

**НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ ІМЕНІ ІГОРЯ
СІКОРСЬКОГО»**

Навчально-науковий інститут атомної та теплової енергетики
Кафедра інженерії програмного забезпечення в енергетиці

ДО ЗАХИСТУ ДОПУЩЕНО

В.о. завідувача кафедри
Олександр КОВАЛЬ

«__» _____ 2026 р.

Дипломна робота

**на здобуття ступеня бакалавра
за освітньо-професійною програмою «Інженерія програмного забезпечення
інтелектуальних кібер-фізичних систем в енергетиці»
спеціальності 121 Інженерія програмного забезпечення
на тему: «Розробка програмних засобів генерації 3D-моделей обличчя»**

Виконав:

студент IV курсу, групи ТВ-21

Цвігун Богдан Володимирович

(прізвище, ім'я, по батькові)

(підпис)

Керівник:

Доцент ІПЗЕ Залевська Ольга Валеріївна

(посада, науковий ступінь, вчене звання, прізвище та ініціали)

(підпис)

Консультант:

(посада, науковий ступінь, вчене звання, прізвище та ініціали)

(підпис)

Рецензент:

Доцент ІПІ Фіногенов Олексій Дмитрович

(посада, науковий ступінь, вчене звання, прізвище та ініціали)

(підпис)

Засвідчую, що у цій дипломній роботі
немає запозичень із праць інших авторів
без відповідних посилань.

Студент _____
(підпис)

Київ – 2026

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

Навчально-науковий інститут атомної та теплової енергетики
Кафедра інженерії програмного забезпечення в енергетиці
Рівень вищої освіти перший (бакалаврський)
Спеціальність 121 Інженерія програмного забезпечення
Освітньо-професійна програма «Інженерія програмного забезпечення
інтелектуальних кібер-фізичних систем в енергетиці»

ЗАТВЕРДЖУЮ

В.о. завідувача кафедри
Олександр КОВАЛЬ

«__» _____ 2026р.

ЗАВДАННЯ

на дипломну роботу студенту

Цвігуну Богдану Володимировичу
(прізвище, ім'я, по батькові)

1. Тема роботи: Розробка програмних засобів генерації 3D-моделей обличчя

керівник роботи Доцент кафедри ІПЗЕ Залевська Ольга Валеріївна
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)

затверджені наказом по університету від «__» _____ 2026 р. № _____

2. Строк подання студентом роботи «10» червня 2026 р.

3. Вихідні дані до роботи: Технічне завдання, базова 3D-модель голови людини, набір параметрів для редагування геометрії обличчя, вимоги до програмного засобу, мова Python та бібліотеки для роботи з тривимірною графікою.

4. Зміст (дипломної роботи) пояснювальної записки (перелік завдань, які потрібно розробити): Аналіз предметної області та існуючих засобів створення 3D-моделей обличчя. Постановка задачі розробки програмного засобу. Проектування структури програмної системи. Розробка програмного засобу генерації та редагування 3D-моделей обличчя. Опис інтерфейсу користувача та функціональних можливостей системи. Тестування роботи програмного засобу. Висновки.

5. Перелік ілюстративного матеріалу: _____

6. Дата видачі завдання «31» жовтня 2025 р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів виконання дипломної роботи	Строки виконання етапів роботи	Примітка
1	Отримання завдання	31.10.2025	
2	Дослідження предметної області	01.11.2025 – 31.11.2025	
3	Дослідження існуючих рішень	01.12.2025 – 31.12.2025	
4	Постановка вимог до проектування системи	01.01.2026 – 31.01.2026	
5	Розробка програмного продукту	01.02.2026 – 03.05.2026	
6	Тестування	04.05.2026 – 10.05.2026	
7	Захист програмного продукту	11.05.2026 – 15.05.2026	
8	Оформлення дипломної роботи	18.05.2026 – 31.05.2026	
9	Передзахист	01.06.2026 – 05.06.2026	
10	Захист	15.06.2026 – 19.06.2026	

Студент

(підпис)

Богдан ЦВІГУН

(ім'я, ПРІЗВИЩЕ)

Керівник роботи

(підпис)

Ольга ЗАЛЕВСЬКА

(ім'я, ПРІЗВИЩЕ)

РЕФЕРАТ

Структура та обсяг дипломної роботи. Робота містить 69 сторінок, 2 додатки.

Метою роботи є розробка програмної системи для процедурної генерації та параметричного редагування тривимірної моделі людського обличчя. Розроблена система призначена для автоматизованого створення базової 3D-моделі голови, формування основних елементів обличчя та надання користувачу можливості змінювати геометричні параметри моделі за допомогою графічного інтерфейсу.

У дипломній роботі розглянуто основні поняття тривимірного моделювання, особливості побудови полігональних сіток, принципи роботи з вершинами, гранями та групами вершин. Проведено аналіз існуючих програмних рішень для створення 3D-моделей, зокрема професійних 3D-редакторів, генераторів персонажів та методів параметричного редагування. На основі проведеного аналізу обґрунтовано доцільність використання процедурної генерації для створення базових моделей обличчя.

Практичне значення одержаних результатів полягає у створенні програмного продукту, який може бути використаний для автоматизованої генерації та редагування базових тривимірних моделей людського обличчя. Розроблена система може застосовуватися у навчальних, дослідницьких та прикладних проєктах, пов'язаних із комп'ютерною графікою, 3D-моделюванням, створенням персонажів, анімацією та візуалізацією.

Ключові слова: ТРИВИМІРНЕ МОДЕЛЮВАННЯ, 3D-МОДЕЛЬ ОБЛИЧЧЯ, ПРОЦЕДУРНА ГЕНЕРАЦІЯ, ПАРАМЕТРИЧНЕ РЕДАГУВАННЯ, ПОЛІГОНАЛЬНА СІТКА, PYTHON, NUMPY, TRIMESH, PYQT5, OPENGL.

ABSTRACT

Structure and scope of the thesis. The work contains 69 pages, 2 appendices.

The purpose of the work is to develop a software system for procedural generation and parametric editing of a three-dimensional human face model. The developed system is intended for automated creation of a basic 3D head model, formation of the main facial elements, and providing the user with the ability to modify the geometric parameters of the model through a graphical user interface.

The thesis considers the basic concepts of three-dimensional modeling, the features of polygonal mesh construction, and the principles of working with vertices, faces and vertex groups. Existing software solutions for creating 3D models were analyzed, including professional 3D editors, character generators and parametric editing methods. Based on the analysis, the use of procedural generation for creating basic face models was substantiated.

The software system was implemented using the Python programming language and the NumPy, Trimesh, PyQt5 libraries, as well as OpenGL visualization tools. The system has a modular structure and includes components for basic model generation, geometric data processing, parametric editing and graphical visualization of the result. The user can modify individual facial features, including the shape of the nose, jaw, eyes, ears and other parameters, while the editing result is displayed in real time.

