Національний технічний університет України

«Київський політехнічний інститут імені Ігоря Сікорського»

Навчально-науковий інститут аерокосмічних технологій

Кафедра авіа- та ракетобудування

До захисту допущено
В. о. завідувача кафедри
Олександр БОНДАРЕНКО

«__» _____ 2023 р.

Дипломний проект

на здобуття ступеня бакалавра
за освітньо-професійною програмою «Літаки і вертольоти»
спеціальності 134 «Авіаційна та ракетно-космічна техніка»
на тему: «Вибір основних конструктивно -технологічних параметрів орнітоптера»

Виконав:
студент IV курсу, групи АЛ-91
Чижик Даніїл Олегович _____

Керівник:
Доцент, к.т.н.
Бондаренко Олександр Миколайович _____

Рецензент:
Асистент
Осокін Владислав Сергійович _____

Засвідчую, що у цьому дипломному проєкті немає запозичень з праць інших авторів без відповідних посилань.
Студент _____

Київ – 2023 року
Національний технічний університет України

«Київський політехнічний інститут імені Ігоря Сікорського»

Навчально-науковий інститут аерокосмічних технологій

Кафедра авіа- та ракетобудування

Рівень вищої освіти –перший (бакалаврський)
Спеціальність –134 «Авіаційна та ракетно-космічна техніка»
Освітньо-професійна програма «Літаки і вертольоти»

ЗАТВЕРДЖУЮ
В. о. завідувача кафедри
_____ Олександр БОНДАРЕНКО
«____» _____ 2022 р.

ЗАВДАННЯ

на дипломний проєкт студенту

Чижику Даніілу Олеговичу

(прізвище, ім'я, по батькові)

1. Тема проєкту «**Вибір основних конструктивно -технологічних параметрів орнітоптера**», керівник проєкту Бондаренко Олександр Миколайович, к.т.н., затверджені наказом по університету від «___»_____ 2023 р. №_____

2. Термін подання студентом проєкту *16 червня 2023 р.*

3. Вихідні дані до проєкту:_____

3.1 Висота польоту H=0…300м.

*3.2. Максимальна швидкість польоту V= 20 км/год.*_

3.3. Польотна вага не більше $m_{пол}$ *= 2 кг.*

3.4. Тривалість польоту не менше $t_{пол}$ *= 20 хв.*

4. Зміст пояснювальної записки:_____

4.1. Вибір варіанту побудови.

4.2. Опис функціональної схеми та конструкції.

4.3. Моделювання теоретичної поверхні.

4.4. Розроблення програмного інтерфейсу для моделювання та розрахунку.

4.5. Розрахунок аеродинамічних характеристик

4.6. Оцінка льотно-технічних характеристик.

4.7. Розроблення технології та оснащення.

5. Перелік графічного (ілюстративного) матеріалу (із зазначенням обов'язкових креслеників, плакатів, презентацій тощо):

5.1 Огляд аналогів.

5.2 Моделювання крила .

5.3 Результати розрахунку.

5.4 Конструкція літального апарату.

6. Консультанти розділів проєкту

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

7. Дата видачі завдання: *1 лютого 2023 р.*

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів виконання дипломного проєкту	Термін виконання етапів проєкту	Примітка
1.	*Підбір та аналіз літератури*	*до 15.03.2022 р.*	
2.	*Огляд та аналіз даних аналогічних літальних апаратів*	*до 25.03.2022 р.*	
3.	*Моделювання теоретичної поверхні*	*до 10.04.2022 р.*	
4.	*Розрахунок аеродинамічних характеристик*	*до 20.04.2022 р.*	
5.	*Проектування розрахункового інтерфейсу*	*до 5.05.2022 р.*	
6.	*Розрахунок льотно-технічних характеристик*	*до 15.05.2022 р.*	
7.	*Проєктування конструкції, розробка технології виготовлення*	*до 24.05.2022 р.*	
8.	*Оформлення пояснювальної записки та графічних матеріалів*	*до 27.05.2022 р.*	
9.	*Оформлення пояснювальної записки та графічних матеріалів. Відгук керівника ДП*	*до 06.06.2022 р.*	
10.	*Перевірка ПЗ до дипломного проекту на академічну доброчесність (плагіат)*	*до 10.06.2022 р.*	
11.	*Подання дипломного проекту на рецензування*	*до 12.06.2022 р.*	
12.	*Захист дипломного проекту*	*з 14.06.2022 р. по 17.06.2022 р*	

Студент _____ Данііл ЧИЖИК

Керівник _____ Олександр БОНДАРЕНКО

ВІДОМІСТЬ ДИПЛОМНОГО ПРОЕКТУ

№ з/п	Формат	Позначення	Найменування	Кількість аркушів	Примітка
1	А4		Завдання на дипломний проект	2	
2	А4	АЛ9117.17.00.00.00 ПЗ	Пояснювальна записка	63	
3	А1	АЛ9117.17.00.00.01	Блок схема алгоритму візуалізації	1	
4	А1	АЛ9117.17.00.00.02	Огляд анімованої моделі	1	
5	А1	АЛ9117.17.00.00.03	Результати аеродинамічної симуляції	1	

		П.І.Б.	Підп.	Дата	АЛ9117.17.00.00.00
Розробив		Чижик Д.О.			Відомість дипломного проекту
Перевірив		Бондаренко О.М.			Аркуш 4 / Аркушів 66
					КПІ ім. Ігоря Сікорського
Н. контр.		Поваров С.А.			Каф. АРБ Гр. АЛ-91
Зав. каф.		Бондаренко О.М.			

Реферат

Пояснювальна записка до диплому з теми «Вибір основних конструктивно - технологічних параметрів орнітоптера» містить 66 сторінок, 50 ілюстрацій, 1 таблицю, 9 використаних джерел та три додатки.

Метою дипломного проекту є дослідження та розробка конструкції орнітоптера та визначення його основних характеристик, а саме: конструктивних, технологічних та аеродинамічних.

У дипломному проекті був проведений аналіз характеристик існуючих орнітоптерів, на його основі був обраний прототип для подальшого проектування. Була розроблена модель орнітоптера, проведено глибокий аналіз аеродинамічних характеристик, використовуючи різноманітне програмне забезпечення. Для проектування та проведення подальших розрахунків була створена 3D модель літального апарату та додатки для візуалізації.

Ключові слова: ЛА, орнітоптер, візуалізація, триангуляція, моделювання, симуляція.

Abstract

The explanatory note to the diploma on the topic "Selection of the main structural and technological parameters of the ornithopter" contains 66 pages, 50 illustrations, 1 table, 9 references and three appendices.

The purpose of the diploma project is to research and develop the design of the ornithopter and determine its main characteristics, namely: structural, technological and aerodynamic.

In the thesis project, the characteristics of existing ornithopters were analyzed, and a prototype was selected for further design. A model of the ornithopter was developed, and an in-depth analysis of aerodynamic characteristics was conducted using a variety of software. A 3D model of the aircraft and visualization applications were created for design and further calculations.

Keywords: Aircraft, ornithopter, visualization, triangulation, modeling, simulation.

Зміст

Список умовних позначень та скорочень....7
Вступ....7
1 Аналіз літератури....8
1.1 Стан проблеми та напрямки її розвитку....9
2 Огляд аналогів....10
2.1 Орнітоптер Snowbird....11
2.2 Біонічний орнітоптер Festo BionicSwift....12
2.3 Орнітоптер розроблений JSK-koubou....15
2.4 Таблиці основних характеристик аналогів....17
3 Вибір варіанту побудови....17
3.1 Опис функціональної схеми та конструкції....18
4 Розроблення програмного інтерфейсу для моделювання та розрахунку....18
4.1 Апроксимація за допомогою триангуляції....18
4.1.1 Динамічне створення інтерфейсних елементів....21
4.2 Вибір мови програмування C++ для візуалізації: критичні фактори та переваги....28
4.3 Візуалізація триангуляції Делоне за допомогою C++....29
5 Моделювання 3D моделі....40
5.1 Організація нового проекту....40
5.2 Завантаження референсів....41
5.2.1 Створення примітивних форм на фоні референсу....42
5.3 Підготовка моделі до накладання шейдерів....49
5.4 Створення скелета за допомогою джоінтів....54
5.4.1 Прив’язка джоінтів до примітивів та встановлення функціоналу згину суставів....58
5.5 Анімація моделі....59
6 Симуляція аеродинаміки отриманої моделі....61
Висновок....65
Список використаних джерел....65

Змн.	Арк.	№ докум.	Підпис	Дата			
Розробив		*Чижик Д.О.*			***Вибір основних конструктивно - технологічних параметрів орнітоптера***	Літ.	Арк. 1 / Акрушів
Перевірив		*Бондаренко О. М.*					
						НН ІАТ НТУУ «КПІ» ім. І.Сікорського	

Список умовних позначень та скорочень

ICP - Інтегроване середовище розробки

ЛА – Літальний апарат

Референс (reference) – передній вид та вид зліва\зправа нашого теоретичного об’єкта

Фліп (flip) – поворот моделі чи її частини

Джоінт (joint) – вузол, точка зв’язки, сполучник об’єктів, - основні рухомі точки, з яких складається скелет (рухи згину, обертання, переміщення, тощо...)

Рут (root) – корінь, корінний джоінт

Бінд (bind) – окреслені залежності, встановлені людиною чи комп’ютером алгоритми, за якими буде слідувати програма

Шейдери (shaders) - Алгоритм для відеокарти, який створює візуалізацію поверхні для людського ока, простіше кажучи, код, який розуміє відеокарта

Вступ

На даний час в Україні є велика потреба у проектуванні та складанні малих легких літальних апаратів які можуть виконувати багато різних задач. Одним з найактуальніших напрямів використання є аеророзвідка, що в свою чергу включає таку задачу як спостереження за конкретними об’єктами або певною територією короткий час. Новим типом легких, в тому числі, розвідувальних літальних апаратів можуть стати орнітоптери.

Метою дипломного проекту було визначення основних конструктивно-технологічних характеристик орнітоптера. Впродовж виконання роботи використовувались різні методи розробки та конструктивні рішення, що застосовувались у подібних літальних апаратів.

В першому розділі був проведений комплексний аналіз літератури, проблем та їх розвиток.

В другому розділі здійснено широкий огляд конструкцій подібних літальних апаратів, обрано прототип для подальшого моделювання та симуляції.

В третьому розділі описано вибір варіанту орнітоптера та описано функціональну схему і конструкцію.

В четвертому розділі було розроблено декілька додатків для проведення триангуляції та візуалізації.

В п’ятому розділі представлено повний шлях розробки моделі.

В шостому розділі на основі попереднього моделювання було проведено симуляції аеродинаміки моделі.

1 Аналіз літератури

Орнітоптери - це безпілотні ЛА, що імітують політ птахів. Вони привертають увагу завдяки своєму потенціалу для використання в різних галузях, включно з дослідженнями, спостереженням, пошуком і порятунком. Однак для забезпечення ефективної роботи орнітоптерів необхідно вирішити низку складних питань, серед яких вирішальне значення мають базова конструкція та вибір технічних параметрів.

Вибір конструкції та технічних параметрів визначає характеристики та експлуатаційні якості орнітоптера, такі як маневреність, швидкість, стійкість і дальність польоту. Правильний вибір цих параметрів - важливе завдання, що вимагає детального аналізу, досліджень і експериментів.

Поточні дослідження в цій галузі спрямовані на розв'язання цих проблем шляхом розроблення нових методів, алгоритмів і технологій. Засоби комп'ютерного моделювання та імітації можуть бути використані для швидкої та ефективної оцінки впливу різних параметрів конструкції на характеристики орнітоптера. Дослідження нових матеріалів і технологій конструювання спрямовані на зниження ваги, підвищення міцності та стійкості до зовнішніх

впливів. Для підвищення енергоефективності, точності та стабільності польоту можуть бути розроблені нові приводи і системи управління.

1.1 Стан проблеми та напрямки її розвитку

Сьогодні орнітоптери мають кілька важливих конструктивних і технічних параметрів, які потребують уваги та досліджень. Ось деякі з них:

Розмір і форма крил важливі для ефективного польоту орнітоптера. Неправильний вибір цих параметрів може призвести до нестійкого польоту, поганої маневреності або обмеженого чи навіть неможливого польоту.

Вибір відповідних матеріалів для конструкції орнітоптера визначає його вагу, міцність і стійкість до зовнішніх впливів. Розробка та використання новітніх матеріалів з високою міцністю і низькою вагою є важливим аспектом розвитку орнітоптера.

Конструкція приводу(двигун та механізм). Двигун для орнітоптера, чи то електричний, гібридний або паливний, визначає його енергоефективність, швидкість і дальність польоту. Також є важливим сам механізм перетворення потужності двигуна у послідовні рухи крила. Використання та розробка нових, ефективних двигунів та механізмів перетворення важлива для поліпшення характеристик орнітоптера.

Ефективна система управління відіграє важливу роль у забезпеченні стабільного і точного польоту орнітоптера. Розробка нових алгоритмів керування та автоматичної стабілізації є важливим аспектом вдосконалення цих систем.

Для подальшого розвитку орнітоптера та розв'язання проблем, пов'язаних із проектуванням і вибором технічних параметрів, можна виокремити декілька напрямів досліджень, а саме:

Моделювання та симуляція. Використання засобів комп'ютерного моделювання та симуляції дає змогу швидко та ефективно оцінити вплив різних параметрів конструкції на характеристики орнітоптера. Це дає змогу

точніше підібрати параметри та скоротити час і вартість експериментальних досліджень.

Розробка нових приводів. Виробництво більш ефективних і легких приводів на основі новітніх технологій дасть змогу поліпшити швидкість, дальність і стабільність польоту.

Дослідження алгоритмів управління. Розробка нових алгоритмів управління та автоматичної стабілізації дасть змогу забезпечити більш точний і стабільний політ орнітоптерів. Використання штучного інтелекту та машинного навчання може покращити автономність та адаптивність системи управління.

2 Огляд аналогів

Політ птахів часто порівнюють із польотом літаків. Це порівняння можна проводити лише до певної міри, оскільки між польотом на крилі, що махає, і польотом на нерухомому крилі існує безліч відмінностей. Це пов'язано з аеродинамічними силами, які виникають під час руху вперед, як і в літаку. Ці сили мають дві складові: опір, який намагається сповільнити рух уперед, і підйомна сила, яка піднімає крила і тіло птаха.

Підйомна сила створюється здебільшого за рахунок вигину тіла і зап'ястя, а саме зап'ястя з довгим пір'ям крил злегка перевертається при закручуванні в повітря під час польоту, створюючи тягу. Таким чином, основна відмінність між птахом і літаком полягає в тому, що крила птаха поєднують у собі функцію повітряного гвинта літака і несучої поверхні для створення тяги. Таким чином, основна відмінність між птахами і літаками полягає в тому, що крило птаха поєднує в собі функцію пропелера літака з функцією несучої поверхні, що створює підйомну силу. Не вдаючись у подробиці, слід зазначити, що ідея про те, що птахи піднімаються, опускаючи крила, і опускаються, піднімаючи крила, абсолютно невірна.

Тому в птахів немає пристосувань для зменшення опору повітря під час підйому крил. Під час підйому й опускання крил сила діє проти сили тяжіння.

Але тяга виникає тільки під час опускання крил, і тільки на кінчиках крил, і врівноважується опором, що виникає під час змаху.

2.1 Орнітоптер Snowbird

З незапам'ятних часів незліченні ентузіасти мріяли літати як птах, використовуючи власну силу. Сотні й навіть тисячі намагалися здійснити її. У 2010 році в Канаді було розроблено апарат, який літає за рахунок руху крил, так званий орнітоптер (махоліт). Він може літати не завдяки пропелерам чи двигунам, а завдяки руху крил. Формула обчислення потужності, необхідної для підтримки планера повітря, виглядає так: $N = \frac{G \times \omega}{75}$ де,

G - маса ЛА разом з пілотом, кг;

ω - швидкість зниження апарату, м/с;

Канадський вчений Тодд Райкхарт, доктор інженерних наук аерокосмічного інституту університету Торонто, все ж таки зміг зробити працюючий орнітоптер (рис 1.1) і навіть випробував його в роботі. Літальний апарат отримав назву Snowbird. Він важить 42 кг і створений із вуглеволокна, піноматеріалу та коркового дерева. Розмах крил орнітоптерів – 32 м. Профіль має 4 зчленування у поперечній площині.

Прискорення оригінального літака Snowbird робив автомобіль, який виконував функцію тягача, але потім відірвавшись від землі, орнітоптер летів сам. Помахи крил устрою забезпечував сам Райкхарт, який крутив педалі. Апарат подолав таким чином відстань майже 230 метрів зі швидкістю 25 кілометрів на годину, а молодий вчений своїм прикладом довів, що нічого неможливого немає.

Рис. 1.1 – Орнітоптер Snowbird

2.2 Біонічний орнітоптер Festo BionicSwift

У 2020 році компанією Festo було створено орнітоптер у вигляді стрижа (Рис 2.1 (а, б, в, г)). При конструюванні цього орнітоптера особлива увага приділялася використанню легких та надлегких конструкцій і матеріалів. В техніці, як і в природі, чим менша вага для переміщення, тим менше витрата матеріалів та енергії. Тому вага цього механічного птаха всього 42 грами при довжині тіла 44.5 сантиметрів.

Рис. 2.1 (а) – Вид знизу

Рис. 2.1 (б) – Вид зверху

Рис. 2.1 (в) – Вид ззаду

Рис. 2.1 (г) – Вид попереду

Щоб маневри здійснити максимально реалістично, крила створені за зразком оперення птахів. Окремі планки виготовлені з ультралегкого, гнучкого, але дуже міцного пінопласту і лежать одна на одній, як черепиця. Коли крило піднято, окремі планки роздуваються(Рис 2.2), щоб повітря могло протікати через крило. В результаті птахам потрібно менше зусиль, щоб підтягнути крило вгору. Під час ходу вниз планки закриваються, щоб літаючі роботи могли літати потужніше.

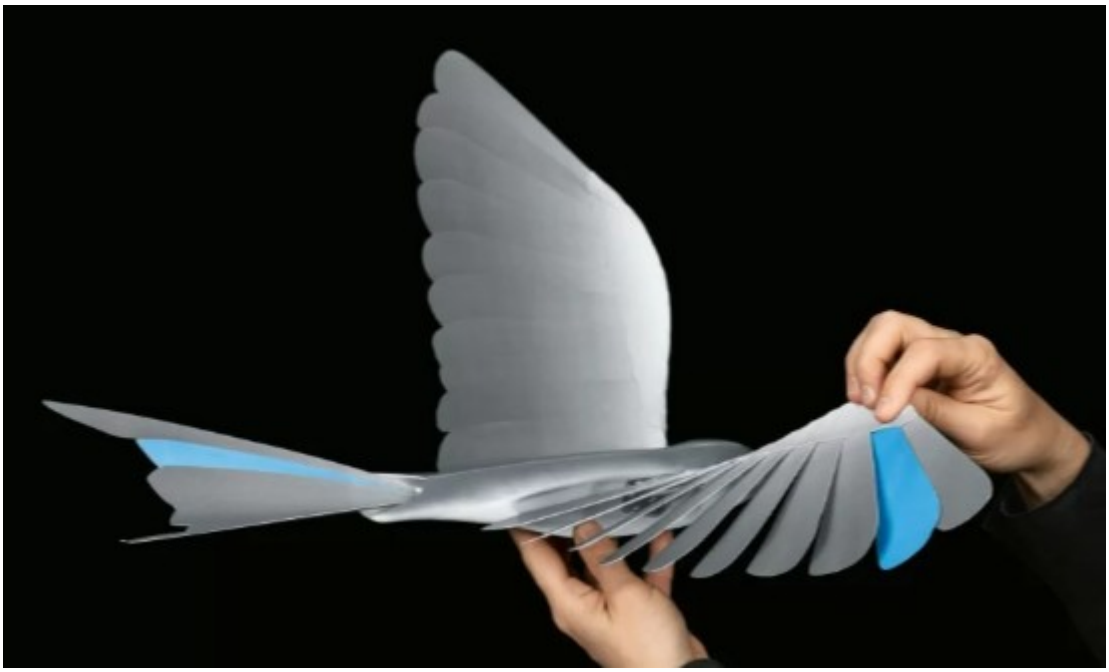

Рис. 2.2 – Крутильна здатність крил

Тіло птаха містить компактну конструкцію механізму махового крила, комунікаційну технологію та компоненти керування маханнями крил і поворотом хвоста. Двигун, два сервомотори, акумулятор, передатчики та різні плати для зв'язку і керування встановлені на дуже маленькому просторі. Інтелектуальна взаємодія двигунів і механіки дозволяє, наприклад, точно регулювати частоту помахів крил і кут атаки для різних маневрів.

Внутрішній радіоприймач GPS забезпечує скоординований та безпечний політ BionicSwifts. Для цього в кімнаті встановлюють кілька радіо модулів. Потім ці модулі знаходять один одного і визначають контрольований повітряний простір. Крім того, кожна пташка-робот оснащена радіомаркером. Це посилає сигнали на модулі, які потім можуть визначити точне положення птаха та відправити зібрані дані на центральний комп’ютер(Рис. 2.3), який діє як навігаційна система.

Рис. 2.3 - Головний комп'ютер, радіо модуль і літаючі об'єкти у взаємодії

На цьому можна здійснити планування маршруту, щоб заздалегідь запрограмовані шляхи вказували маршрут польоту птахів. Якщо птахи відхиляються від своєї траєкторії польоту через раптово змінювані впливи навколишнього середовища, такі як вітер або повітряні потоки, вони негайно коригують свій шлях польоту самі і автономно втручаються в цю ситуацію — без людини-пілота. Радіозв'язок дає змогу точно визначати місцезнаходження, навіть якщо візуальний контакт частково переривається. Інтелектуальна мережа літаючого об'єкта та GPS-маршрутів створює 3D-навігаційну систему, по якій можливо визначити координати об'єктів у будь який момент часу.

2.3 Орнітоптер розроблений JSK-koubou

Компанія JSK-koubou створила орнітоптер, який схожий на механічного птаха, в якому здійснюється перелом поздовжньої осі крила (Рис. 2.4(а.б,в)). Рух здійснює відокремленою частиною крила. Хвіст має дві осі руху, внутрішня механіка виготовлена на 3D-принтері. Для підйому крила вгору-вниз застосовується важіль який закріплений на шестеренному механізмі(Рис. 2.5). Крім того, відокремлена частина має можливість обертатися навколо власної осі.

