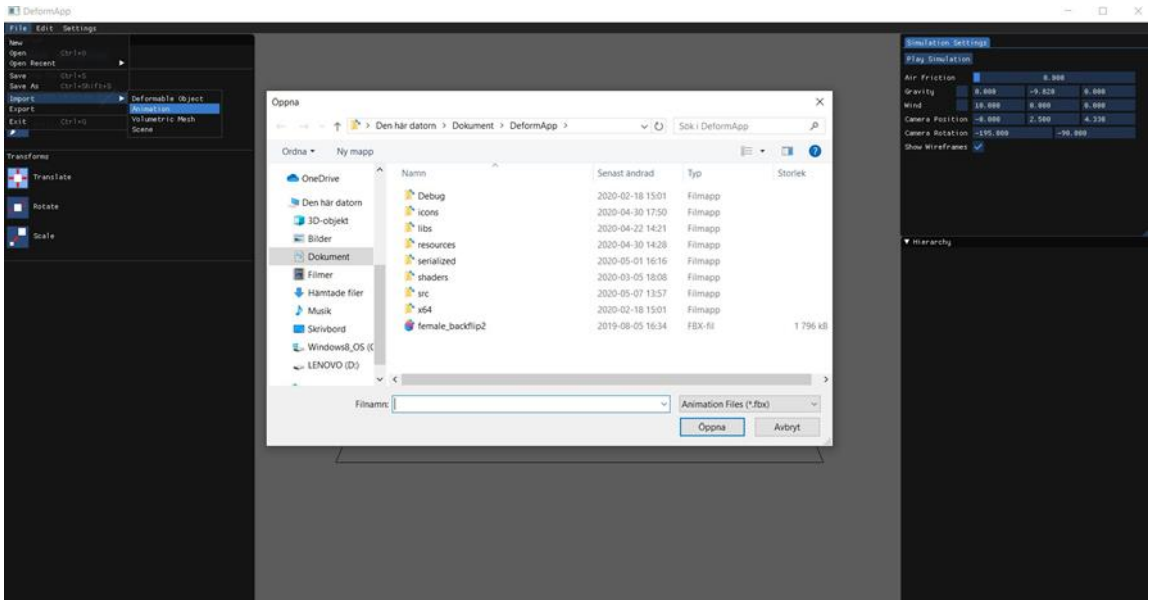

Рисунок 3.8 - На рисунку показаний діалогове меню відкриття файлу Windows, яке з'являється при спробі імпортувати анімацію

Імпорт анімації виконується клацанням по пункту «Анімація» в підміню, що відкривається при натисканні на «Імпорт» в меню «Файл». При натисканні по елементу «Анімація» з'явиться стандартний діалог файлів Windows, що пропонує користувачеві вибрати файл. Можуть бути обрані тільки файли з відповідними закінченнями, в даному випадку файли .fbx, які є поширеним форматом файлів для анімації. Це показано на рис. 3.8. Як тільки файл анімації був відкритий, або вибравши елемент і натиснувши «Відкрити», або двічі клацнувши елемент, то вони будуть додані в сцену, якщо файл містить дійсні дані. Цей крок ідентичний кроків для імпорту інших об'єктів, різниця полягає в закінченні файлу.

Коли анімація успішно імпортована в додаток, вона буде відображатися як статичний об'єкт в центрі сцени. Перегляд відомостей про об'єкт для імпортованих анімації буде містити повзунок, який представляє час анімації в її поточному стані, як показано на малюнку 5.9. Повзунок дозволить користувачеві візуалізувати всю анімацію, кадр за кадром.

Інший настроюється параметр, специфічний для анімації, - це можливість завантажувати серіалізовані дані, що також можна побачити зліва на рис. 3.9. Ця функція дозволяє користувачам створювати власні, що містить серіалізовані дані про одяг, який повинен використовувати анімований персонаж. Подібно до функцій імпорту, натискання кнопки «Завантажити серіалізовані дані» відкриє діалогове вікно файлу Windows, що пропонує можливість користувачам створювати власні з відповідним закінченням файлу. Замість кнопки буде відображена успішно завантажена конфігурація, як показано праворуч на рис 3.9. Значок кошика поруч з ім'ям завантаженого файлу даних дозволяє користувачеві видалити завантажені дані.

Щоб відтворити імпортовану анімацію, користувач повинен запустити всю симуляцію, натиснувши кнопку «Відтворити симуляцію».