The practical value of the obtained results lies in the creation of a software product that can be used for automated generation and editing of basic three-dimensional human face models. The developed system can be applied in educational, research and applied projects related to computer graphics, 3D modeling, character creation, animation and visualization.

Keywords: THREE-DIMENSIONAL MODELING, 3D FACE MODEL, PROCEDURAL GENERATION, PARAMETRIC EDITING, POLYGONAL MESH, PYTHON, NUMPY, TRIMESH, PYQT5, OPENGL.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СКОРОЧЕНЬ І ТЕРМІНІВ.....	8
ВСТУП.....	9
1 МЕТА ТА ОСНОВНІ ЗАВДАННЯ ДИПЛОМНОЇ РОБОТИ.....	11
1.1 Мета дипломної роботи.....	11
1.2 Основні завдання роботи.....	11
1.1 Мета дипломної роботи.....	13
Висновки до розділу 1	13
2 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ.....	16
2.1 Загальна характеристика тривимірного моделювання.....	16
2.2 Особливості створення 3D-моделі людського обличчя.....	17
2.3 Використання процедурної генерації у 3D моделюванні.....	18
Висновки до розділу 2	19
3 АНАЛІЗ ІСНУЮЧИХ СИСТЕМ ТА МЕТОДІВ.....	21
3.1 Професійні 3D-редактори	21
3.2 Генератори персонажів та обличчя	21
3.3 Методи параметричного редагування 3D-моделей	22
3.4 Аналіз недоліків сучасних рішень та постановка задачі	22
Висновки до розділу 3	22
4 ЗАСОБИ РОЗРОБКИ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ	25
4.1 Апаратні засоби розробки	25
4.2 Програмні засоби розробки.....	26
Висновки до розділу 4	27
5 ОПИС ПРОГРАМНОЇ РЕАЛІЗАЦІЇ	29
5.1 Загальна структура програмної системи	29
5.2 Модуль процедурної генерації базової моделі голови.....	30
5.3 Модуль обробки геометричних даних	30
5.4.Візуалізація 3D моделі у графічному інтерфейсі	31
Висновки до розділу 5	32
ВИСНОВКИ.....	35

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ	38
ДОДАТОК А Лістинг розробленої системи.....	39
ДОДАТОК Б Презентація.....	65

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СКОРОЧЕНЬ І ТЕРМІНІВ

3D	Тривимірний простір або тривимірний об'єкт, що має ширину, висоту та глибину
3D-модель	Цифрове представлення об'єкта у тривимірному просторі.
3D-моделювання	Процес створення, редагування та візуалізації тривимірних об'єктів.
Полігональна сітка	Структура 3D-моделі, що складається з вершин, ребер і граней.
Процедурна генерація	Створення об'єктів за допомогою програмних алгоритмів і заданих параметрів
Параметричне редагування	Зміна форми моделі за допомогою числових параметрів або елементів керування

ВСТУП

У сучасних умовах стрімкого розвитку цифрових технологій, комп'ютерної графіки та тривимірного моделювання особливої актуальності набувають питання автоматизації процесів створення 3D-моделей людини. Тривимірні моделі обличчя широко використовуються у сфері комп'ютерних ігор, анімації, кіноіндустрії, віртуальної та доповненої реальності, медичних симуляціях, системах ідентифікації, навчальних та дослідницьких проєктах. Зростання попиту на цифрові тривимірні об'єкти зумовлює необхідність створення зручних програмних засобів, які дають змогу швидко формувати базові моделі обличчя та виконувати їх подальше редагування.

Сучасні професійні програмні комплекси для 3D-моделювання забезпечують широкі можливості для створення високоякісних моделей, однак їх ефективне використання часто потребує від користувача спеціальних знань у сфері комп'ютерної графіки, анатомії обличчя, топології тривимірної сітки та роботи з графічними редакторами. Для створення навіть базової моделі користувач має виконувати значну кількість ручних операцій, що ускладнює процес розробки та збільшує витрати часу. У зв'язку з цим виникає потреба у програмних рішеннях, які спрощують процес генерації тривимірних моделей обличчя та надають можливість змінювати їх основні параметри за допомогою зрозумілого інтерфейсу.

Актуальність теми дипломної роботи полягає у необхідності розробки програмного засобу, який дозволяє автоматизувати створення базової 3D-моделі обличчя, забезпечити її параметричне редагування та візуальний перегляд результату в режимі реального часу. Такий підхід дає змогу зменшити складність процесу моделювання, зробити роботу з тривимірною геометрією більш доступною для користувача та створити основу для подальшого використання отриманих моделей у навчальних, дослідницьких або прикладних задачах комп'ютерної графіки.

Об'єктом дослідження є процес створення та редагування тривимірних моделей обличчя засобами комп'ютерної графіки.

Предметом дослідження є методи та програмні засоби процедурної генерації, параметричного редагування і візуалізації 3D-моделі обличчя на основі тривимірної сітки.

Метою дипломної роботи є розробка програмної системи для генерації та параметричного редагування 3D-моделі обличчя, яка дозволяє створювати базову модель голови, змінювати її основні риси та переглядати результат у режимі реального часу.

Для досягнення поставленої мети необхідно вирішити такі завдання: проаналізувати особливості побудови тривимірних моделей обличчя; дослідити принципи роботи з полігональними сітками, вершинами та гранями; розробити структуру програмної системи; реалізувати механізми зміни параметрів моделі; створити графічний інтерфейс користувача; забезпечити візуалізацію результатів редагування у режимі реального часу; перевірити працездатність розробленого програмного засобу.

Реалізація програмної системи здійснюється з використанням мови програмування Python, бібліотек для роботи з тривимірною геометрією, масивами даних та графічним інтерфейсом користувача. Для збереження й обробки геометричних даних використовуються масиви вершин і граней тривимірної сітки, а для візуалізації моделі застосовується OpenGL-середовище. Розроблена система забезпечує генерацію базової моделі обличчя, редагування її параметрів, візуальний перегляд змін у режимі реального часу та підготовку результату для подальшого використання у сфері 3D-графіки.

1 МЕТА ТА ОСНОВНІ ЗАВДАННЯ ДИПЛОМНОЇ РОБОТИ

В наступому пункті буде детально описано мету даної роботи, її основні завдання та практичне значення розробленої система у майбутньому.

1.1 Мета дипломної роботи 1

Метою даної дипломної роботи є створення програмної системи для процедурної генерації та параметричного редагування тривимірної моделі людського обличчя. Основне призначення системи полягає у забезпеченні автоматизованого створення базової 3D-моделі голови, формуванні основних елементів обличчя, а також у наданні користувачу можливості змінювати геометричні параметри моделі за допомогою зручного графічного інтерфейсу.