Рис. 2.4(а) – Вид позаду з переламаним крилом

Рис. 2.4(б) – Вид позаду з прямим крилом

Рис. 2.4(в) – Вид попереду

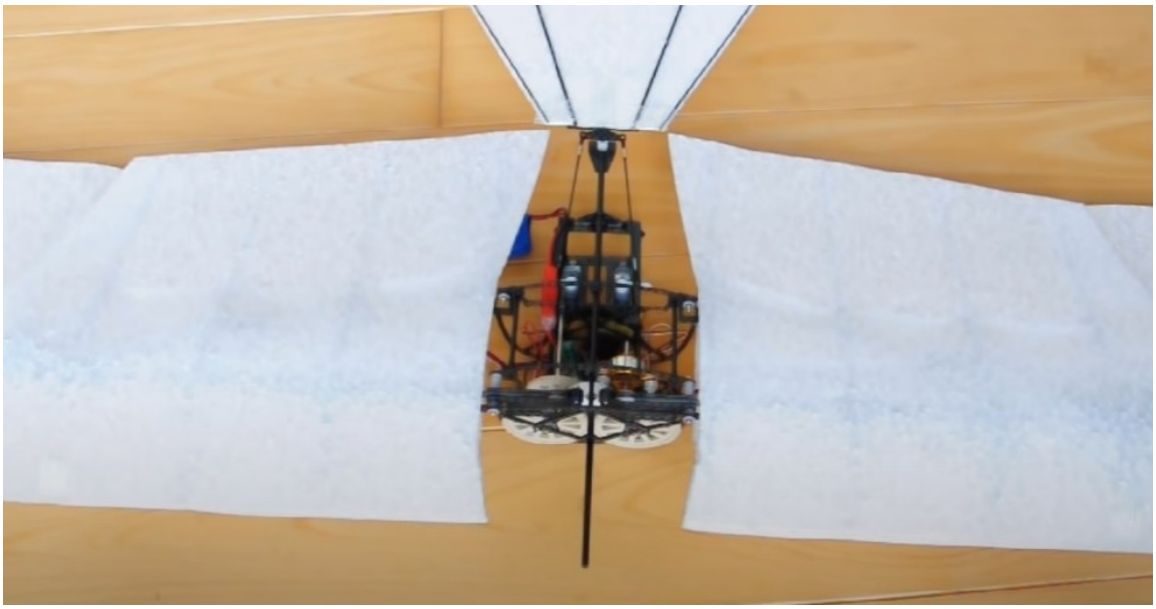

Рис. 2.5 – Внутрішня конструкція та механізм приводу

Конструкція крила(Рис. 2.6) складається з кількох стержнів скріплених між собою двох важільним одноступінчастим поворотним механізмом, а обмежувачем є сама конструкція механізму. Запуск здійснюється з руки, додаткових режимів польоту не передбачено, тільки політ, що махає, і кружляння.

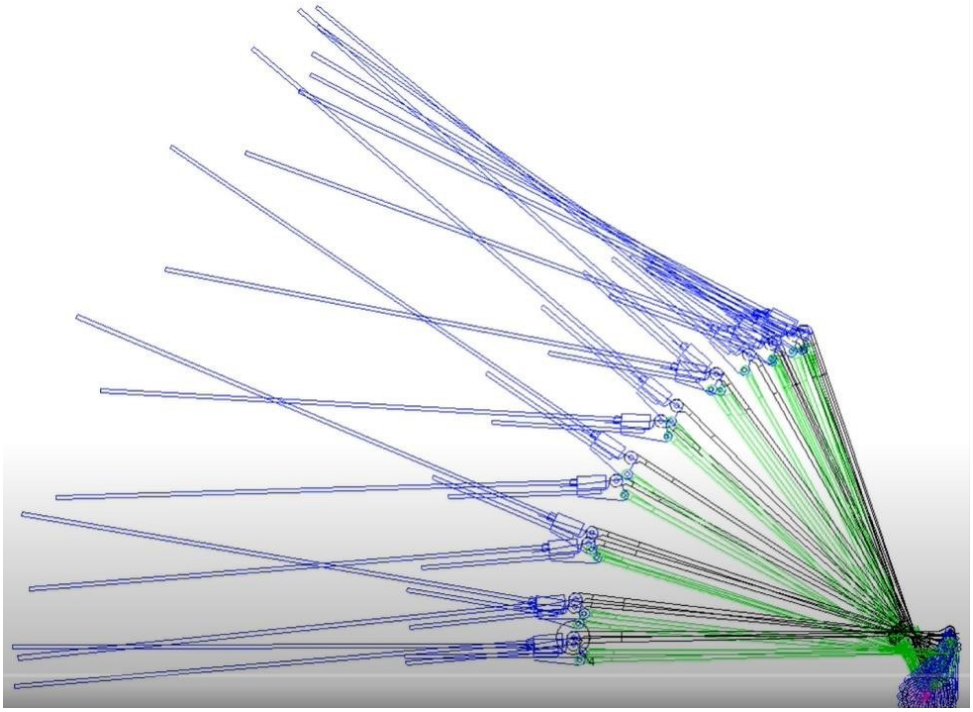

Рис. 2.6 – Механіка перелому крила

2.4 Таблиці основних характеристик аналогів

	Snowbird	BionicSwift	JSK-koubou
Привід	Мускульна сила	Електричний двигун	Електричний двигун
Розмах крил	32 м	68 см	143 см
Довжина	~ 10 м	44,5 см	61 см
Вага	42 кг	42 гр	165 гр
Час польоту	~ 1 хв	~ 7 хв	~ 5-15 хв
Швидкість польоту	25км/год	Невідомо, <15км/год	Невідомо, <20км/год
Матеріали	Вуглеволокно, піноматеріал та коркове дерево	Вуглеволокно, піноматеріал та 3Д друкований пластик	Текстоліт, вуглеволокно та епоксидна смола

3 Вибір варіанту побудови

Дослідивши показані вище аналоги було обрано як приклад для проектування орнітоптер від компанії Festo. Його продумана конструкція ідеально підходить для наших цілей.

3.1 Опис функціональної схеми та конструкції

До складу орнітоптера буде входити крила, хвостова частина та основний фюзеляж. У якому буде знаходитись мотор, шестерневий механізм для контрольованого руху крил, електронні компоненти управління польотом, сервоприводи хвостової частини та батарея.

Основна конструкція та шестерневий механізм планується бути зробленою з 3Д друкованого матеріалу. Крила та хвостова частина планується з надлегкого піноматеіалу з посиленням пластиком.

4 Розроблення програмного інтерфейсу для моделювання та розрахунку

4.1 Апроксимація за допомогою триангуляції

В цьому випадку повинні бути задані три вектори однакової довжини: вектор абсцис x, вектор ординат y вектор аплікат z. Таким чином, кожному вузлу координатної площині (xi, yi) зіставляється своє значення zi. Ці дані мають бути апроксимовані деякою досить простою функцією z = f (x, y). Найпростіша апроксимація - шматковолінійна. Її обчислення включає кілька етапів. Спочатку на безлічі вузлів (xi; yi) будується тріанангуляція (розбиття площини на трикутники). Для заданої множини вузлів (якщо їх більше трьох) існує багато різних тріангуляції. Найкраща з них у певному сенсі, так звана тріангуляція Делоне, — характеризується тим, що мінімальний по всіх трикутниках кут досягає максимального значення.

Тріангуляцію Делоне за заданими векторами x і y можна побудувати за допомогою функції delaunay (x, y), що повертає масив, кожен рядок якого містить трійку номерів вузлів, що утворюють черговий трикутник. Відповідний приклад показує програма prog2_1.m (Приклад 4.1).

Приклад 4.1. Триангуляція Делоне

```
function prog13_10 x=[-2,-1,0,0,2, 3,3,4, 6, 6]; y=[ 2,-2,0,4,2,-2,5,3,-1, 1];
axes('XLim',[-2.5 6.5], 'YLim',[-2.5 5.5])
```

```
tri = delaunay(x,y); s=size(tri); hold on;
grid off; xlabel('\itx'); ylabel('\ity');
title('Триангуляція Делоне','FontName','Courier')
for t=l:s(l) i=tri(t,l); j=tri(t,2); k=tri(t,3);
xt=[x(i),x(j),x(k),x(i)];
yt=[y(i),y(j),y(k),y(i)];
plot( xt, yt,'k') ;
end
```

Функція delaunay побудувала масив tri з 11 трійок, що містять номери вершин відповідних трикутників (рис. 4.1):

```
3  6  25  6  32  1  36 10  95 10  6
5  8 101  4  34  5  35  7  84  7  5
7 10  8
```

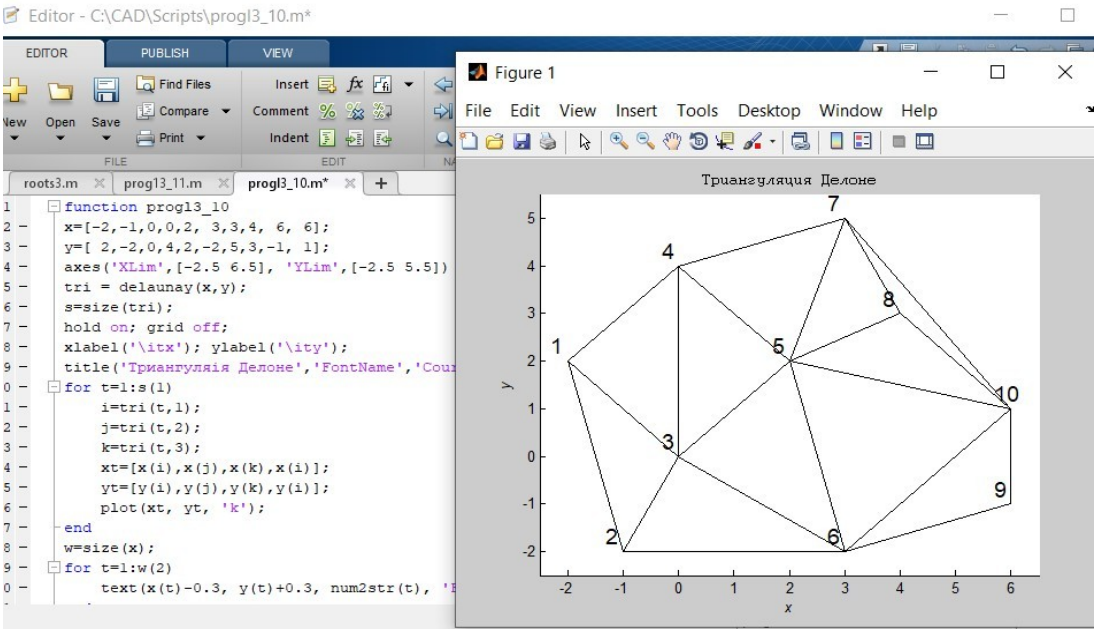

Рис. 4.1 - Триангуляція Делоне

Найпростіший приклад поверхні із трикутними гранями (рис. 4.2) будує програма prog2_2.m .

```
function prog2_2 x=[-2,-1,0,0,2, 3,3,4, 6, 6]; y=[ 2,-2,0,4,2,-2,5,3,-1, 1]; z=[ 1, 2,3,4,5, 6,7,8, 9,10];
tri = delaunay(x,y);
trimesh(tri,x,y,zeros(size(x))); hold on;
trimesh (tri, x, y, z) ; xlabel('x'); ylabel('y');
zlabel('z') colormap copper title('Поверхня із трикутними гранями','FontName','Courier');
```

У цій програмі використана функція trimesh, яка будує зображення поверхні з трикутними гранями по масивах x, y, z та опису тріангуляції Делоне (масив tri). Так як кожній вершині (xi; yi) відповідає апліката zu, то за вихідною табличною функцією ми фактично побудували поверхню, складену з трикутних граней. Тепер у будь-якій точці (x, y), що знаходиться всередині оригінального багатокутника, можна відновити перпендикуляр до перетину з відповідною гранню і знайти значення нашої функції z (x, y) у проміжних точках. Всі ці обчислення виконуються за допомогою функції griddata, яка для заданих табличних значень (вектори x, y, z) та масивів проміж точних точок (xx, yy) обчислює значення відповідних аплікат

zz = griddata(x,y,z,xx,yy);

```
x=[-2,-1,0,0,2,3,3,4,6,6];
y=[2,-2,0,4,2,-2,5,3,-1,1];
c=[1,2,3,4,5,6,7,8,9,10];
tri = delaunay(x,y);
trimesh(tri,x,y,zeros(size(x)));
hold on;
trimesh(tri,x,y,z);
xlabel('x'); ylabel('y'); zlabel('z')
colormap copper
title('Поверхность с треугольными гранями','FontName','Courier');
zz = griddata(x,y,z,xx,yy,method);
```

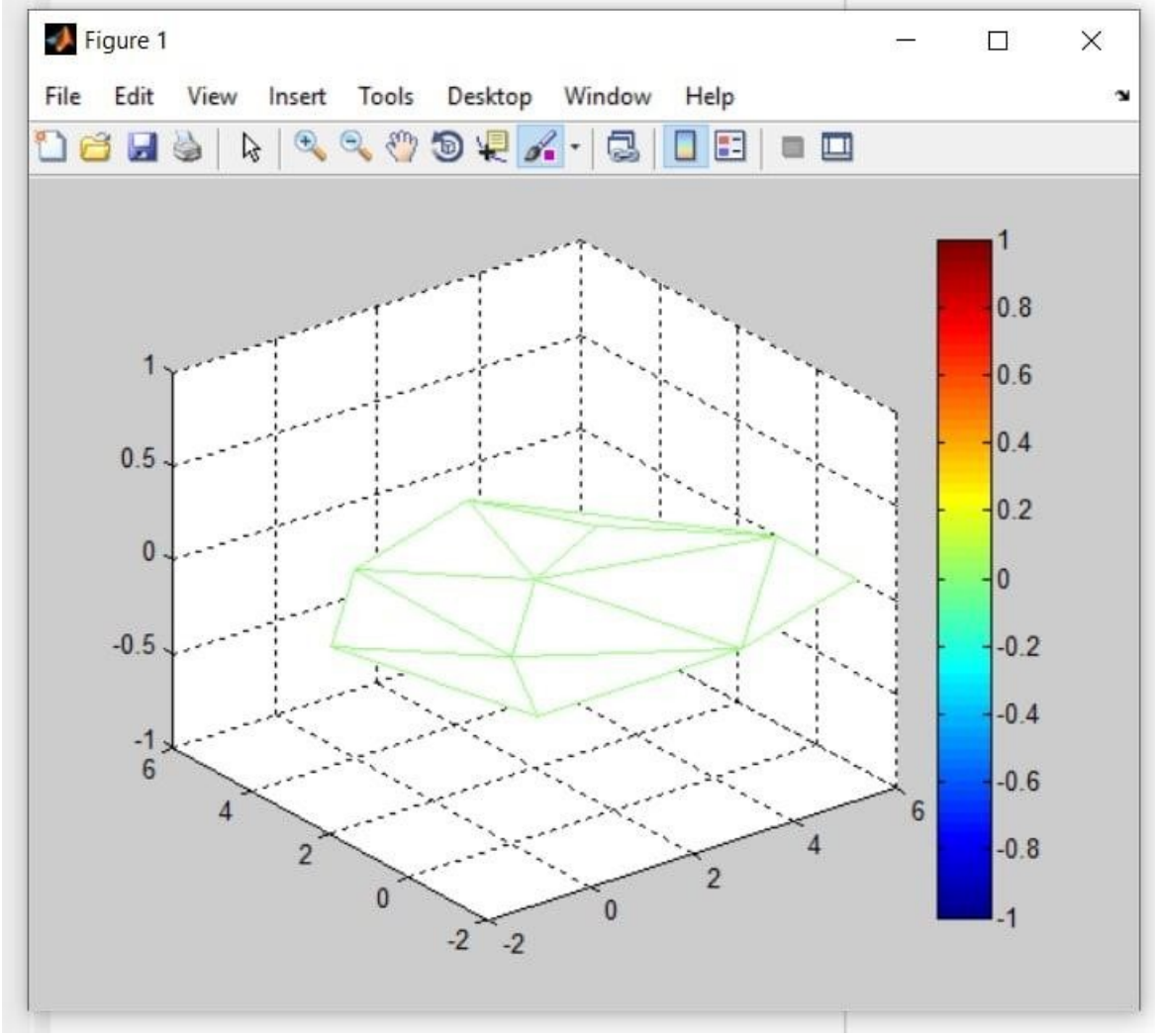

Рис. 4.2. - Поверхня із трикутними гранями

4.1.1 Динамічне створення інтерфейсних елементів

Для створення будь-якого інтерфейсного компонента із заданими властивостями використовується функція uicontroi, що повертає покажчик на компонент, що формується.

hUIC = uicontroi([hFig,] 'Style','тип_компонента', 1 Властивість_1',Значення_1,... 'Властивість 2',Значення 2,... 'Властивість к', Значення к);

Перший аргумент функції uicontrol не є обов'язковим, і якщо він відсутній, то батьком (власником) компонента, що створюється, є поточний графічний об'єкт. У існуючого інтерфейсного об'єкта можна змінити ті чи інші властивості за допомогою функції set:

set(UIC,'Властивість _1',Значение_1,... 'Властивість _2',Значення_2,...

'Властивість _к', Значення_к)

Пояснимо сказане на прикладі розв'язання квадратного рівняння:

$ax^2 + bx + c = 0.$

Припустимо, що нам хотілося б бачити на формі головного додатка три вікна для введення коефіцієнтів рівняння, два вікна для відображення значень знайдених коренів і кнопку для запуску алгоритма обчислення коренів після введення чергових значень коефіцієнтів (Приклад 4.3). У цьому додатку дуже корисним виявилося б вікно, що відтворює графік функції $y=ax^2 + bx + c$, що візуально підтверджує розташування знайденого коріння.

Для контролю роботи нашої майбутньої програми попередньо проробимо всі обчислення в командному вікні. Набір коефіцієнтів (a, b, c) доцільно оформити у вигляді вектора, який знадобиться як для виклику функції roots, що обчислює корені полінома, так і для функції polyval, що обчислює значення полінома у заданій точці. Якщо результати обчислень, виведені за форматом short, нам здадуться не дуже задовільними, можна скористатися форматом long, щоб побачити значення коренів і полінома з вищою точністю.

Приклад 4.3. Пошук розв'язання квадратного рівняння

```
» a=1.5;
» Ь=-2.4;
» c=0.75;
» p=[a b c]; » x=roots(p)
X =     1.1742
        0.4258
```

» polyval(p , 1.1742) ans =
3.8460e-005
» polyval(p, 0.4258) ans =
3.8460e-005
» format long
» x
X = 1.17416573867739
0.42583426132261
» polyval(p, 1.17416573867739) ans =
-4.329869796038111e-015
» polyval(p, 0.42583426132261) ans =
-4.551914400963142e-015

Тепер побудуємо графік нашої функції, поклавши як граничні інтервали по осі X значення xmin=min(real(x(1)), real(x(2)))-0.5 і xmax=max(real(x(l)),real(x(2)))+0.5:

```
» xrl=real(x(1));
» xr2=real(x(2));
» xmin=min(xrl,xr2)-0.5;
» xmax=max(xrl,xr2)+0.5;
» xx=xmin:0.1:xmax;
» yy=polyval(p,xx);
» plot (xx, yy)
» grid on
```

Вікно програми з графіком нашої функції наведено на Рис. 4.3.

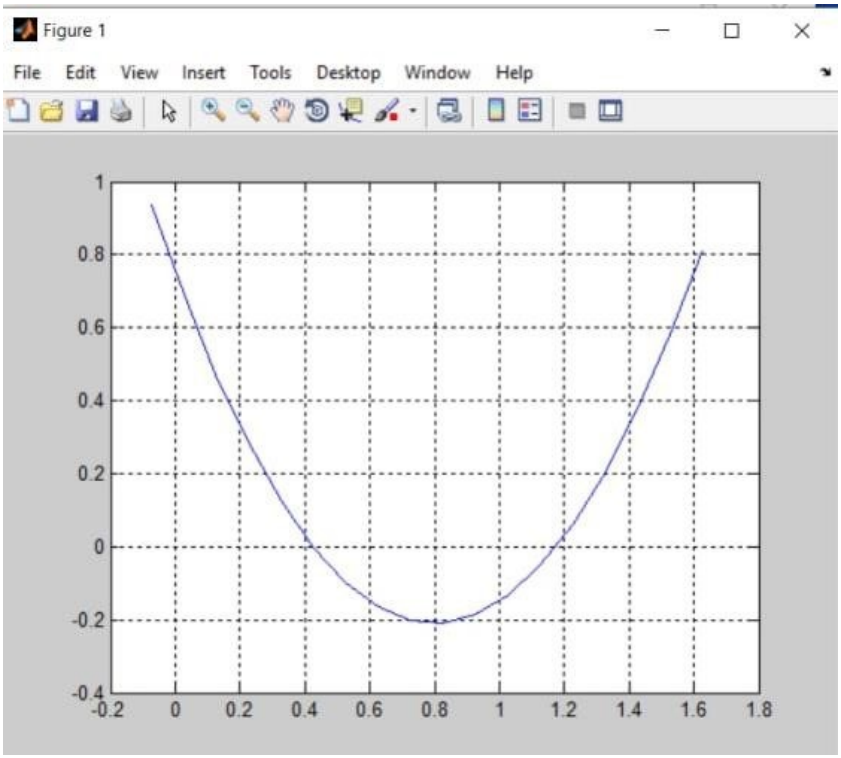

Рис. 4.3 – Графік функції $ax^2 + bx + c = y$

Будь-які спроби розширити створене вікно, щоб звільнити простір для передбачуваних полів введення та кнопки, ні до чого не призведуть. При зміні розмірів фігури графік функції автоматично підлаштовується під нові розміри.

Тому доведеться примусово задавати параметри вікна, поля графіка та всіх інтерфейсних компонентів на стадії виконання програми. Попередній дизайн форми можна провести на міліметрівці та не забути після цього перевести міліметри в пікселі (коефіцієнт перерахунку для різних моніторів та встановленого дозволу може бути різним). Числові дані, якими ми користувалися для побудови макету форми (рис. 4.4), зберігаються у файлі roots2.ni (Приклад 4.4).