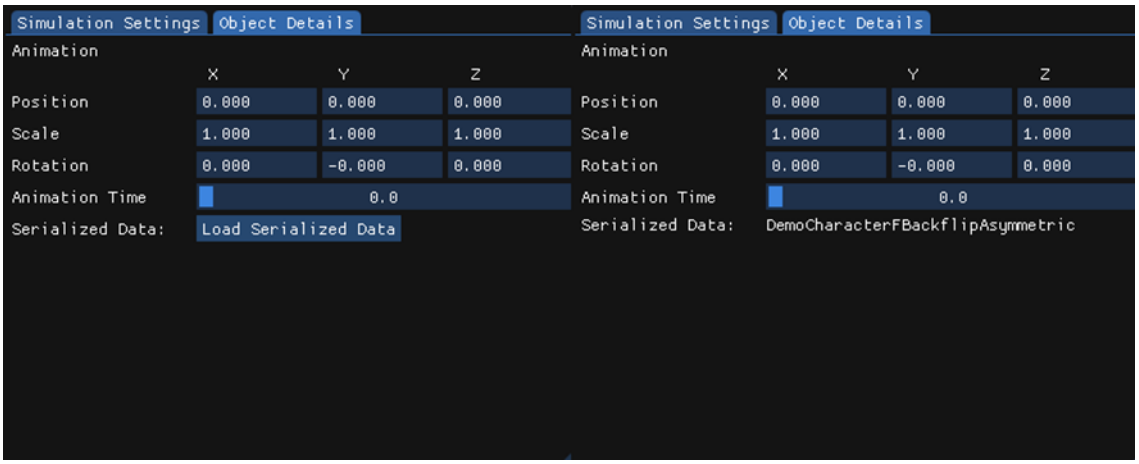

Рисунок 3.9 - Ліва частина рисунку показує деталі анімації без завантажених серіалізованих даних. У правій частині показаний той же вид після завантаження серіалізованих даних

3.1.9 Редагувати

Пункт меню «Редагувати» відкриває спливаюче меню, що містить деякі зручні для користувача функції, показані на рис 3.10. Ці функції - відміна, яка скасовує останню дію, повторення, яке повторює останню скасовану дію,

вирізання, яке копіює і видаляє вибрані об'єкти, копіювання, яке копіює вибрані об'єкти, вставка, яка вставляє скопійовані об'єкти, і видалення, яке видаляє вибрані об'єкти зі сцени . Ярлики клавіатури для кожної функції вказані поруч з їх назвою.

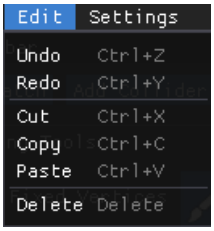

Рисунок 3.10 - На рисунку показано спливаюче меню, що з'являється при виборі пункту меню «Редагувати» в верхньому рядку меню

ВИСНОВОК

Тривимірне моделювання - це велика область, в якій концепції варіюються від створення анімованих моделей персонажів до відтворення різних фізичних ефектів в анімації. Сьогодні він широко використовується в різних галузях, таких як ігрова індустрія, кіноіндустрія і архітектура, що збільшує попит на програмне забезпечення для 3D-моделювання.

Метою даного дипломного проєкту було спробувати відповісти на наступний дослідницький питання:

Що слід враховувати при розробці та впровадженні інтерактивного інструменту тривимірного моделювання для створення універсальних м'яких тіл і управління ними?

Для цього був розроблений інтерактивний інструмент тривимірного моделювання для створення універсальних м'яких тіл і управління ними відповідно до досліджень за допомогою методології проектування. Однією з основних функцій інструменту є можливість створювати симуляції, включаючи реалістично виглядають фізику тканини. Результуючі анімації можуть бути простими, наприклад, зіткнення ділянок тканини з різними геометричними фігурами, або більш складними, з анімованими одягненими персонажами, які виконують деякі фізичні дії. Останній варіант зажадає імпорту анімованого персонажа і його одягу з локального файлу. Результуюча анімація імітує ефекти реальної фізики з правдоподібним результатом з надзвичайно високою швидкістю.

Перед початком розробки були ретельно проведені дослідження в цій області, щоб отримати гарне уявлення про все, що стосується інструментів 3D-моделювання. Це дослідження включало вивчення не тільки існуючих в даний час інструментів, але також відповідної літератури і попередніх досліджень, проведених в цій галузі. Це дослідження, поряд зі знаннями, отриманими при розробці інструменту 3D-моделювання, потім було використано в якості основи для відповіді на питання дослідження.