У сучасних сферах комп'ютерної графіки, ігрової розробки, анімації, віртуальної реальності та навчального моделювання тривимірні моделі обличчя використовуються досить широко. Проте створення якісної 3D-моделі зазвичай потребує значних часових витрат, спеціальних знань з анатомії, топології сітки та навичок роботи з професійними редакторами тривимірної графіки. Саме тому використання процедурної генерації дозволяє спростити процес створення базової моделі, автоматизувати побудову основних частин обличчя та забезпечити можливість швидкого редагування форми моделі за заданими параметрами.

1.2 Основні завдання роботи

Для досягнення поставленої мети у межах дипломної роботи було визначено ряд основних завдань. Першочергово проведено аналіз предметної області та дослідження особливостей створення тривимірних моделей людського обличчя. Особлива увага приділялася принципам побудови 3D-сітки, структурі вершин і

граней, а також способам формування основних елементів обличчя, таких як очі, ніс, рот, щелепа та вуха.

Наступним етапом стало дослідження існуючих програмних засобів для створення та редагування тривимірних моделей. Було розглянуто сучасні 3D-редактори, генератори персонажів та інструменти процедурного моделювання, визначено їх переваги та недоліки. У результаті аналізу встановлено, що більшість професійних рішень потребує значного досвіду роботи з 3D-графікою, а простіші інструменти часто мають обмежені можливості для гнучкого редагування геометрії моделі.

Під час виконання роботи також було виконано проєктування архітектури програмної системи. Розроблена система має модульну структуру та складається з компонентів генерації базової моделі, обробки геометричних даних, параметричного редагування, візуалізації тривимірного об'єкта та графічного інтерфейсу користувача. Такий підхід дозволяє розділити логіку створення моделі, її деформації та відображення, що спрощує подальше розширення функціональності системи.

Одним із ключових завдань стало створення алгоритму процедурної генерації базової 3D-моделі голови. У системі реалізовано механізми формування тривимірної сітки на основі масивів вершин і граней, а також створення окремих зон моделі, які відповідають за основні частини обличчя. Для подальшого редагування було передбачено використання груп вершин, що дозволяє змінювати форму окремих елементів моделі без необхідності повного ручного редагування всієї сітки.

Окрему увагу приділено реалізації модуля параметричного редагування 3D-моделі. У межах роботи було створено систему повзунків, за допомогою яких користувач може змінювати окремі риси обличчя, зокрема форму носа, ширину щелепи, положення очей, розмір вух та інші геометричні параметри. Також було враховано можливість симетричного редагування моделі, що є важливим для збереження природної форми людського обличчя під час зміни його параметрів.

Завершальним етапом роботи стало тестування функціонування програмної системи. Під час тестування було перевірено коректність генерації базової моделі, роботу механізмів деформації, стабільність графічного інтерфейсу, а також правильність відображення змін у режимі реального часу. Отримані результати підтвердили ефективність використання процедурного підходу для створення базових 3D-моделей обличчя та можливість подальшого розвитку системи у напрямі більш детального моделювання персонажів.

1.3 Практичне значення роботи

Практичне значення дипломної роботи полягає у створенні програмного продукту, який може бути використаний для автоматизованої генерації та параметричного редагування тривимірних моделей людського обличчя. Розроблена система дозволяє спростити процес створення базової 3D-моделі, забезпечити можливість зміни окремих рис обличчя та переглядати результат у режимі реального часу. Крім того, система може бути використана у навчальних, дослідницьких та прикладних проєктах, пов'язаних із комп'ютерною графікою, 3D-моделюванням, розробкою персонажів, анімацією та візуалізацією.

Використання процедурної генерації дозволяє зменшити час, необхідний для створення базової моделі обличчя, знизити залежність від ручного моделювання та зробити процес редагування більш зручним для користувача. Результати роботи можуть бути використані для подальших досліджень у сфері процедурного моделювання, генерації персонажів, комп'ютерної графіки та розробки програмних інструментів для роботи з тривимірною геометрією.

Висновки до розділу 1

У першому розділі було визначено мету дипломної роботи, яка полягає у створенні програмної системи для процедурної генерації та параметричного редагування тривимірної моделі людського обличчя. Обґрунтовано актуальність обраної теми, оскільки створення 3D-моделей обличчя є важливим завданням у сферах комп'ютерної графіки, ігрової розробки, анімації, віртуальної реальності, навчального моделювання та візуалізації.

Було встановлено, що використання професійних програмних засобів для 3D-моделювання часто потребує значних часових витрат, спеціальних знань з анатомії, топології сітки та навичок роботи з редакторами тривимірної графіки. У зв'язку з цим доцільним є застосування процедурного підходу, який дозволяє автоматизувати процес створення базової моделі обличчя та забезпечити можливість її подальшого редагування за допомогою параметричних інструментів.

У межах розділу було визначено основні завдання дипломної роботи, зокрема аналіз предметної області, дослідження існуючих програмних засобів для створення 3D-моделей, проєктування архітектури програмної системи, розробку алгоритмів процедурної генерації, реалізацію модуля параметричного редагування, створення графічного інтерфейсу користувача та тестування роботи системи.

Також було окреслено загальну структуру майбутньої програмної системи, яка має включати модулі генерації базової моделі, обробки геометричних даних, параметричного редагування, візуалізації тривимірного об'єкта та взаємодії з користувачем. Такий підхід дозволяє забезпечити зручність використання системи, гнучкість її подальшого розвитку та можливість розширення функціональності.

Отже, у результаті проведеного аналізу було підтверджено доцільність розробки програмного засобу для процедурної генерації та параметричного редагування 3D-моделі обличчя. Запропонований підхід дозволяє спростити процес створення базової тривимірної моделі, зменшити залежність від ручного

модельювання та забезпечити користувачу можливість швидкого редагування основних рис обличчя у режимі реального часу.

2 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

У наступному розділі проведено детальний аналіз предметної області, описані використані технології та основні принципи роботи програмної системи для процедурної генерації та параметричного редагування 3D-моделі обличчя. Також розглянуто особливості створення тривимірної геометрії, принципи побудови сітки моделі, методи деформації окремих частин обличчя, структуру програмної системи та основні компоненти, що забезпечують візуалізацію й редагування моделі у режимі реального часу.

2.1 Загальна характеристика тривимірного моделювання

Тривимірне моделювання є сучасним напрямом комп'ютерної графіки, який пов'язаний зі створенням цифрових об'єктів у тривимірному просторі. Такі об'єкти мають геометричну форму, що описується за допомогою вершин, ребер та граней, а також можуть містити додаткові параметри, зокрема матеріали, текстури, кольори, освітлення та анімацію. Тривимірні моделі широко використовуються у комп'ютерних іграх, кіноіндустрії, архітектурній візуалізації, віртуальній реальності, медицині, освіті та інженерному проєктуванні.