Приклад 4.4. Функція roots2

```
function roots2 %
глобальні змінні
global hFig hAxes hBtn
```

```
global hTxta hTxtb hTxtc hTxtxl hTxtx2global hEda hEdb hEdc hEdxl hEdx2
hFig=figure('Position',[50,50,480, 300]) ; hAxes=axes;
set(hAxes,'Unit','pixels','Position',[30,70,280,220])
hTxta=uicontrol(hFig,'Style','text','String','a=','Position', [340,253,30,21] ,
'BackgroundColor', [1 1 1] );
hTxtb=uicontrol(hFig,'Style','text', 'String','b=','Position',
[340,218,30,21],'BackgroundColor', [1 1 1] );
hTxtc=uicontrol(hFig,'Style','text','String','c=','Position',[340,188,30,21],'Backgrou ndColor',[1 1
1]); hTxtxl=uicontrol(hFig,'Style','text','String','xl=','Position',[340,113,30,21],'Backgr oundColor',
[1 1 1]);
hTxtx2=uicontrol(hFig,'Style','text','String','x2=','Position',
[340,78,30,21],'BackgroundColor',[1 1 1]);
hEda=uicontrol(hFig,'Style','edit','Position',[380,250,100,25],'BackgroundColor',[1 1
1],'HorizontalAlignment','left');
hEdb=uicontrol(hFig,'Style','edit','Position',[380,215,100,25], 'BackgroundColor',[1 1
1],'HorizontalAlignment','left');
hEdc=uicontrol(hFig,'Style','edit','Position', [380,185,100,25] , 'BackgroundColor',[1
1 1],'HorizontalAlignment','left');
hEdxl=uicontrol(hFig,'Style','edit','Position', [380,110,100,25], 'BackgroundColor',
[1 1 1],'HorizontalAlignment','left');
hEdx2=uicontrol(hFig,'Style','edit','Position',[380,75,100,25], 'BackgroundColor',[1 1
1],'HorizontalAlignment','left');
hBtn=uicontrol(hFig,'Style','pushbutton','String','Решение','Position',[340,150,140,25
],'Callback','roots3');
```

З кнопкою «Рішення» асоційовано обробник подій (callback-функція), представлений файлом roots3.m. Текст цієї функції наведено у прикладі 4.5. Рядки функції пронумеровані лише для довідки

Приклад 4.5

```
1.  function roots3
2.  global hAxes
```

```
3.  global hEda hEdb hEdc hEdxl hEdx2
4.  axes(hAxes);
5.  cla;
6.  str=get(hEda,'String');
7.  a=str2num(str);
8.  str=get(hEdb,'String');
9.  b=str2num(str);
10. str=get(hEdc,'String');
11. c=str2num(str);
12. p=[a b c];
13. x=roots(p)
14. xrl=real(x(l));
15. xr2=real(x(2));
16. xmin=min(xrl,xr2)-0.5;
17. xmax=max(xrl,xr2)+0.5;
18. xx=xmin: 0.1: xmax;
19. yy=polyval(p,xx);
20. plot(xx,yy)
21. grid on
22. set(hEdxl,'String',num2str(x(1)));
23. set(hEdx2,'String',num2str(x(2)));
```

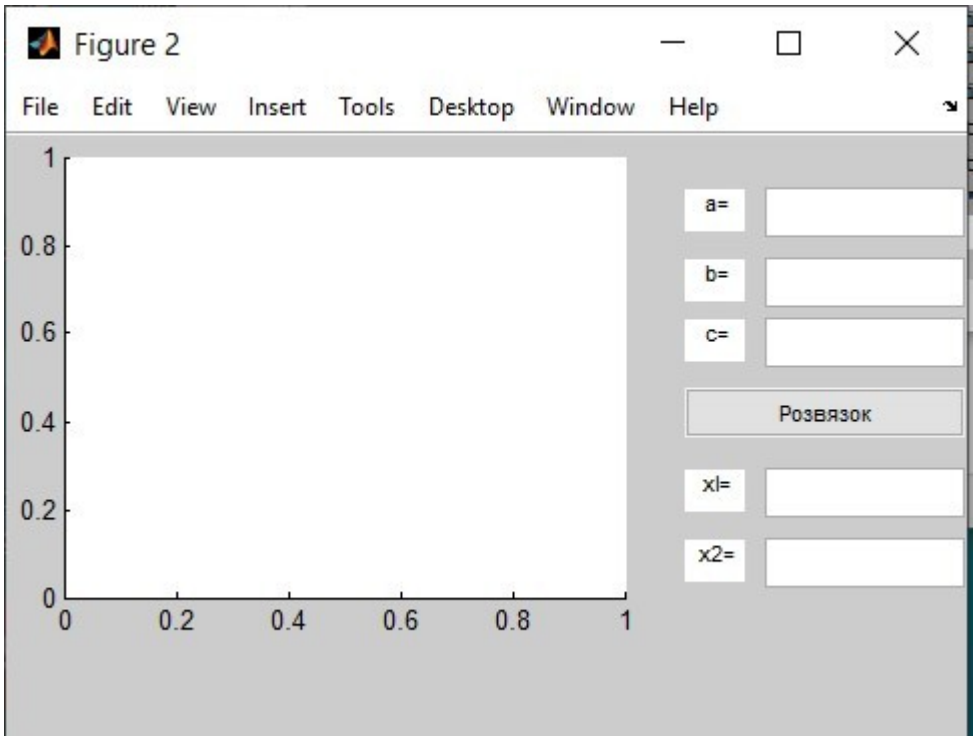

Рис. 4.4 - Ескіз вікна гаданої програми

Після набору значень коефіцієнтів a, b, c в редагованих полях введення і кліку по кнопці Розв'язок виникає подія, для обробки якої автоматично викликається Callback-функція root3.m. Для страховки четвертим рядком активізується вікно графіка, в яке дивиться покажчик hAxes. У нашому додатку таке вікно єдине, тому саме воно і є поточним, отже, у зверненні до функції axes особливої потреби немає. У п'ятому рядку здійснюється очищення поточного поля графіка. Потім у рядках 6-11 виконується зчитування вмісту тестових вікон та перетворення витягнутих рядків у числовий формат. Рядки 12-21 повторюють обчислювальну схему, опробовану нами у командному вікні. Знайдені значення коренів перетворюються з числового формату символьний вигляд і заносяться у відповідні вікна виведення. Результат роботи програми, представленої файлами roots2.m та roots3.m, наведено на Рис. 4.5 .

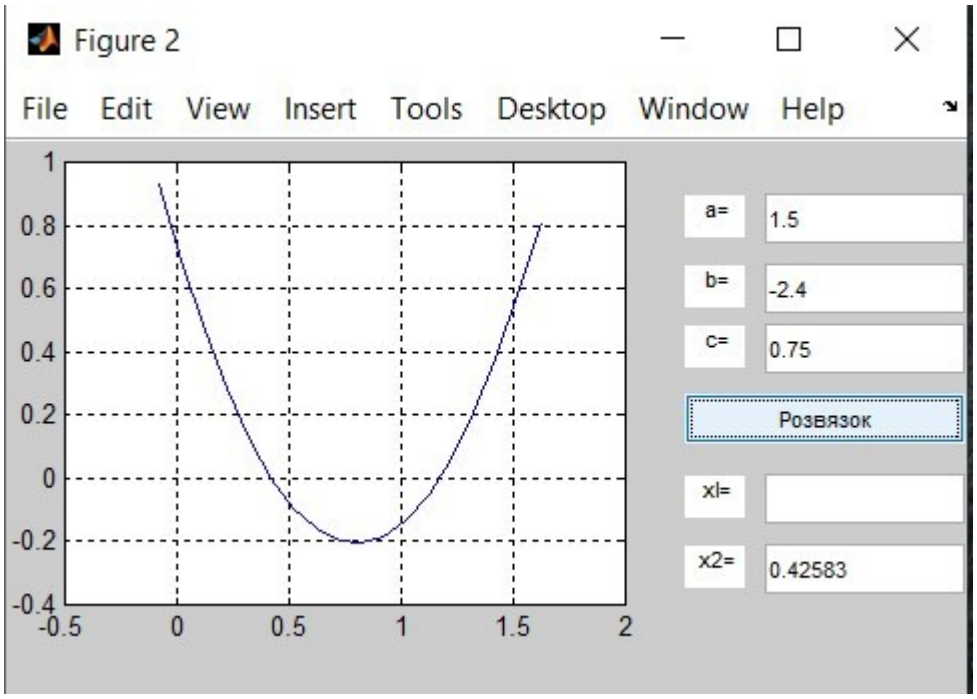

Рис. 4.5 - Розв'язання квадратного рівняння

4.2 Вибір мови програмування C++ для візуалізації: критичні фактори та переваги

Вибір мови програмування C++, для виконання візуалізації було обумовлене декількома критичними факторами, такими як.

C++ є високопродуктивною мовою програмування, що дозволяє оптимізувати розрахунки та візуалізацію для великих об'ємів даних. Вона набагато швидша за багато інших мов програмування, що дозволяє вам працювати зі складними алгоритмами триангуляції на великих наборах даних без значного збільшення часу обчислення.

Іншим важливим фактором є прямий контроль над управлінням пам'яттю, що дозволяє оптимізувати використання пам'яті та уникнути зайвих виділень та звільнень пам'яті. Це особливо важливо для обробки великих наборів даних, де ефективне використання пам'яті може значно покращити продуктивність програми. Та багатий набір інструментів в ICP для налагодження, що дозволяє виявляти та усувати помилки в програмі швидше. Ви можете використовувати дебагери та інші інструменти для аналізу та оптимізації вашого коду, що полегшує процес розробки.

Також C++ є мовою програмування з великою кількістю функціональних можливостей. Ви можете створювати власні класи, функції та алгоритми для візуалізації та обробки триангуляції, що дозволяє вам адаптувати рішення під ваші конкретні потреби.

Найкритичнішим фактором стала наявність широкого вибору бібліотек для візуалізації та обробки графіки. Наприклад, OpenCV, OpenGL, Qt, GLUT і багато інших. Ці бібліотеки надають потужні інструменти для створення візуалізаційних додатків, що допомагають вам легко створювати та керувати вікнами, графікою, взаємодією користувача та іншими елементами.

4.3 Візуалізація триангуляції Делоне за допомогою C++

Напочатку треба підключити необхідні для роботи заголовочні файли

```
#include <CGAL/Exact_predicates_inexact_constructions_kernel.h>
#include <CGAL/Delaunay_triangulation_3.h>
#include <iostream>
#include <fstream>
#include <vector>
#include <GL/glut.h>
```

CGAL/Exact_predicates_inexact_constructions_kernel.h: Це бібліотека CGAL, яка містить відомості про точки, вектори та різні математичні обчислення, що потрібні для триангуляції. Використовується для визначення типу K, який представляє ядро точок та операцій над ними.

CGAL/Delaunay_triangulation_3.h: Це бібліотека CGAL, яка містить алгоритм триангуляції Делоне у тривимірному просторі. Використовується для визначення типу Delaunay, який представляє об'єкт триангуляції Делоне.

GL/glut.h: Це бібліотека OpenGL, яка містить функції та типи для візуалізації графіки. Використовується для створення вікна, обробки подій та візуалізації триангуляції.

iostream, fstream, vector: Ці стандартні бібліотеки C++ використовуються для зчитування точок з файлу та зберігання їх у векторі.

Далі описуемо оголошення типів та змінних:

```
typedef CGAL::Exact_predicates_inexact_constructions_kernel K;
typedef CGAL::Delaunay_triangulation_3<K> Delaunay;
int window_width = 800;
int window_height = 600;
double camera_x = 4.0;
double camera_y = 4.0;
double camera_z = 4.0;
double camera_yaw = 45.0;
std::vector<K::Point_3> points = { ... };
Delaunay dt;
```

K - тип ядро CGAL, який використовується для точної обробки геометричних об'єктів.

Delaunay - тип триангуляції Делоне.

window_width та window_height - розміри вікна для відображення.

camera_x, camera_y, camera_z – початкові координати камери в тривимірному просторі.

camera_yaw – початковий кут повороту камери навколо осі Z.

points - масив точок, які будуть тріангульовані.

dt - об'єкт триангуляції Делоне.

Функція loadPointsFromFile завантажує точки з вхідного файлу:

```
void loadPointsFromFile(const std::string& filename)
{
```

```
    std::ifstream file(filename);
    if (!file.is_open()) {
        std::cout << "Failed to open file: " << filename << std::endl;
        return;
    }
    std::string line;
    while (std::getline(file, line)) {
        std::string::size_type pos = line.find('=');
        if (pos == std::string::npos)
            continue;
        std::string prefix = line.substr(0, pos);
        std::string pointsStr = line.substr(pos + 1);
        std::istringstream iss(pointsStr);
        double x, y, z;
        while (iss >> x) {
            iss.ignore();
            iss >> y;
            iss.ignore();
            iss >> z;
            points.push_back(K::Point_3(x, y, z));
        }
    }
    file.close();
}
```

Ця функція відкриває файл за заданим ім'ям, перевіряє, чи відкриття пройшло успішно, а потім зчитує точки з файлу і додає їх до масиву points. Координати точок зберігаються у тривимірному варіанті.

Функція handleKeypress обробляє введення з клавіатури:

```
void handleKeypress(unsigned char key, int x, int y)
{
    const double camera_speed = 0.1;
    const double camera_rotation_speed = 1.0;
```

```
    switch (key) {
        case 'w':
        case 'W':
            camera_z -= camera_speed;
            break;
        case 's':
        case 'S':
            camera_z += camera_speed;
            break;
        case 'a':
        case 'A':
            camera_x -= camera_speed;
            break;
        case 'd':
        case 'D':
            camera_x += camera_speed;
            break;
        case 'q':
        case 'Q':
            camera_yaw -= camera_rotation_speed;
            break;
        case 'e':
        case 'E':
            camera_yaw += camera_rotation_speed;
            break;
        default:
            break;
    }
    glutPostRedisplay();
}
```

У цій функції використовуються наступні змінні:

camera_speed: Змінна, що визначає швидкість переміщення камери. Значення 0.1 вказує на те, що камера буде переміщуватись на 0.1 одиниці по вісі X або Z за одне натискання клавіші.

camera_rotation_speed: Змінна, що визначає швидкість обертання камери. Значення 1.0 вказує на те, що камера буде обертатись на 1.0 градус за одне натискання клавіші.

У функції використовується оператор switch, який перевіряє значення key (код натиснутої клавіші). Залежно від значення key виконується певна дія:

Якщо натиснута клавіша 'w' або 'W', то зменшується значення camera_z, що призводить до переміщення камери у напрямку спереду.

Якщо натиснута клавіша 's' або 'S', то збільшується значення camera_z, що призводить до переміщення камери у напрямку назад.

Якщо натиснута клавіша 'a' або 'A', то зменшується значення camera_x, що призводить до переміщення камери вліво.

Якщо натиснута клавіша 'd' або 'D', то збільшується значення camera_x, що призводить до переміщення камери вправо.

Якщо натиснута клавіша 'q' або 'Q', то зменшується значення camera_yaw, що призводить до повороту камери проти годинникової стрілки.

Якщо натиснута клавіша 'e' або 'E', то збільшується значення camera_yaw, що призводить до повороту камери за годинниковою стрілкою.

Після зміни положення камери або кута повороту, викликається функція glutPostRedisplay(), яка запускає перерисування вікна, щоб зміни були відображені.

Функція display відображає результат триангуляції та точки на екрані:

```
void display()
```

```
{
    glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT);
    glMatrixMode(GL_PROJECTION);
    glLoadIdentity();
    gluPerspective(60.0, (GLfloat)window_width / (GLfloat)window_height, 0.1, 100.0);
    glMatrixMode(GL_MODELVIEW);
    glLoadIdentity();
    gluLookAt(camera_x, camera_y, camera_z, 0, 0, 0, 0, 0, 1);
    glRotatef(camera_yaw, 0, 0, 1);
    glPointSize(5.0f);
    glBegin(GL_POINTS);
    glColor3f(1.0f, 0.0f, 0.0f);
    for (auto p : points) {
        glVertex3f(p.x(), p.y(), p.z());
    }
    glEnd();
    glLineWidth(1.0f);
    glBegin(GL_LINES);
    glColor3f(0.0f, 0.0f, 1.0f);
    for (auto e = dt.finite_edges_begin(); e != dt.finite_edges_end(); ++e) {
        auto v1 = (*e).first->vertex((*e).second)->point();
        auto v2 = (*e).first->vertex((*e).third)->point();
        glVertex3f(v1.x(), v1.y(), v1.z());
        glVertex3f(v2.x(), v2.y(), v2.z());
    }
    glEnd();
    glutSwapBuffers();
}
```

У цій функції використовуються наступні операції та функції:

glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT): Ця операція очищає буфер кольору та буфер глибини. Вона виконується на початку кожного кадру для очищення попереднього зображення.

glMatrixMode(GL_PROJECTION): Встановлює активну матрицю на матрицю проекції.

glLoadIdentity(): Скидає активну матрицю на одиничну матрицю.

gluPerspective(60.0, (GLfloat)window_width / (GLfloat)window_height, 0.1, 100.0): Встановлює перспективну проекцію. Параметри функції вказують на поля зору 60 градусів, співвідношення сторін вікна, мінімальну та максимальну відстані до об'єктів, які будуть видимі.

glMatrixMode(GL_MODELVIEW): Встановлює активну матрицю на матрицю моделі-огляду.

gluLookAt(camera_x, camera_y, camera_z, 0, 0, 0, 0, 0, 1): Встановлює положення та орієнтацію камери у просторі. Параметри функції вказують на координати позиції камери (camera_x, camera_y, camera_z), координати точки, на яку спрямована камера (0, 0, 0), та вектор, що вказує напрямок вгору (0, 0, 1).

glRotatef(camera_yaw, 0, 0, 1): Виконує поворот сцени навколо осі Z на кут camera_yaw. Це дозволяє обертати сцену навколо вертикальної осі.

glPointSize(5.0f): Встановлює розмір точок, які будуть малюватись.

glBegin(GL_POINTS): Починає блок малювання точок.

glColor3f(1.0f, 0.0f, 0.0f): Встановлює колір точок, які будуть малюватись. У цьому випадку, точки мають червоний колір.

glVertex3f(p.x(), p.y(), p.z()): Додає точку з координатами (p.x(), p.y(), p.z()) до поточного блоку малювання точок.

glEnd(): Закінчує блок малювання точок.

glLineWidth(1.0f): Встановлює товщину ліній, які будуть малюватись.

glBegin(GL_LINES): Починає блок малювання ліній.

glColor3f(0.0f, 0.0f, 1.0f): Встановлює колір ліній, які будуть малюватись. У цьому випадку, лінії мають синій колір.

glVertex3f(v1.x(), v1.y(), v1.z()): Додає початкову точку лінії з координатами (v1.x(), v1.y(), v1.z()) до поточного блоку малювання ліній.

glVertex3f(v2.x(), v2.y(), v2.z()): Додає кінцеву точку лінії з координатами (v2.x(), v2.y(), v2.z()) до поточного блоку малювання ліній.

glEnd(): Закінчує блок малювання ліній.

glutSwapBuffers(): Змінює буфер відображення, що дозволяє відобразити зображення, яке було створено у пам'яті, на екрані.

Отримана графічна сцена містить відображення точок у червоному кольорі та Делоне-триангуляції у синьому кольорі. Перед відображенням камера переорієнтується та переміщується відповідно до заданих значень camera_x, camera_y, camera_z та camera_yaw.

Функція reshape викликається при зміні розмірів вікна:

```
void reshape(int width, int height)
{
   window_width = width;
   window_height = height;
   glViewport(0, 0, width, height);
   glutPostRedisplay();
}
```

У цій функції використовуються наступні операції та функції:

window_width = width та window_height = height: Оновлюються глобальні змінні window_width та window_height з новими розмірами вікна.

glViewport(0, 0, width, height): Встановлює розмір та положення порту виведення. У цьому випадку, порт виведення розміщується у верхньому лівому куті вікна з розмірами width та height.

glutPostRedisplay(): Повідомляє системі про необхідність повторного відображення сцени. Це призводить до виклику функції display для оновлення зображення з новими розмірами вікна.

Ця функція викликається при зміні розмірів вікна і забезпечує адаптацію розмірів вікна для коректного відображення графічної сцени.

Заключна частина коду є найголовнішою, в ній знаходится функція main:

```
int main(int argc, char** argv)
{
   if (argc < 2) {
      std::cerr << "Usage: " << argv[0] << " <input_file>" << std::endl;
      return 1;
   }
   loadPointsFromFile(argv[1]);
   if (points.empty()) {
      std::cout << "No points to triangulate." << std::endl;
      return 1;
   }
   dt = Delaunay();
   dt.insert(points.begin(), points.end());
   glutInit(&argc, argv);
   glutInitDisplayMode(GLUT_DOUBLE | GLUT_RGB | GLUT_DEPTH);
   glutInitWindowSize(window_width, window_height);
   glutCreateWindow("Delaunay Triangulation");
   glEnable(GL_DEPTH_TEST);
   glutDisplayFunc(display);
   glutReshapeFunc(reshape);
   glutKeyboardFunc(handleKeypress);
   glutMainLoop();
   return 0;
}
```

У цій функції використовуються наступні операції та функції:

Перевіряється кількість аргументів командного рядка. Якщо кількість аргументів менше 2 (що вказує на відсутність вказаного вхідного файлу), виводиться повідомлення про використання програми та повертається значення 1, що свідчить про помилку.

Викликається функція loadPointsFromFile(argv[1]), яка завантажує точки з вказаного вхідного файлу у вектор points.

Перевіряється, чи points не є порожнім. Якщо вектор points не містить жодної точки, виводиться повідомлення про відсутність точок для тріангуляції та повертається значення 1.

Створюється об'єкт dt типу Delaunay, який використовується для збереження та обробки тріангуляції Делоне.

Викликаються функції glutInit, glutInitDisplayMode, glutInitWindowSize та glutCreateWindow для ініціалізації вікна та контексту OpenGL.

Увімкнений режим GL_DEPTH_TEST, який забезпечує обробку глибини для відображення об'єктів.

Встановлюються функції зворотного виклику glutDisplayFunc, glutReshapeFunc та glutKeyboardFunc, які будуть викликатися відповідно для відображення, зміни розміру вікна та обробки натискання клавіш.

Викликається функція glutMainLoop, яка розпочинає основний цикл обробки подій OpenGL.

Після завершення циклу glutMainLoop, програма повертається до функції main, де повертається значення 0, що свідчить про успішне завершення програми.

Роботу програми узагальнено у блок схемі(Рис. 4.6).

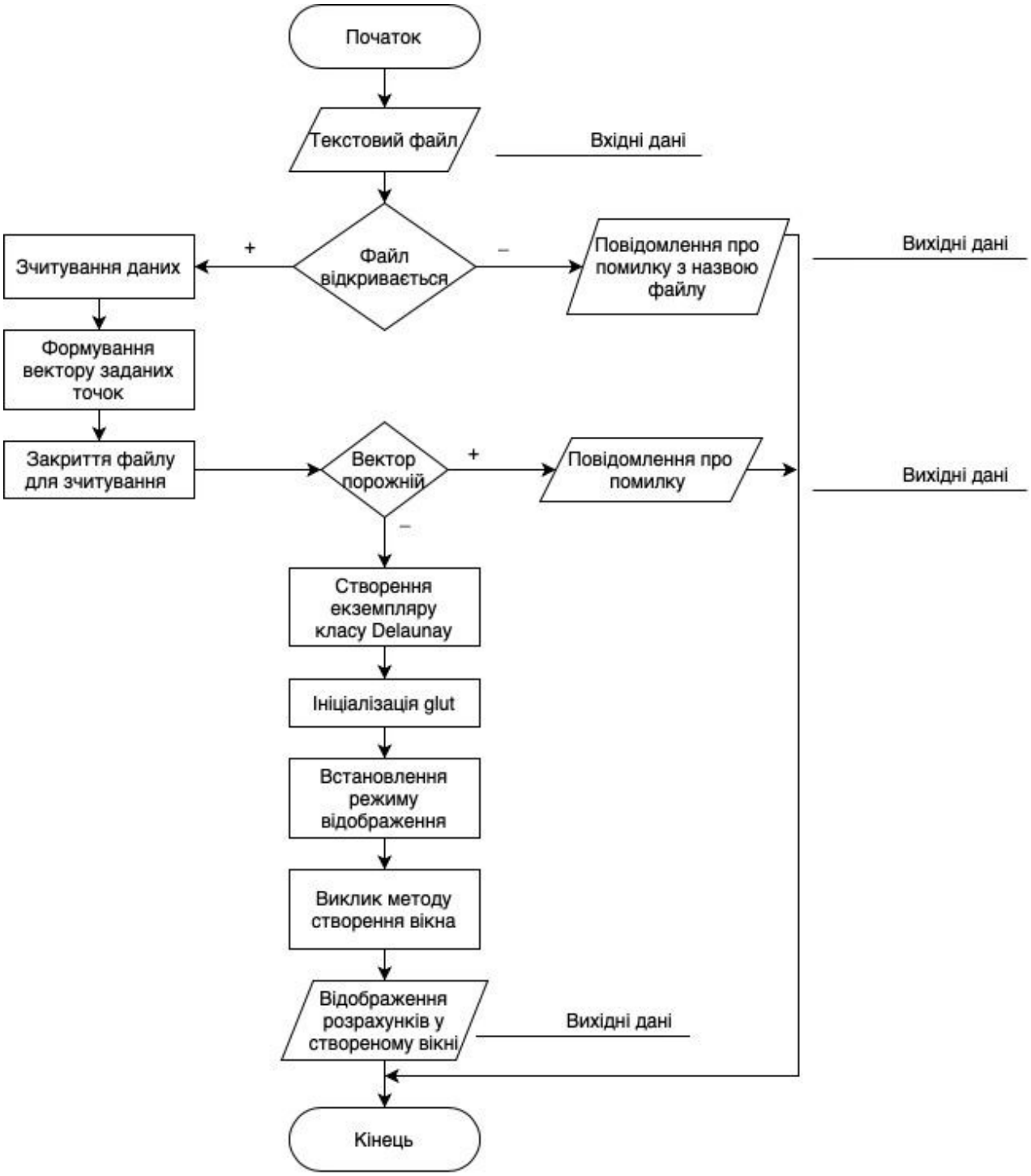

Рис. 4.6 – Блок схема алгоритму візуалізації триангуляції

5 Моделювання 3D моделі

5.1 Організація нового проекту

Аби почати моделювання, нам потрібен новий проект, для цього - заходимо у вкладку File >> Project Window і нажимаємо кнопку New. У поле Current Project вписуємо назву нового проекту, а в полі Location – локалізацію

нового проекту. Таким чином, ми створили декілька зрозумілих для MAYA папок, у яких буде зберігатися інформація для нашого проекту. Ми будемо використовувати 2 папки під назвою sourceimages та scenes. Інші папки можна видалити, так як вони не будуть використовуватися, але вони не займають багато місця, тому, рекомендовано, їх залишити.