Як і у випадку з більшістю інших тем в області дизайну взаємодії, на таке питання ніколи не може бути однозначної відповіді. Замість цього результатом цього проекту є набір керівних принципів, які слід враховувати при розробці іншими інструментами. Ці керівні принципи, які слід враховувати, можна резюмувати наступним чином: які зовнішні бібліотеки використовувати, який робочий процес використовувати, які функції повинні бути доступні, як проектувати для різних груп користувачів і як зробити код модульним. Хоча ці рекомендації були результатом розробки цього інструменту, вони жодним чином не можна назвати єдиними остаточними рекомендаціями, які можуть виникнути при розробці інструменту 3D-моделювання. Замість цього вони вважаються хорошим орієнтиром для тих, хто розробляє власне програмне забезпечення для 3D-моделювання. Незважаючи на те, що багато функцій поновлюваного додатки Що стосується анімації одягу, то вважається, що отримані рекомендації застосовні до всіх інструментів тривимірного моделювання.

ПЕРЕЛІК ПОСИЛАНЬ

1. 3D – моделирование в Google SketchUp – от простого к сложному. / Петлин А. Издательство ДМК, г. Москва: 2014. – 344 с.
2. Самоучитель Blender 2.7. / Прахов А. БХВ Петербург, г. Санкт-Петербург: 2016. – 399 с.
3. Введение в ZBrush 4. / Эрик Келлер. Издательство ДМК, г. Москва: 2012. – 769 с.
4. Самоучитель 3ds Max. / Горелик Александр БХВ Петербург, г. Санкт-Петербург: 2018. – 529 с.
5. 3ds Max. Трехмерное моделирование и анимация на примерах. / Тозик В., Меженин А., Звягин К. БХВ Петербург, г. Санкт-Петербург: 2008 – 876 с.
6. 3DToDay [Електронний ресурс] – Режим доступу: https://3dtoday.ru
7. LiveJournal [Електронний ресурс] – Режим доступу: https://livejournal.com
8. MOTOCMS [Електронний ресурс] – Режим доступу: https://www.motocms.com
9. BRAINRAIN [Електронний ресурс] – Режим доступу: https://brainrain.com.ua
10. Программирование. Принципы и практика использования C++. / Бьерн Страуструп. Издательский дом “Вильямс”, г. Москва, Санкт-Петербург, Киев: 2011. – 1223 с.
11. Язык программирования C. / Брайан Керниган и Деннис Ритчи. Издательский дом “Вильямс”, г. Москва, Киев: 2007. – 304 с.
12. Изучаем программирование на Python. / Пол Бэрри. Издательство «Э», г.Москва: 2017. – 611 с.
13. Глубокое обучение на Python. / Шолле Франсуа. Издательство «Питер», г.Москва: 2018. – 400 с.

14. Python. Книга рецептов. / Брайан К. Джонс и Дэвид Бизли. ДМК Пресс, г.Москва: 2019. – 650 с.

15. A Byte of Python [Електронний ресурс] – Режим доступу: https://python.swaroopch.com

16. Learnpython [Електронний ресурс] – Режим доступу: https://www.learnpython.org

17. Udemy [Електронний ресурс] – Режим доступу: https://www.udemy.com

18. CodeCademy [Електронний ресурс] – Режим доступу: https://www.codecademy.com

19. Learn-Js [Електронний ресурс] – Режим доступу: https://www.learn-js.org

20. JavaScript [Електронний ресурс] – Режим доступу: https://learn.javascript.ru

21. wikipedia [Електронний ресурс] – Режим доступу: https://ru.wikipedia.org

22. 3DPulse [Електронний ресурс] – Режим доступу: https://www.3dpulse.ru

23. 3D Mini Factory [Електронний ресурс] – Режим доступу: https://3dmf.ru

24. Двухмерное и трехмерное моделирование в 3ds max. / Аббасов И.Б. Издательство ДМК, г.Москва: 2012. – 176 с.

25. Maya 6 для windows и macintosh. / Риддел Д. Издательство ДМК, г.Москва: 2013. – 592 с.

26. V-ray для autodesk maya. Руководство по визуализации. / Чехлов Д.А. Издательство ДМК, г.Москва: 2020. – 808 с.

27. Шаг комп’ютерна Академія [Електронний ресурс] – Режим доступу: https://sumy.itstep.org