Основною особливістю тривимірного моделювання є поєднання математичного опису форми об'єкта з його візуальним представленням. Будь-яка 3D-модель складається з набору геометричних даних, які визначають її структуру у просторі. Найпоширенішим способом представлення таких моделей є полігональна сітка, що складається з вершин і граней. Саме за допомогою цієї сітки формується загальна форма об'єкта, а подальше редагування дозволяє змінювати його пропорції, деталізацію та зовнішній вигляд.

Простими словами, тривимірне моделювання — це процес створення цифрового об'єкта, який можна обертати, масштабувати, редагувати та переглядати з різних сторін. У випадку моделювання обличчя це означає створення

віртуальної голови людини, у якій можна змінювати форму носа, очей, рота, щелепи, вух та інших частин. Такий підхід дозволяє не малювати кожен модель вручну з нуля, а створювати її за допомогою програмних алгоритмів і налаштовуваних параметрів.

2.2 Особливості створення 3D-моделі людського обличчя

У межах даної роботи досліджується процес створення тривимірної моделі людського обличчя за допомогою програмних алгоритмів та засобів процедурної генерації. 3D-модель обличчя є складним об'єктом комп'ютерної графіки, оскільки вона повинна мати впізнавану анатомічну форму, правильне розташування основних елементів та можливість подальшого редагування. До таких елементів належать очі, ніс, рот, щелепа, вуха, лоб, щоки та загальна форма голови. Основа створення такої моделі полягає у побудові тривимірної полігональної сітки, яка складається з вершин, ребер та граней.

Важливою особливістю 3D-моделі людського обличчя як об'єкта даного дослідження є наявність великої кількості параметрів, що визначають її зовнішній вигляд. До них належать ширина та висота голови, форма носа, розмір і положення очей, ширина щелепи, розмір вух, форма рота та інші геометричні характеристики. Контроль і зміна цих параметрів дозволяє створювати різні варіанти обличчя, змінювати пропорції моделі та швидко оцінювати результат без необхідності ручного редагування кожної вершини сітки. Саме тому у роботі особлива увага приділяється параметричному підходу, який дає змогу керувати формою моделі за допомогою програмних налаштувань.

2.3 Використання процедурної генерації у 3D-моделюванні розділу

Під час виконання роботи було проведено дослідження сучасних підходів до створення тривимірних моделей та методів автоматизованої побудови геометрії об'єктів. Особливу увагу приділено використанню процедурної генерації, яка передбачає створення 3D-моделі за допомогою програмних алгоритмів і заданих параметрів. Це дозволяє зменшити час створення базової моделі, підвищити гнучкість редагування та швидко формувати різні варіанти обличчя.

Для реалізації процедурного підходу у роботі використовується мова програмування Python та бібліотеки для роботи з тривимірною геометрією і графічною візуалізацією. На цьому рівні виконуються такі операції:

1. створення базової геометрії голови у вигляді тривимірної сітки;
2. формування основних елементів обличчя, таких як ніс, очі, рот та вуха;
3. визначення груп вершин для редагування окремих зон моделі;
4. відображення зміненої моделі у графічному інтерфейсі.

Після виконання цих процесів користувач отримує модель, яку можна змінювати за допомогою налаштувань. У системі реалізовано модулі генерації, редагування та візуалізації 3D-моделі, а результати роботи підтверджують ефективність використання процедурної генерації для створення базових моделей людського обличчя.

Висновки до розділу 2

У другому розділі було розглянуто основні поняття тривимірного моделювання та його роль у сучасній комп'ютерній графіці. Встановлено, що 3D-моделі є важливим інструментом для створення цифрових об'єктів, які використовуються у комп'ютерних іграх, анімації, віртуальній реальності, медицині, освіті, інженерному проектуванні та інших сферах. Основою таких моделей є геометричне представлення об'єкта у тривимірному просторі за допомогою вершин, ребер і граней.

Було визначено, що найбільш поширеним способом представлення тривимірних об'єктів є полігональна сітка. Саме вона формує просторову структуру моделі та дозволяє змінювати її форму, пропорції й деталізацію. У випадку моделювання людського обличчя полігональна сітка має особливе значення, оскільки повинна забезпечувати правильне розташування основних елементів обличчя, зокрема очей, носа, рота, щелепи, вух, лоба та щік.

У межах розділу було проаналізовано особливості тривимірної моделі людського обличчя як об'єкта дослідження. Встановлено, що така модель є складним об'єктом комп'ютерної графіки, оскільки її зовнішній вигляд залежить від великої кількості геометричних параметрів. До них належать форма голови, розмір і положення очей, форма носа, ширина щелепи, розмір вух, форма рота та інші характеристики.

Також було обґрунтовано доцільність використання параметричного підходу до редагування 3D-моделі обличчя. Зміна окремих параметрів дозволяє швидко створювати різні варіанти моделі без необхідності ручного редагування кожної вершини сітки. Це робить процес моделювання більш зручним, гнучким та доступним для користувача.

Окрему увагу було приділено процедурній генерації як методу автоматизованого створення тривимірної геометрії. Встановлено, що використання програмних алгоритмів і заданих параметрів дозволяє зменшити час створення

базової моделі, спростити процес формування основних елементів обличчя та забезпечити можливість подальшого редагування отриманого результату.

У результаті розгляду теоретичних основ було визначено основні операції, необхідні для реалізації програмної системи: створення базової геометрії голови, формування основних елементів обличчя, визначення груп вершин для редагування окремих зон моделі та відображення результату у графічному інтерфейсі. Отже, проведений аналіз підтвердив доцільність використання процедурної генерації та параметричного редагування для створення базових тривимірних моделей людського обличчя.

3 АНАЛІЗ ІСНУЮЧИХ СИСТЕМ ТА МЕТОДІВ

У процесі розробки програмної системи для генерації та редагування 3D-моделі обличчя було проведено аналіз існуючих програмних рішень та сучасних методів тривимірного моделювання. Основну увагу приділено професійним 3D-редакторам, генераторам персонажів та методам параметричного редагування, які використовуються для створення моделей людини, зміни геометрії обличчя та візуалізації результатів у тривимірному просторі.

3.1 Професійні 3D-редактори

Професійні 3D-редактори є одним із найбільш поширених підходів до створення тривимірних моделей. Вони забезпечують побудову геометрії об'єктів, редагування полігональної сітки, налаштування матеріалів, текстур, освітлення та підготовку моделей до подальшого використання. Перевагами таких редакторів є широкий набір інструментів, висока якість результату та можливість детального ручного моделювання. Однак такі системи мають ряд суттєвих недоліків, серед яких складність освоєння, потреба у спеціальних навичках роботи з 3D-графікою та значні часові витрати на створення моделі обличчя з нуля.

3.2 Генератори персонажів та обличчя

Окрему категорію сучасних рішень становлять генератори персонажів та обличчя, які дозволяють створювати тривимірні моделі людини за допомогою готових шаблонів і набору параметрів. Такі системи дають змогу змінювати форму голови, риси обличчя, зачіску, пропорції тіла та інші зовнішні характеристики моделі. Основними перевагами таких рішень є зручність використання, швидкість створення базової моделі та можливість отримання візуально якісного результату без повного ручного моделювання. Разом з тим, більшість генераторів має

обмежену гнучкість, залежить від готових пресетів і не завжди дозволяє детально редагувати геометрію моделі на рівні окремих вершин або зон обличчя.