У такий спосіб можна зорганізувати робочий простір, аби усунути проблеми, що можуть виникнути підчас відкриття проекту на іншому комп'ютері.

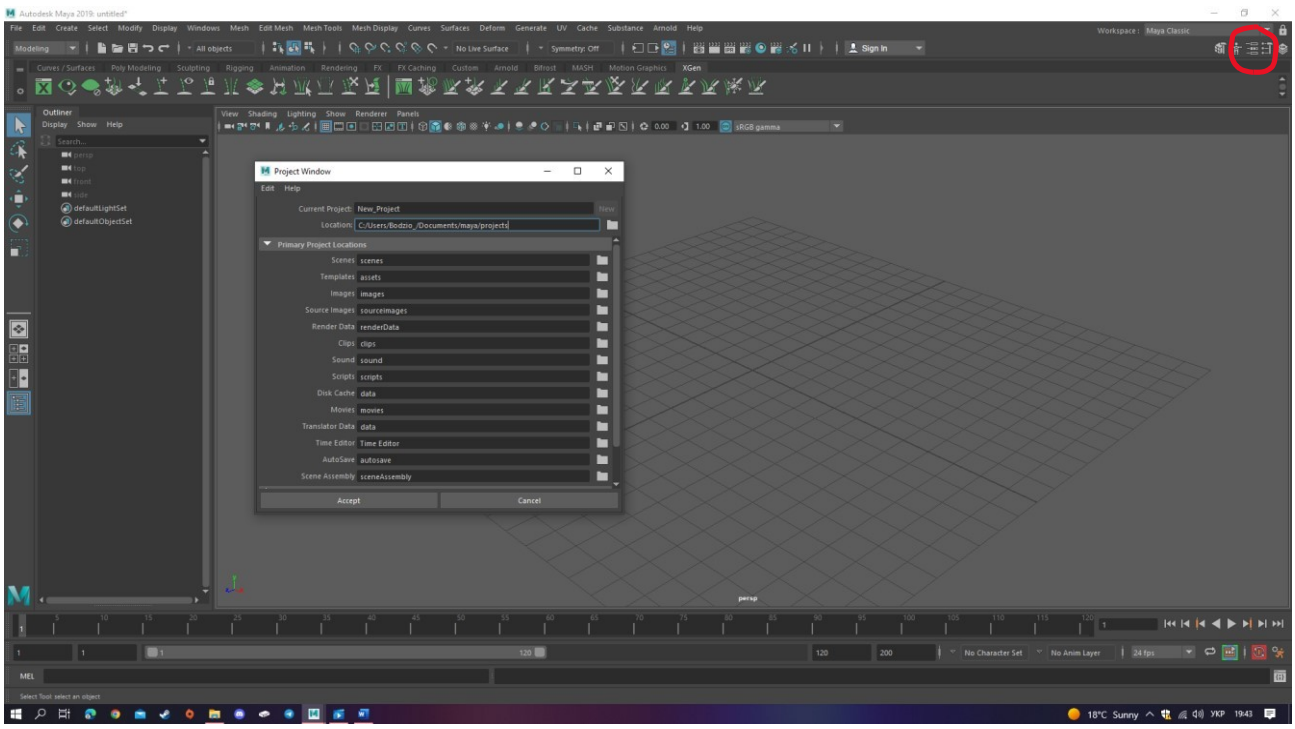

Рис. 5.1 – Створення проекту і файлової стежки

5.2 Завантаження референсів

Перш ніж долучати до проекту референси, потрібно їх створити. Це може бути фото, сканований ручний ескіз, що небудь, що має форму нашого теоретичного об'єкта. У даному проекті було використано фото з інтернету.

Отже, є 2 варіанти: 1 – із прив'язкою фото-файлу до камери, що ми не будемо використовувати, і 2 – додання референс файлу як об'єкта сцени.

Знаходимо\креслимо\рисуємо необхідний референс і переносимо до папки sourceimages у новоствореному проекті.

Вкладка Create >> Free Image Plane.

Далі, нам необхідно відкрити Attribute editor, кнопка якого знаходиться вверху з правої сторони вікна програми MAYA, як це видно на Рисунку 5.1, клікаємо на знак папки біля поля Image Name і шукаємо референс, який має знаходитися в папці з проектом, який ми створили, а саме, в підпапці sourceimages. Вибираємо потрібний референс. У назві референс-файлу мають міститися тільки латинські літери алфавіту, інакше виникне помилка.

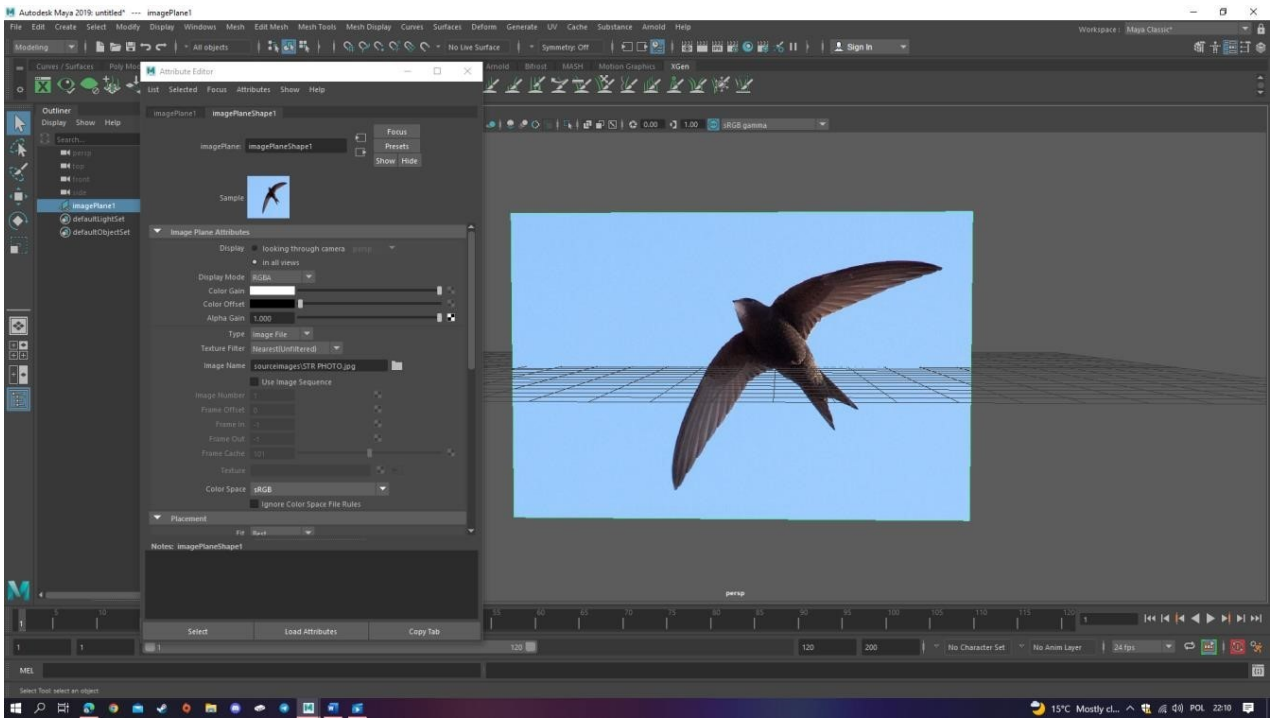

Рис. 5.2 – Референс перенесений до MAYA

5.2.1 Створення примітивних форм на фоні референсу

Використовуючи закладку Create >> Polygon Primitives >> Cube, можна поетапно створити примітивну форму птаха. Почнемо з шиї.

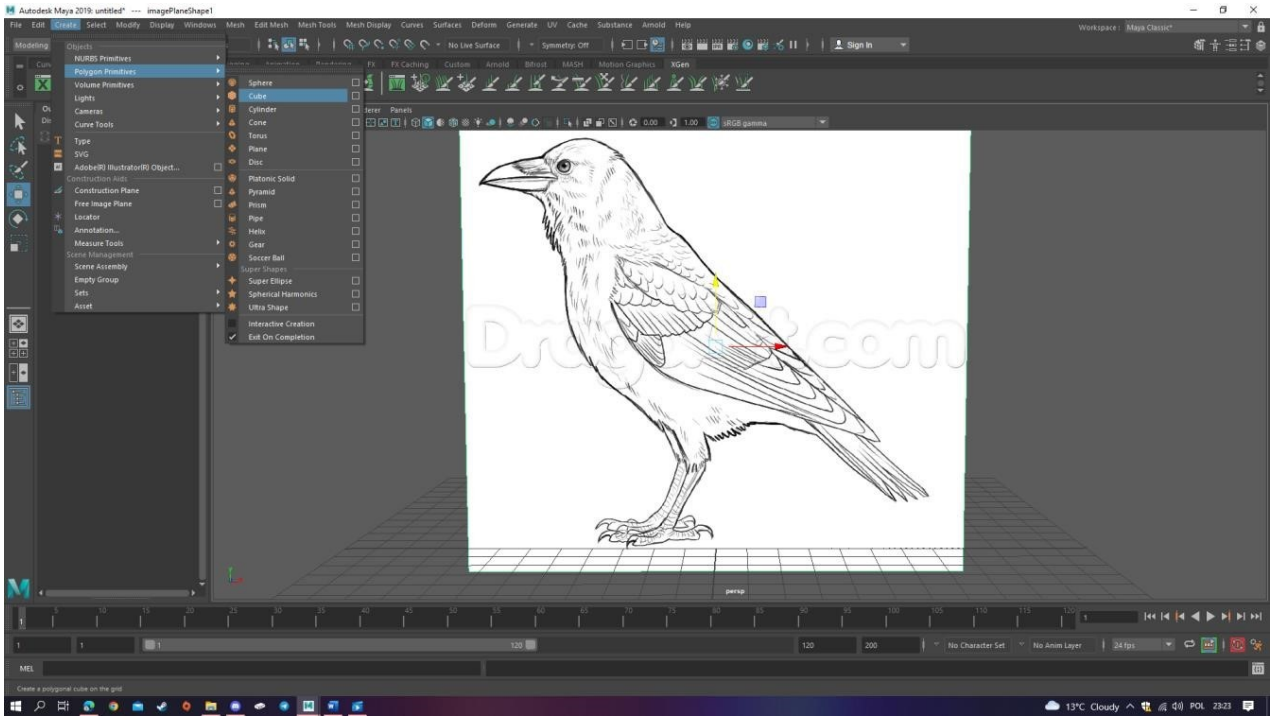

Рис. 5.3

Аби перейти до Front view, використовуємо кнопку ‘space’ на клавіатурі, наводимо курсор на потрібний вид і знову нажимаємо ‘space’. Для переміщення, масштабування та обертання об’єктів використовуємо панель, яка знаходиться скраю зліва.

Отже, намагаємося відтворити контури об’єкта.

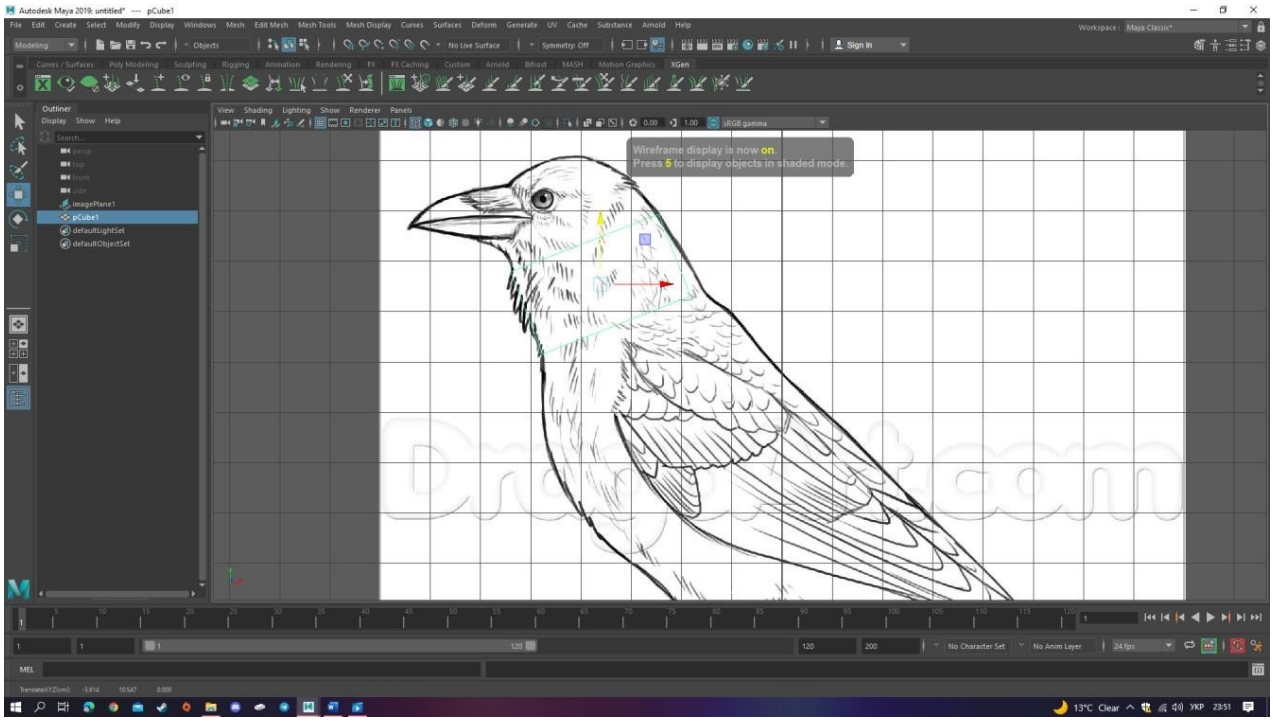

Рис. 5.4

Після створення куба, допасуємо один його край до лівого краю референсу птиці, як показано на рисунку 5.4. Далі використовуємо спеціальний інструмент MAYA для роботи з поверхнями, ребрами та точками фігур.

Наводимо курсор на пусте поле середовища MAYA, нажимаємо праву кнопку миші і тримаємо її, пересуваючи, вибираємо інструкцію Face. Переходимо у Perspective mode за допомогою клавіші ‘space’ і вибираємо нижню поверхню куба. Як тільки поверхня куба виділена, можемо вибрати функцію розширення фігури і «розтягнути», після цього, можемо використати функцію переміщення об’єкта аби допасувати куб до краю нарисованого тіла.

На даному етапі не рекомендовано вдаватися в деталі моделі, її редагування буде можливе пізніше.

Другим кроком є використання функції ‘Extrude’. Вкладка Edit Mesh >> Extrude. Ця функція має бути використана при виділеній нижній поверхні куба. Тягнемо за стрілку ОУ і маємо ще один куб поєднаний з тим, що ми вже створили. Повторюємо попередній крок, і як тільки допасуємо новий куб до країв моделі, тримаємо клавішу ‘Shift’ і тягнемо за стрілку переміщення, аби повторити попередню дію, тобто ‘Extrude’. У місці переходу до ніг, потрібно зробити куб, з бокової поверхні якого, можна витягнути куб для ніг птаха, як показано на рисунку 5.5.

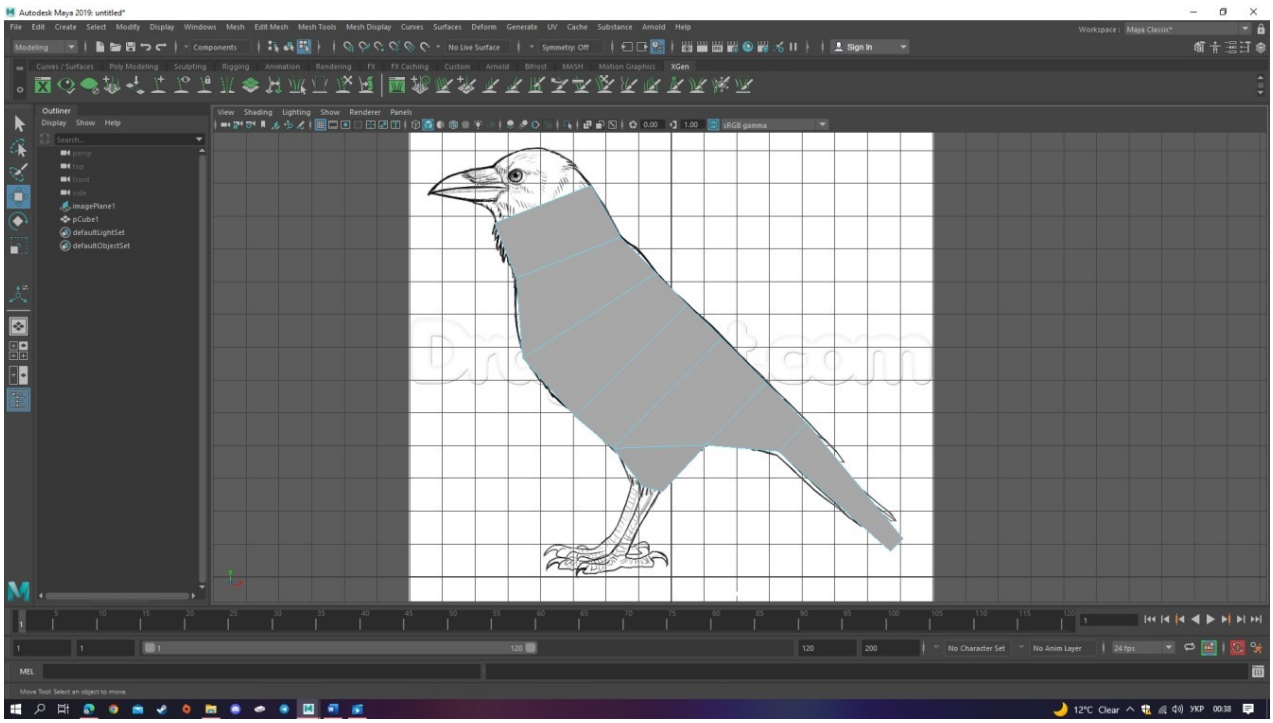

Рис. 5.5

За такою самою схемою робимо ноги та пальці і стягуємо квадратну поверхню кінців пальців за допомогою функції масштабування на лівій панелі, аби зробити їх гострішими (клікаючи на поверхні кінців пальців із зажатою клавішою ‘Shift’ можна виділити 3 площини одразу та редагувати їх одночасно). Результат ми бачимо на рисунку 5.6.

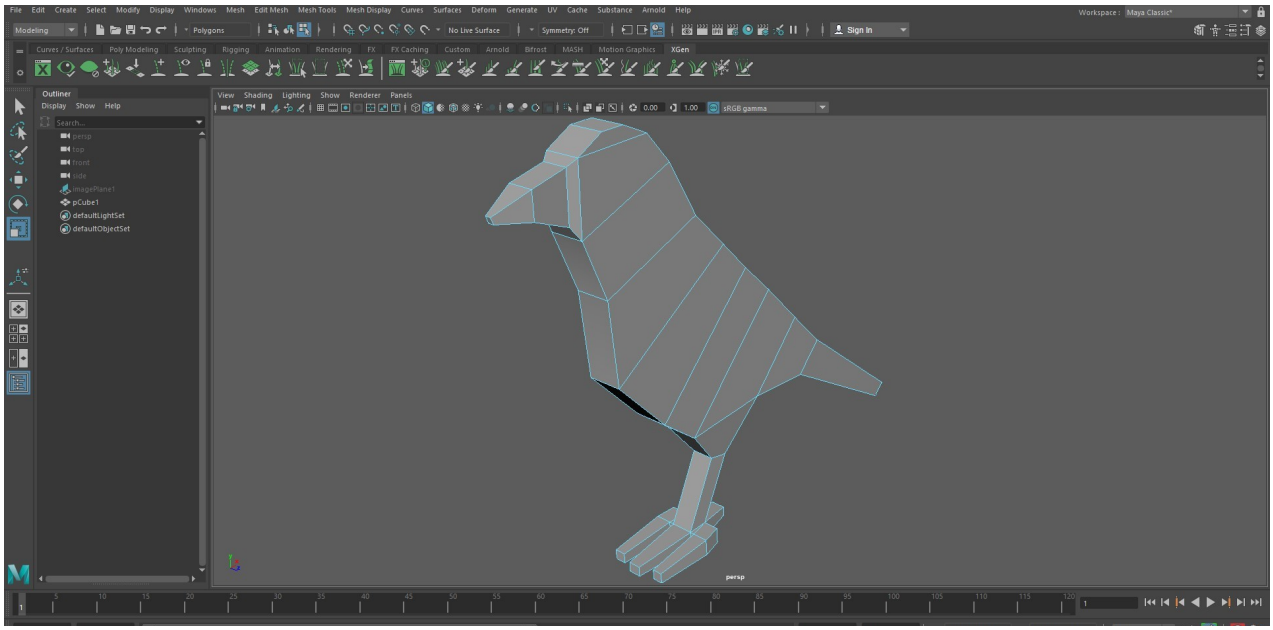

Рис. 5.6

Аби злити деякі точки в одну, наприклад, на лапках птаха, потрібно застосувати функцію під назвою Target Weld. Клікаємо виділену червоним кольором на рисунку 5.7 функцію та, клікаючи на потрібну точку, яку хочемо

злити, перетягуємо на іншу, яку теж хочемо злити з першою, так як це показано на рисунку 5.7. Аби вимкнути функцію, потрібно клікнути на звичайний курсор, який розташований зліва у вікні нашого середовища.