3.3 Методи параметричного редагування 3D-моделей

Також було проаналізовано методи параметричного редагування 3D-моделей, які використовуються для зміни форми об'єкта за допомогою заданих числових параметрів. Такі методи дозволяють керувати окремими частинами моделі, наприклад змінювати ширину носа, форму щелепи, розмір очей або положення вух. Основними перевагами параметричного підходу є зручність редагування, швидке отримання різних варіантів моделі та можливість автоматизувати процес зміни геометрії. Основними недоліками таких методів є потреба у правильному визначенні зон редагування, налаштуванні груп вершин та обмеження у деталізації порівняно з ручним моделюванням.

3.4 Аналіз недоліків існуючих рішень та постановка задачі

У результаті проведеного аналізу встановлено, що існуючі системи для створення 3D-моделей обличчя мають низку спільних недоліків. До них належать складність освоєння професійних редакторів, значні часові витрати на ручне моделювання, обмежена гнучкість готових генераторів персонажів, а також залежність від попередньо створених шаблонів і пресетів. У випадку необхідності швидко створити базову модель обличчя та змінити її основні параметри такі рішення не завжди є достатньо зручними або ефективними.

Висновки до розділу 3

У третьому розділі було проведено аналіз існуючих програмних рішень та сучасних методів тривимірного моделювання, які можуть застосовуватися для створення та редагування 3D-моделей людського обличчя. Розглянуто основні

категорії таких засобів, зокрема професійні 3D-редактори, генератори персонажів та методи параметричного редагування геометрії моделі.

Було встановлено, що професійні 3D-редактори забезпечують широкий набір інструментів для створення якісних тривимірних моделей, редагування полігональної сітки, налаштування матеріалів, текстур та візуалізації. Водночас їх використання потребує спеціальних знань, практичного досвіду роботи з 3D-графікою та значних часових витрат. Це ускладнює процес створення базової моделі обличчя для користувачів, які не мають достатньої підготовки у сфері тривимірного моделювання.

Також було розглянуто генератори персонажів та обличчя, які дозволяють швидко створювати моделі людини на основі готових шаблонів, пресетів і набору параметрів. Перевагами таких систем є зручність використання, висока швидкість створення моделі та можливість отримання візуально якісного результату без повного ручного моделювання. Однак такі рішення часто мають обмежену гнучкість, залежать від попередньо підготовлених шаблонів і не завжди дозволяють детально змінювати геометрію моделі на рівні окремих зон обличчя.

У межах розділу було визначено, що параметричне редагування є доцільним підходом для зміни форми 3D-моделі обличчя, оскільки дозволяє керувати окремими геометричними характеристиками за допомогою числових параметрів або елементів графічного інтерфейсу. Такий підхід спрощує процес редагування, дає змогу швидко створювати різні варіанти моделі та зменшує потребу у ручному коригуванні кожної вершини полігональної сітки.

Разом з тим було встановлено, що параметричні методи потребують правильного визначення зон редагування, груп вершин та логіки деформації моделі. Від якості цих налаштувань залежить коректність зміни окремих елементів обличчя, збереження пропорцій моделі та природність отриманого результату.

Отже, результати аналізу показали, що існуючі програмні рішення не завжди є достатньо зручними для швидкого створення базової 3D-моделі обличчя з можливістю простого параметричного редагування. Професійні редактори є

складними для освоєння, а готові генератори персонажів часто обмежують користувача набором шаблонів і пресетів. Саме тому доцільною є розробка власної програмної системи, яка поєднує процедурну генерацію базової моделі, параметричне редагування її основних елементів та візуалізацію результату в режимі реального часу.

4 ЗАСОБИ РОЗРОБКИ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

Для реалізації програмної системи процедурної генерації та параметричного редагування 3D-моделі обличчя використано комплекс програмних засобів, які забезпечують створення геометрії, обробку тривимірних даних, візуалізацію моделі та взаємодію користувача з системою.

Основною вимогою до засобів розробки була можливість роботи з тривимірною сіткою, редагування координат вершин, відображення моделі у графічному інтерфейсі та швидке оновлення результату після зміни параметрів. Також важливим було забезпечення модульної структури програми, щоб окремо реалізувати генерацію моделі, її деформацію та візуальне представлення.

4.1 Апаратні засоби розробки

Для розробки та тестування програмної системи використано персональний комп'ютер, який виконує роль основної апаратної платформи. Даний пристрій забезпечує запуск середовища розробки, виконання програмного коду, обробку тривимірної геометрії та відображення 3D-моделі у графічному інтерфейсі користувача. Використання персонального комп'ютера є доцільним, оскільки робота з тривимірними об'єктами потребує достатньої продуктивності процесора, оперативної пам'яті та графічної підсистеми.

Завдяки використанню персонального комп'ютера було забезпечено можливість швидкого тестування алгоритмів процедурної генерації, редагування координат вершин моделі та перевірки роботи інтерфейсу в режимі реального часу.

Для роботи системи використовувалася операційна система Windows. Дана система забезпечує стабільну роботу середовища розробки, підтримку необхідних бібліотек Python, засобів візуалізації OpenGL та запуск графічного інтерфейсу

користувача. Використання Windows дозволило зручно організувати процес розробки, тестування та налагодження програмної системи.

4.2 Програмні засоби розробки

Основною мовою програмування для реалізації програмної частини системи було обрано Python. Даний вибір обумовлений простотою синтаксису, зручністю роботи з математичними обчисленнями та наявністю великої кількості бібліотек для обробки тривимірної геометрії, масивів даних і створення графічного інтерфейсу. Python використовується для реалізації модулів генерації базової моделі голови, редагування параметрів обличчя, обробки координат вершин та взаємодії з інтерфейсом користувача.

Для роботи з числовими даними та геометрією моделі використано бібліотеку NumPy. Вона дозволяє ефективно обробляти масиви координат вершин, виконувати математичні операції над ними та змінювати форму окремих частин 3D-моделі. Це є важливим для реалізації процедурної генерації та параметричного редагування обличчя.

Для роботи з тривимірними сітками використано бібліотеку Trimesh. Вона забезпечує створення, збереження та обробку 3D-моделей, які складаються з вершин і граней. За допомогою Trimesh у системі реалізовано роботу з геометрією моделі та підготовку даних для подальшого відображення у графічному середовищі.

Для створення графічного інтерфейсу користувача використано PyQt5. Дана бібліотека дозволяє створювати вікна програми, кнопки, повзунки, перемикачі та інші елементи керування. За допомогою PyQt5 користувач може змінювати параметри моделі обличчя, скидати налаштування та взаємодіяти з програмною системою у зручному форматі.