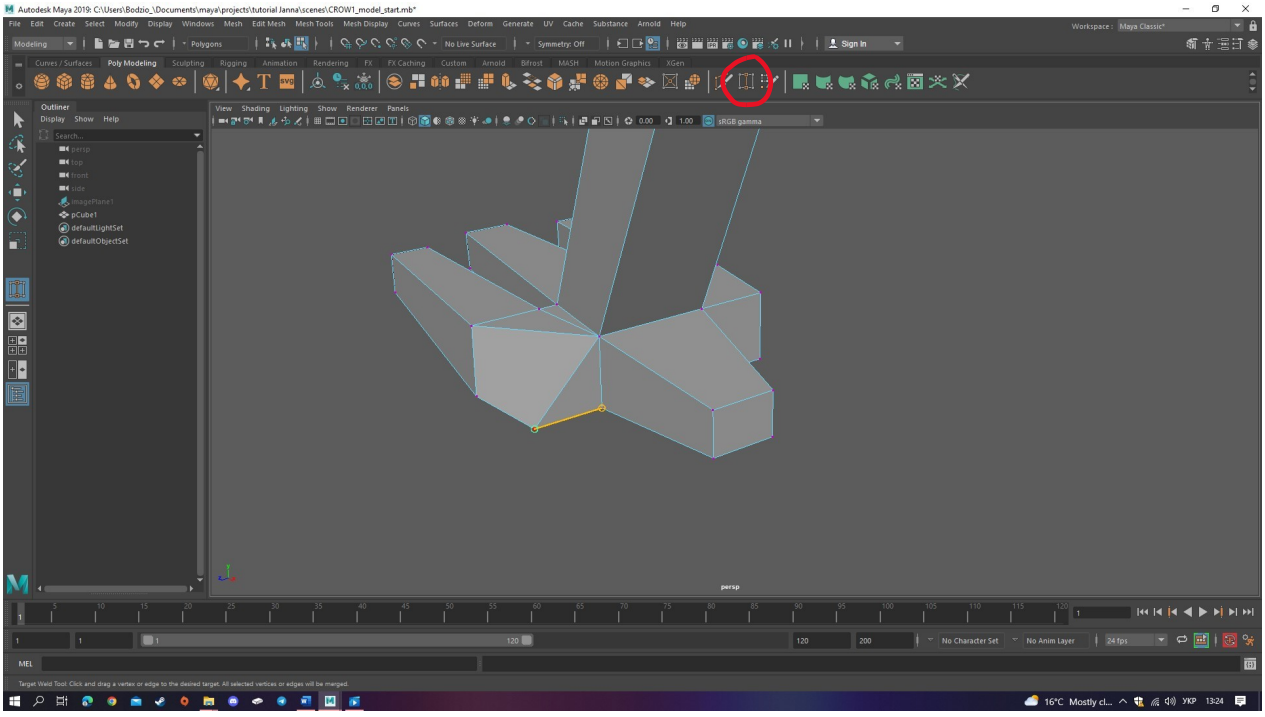

Рис. 5.7

Для того, аби відобразити модель дзеркально у майбутньому, потрібно пересунути модель на перед референсу, та видалити площини, які утворилися ззаду моделі (дивитися рисунок 5.8). (Аби сховати наш референс вибираємо його в Outliner'і зліва,

Display >> Hide >> Hide Selection. Зробити видимим - Display >> Show >> Show

Selection), аби зробити наш референс прозорим – виділяємо його, клікаємо на клавішу(Channel Box), яка виділена червоним на рисунку 5.8 зверху зправа, та знаходимо поле Alpha Gain змінюючи значення з одиниці на 0.15. чи будь-яке зручне нам значення.

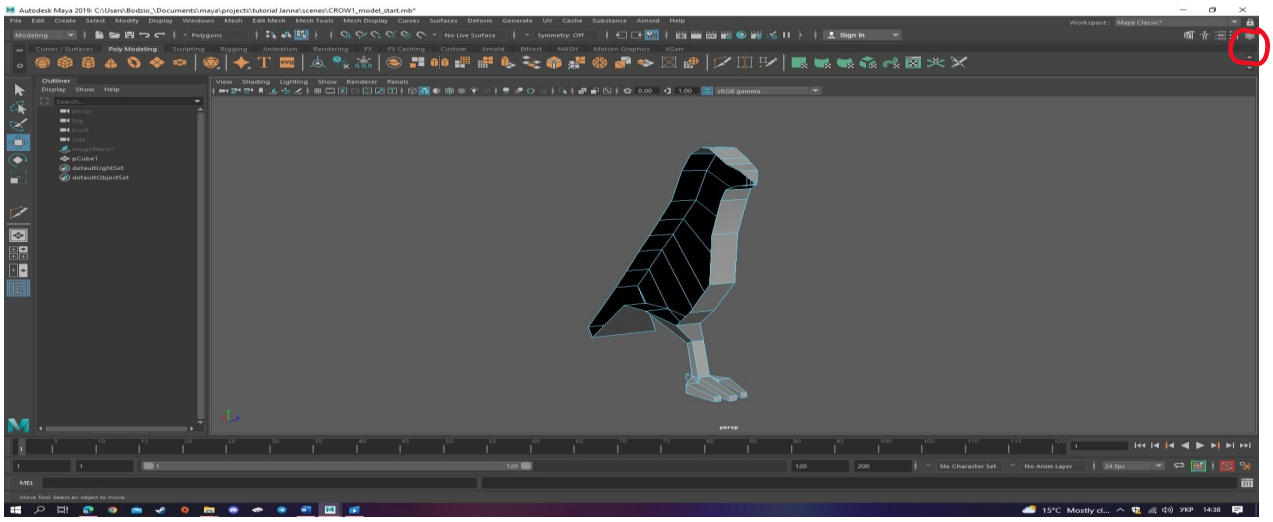

Рис. 5.8

Перед початком наступної дії очистимо історію змін та заморозимо трансформації аби ліквідувати можливі проблеми: Edit >> Delete By Type >> History; Modify >> Freeze Transformations;

Наступним кроком є завантаження фронт референсу до майї, як було зроблено у випадку першого референсу, регулюючи налаштування в ‘Attribute editor’. Після цієї дії, вкладка Mesh Tools >> Insert Edge Loop клікаємо на квадратик збоку, аби відкрилася панель налаштувань цієї функції. За допомогою ‘Edge Flow’ регулюємо силу випуклості при додаванні нових країв до нашої моделі. Додаємо 2 – 3 лінії вздовж тіла, аби округлити його, і, тим самим, створюємо плацдарм для видовження крила з полігонів тіла птаха. Дотримуючись геометрії тіла референсу, формуємо крила(Extrude). Результат має бути приблизно таким, як на рисунку 5.9.

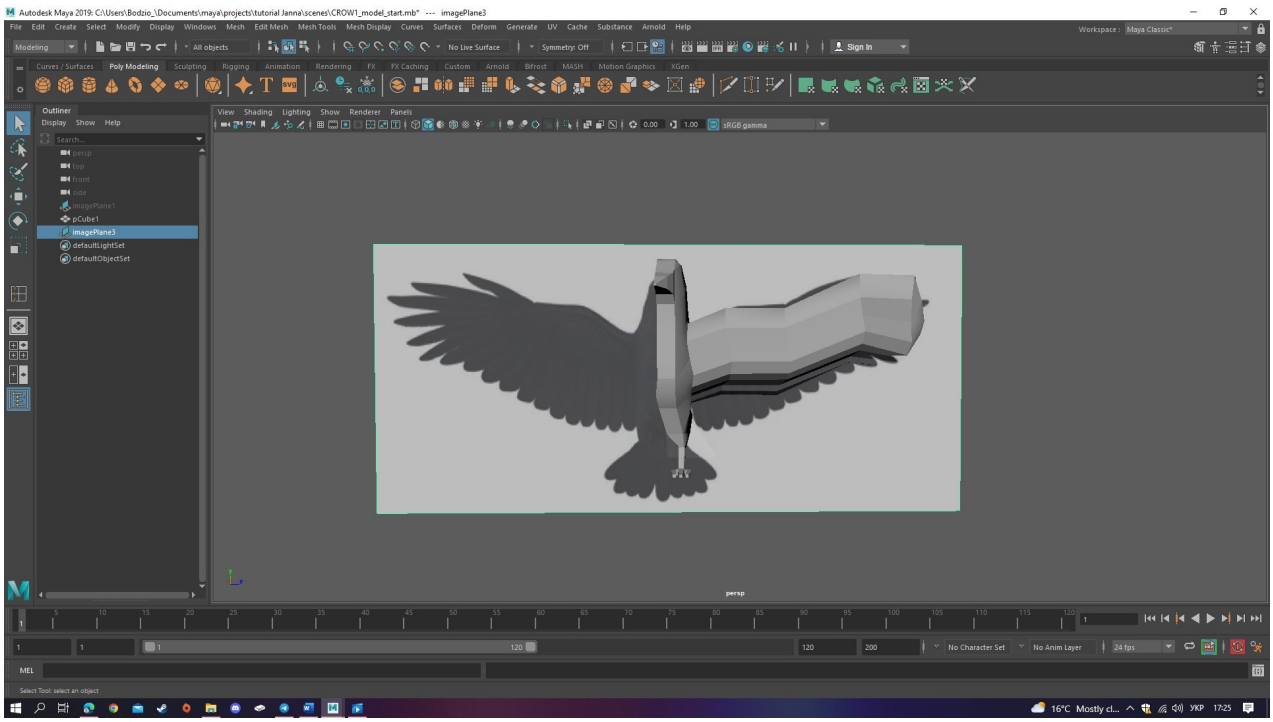

Рис. 5.9

Так як наш птах стоїть дещо зігнутим, крила не зовсім пасують до референсу, але якщо він випрямиться, крила будуть такої самої форми як на ескізі. Mesh Display >> Soften Edge – зробить модель більш приязною для ока.

Наступним кроком є створення повноцінної моделі з однієї її половини за допомогою дзеркала. Виділяємо нашу половину як об’єкт, Mesh >> Mirror клікаємо на квадратик аби підправити змінні – обираємо у з’явившомуся вікні потрібну ось відображення (подивитися свою ось можна у лівому нижньому кутку інтерфейсу MAYA) і нажимаємо ‘Apply’ або ‘Mirror’.

Перед цим, потрібно обов’язково перевірити чи край нашої моделі знаходиться на одній лінії, аби усунути подальші можливі проблеми. Якщо лінія половини вийшла не рівною – повністю виділяємо край розрізу, тримаємо клавішу ctrl і нажимаємо ПКМ, тримаючи, переводимо курсор на ‘To Vertices’ і відпускаємо, натискаємо кнопку в інтерфейсі MAYA, який виглядає як магніт з # в другому ряді інтерфейсу. Переміщаємо і знову вертаємось до центральної осі – проблема виправлена.

Якщо ж виникнуть проблеми із дзеркалом – переверніть його за допомогою характеристик в його Channel Box'і або Layer Editor (зверху справа екрану) на 180% і не забудьте вимкнути магніт до сітки простору MAYA.

Результат усіх вище перечислених дій ми можемо побачити на рисунку 5.10. Аби редагувати одночасно обидві частини моделі, нажимаємо крайній лівий знак згори справа екрану майя, аби відкрити інструмент моделювання . Виділяємо лише одну лінію, яка знаходиться на симетричному розрізі нашого птаха і нажимаємо кнопку 'Symmetry' у відкрившомуся вікні. Тепер, вибираючи одну точку, можна редагувати одразу обидві точки, які розташовані на одинаковій відстані від виділеного умовного перерізу моделі. Аби вимкнути функцію – знову нажимаємо 'Symmetry'.

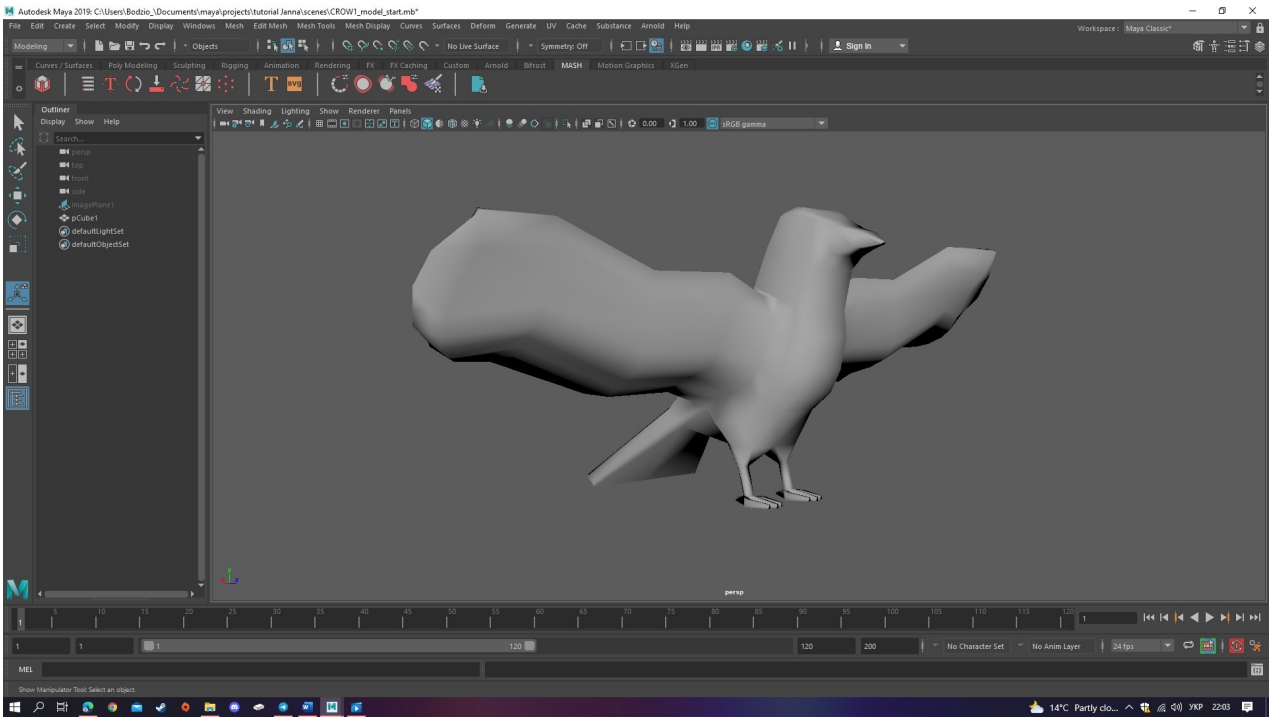

Рис. 5.10

Застосовуючи Mesh Tools >> Sculpting Tool >> Pinch Tool та Smooth Tool згладжуємо грубі краї моделі та подекуди їх, навпаки, витягуємо. За допомогою рейок в налаштуваннях функції, можемо регулювати силу витягування/згладжування. Це дуже ефективний інструмент, який заощаджує багато часу, замість редагування локалізації точок поверхні птиці. Його слід застосовувати після корекції точок поверхні, адже саме так, точність моделі буде найвищою(Рис. 5.11).

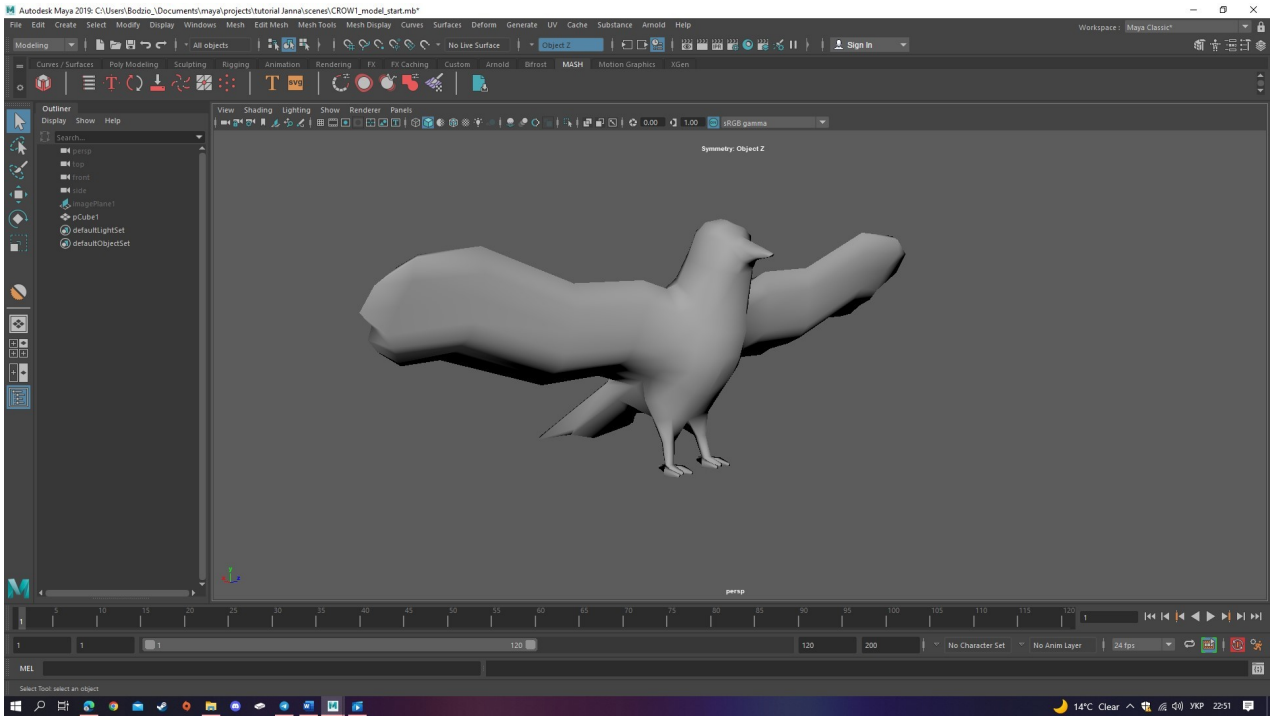

Рис. 5.11

В результаті, отримуємо трохи кращу, для людського ока, модель. Очистимо історію і заморозимо трансформацію.

5.3 Підготовка моделі до накладання шейдерів

Змінюємо ‘MAYA Workspace’ на ‘UV Editing’ у верхньому правому кутку нашого робочого простору. Процес буде перебігати тільки на правій частині моделі, а потім віддзеркалиться на інші поверхні. Отже, виділяємо усю поверхню правого крила, направляємо перспективну камеру так, аби проектант дивився на модель максимально перпендикулярно найбільшій площі правого крила. У вікні UV Editor’а вкладка Create >> Camera Based.

Пересуваємо новостворені і виділені частини за допомогою стрілок на вільне місце в ‘UV Editor’ просторі, незалежно від значень на осі XY. Виділяємо лінію перерізу крила разом із лінією, що є границею крила, у звичайному середовищі MAYA, як це показано на рисунку 5.12. Відділяємо передню частину крила від задньої за допомогою вкладки Cut/Sew >> Cut в середовищі інструменту ‘UV Editor’.

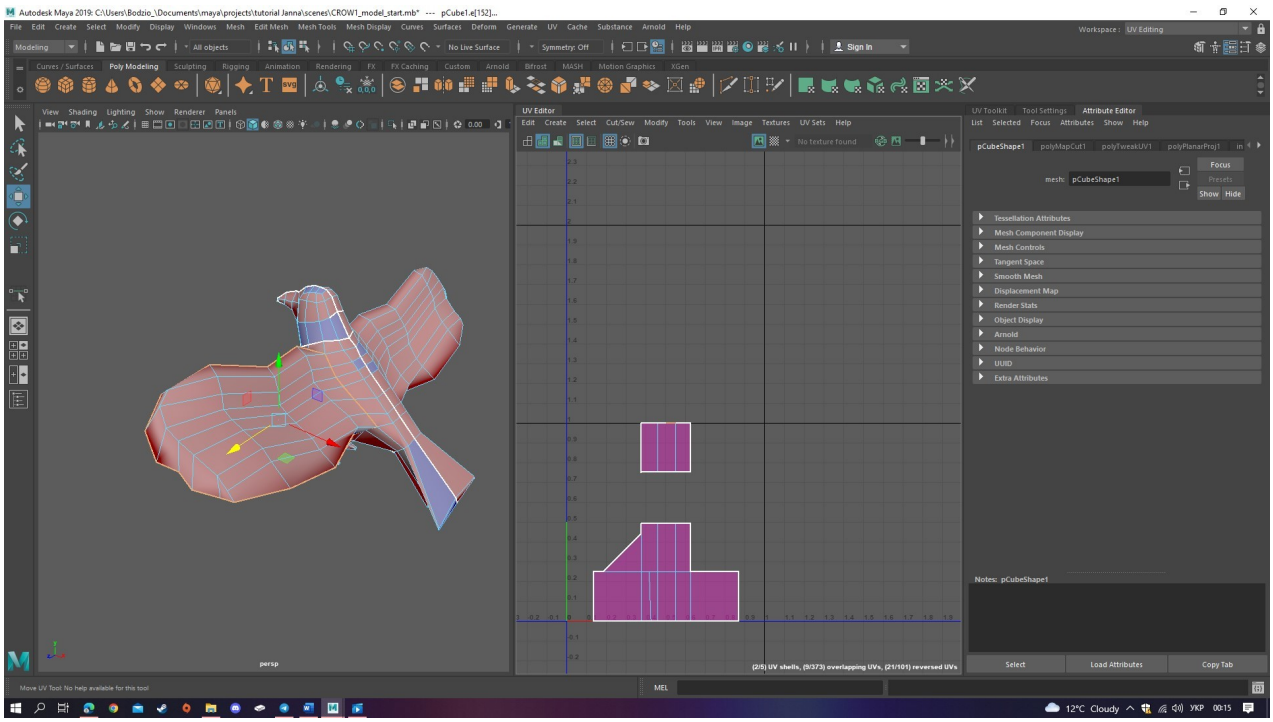

Рис. 5.12

Після цього, у тому ж просторі, затримуємо ПКМ і перетягуємо курсор на UV, виділяємо будь яку точку створених поверхонь крил і виконуємо дії в UV Editor: вкладка Select >> Convert Selection >> UV Shell, перетягуємо виділені точки цілісно на вільне місце біля іншої частини крила. Виділяємо одну цілісну частину крила, переходимо у вкладку редактора UV Toolkit та клікаємо дві кнопки, а саме Optimize та Unfold, як це зображено на рисунку 5.13 червоним кольором. Виділяємо іншу частину крила та повторюємо дії.

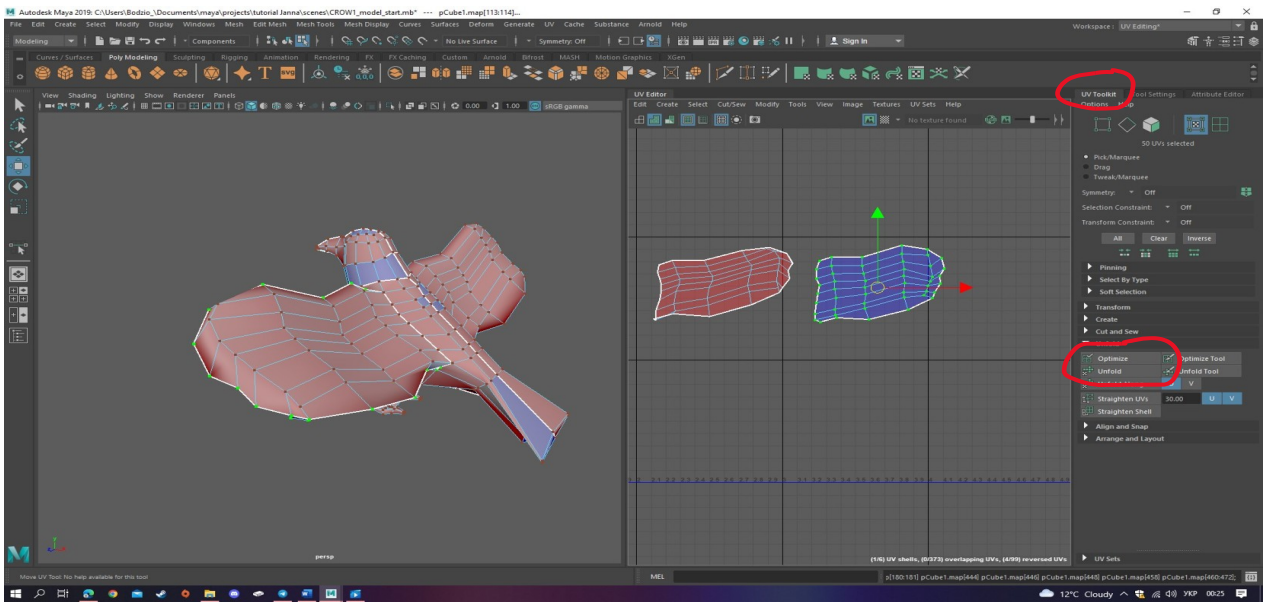

Рис. 5.13

У сухому залишку маємо одне оптимізоване крило, залишилося повторити схожі дії відносно інших частин тіла. Бажано, відділяти їх згідно з різними текстурами, наприклад: пальці будуть мати текстуру шкіри, ноги – текстуру пухкого пір’я, крила – велике пір’я, клюв та груди – середнє за величиною пір’я.

У даному проекті - це не грає великої ролі, адже шейдери будуть достатньо темні, аби не помічати різниці. Якщо маємо справу зі світлими об’єктами – даний етап буде достатньо важливий для проекту в цілому.

Повертаючись до теми, різниця процесів оптимізації частин моделі полягає в тому, якої форми ця частина, що ми намагаємося оптимізувати. Якщо це пальці – вкладка Create >> Cylindrical (form). Для крил ми вибрали прив’язку до камери, хоча можна було теж Create >> Planar замість Create >> Camera Based. Усе залежить від форми об'єкта, для якого створюється проекція конкретної форми.

Таким чином, для тіла, голови та клюва - слід використати Create >> Planar. Для пальців та ніг – Create >> Cylindrical. Теж, слід зауважити, що не варто робити проекцію для усієї ноги повністю, слід брати кожен палець окремо, інакше - текстура буде виглядати жахливо, якщо це деталізована фігура а не ‘low detalization’ туторіал. Варто теж розрізати модель в місцях, де ряд полігонів зациклений у коло, наприклад: палець. Паралелепіпед ніколи не вдасться розвернути на 2D поверхню без викривлення проекції, тому, варто причіпляти ‘кришки’ до циліндрів лише однією стороною за допомогою функції Cut/Sew >> Move And Saw. Для легшого розуміння, спробуйте вдома розвернути жерстяну банку на 2D поверхню так, аби її кришка залишилася закритою.

Тож, відповіді немає – це неможливо. Можливо тільки при відкривши банку, розрізати її один раз і залишивши тільки одну дотичну точку кришки з краями банки. Аби знати який край до якого прикріпити потім, можна просто

навести курсором на кожен край ‘банки’ і тоді спільні краї кришки та банки підсвітяться, після чого, можна буде застосувати функцію Move And Saw.

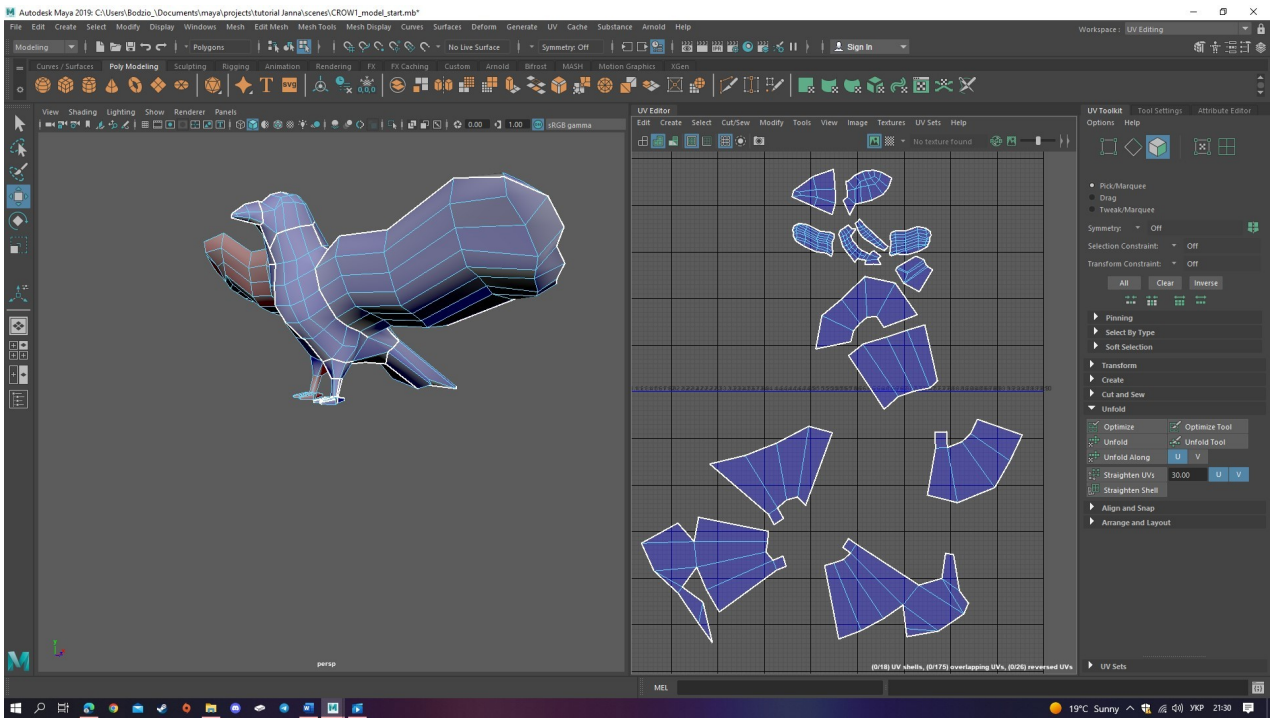

Рис. 5.14

Клопітливою працею завершено роботу над половиною птаха, залишилося відобразити все що ми зробили на іншу половину птаха та злити краї полігонів у потрібних місцях, аби спина була разом, груди разом, перехід до ніг зі спільним краєм. Видаляємо історію об’єкта.

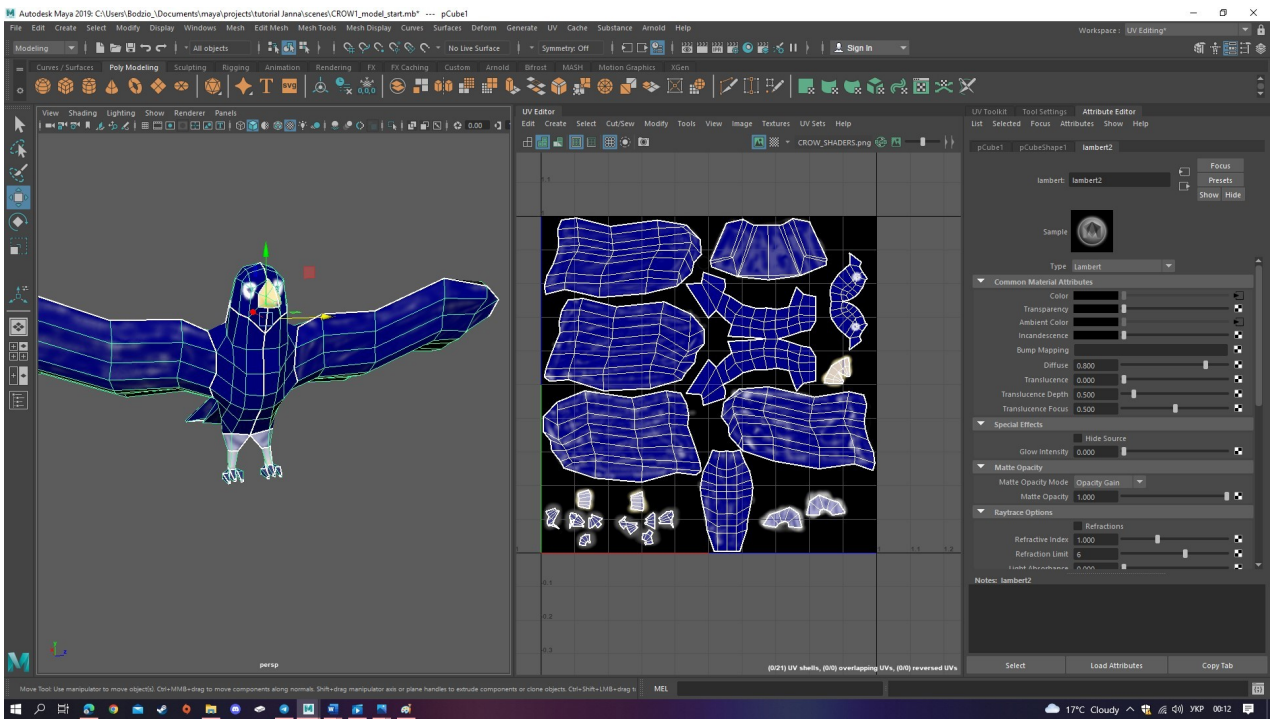

Рис. 5.15

Аби відобразити всі перспективи, створюємо нове дзеркало Mesh >> Mirror, зазначаємо галочку на Flip UV’s розбираємося із залишившимися деталями, переходимо у вкладку справа в UV Editor’і, яка називається Arrange And Layout, нажимаємо кнопку Layout, редагуємо трохи те, що вийшло, фліпаємо*, видаляємо історію.

В вікні UV Editor вкладка Image >> UV Snapshot, ставимо рейку на максимум, вибираємо формат JPEG, змінюємо ім’я файла та локалізуємо його до нашого проекту в фолдер sourceimages, затверджуємо. Робимо шейдери вирізаючи з фото потрібні форми або малюємо їх самі.

Після виконання цієї роботи: ПКМ >> Assign Favorite Material >> Lambert. Заходимо в ламберт в історії моделі, клікаємо на шаховий значок біля поля Color >> File, обираємо створені шейдери, шукаємо знову Ambient Color і повторюємо дії.

Таким чином, шейдери накладуться двічі і будуть менш прозорими і більш контрастними.

Так як для нашого туторіалу був створений ще один, інший птах – створення шейдерів було максимально спрощено до рівня анімованих мультфільмів, - процес шейдерінгу перебігав у програмі Paint 3D. Вийшло доволі симпатично, як на мій погляд.

Чистимо історію, зберігаємо проект.

Аби відобразити текстуру поверхні, потрібно нажати клавішу 6. 1 – звичайний вид, 2 – частково згладжений режим відображення, 3 – згладжений, 4 – сітка-мод.

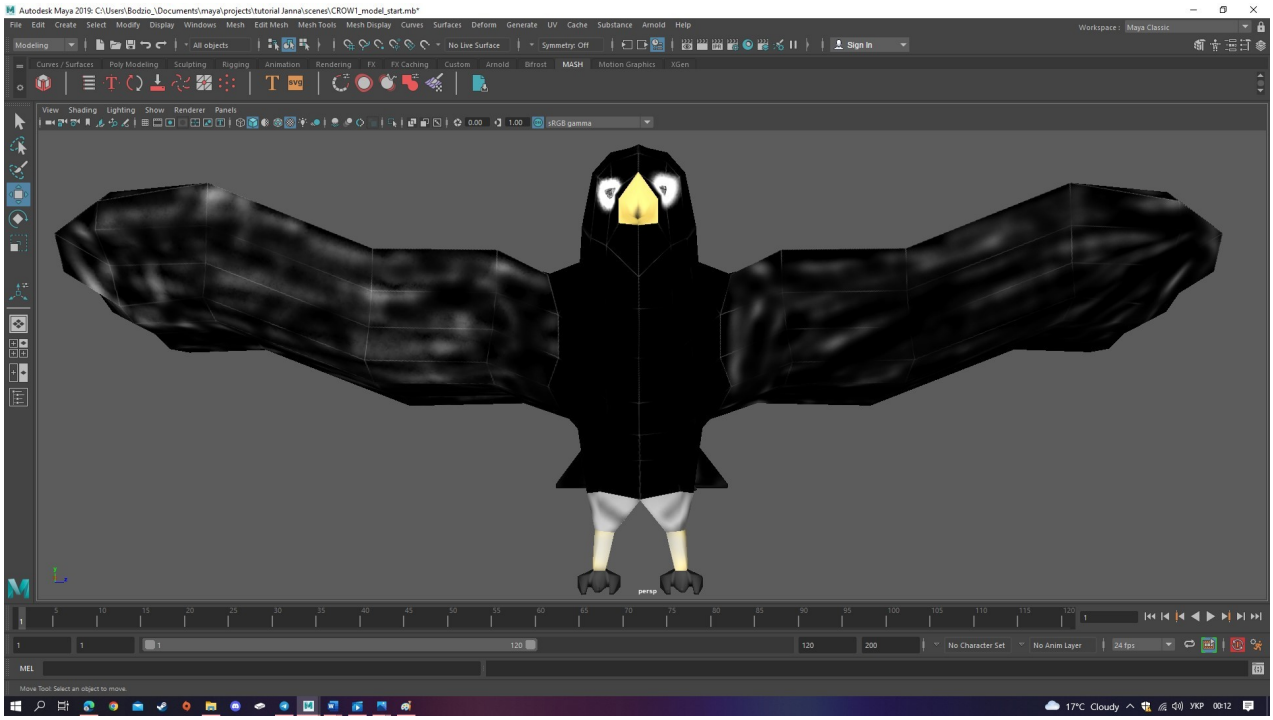

Рис. 5.16

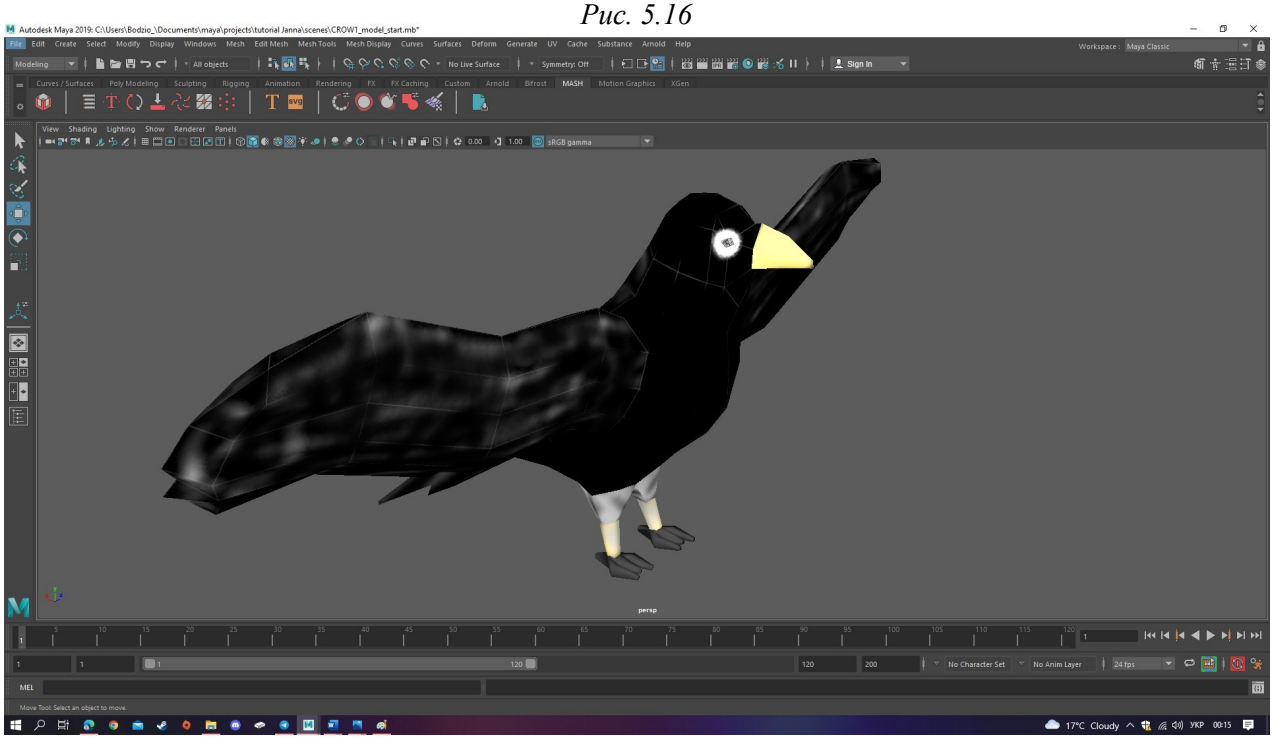

Рис. 5.17

5.4 Створення скелета за допомогою джоінтів

На початку – декілька гарячих клавіш для MAYA: виділяємо 2 джоінти, нажимаємо клавішу P – останній джоінт стане материнським для всіх джоінтів виділених до нього.

G – перейти до останньої застосованої функції.

При переміщенні об’єктів затримуємо клавішу V аби вирівняти об’єкт відносно інших ліній, площин, тощо. Важливо, аби джоінти були посередині тіла птаха у фронтальній перспективі.

Отже, переходимо в режим ‘Animation’, для цього потрібно натиснути кнопку ‘Modeling’ зверху зліва та вибрати ‘Animation’. Переходимо до фронтальної камери, вкладка Skeleton >> Create Joints. Краще за все, рухомі частини тіла робити у горизонтальній позиції. Тобто два джоінти мають знаходитися на одній площині. Для цього, вирівнюємо джоінт голови до джоінта шиї, аби вони лежали обидва на осі на спільних координатах осі ОУ.

Те саме стосується пальців ніг, вирівнюємо джоінти за допомогою гарячих клавіш описаних вище. Важливо будувати джоінти від рутового* (root) джоінта що буде знаходитися приблизно в центрі тулуба на однаковій відстані від голови до хвоста, але не на одній прямій. Після цього, прив’язуємо джоінти до рута за допомогою гарячої клавіші (P). Для ніг, буде додана окрема ілюстрація структури джоінтів.

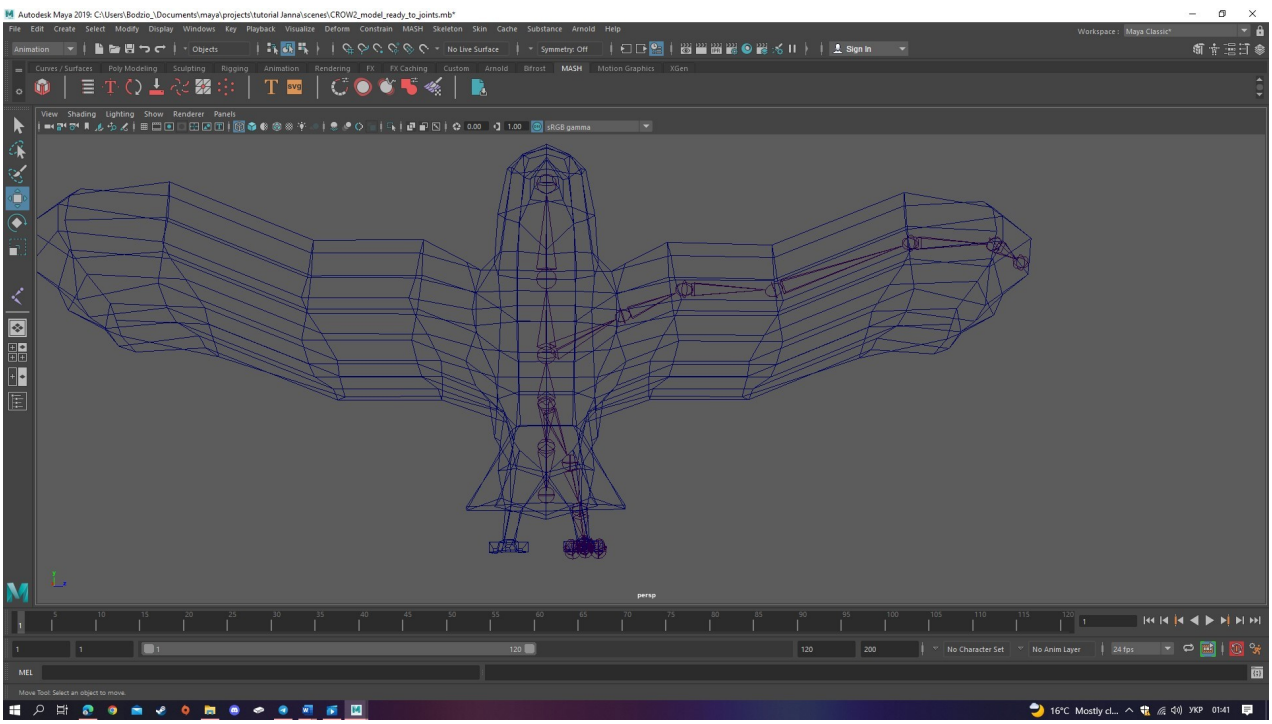

Рис. 5.18 - Структура джоінтів моделі птаха

На пальцях ми бачимо по 2 джоінти та 1 джоінт лапи, що зв’язує інші як перент*.

Виділяємо нашого птаха як об’єкт, Modify >> Freeze Transformations (квадратик аби відкрити прев’ю функції), ставимо пташку біля Joint Orient. Нажимаємо затвердити.

Наступною дією є виділення root вузла, Skeleton >> відкриваємо налаштування функції Orient Joint. Зазначаємо першу ось OY, у другій і третій секції зазначаємо завжди одну й ту саму пряму. У даному випадку зазначаємо пряму, на яку спрямований дзьоб птаха. Якщо він дивиться у від’ємні значення вашої осі – зазначаємо знак мінус у полі для вибору знаку. Нажимаємо затвердити. Виділяємо основний джоінт на початку крила, першою прямою обираємо напрямок, у який дивиться крило нашого птаха, другим та третім напрямком обираємо ось, куди дивиться ваш птах(куди буде скручуватись крило). Не можна забувати про знак осі.