Для візуалізації тривимірної моделі використано OpenGL-засоби, зокрема компонент GLWidget з бібліотеки pyqtgraph. Це дозволяє відображати 3D-

модель у вікні програми, обертати її, змінювати масштаб та переглядати результат редагування у режимі реального часу.

У процесі розробки програмного забезпечення також використовувалась система контролю версій Git. Вона дозволяє відстежувати зміни у програмному коді, зберігати різні етапи розробки та спрощує роботу над структурою проєкту

У межах виконання даного етапу роботи було проведено дослідження засобів процедурної генерації 3D-моделей, реалізовано механізми створення базової моделі голови, редагування її параметрів та візуалізації результату. Особливу увагу приділено використанню Python, NumPy, Trimesh, PyQt5 та OpenGL як основних інструментів для створення програмної системи генерації 3D-моделі обличчя.

Висновки до розділу 4

У четвертому розділі було розглянуто програмні та апаратні засоби, використані для реалізації системи процедурної генерації та параметричного редагування 3D-моделі людського обличчя. Було визначено, що для розробки такої системи необхідні інструменти, які забезпечують роботу з тривимірною геометрією, обробку масивів вершин і граней, візуалізацію моделі у графічному інтерфейсі та швидке оновлення результату після зміни параметрів.

Як основну апаратну платформу було використано персональний комп'ютер, який забезпечує запуск середовища розробки, виконання програмного коду, обробку тривимірної сітки та відображення моделі у режимі реального часу. Використання операційної системи Windows дозволило організувати стабільний процес розробки, тестування та налагодження програмної системи, а також забезпечити підтримку необхідних бібліотек і засобів візуалізації.

Основною мовою програмування для реалізації системи було обрано Python. Його використання є доцільним завдяки простому синтаксису, зручності роботи з математичними обчисленнями та наявності бібліотек для обробки числових даних, тривимірної геометрії й створення графічного інтерфейсу користувача. Саме

Python став основою для реалізації модулів генерації базової моделі голови, параметричного редагування, обробки координат вершин і взаємодії користувача із системою.

Для роботи з числовими даними та масивами координат було використано бібліотеку NumPy, яка забезпечує ефективне виконання математичних операцій над вершинами тривимірної сітки. Для створення, збереження та обробки полігональних моделей використано бібліотеку Trimesh, що дозволило організувати роботу з геометрією моделі на рівні вершин і граней.

Графічний інтерфейс користувача було реалізовано за допомогою бібліотеки PyQt5. Вона дала змогу створити вікно програми, елементи керування, повзунки та кнопки для зміни параметрів моделі. Для візуалізації тривимірного об'єкта використано OpenGL-засоби, зокрема GLWidget з бібліотеки pyqtgraph, що забезпечило можливість перегляду моделі, її обертання, масштабування та відображення змін у режимі реального часу.

Також у процесі розробки було використано систему контролю версій Git, яка дозволила відстежувати зміни у програмному коді, зберігати проміжні етапи реалізації та впорядкувати структуру проєкту. Це сприяло більш зручній організації роботи над програмною системою та її подальшому розвитку.

Отже, обраний комплекс програмних засобів є доцільним для реалізації системи генерації та редагування 3D-моделі обличчя. Використання Python, NumPy, Trimesh, PyQt5, OpenGL та Git дозволило забезпечити створення базової геометрії, параметричне редагування моделі, візуалізацію результату та зручну взаємодію користувача з програмною системою.

5 ОПИС ПРОГРАМНОЇ РЕАЛІЗАЦІЇ

В наступному розділі коротко описано структуру програмного продукту для процедурної генерації 3D-моделі обличчя. Детально описано основні модулі системи, які відповідають за створення базової геометрії голови, обробку тривимірної сітки, параметричне редагування окремих частин обличчя та візуалізацію моделі у графічному інтерфейсі користувача.

5.1 Загальна структура програмної системи

Програмна реалізація системи процедурної генерації та редагування 3D-моделі обличчя побудована за модульним принципом. Такий підхід дозволяє розділити функціональність програмного забезпечення на окремі компоненти, спростити процес розробки та забезпечити можливість подальшого розширення системи.

Основною метою програмної реалізації є забезпечення створення базової тривимірної моделі голови, її обробки, параметричного редагування та відображення у графічному інтерфейсі користувача. Система функціонує на персональному комп'ютері та забезпечує взаємодію між модулями генерації геометрії, обробки даних моделі та візуалізації результату.

До складу програмної системи входять такі основні модулі:

1. `generate_procedural_head.py` — генерація базової 3D-моделі голови;
2. `model_profile.py` — збереження та обробка профілю моделі;
3. `app.py` — головний модуль графічного інтерфейсу та редагування моделі.

Використання модульної структури дозволяє реалізувати окремі етапи роботи системи незалежно один від одного та забезпечує гнучкість подальшого розвитку програмного продукту.

5.2 Модуль процедурної генерації базової моделі голови

Модуль `generate_procedural_head.py` використовується для створення базової тривимірної моделі голови, яка надалі може бути відредагована користувачем. Генерація початкової моделі є важливою складовою процесу розробки, оскільки дозволяє автоматично сформувати основу обличчя без необхідності ручного моделювання у спеціалізованому 3D-редакторі.

У процесі генерації створюється геометрія голови у вигляді полігональної сітки, що складається з вершин та граней. Також формуються базові елементи обличчя, зокрема ніс, очі, рот, щелепа та вуха. Для побудови форми використовуються математичні залежності, координати точок у тривимірному просторі та параметри, які визначають пропорції моделі.

Сформована модель зберігається у вигляді набору геометричних даних та надалі використовується у графічному інтерфейсі програми. Використання процедурної генерації дозволяє створювати базову модель голови автоматично, змінювати її параметри та розширювати систему для формування більш деталізованих варіантів обличчя.

5.3 Модуль обробки геометричних даних

Центральним компонентом програмної системи є модуль **`model_profile.py`**, який відповідає за збереження, завантаження та обробку геометричних даних 3D-моделі. Даний модуль забезпечує роботу з основними елементами тривимірної сітки, а саме з вершинами, гранями та групами вершин, які використовуються для редагування окремих частин обличчя. Використання окремого модуля для роботи з профілем моделі дозволяє відокремити логіку обробки геометрії від графічного інтерфейсу та зробити структуру програми більш зрозумілою.

Модуль працює з файлами, у яких зберігається базова модель голови та додаткова інформація про її частини. Основна геометрія моделі представлена у

вигляді масивів координат вершин і списку граней, що формують полігональну сітку. Окремо зберігаються групи вершин, які відповідають за певні зони обличчя, наприклад ніс, очі, рот, щелепу або вуха. Завдяки цьому система може змінювати не всю модель одразу, а лише вибрані області.