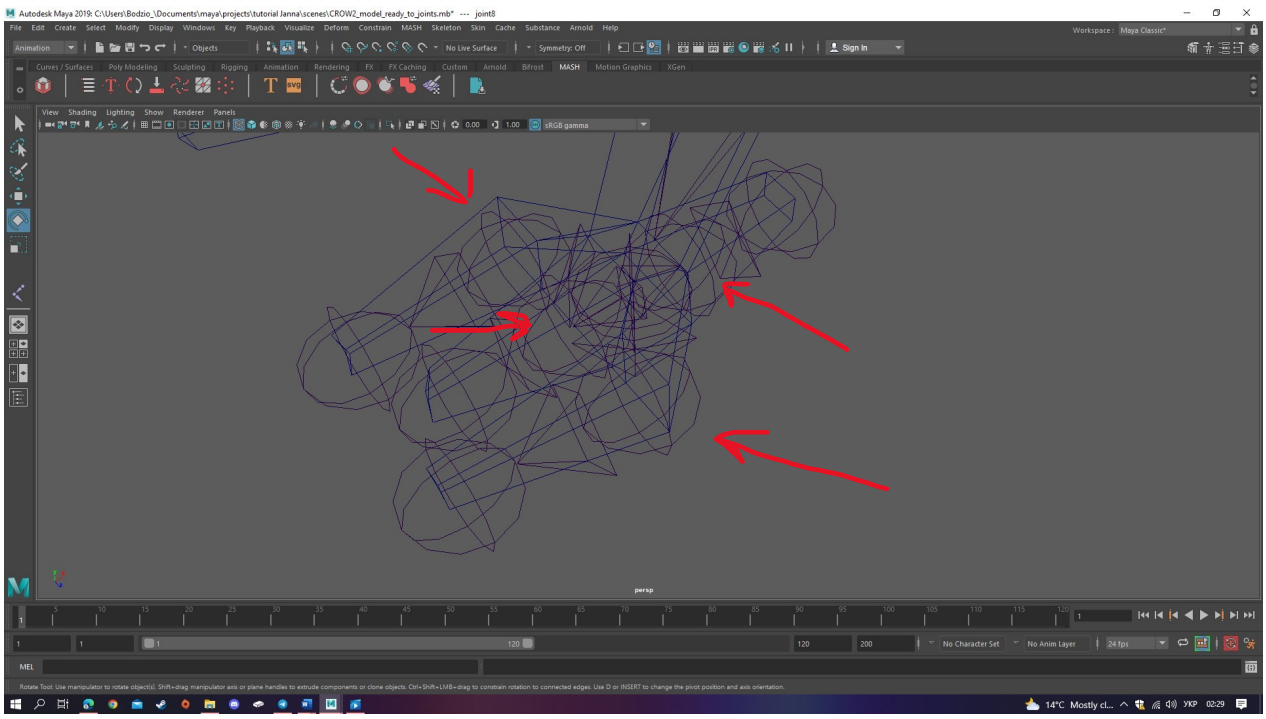

Рис. 5.19 - Структура джоінтів лівої ноги

Виділяємо за допомогою клавіші ‘Shift’ джоінти, які виділені червоними стрілками на рисунку 5.19. Зазначаємо для них першим напрямком пряму, куди дивиться птах, другим і третім напрямком обираємо ось у яку дивиться крило птаха, сторону якого ми редагуємо.

Навіщо ми це зробили? Щоб при переміщенні об’єкта, кінцівки знали куди крутитися, інакше – вони стануть вертітися в різні сторони; непотрібні нам сторони.

Наступним кроком є відображення джоінтів.

Спочатку перейменуємо їх, аби розуміти де який знаходиться. Виділяємо усі джоінти окремої частини тіла, наприклад, крила - по порядку і нажимаємо другу маленьку стрілочку зправа від поля sign in у верхньому ряду функціоналу програми. Клікаємо на значок квадратної сітки та вибираємо Rename. Вписуємо ім’я наприклад left_wing_1. Усі інші джоінти перейменуються автоматично.

Виділяємо джоінт початку крила і виконуємо дії: вкладка Skeleton >> Mirror joint. У першу строку вводимо ключове слово для усіх джоінтів крила, тобто left, а у другому полі пишемо те, на що змінимо його, тобто rght або right. Ставимо пташку на ‘behavior’, обираємо ось, яка послужить відображенням та клікаємо Apply. Те саме робимо із ногою. Відображення завершено.

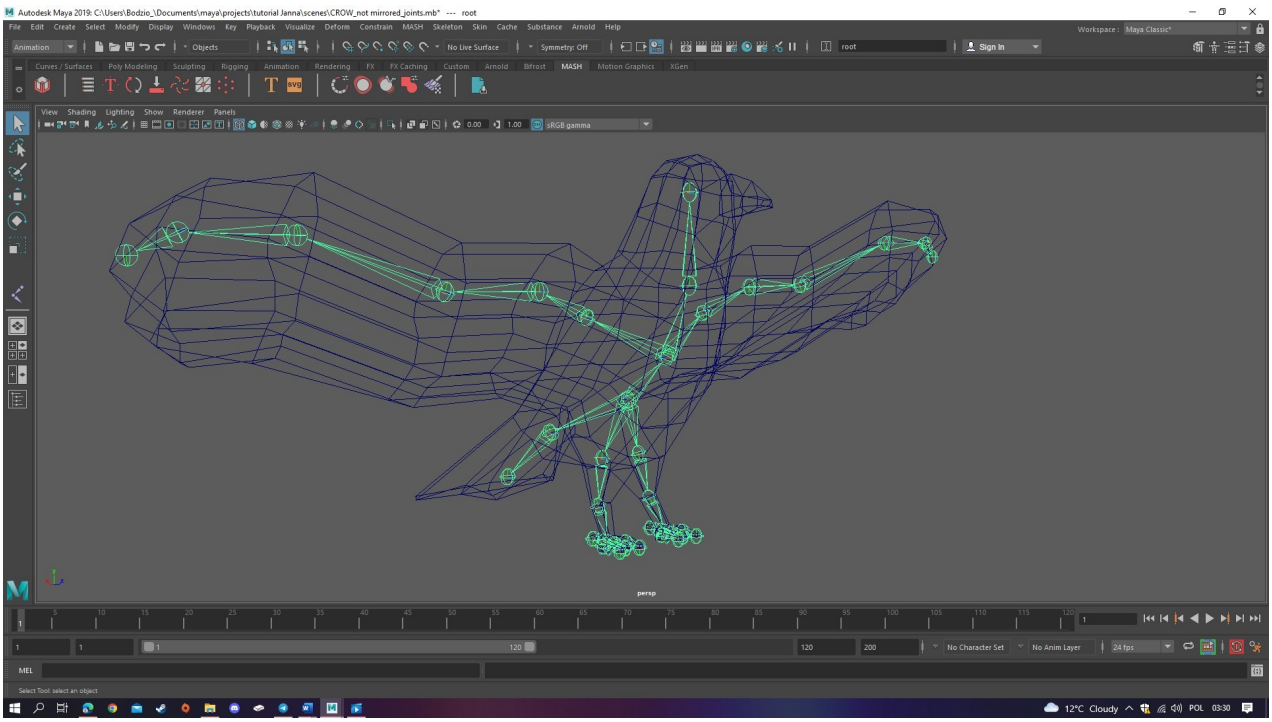

Рис. 5.20

Прив’язуємо модель до джоінтів: клікаємо на рут джоінт, об’єктово виділяємо нашу модель, вкладка Skin >> Bind Skin(квадратик опцій) встановлюємо Normalize Weights на post, Max Influences 5, Heatmap falloff 0.5, затверджуємо.

5.4.1 Прив’язка джоінтів до примітивів та встановлення функціоналу згину суставів

Вкладка Skeleton >> Create IK Handle, клікаємо на джоінт основи правої стопи, клікаємо на джоінт правої ноги, який розташований одразу після рута. Утворилася колінна залежність. Створюємо примітив – коло. Create >> NURBS Primitives >> Circle. Клеїмо його до центральної точки джоінта правої ступні за допомогою гарячої клавіші V. Виділяємо коло, чистимо його історію. Морозимо трансформацію (Modify >> Freeze Transformations).

Виділяємо коло а потім створений ‘IK Handle’ по черзі. Вкладка Constrain >> Point. Після цього одразу вкладка Constrain >> Orient. Для іншої ноги робимо те саме. Не забуваємо відрегулювати ширину кола, аби можна було легко за нього схопитися підчас анімації.

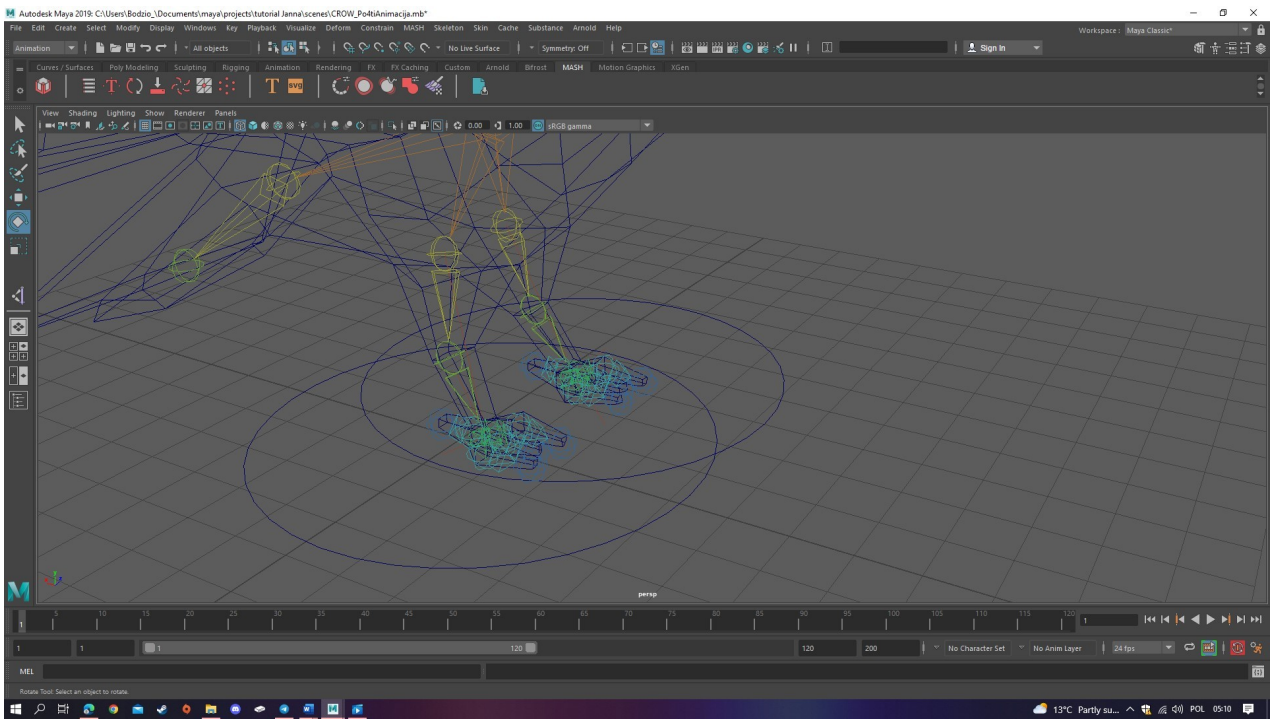

Рис. 5.21

Тепер, беручись за коло, можна крутити саму ногу, та пересувати її. Додаємо декілька складних деталей, що регулюють поведінку пальців. в які ми не будемо поглиблюватися. Від шиї до голови теж створюємо ‘IK Handle’ і додаємо перпендикулярне до площини сцени коло, фрізимо трансформацію, чистимо історію для кола, додаємо до нього залежності Point та Orient.

Створюємо ще одне коло на площині сцени, яке буде рухати цілого птаха. Чистимо історію, фрізимо трансформацію, перентимо усі створені нами кола до головного кола. Модель готова до анімації.

На виході маємо щось подібне:

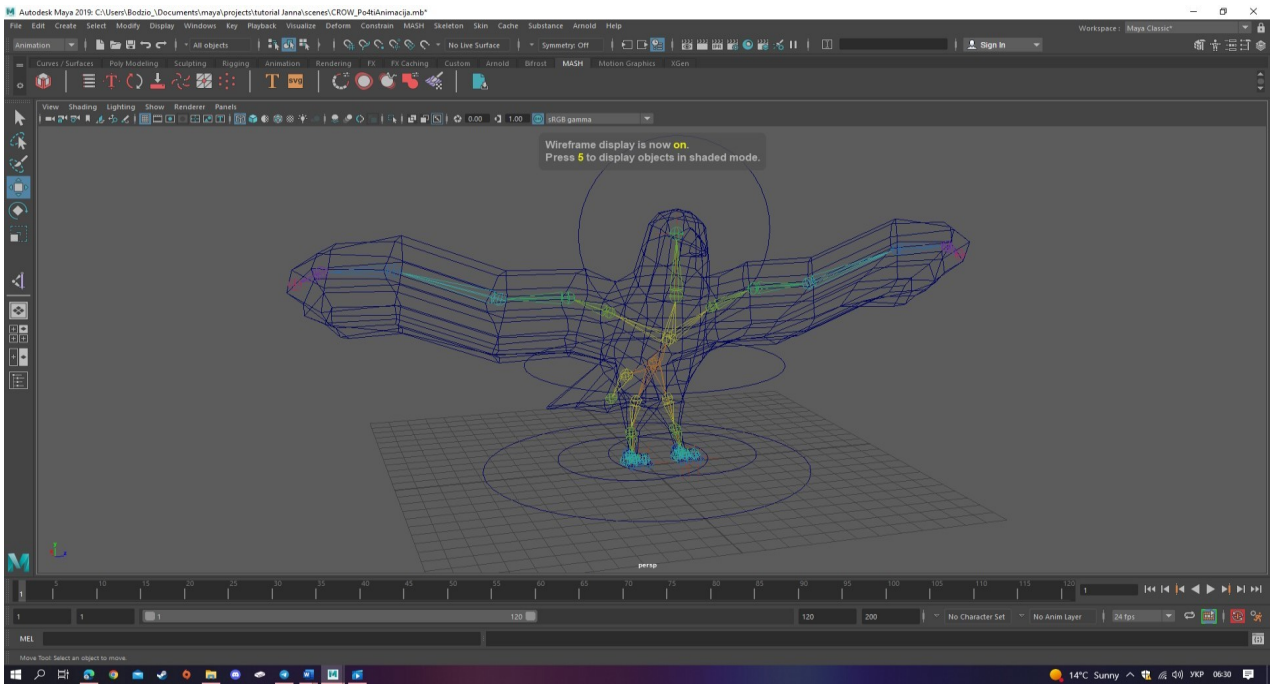

Рис. 5.22

5.5 Анімація моделі

Увесь цей час, на низу вікна програми ми мали рельси анімації і не помічали цього, отож, зараз вони нам знадобляться. Але спочатку шукаємо вкладку Panels >> Panel >> Node Editor, виділяємо усі вузли нашого лівого крила через ‘Shift’, і нажимаємо іконку додати вузли до графу. Розвертаємо їх нажимаючи на три риски, знаходимо у кожному атрибуті Rotation, розгортаємо міні-меню, і від 1 до 2 вузла створюємо зв’язок між атрибутами Rotation X, Y, або Z, в залежності від того на якій осі крила у вашої птиці згинаються в натуральну сторону. Рисунок 23.

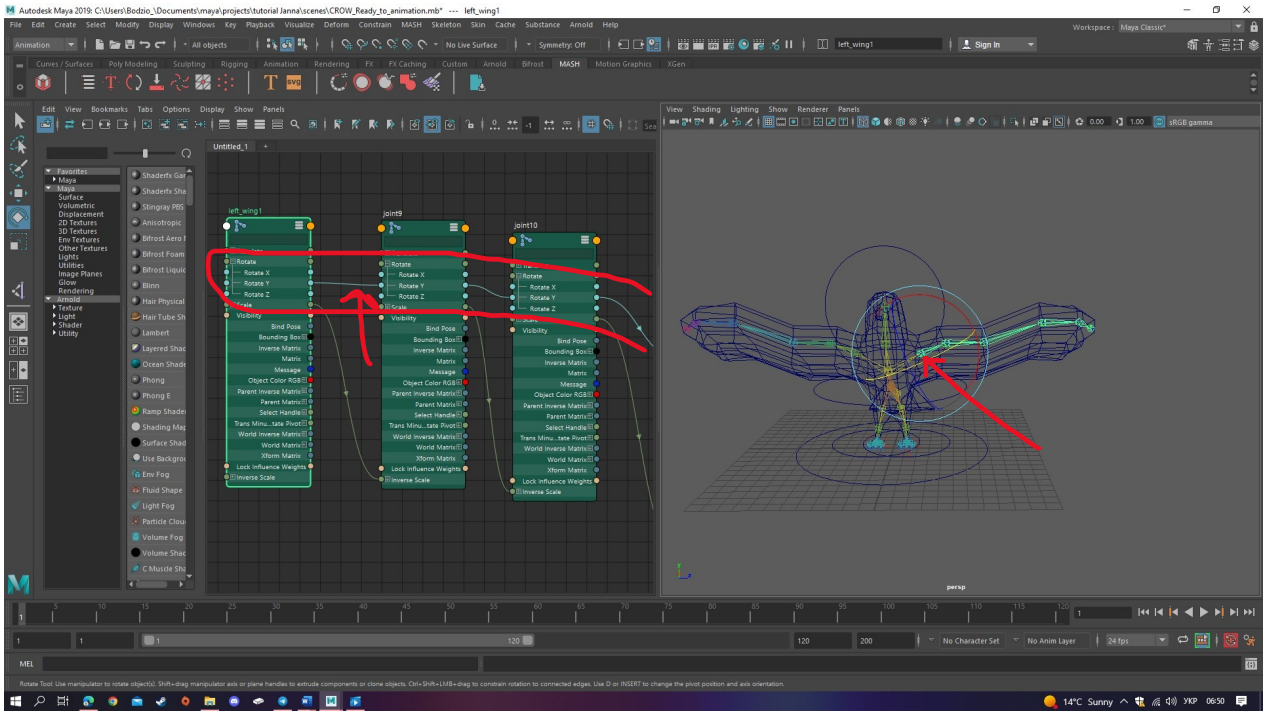

Рис. 5.23

Важливо, аби залежності йшли від основи крила до його кінця. Повторюємо дію для іншого крила. Це було зроблено для того, аби при виділенні двох основних їх джоінтів, виділялась і решта – це знадобиться підчас анімації.

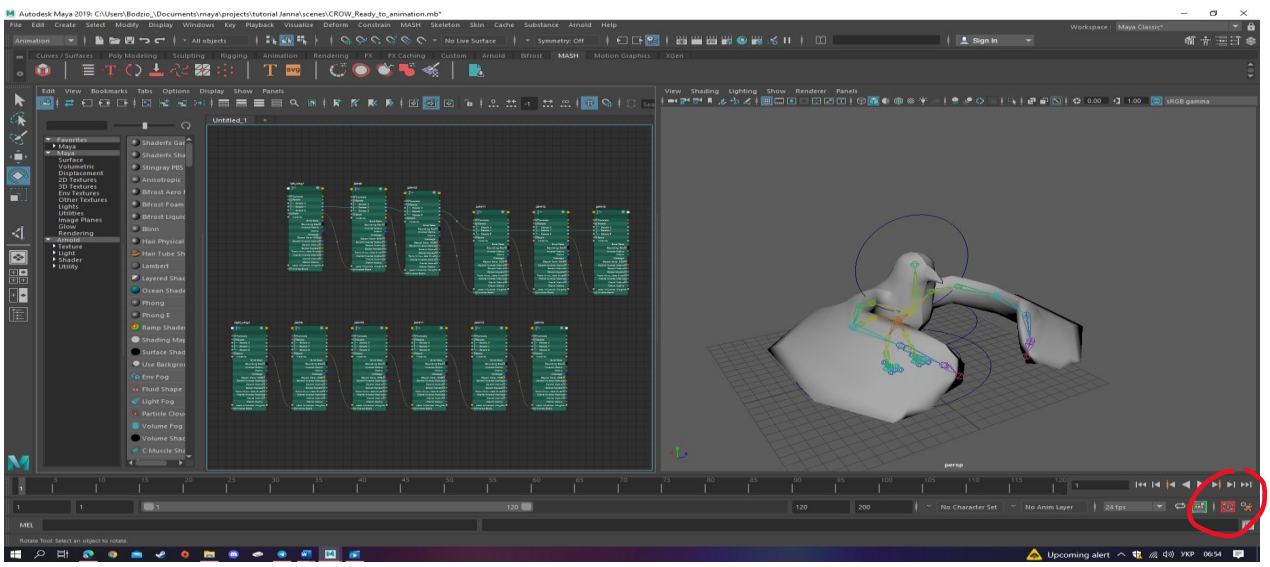

Рис. 5.24

Отже, встановлюємо нашу птицю на статичне положення, у якому ми її змоделювали. Клікаємо на початок прямої анімації, яка знаходиться знизу і пронумерована у порядку зростання, на 1 кадр.

Відкриваємо Channel Box для кожного кола, виділяємо координати Translate і Rotate, нажимаємо Key selected. Впевніться теж, що іконка знизу справа світиться червоним, як це видно на рисунку 24. Отже, сенс анімації заключається в тому, аби переміщати героя на сцені у кадрах і ставити бінди* на них, якщо, наприклад, ми у 5 кадрі поставимо птаха лівіше на 10 см, то від 1 до 5 кадру він пройде 10 сантиметрів. Для кожного кола – свої бінди та залежності, які можна встановити. Для кожного кола ми встановлюємо їх окремо.

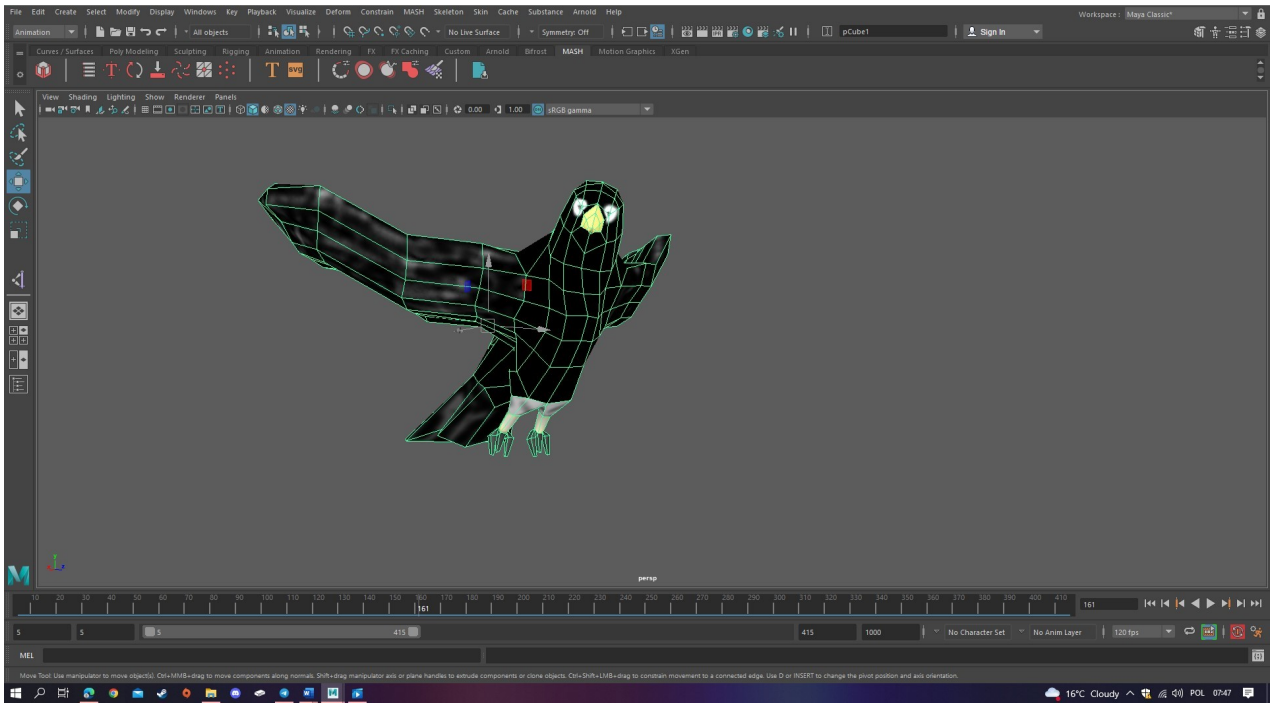

Рис. 5.25

Детальне зображення моделі надано у Додатку Б (АЛ9117.17.00.00.02 Огляд анімованої моделі)

6 Симуляція аеродинаміки отриманої моделі

Для виконання симуляції нам потрібне середовище AutoDesk CFD та модель з минулого розділу(Рис. 6.1).