Основною функцією модуля є підготовка геометричних даних для подальшого редагування та візуалізації. Після завантаження моделі програма отримує доступ до координат вершин, визначає потрібні групи та передає ці дані до головного модуля інтерфейсу. Надалі під час зміни параметрів обличчя система використовує ці групи для застосування деформацій до відповідних частин моделі.

Для редагування геометрії використовується параметричний підхід. Кожен повзунок у графічному інтерфейсі пов'язаний з певною групою вершин і напрямком зміни координат. Наприклад, окремі параметри можуть відповідати за зміну ширини носа, висоти голови, форми щелепи або розміру вух. При зміні значення параметра координати відповідних вершин перераховуються, після чого оновлена модель відображається у вікні програми.

Також модуль дозволяє організувати роботу з різними профілями моделей. Це дає можливість у майбутньому додавати нові базові голови, різні набори параметрів або окремі конфігурації для редагування. Такий підхід робить програмну систему більш гнучкою та придатною для подальшого розвитку.

Поєднання модулів генерації, обробки геометричних даних та графічного інтерфейсу дозволяє забезпечити повний цикл роботи з 3D-моделлю: від створення базової голови до її редагування та перегляду результату у режимі реального часу. Завдяки цьому користувач може швидко змінювати основні риси обличчя без необхідності ручного редагування кожної вершини у професійному 3D-редакторі.

5.4 Візуалізація 3D-моделі у графічному інтерфейсі

Для відображення тривимірної моделі у програмній системі використовується графічний інтерфейс, реалізований за допомогою PyQt5 та

засобів OpenGL-візуалізації. Візуалізація є важливою частиною системи, оскільки дозволяє користувачу одразу бачити результат генерації та редагування моделі обличчя.

Основними перевагами такого підходу є:

1. відображення 3D-моделі у реальному часі;
2. можливість обертання та масштабування моделі;
3. швидке оновлення геометрії після зміни параметрів;
4. зручна взаємодія користувача з елементами редагування.

У системі графічний інтерфейс використовується для перегляду базової моделі голови, зміни параметрів обличчя за допомогою повзунків, а також для скидання моделі до початкового стану. Після зміни значення певного параметра відповідні вершини моделі перераховуються, а оновлена геометрія одразу відображається у вікні програми.

Після завершення редагування користувач може оцінити отриманий результат з різних ракурсів. Це дозволяє швидко перевірити форму носа, очей, щелепи, вух та інших частин обличчя. Завдяки візуалізації у графічному інтерфейсі процес роботи з моделлю стає більш наочним, зручним та зрозумілим для користувача.

Висновки до розділу 5

У п'ятому розділі У даному розділі було розглянуто програмну реалізацію системи процедурної генерації та параметричного редагування 3D-моделі людського обличчя. Встановлено, що використання модульного підходу є доцільним, оскільки дозволяє розділити основні функції системи на окремі компоненти, спростити процес розробки, покращити структуру програмного коду та забезпечити можливість подальшого розширення функціональності.

Було визначено основні модулі програмної системи: модуль генерації базової 3D-моделі голови, модуль збереження та обробки профілю моделі, а також

головний модуль графічного інтерфейсу користувача. Кожен із цих компонентів виконує окрему роль у загальному процесі роботи системи та забезпечує послідовний перехід від створення геометрії до її редагування і візуального відображення.

Модуль `generate_procedural_head.py` забезпечує створення базової тривимірної моделі голови у вигляді полігональної сітки, яка складається з вершин і граней. У межах цього модуля формується початкова геометрія голови та основні елементи обличчя, зокрема ніс, очі, рот, щелепа та вуха. Використання процедурної генерації дозволяє автоматизувати побудову базової моделі та зменшити потребу в ручному моделюванні у спеціалізованих 3D-редакторах.

Модуль `model_profile.py` відповідає за збереження, завантаження та обробку геометричних даних моделі. Завдяки використанню масивів координат вершин, списків граней і груп вершин система отримує можливість працювати не з усією моделлю одночасно, а з окремими зонами обличчя. Це забезпечує основу для параметричного редагування, оскільки кожен параметр може бути пов'язаний з конкретною областю моделі та відповідним напрямком деформації.

У розділі також було розглянуто принцип роботи параметричного редагування. Зміна значень у графічному інтерфейсі призводить до перерахунку координат відповідних вершин, після чого оновлена модель відображається у вікні програми. Такий підхід дозволяє користувачу змінювати форму носа, щелепи, очей, вух та інших частин обличчя без необхідності ручного редагування кожної вершини полігональної сітки.

Окрему увагу було приділено реалізації графічного інтерфейсу користувача та візуалізації 3D-моделі. Використання PyQt5 та OpenGL-засобів дозволило забезпечити відображення моделі у реальному часі, можливість її обертання, масштабування та швидкого оновлення після зміни параметрів. Завдяки цьому процес редагування стає більш наочним і зручним для користувача.

Отже, у результаті програмної реалізації було створено систему, яка забезпечує повний цикл роботи з 3D-моделлю обличчя: генерацію базової

геометрії, збереження та обробку даних моделі, параметричне редагування окремих елементів і візуалізацію результату в режимі реального часу. Запропонована модульна структура робить програмний продукт зрозумілим, гнучким та придатним для подальшого розвитку, зокрема для додавання нових параметрів редагування, профілів моделей і більш деталізованих елементів обличчя.

.

ВИСНОВКИ

У даній дипломній роботі було розроблено програмну систему для процедурної генерації та параметричного редагування тривимірної моделі людського обличчя. Розроблена система дозволяє створювати базову 3D-модель голови, редагувати її основні геометричні параметри та переглядати результат у графічному інтерфейсі користувача в режимі реального часу.

У ході роботи було розглянуто основні поняття тривимірного моделювання, особливості побудови полігональних сіток, принципи роботи з вершинами, ребрами та гранями, а також специфіку створення 3D-моделей людського обличчя. Було встановлено, що модель обличчя є складним об'єктом комп'ютерної графіки, оскільки її зовнішній вигляд залежить від великої кількості параметрів, зокрема форми голови, носа, очей, рота, щелепи, вух та інших елементів.

У межах дипломної роботи було проаналізовано існуючі програмні засоби для створення та редагування тривимірних моделей. Розглянуто професійні 3D-редактори, генератори персонажів та методи параметричного редагування. У результаті аналізу визначено, що професійні редактори забезпечують широкі можливості для створення якісних моделей, однак потребують значних часових витрат і спеціальних знань. Водночас готові генератори персонажів є зручними для швидкого створення моделей, але часто мають обмежену гнучкість і залежать від попередньо підготовлених шаблонів та пресетів.

На основі проведеного аналізу було обґрунтовано доцільність використання процедурної генерації та параметричного редагування для створення базової 3D-моделі обличчя. Процедурний підхід дозволяє автоматизувати процес формування початкової геометрії моделі, а параметричне редагування дає змогу змінювати окремі частини обличчя без необхідності ручного коригування кожної вершини полігональної сітки.