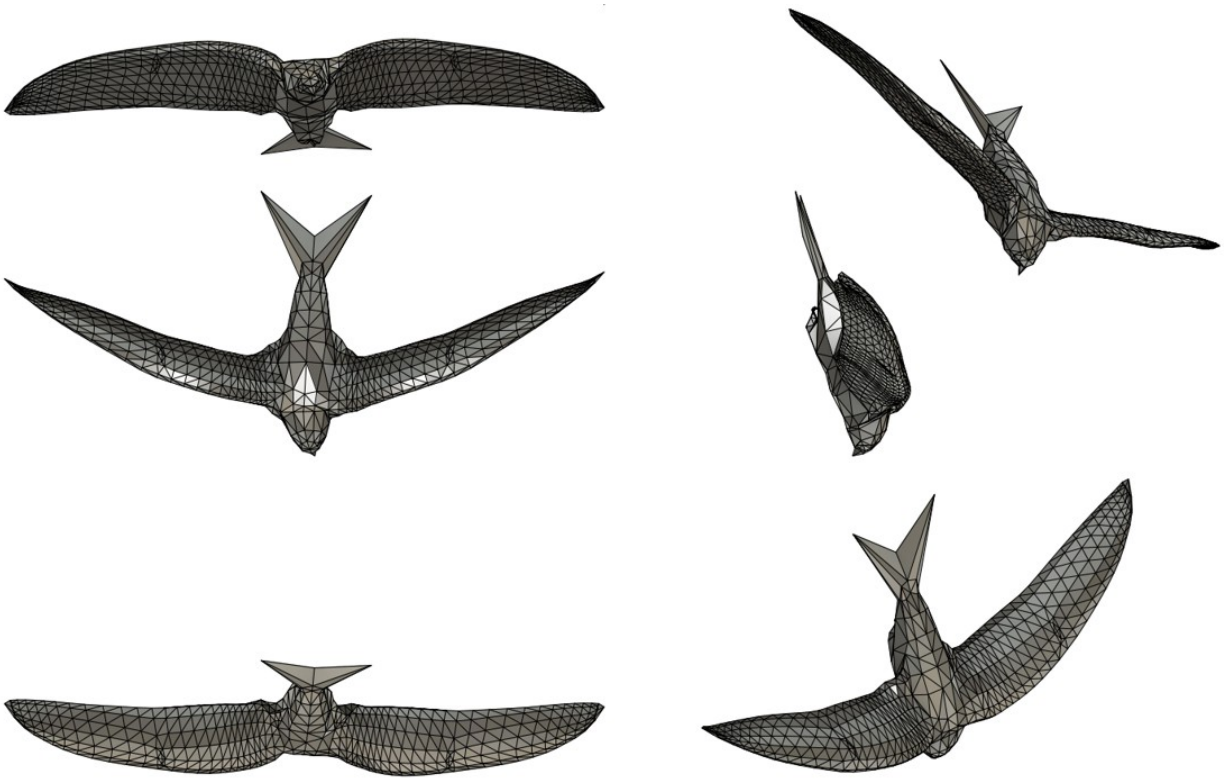

Рис. 6.1 – Статична модель у різних положеннях

Спочатку робится передготовка моделі. Завантажуемо модель у програму та виконуемо необхідні попередні налаштування, такі як встановлення фізичних властивостей матеріалу та параметрів середовища(Рис. 6.2).

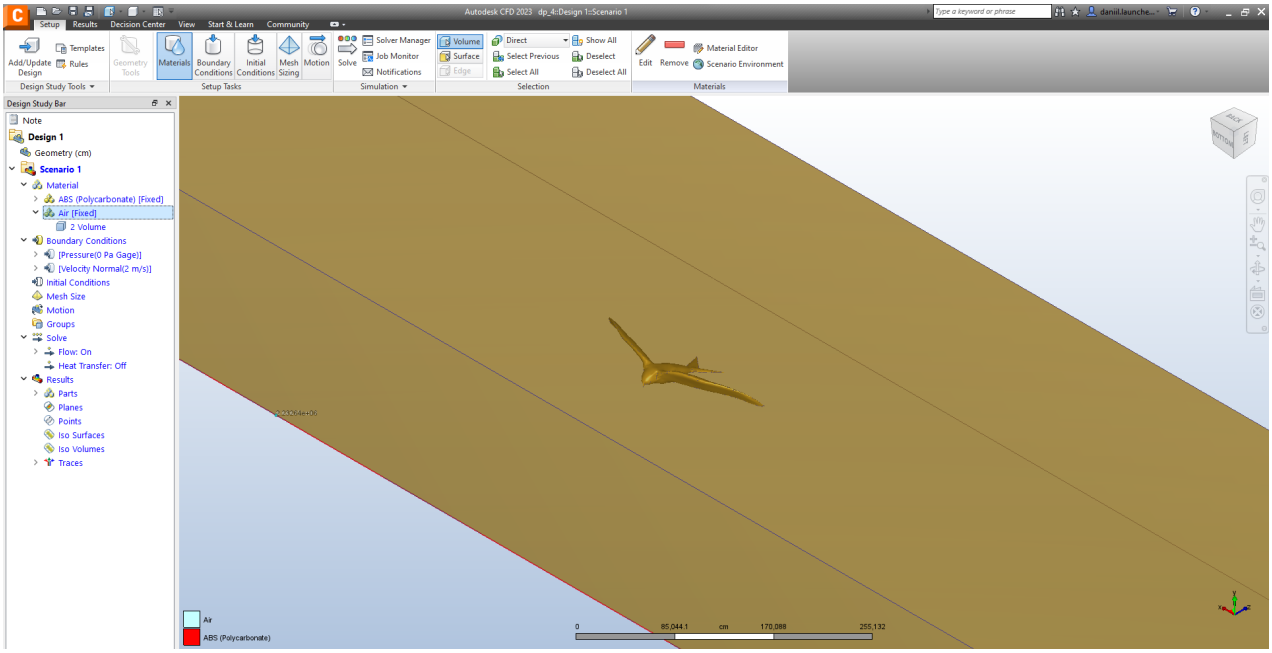

Рис. 6.2 – Вид на модель у прозорому об'ємі повітря

Далі робиться налаштування граничних умов, визначаємо граничні умови для симуляції, такі як швидкість повітряного потоку та тиск. Після цього ми

можемо налаштувати сітку об’єктів(Рис. 6.3), це сама модель та об’єм повітря навколо неї.

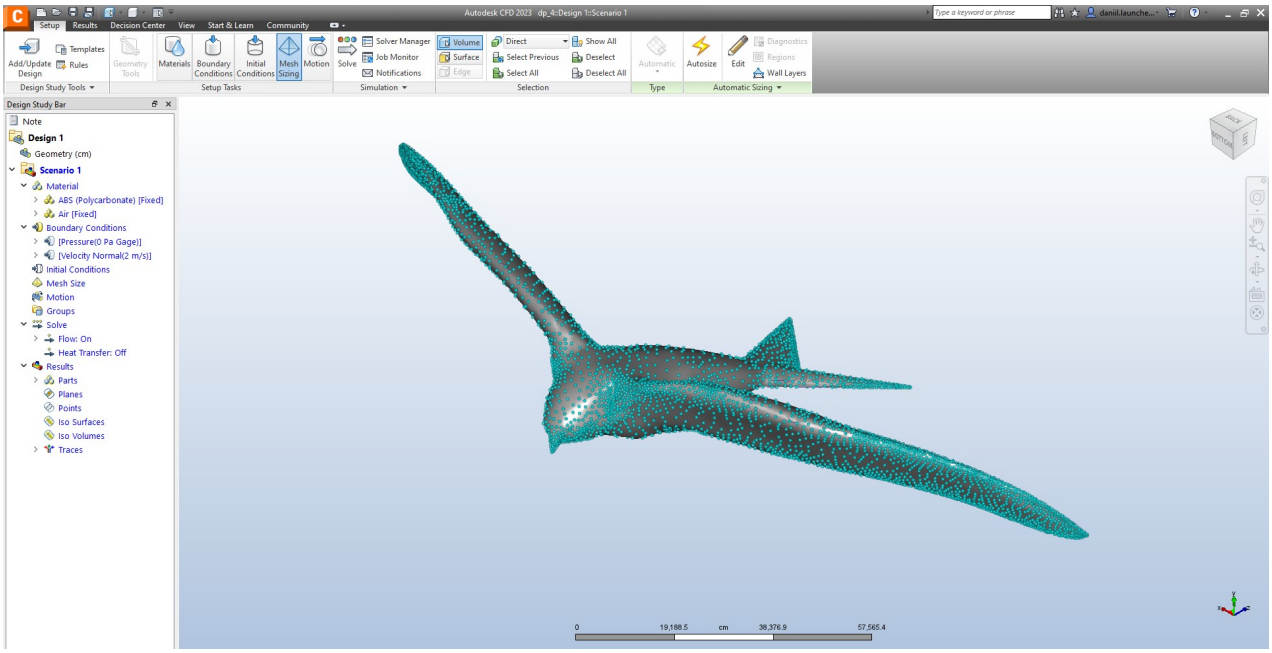

Рис. 6.3 – Авто розташування вузлів сітки об’єкта

Тепер ми можемо запустити симуляцію, для надійних результатів зробимо 100 ітерацій(Рис. 6.4).

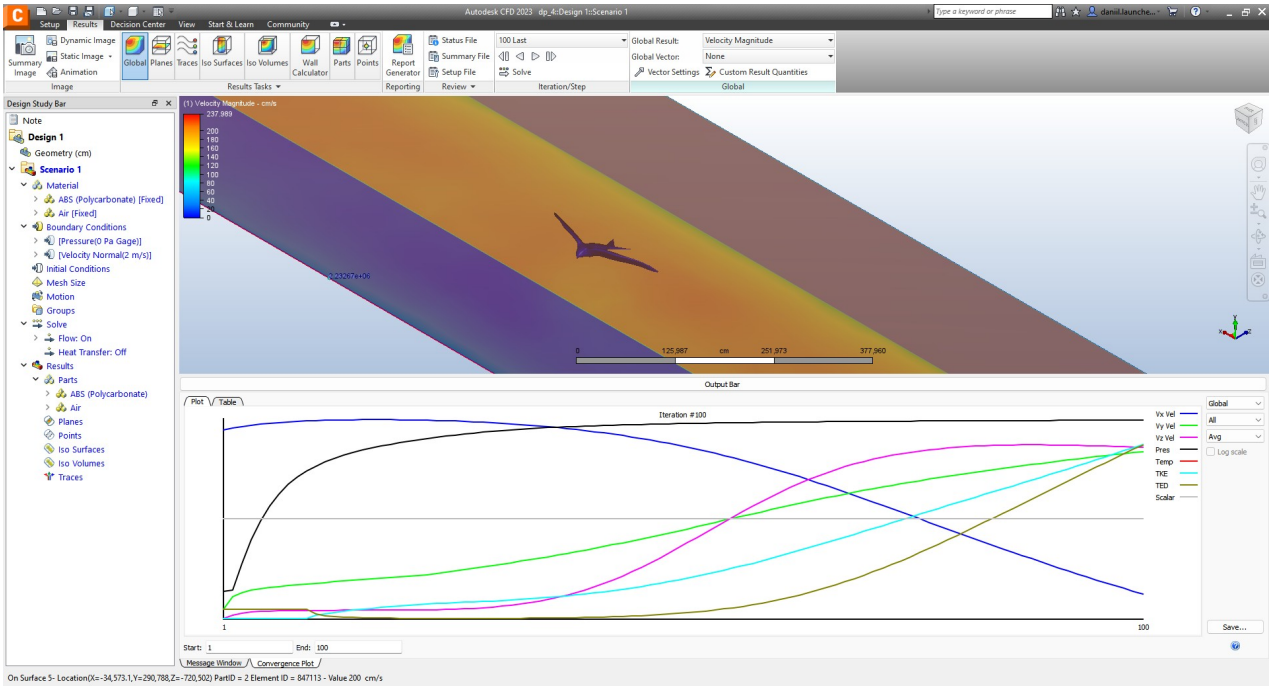

Рис. 6.4 – Загальні результати симуляції

Для подальшого розуміння та якісної обробки результатів була створена силова діаграма(Рис. 6.5)

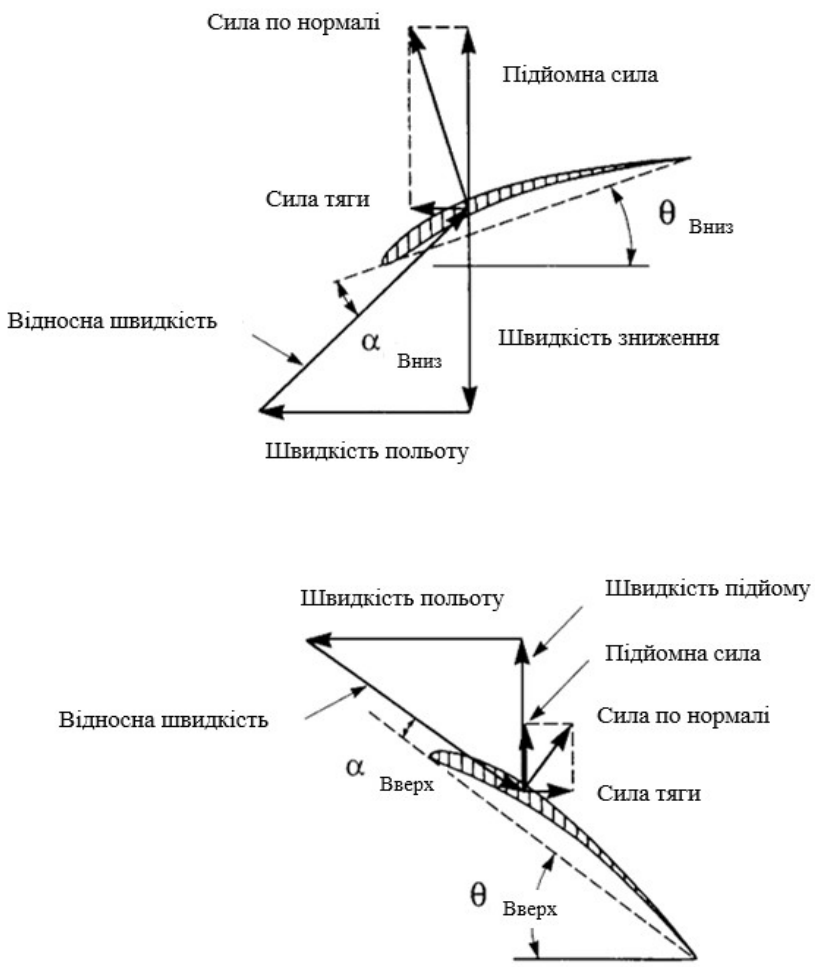

Рис. 6.5 – Силова діаграма польоту

Далі представлений графік середнього значення сили тяги та підйомної сили в залежності від частоти махів крил(Рис. 6.6(а, б)).

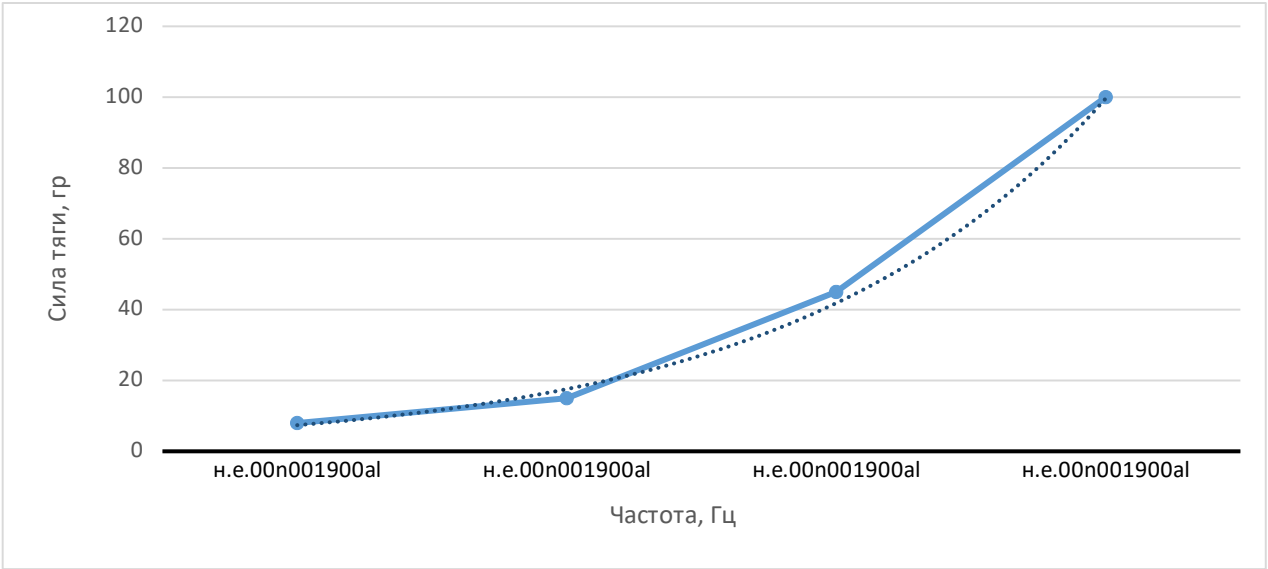

Рис. 6.6(а) – Графік залежності сили тяги від частоти

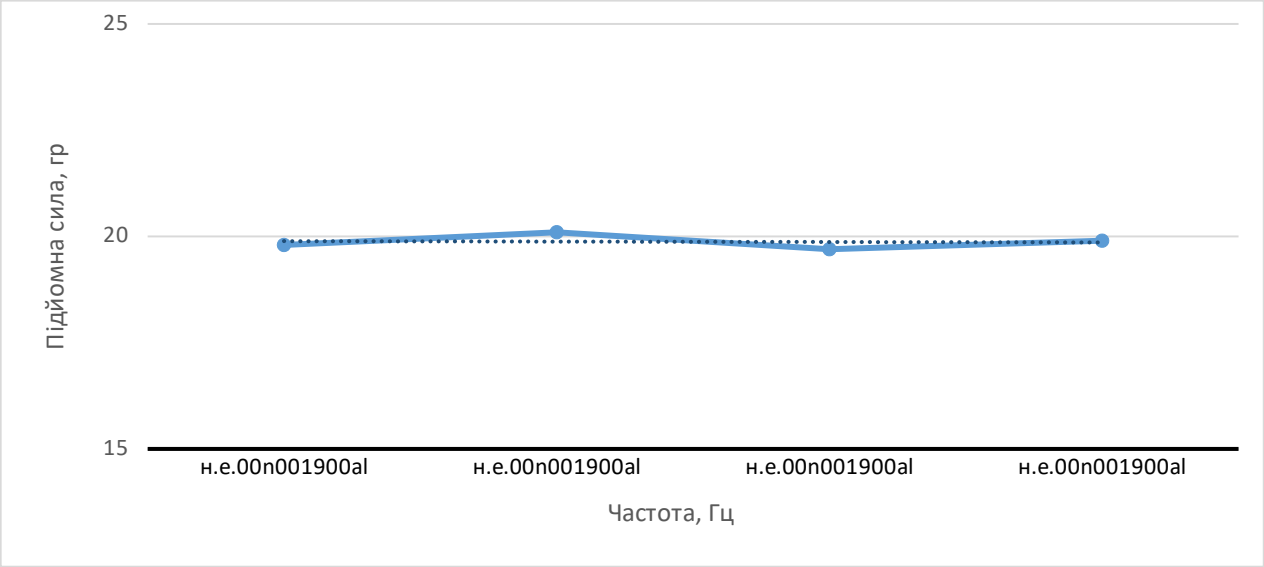

Рис. 6.6(б) – Графік залежності підйомної сили від частоти

Детальний аналіз результатів симуляції наведений у Додатку В (АЛ9117.17.00.00.04 : Результати аеродинамічної симуляції)

Висновок

В ході дипломного проекту було розглянуто три аналоги, визначені їх характеристики. Обрано прототип для подальшого моделювання та розрахунків. Згідно з цим прототипом було обрано конструкцію орнітоптера.

Мною було зроблено застосунки для виконання триангуляції Делоне та її візуалізації, з допомогою яких можливо швидко підібрати потрібні параметри літального апарату.

Далі була змодельована модель на основі аналогу та природної особи. До цієї моделі була зроблена анімація можливих рухів.

Для одержання якісних даних була проведена симуляція аеродинамічних потоків.

Список використаних джерел

1. Бондаренко О.М., Ж. В. Рябіченко. Моделювання кінцево-елементної сітки складних аеродинамічних поверхонь / Механіка гіроскопічних систем. - №

43 – 2022. - С.35-44.

2. Festo BionicSwift [Електронна стаття] – Режим доступу до ресурсу: https://www.festo.com/PDF_Flip/corp/Festo_BionicSwift/en/
3. JSK-koubou Orthopter [Плей лист з відео] – Режим доступу до ресурсу: https://www.youtube.com/playlist?list=PLHFFsWOXN0f9ecPwH_kBTCHSo6a8IoqNl
4. CGAL 5.5.2 Documentation – 3D Triangulations [Програмна документація] – Режим доступу до ресурсу: https://doc.cgal.org/latest/Triangulation_3/classCGAL_1_1Triangulation__3.html
5. OpenGL Utility Toolkit(GLUT) Documentation [Програмна документація] – Режим доступу до ресурсу: https://www.opengl.org/resources/libraries/glut/spec3/spec3.html
6. Snowbird [Електронна стаття] – Режим доступу до ресурсу: http://www.aerovelo.com/ornithopter-summary
7. AnOrnithopterWingDesign [Електронна стаття] – Режим доступу до ресурсу: http://www.ornithopter.net/Publications/AnOrnithopterWingDesign.pdf
8. Designing a Biomimetic Ornithopter Capable of Sustained and Controlled Flight [Електронна стаття] – Режим доступу до ресурсу: https://sci-hub.se/https://link.springer.com/article/10.1016/S1672-6529(08)60005-0
9. ELASTODYNAIMC SYSTEMS: ORNITHOPTERS AND POWER HARVESTING [Електронна стаття] – Режим доступу до ресурсу: інтернет.