Для реалізації програмної системи було використано мову програмування Python та набір бібліотек для роботи з числовими даними, тривимірною

геометрією, графічним інтерфейсом і візуалізацією. Зокрема, бібліотека NumPy застосовувалася для обробки масивів координат вершин, Trimesh — для роботи з полігональною сіткою, PyQt5 — для створення графічного інтерфейсу користувача, а OpenGL-засоби та GLWidget — для відображення тривимірної моделі у вікні програми. Також у процесі розробки використовувалася система контролю версій Git.

Програмна реалізація системи побудована за модульним принципом. Було реалізовано модуль генерації базової 3D-моделі голови, модуль збереження та обробки профілю моделі, а також головний модуль графічного інтерфейсу користувача. Така структура дозволила розділити логіку створення геометрії, обробки даних, параметричного редагування та візуалізації результату, що підвищує зрозумілість програмного коду та створює основу для подальшого розширення системи.

У розробленій системі реалізовано можливість редагування окремих зон моделі за допомогою груп вершин. Користувач може змінювати певні параметри обличчя за допомогою елементів графічного інтерфейсу, після чого координати відповідних вершин перераховуються, а оновлена модель відображається у режимі реального часу. Це дозволяє швидко оцінювати результат змін і робить процес редагування більш наочним та зручним.

Під час тестування програмної системи було перевірено коректність генерації базової моделі, роботу механізмів параметричного редагування, стабільність графічного інтерфейсу та правильність відображення змін у вікні візуалізації. Отримані результати підтвердили працездатність розробленого програмного засобу та доцільність використання процедурного підходу для створення базових тривимірних моделей людського обличчя.

Практичне значення дипломної роботи полягає у створенні програмного продукту, який може бути використаний для автоматизованої генерації та редагування базових 3D-моделей обличчя. Розроблена система може застосовуватися у навчальних, дослідницьких та прикладних проєктах, пов'язаних

із комп'ютерною графікою, 3D-модельованням, створенням персонажів, анімацією та візуалізацією.

Подальший розвиток програмної системи може бути спрямований на підвищення деталізації моделі, додавання нових параметрів редагування, покращення анатомічної точності обличчя, реалізацію роботи з текстурами та матеріалами, а також експорт готових моделей у формати, придатні для використання в інших 3D-редакторах або ігрових рушіях.

Отже, мету дипломної роботи було досягнуто. У результаті роботи створено програмну систему, яка забезпечує процедурну генерацію базової 3D-моделі обличчя, її параметричне редагування та візуалізацію результату в режимі реального часу.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Foley J. D., van Dam A., Feiner S. K., Hughes J. F. Computer Graphics: Principles and Practice. – 1996.
2. Watt A. 3D Computer Graphics. – 2000.
3. Botsch M., Kobbelt L., Pauly M., Alliez P., Lévy B. Polygon Mesh Processing. – 2010.
4. Akenine-Möller T., Haines E., Hoffman N. Real-Time Rendering. – 2018.
5. Ebert D. S., Musgrave F. K., Peachey D., Perlin K., Worley S. Texturing and Modeling: A Procedural Approach. – 2003.
6. Shreiner D., Sellers G., Kessenich J., Licea-Kane B. OpenGL Programming Guide. – 2013.
7. Lutz M. Learning Python. – 2013.
8. Harris C. R., Millman K. J., van der Walt S. J. Array programming with NumPy. – 2020.
9. Dawson-Haggerty M. Trimesh Documentation. – 2024.
10. PyQt5 Documentation. Riverbank Computing. – 2024.
11. Lutz M. Learning Python. – 2013.
12. PyQtGraph Documentation. – 2024.
13. Blender Foundation. Blender Documentation. – 2024.
14. Parke F. I., Waters K. Computer Facial Animation. – 1996..
15. Chacon S., Straub B. Pro Git. – 2014.

ДОДАТОК А Лістинг розробленої системи

Файл app.py

```
0001 | import sys
0002 | import json
0003 | import random
0004 | import numpy as np
0005 | import trimesh
0006 |
0007 | from PyQt5.QtWidgets import (
0008 |     QApplication, QWidget, QVBoxLayout, QHBoxLayout,
0009 |     QPushButton, QLabel, QSlider, QCheckBox,
0010 |     QFileDialog, QGroupBox
0011 | )
0012 |
0013 | from PyQt5.QtCore import Qt
0014 |
0015 | import pyqtgraph.opengl as gl
0016 |
0017 | from core.face_generator import FaceMeshGenerator
0018 | from core.model_profile import ModelProfile
0019 |
0020 |
0021 | class FaceGenerator(QWidget):
0022 |     def __init__(self):
0023 |         super().__init__()
0024 |
0025 |         self.setWindowTitle("Генератор")
0026 |         self.resize(1200, 800)
0027 |
0028 |         #profile = ModelProfile("models/male_average")
0029 |         profile = ModelProfile("models/procedural_head_v3")
0030 |
0031 |         self.base_vertices = profile.vertices.copy()
0032 |         self.faces = profile.faces.copy()
0033 |         self.vertex_groups = profile.vertex_groups
0034 |         self.sliders_config = profile.sliders
0035 |
0036 |         x = self.base_vertices[:, 0].copy()
0037 |         y = self.base_vertices[:, 1].copy()
0038 |         z = self.base_vertices[:, 2].copy()
0039 |
0040 |         self.base_vertices[:, 0] = x
0041 |         self.base_vertices[:, 1] = -z
0042 |         self.base_vertices[:, 2] = y
0043 |
0044 |         self.generator = FaceMeshGenerator(
0045 |             self.base_vertices,
0046 |             self.faces,
0047 |             self.vertex_groups,
0048 |             self.sliders_config
0049 |         )
0050 |
0051 |         self.current_vertices = self.base_vertices.copy()
0052 |
0053 |         self.view = gl.GLViewWidget()
0054 |         self.view.setCameraPosition(distance=1, elevation=30, azimuth=270)
0055 |
0056 |         self.mesh_item = None
0057 |         self.wire_item = None
0058 |
0059 |         self.show_wireframe = QCheckBox("Показати сітку")
0060 |         self.show_wireframe.setChecked(True)
0061 |         self.show_wireframe.stateChanged.connect(self.draw_mesh)
0062 |
0063 |         self.sliders = {}
0064 |
0065 |         controls = QVBoxLayout()
0066 |         category_layouts = {}
0067 |
0068 |         for cfg in self.sliders_config:
0069 |             category = cfg.get("category", "Other")
0070 |
0071 |             if category not in category_layouts:
0072 |                 group_box = QGroupBox(category)
0073 |                 group_layout = QVBoxLayout()
0074 |                 group_box.setLayout(group_layout)
0075 |                 controls.addWidget(group_box)
0076 |                 category_layouts[category] = group_layout
0077 |
0078 |                 category_layouts[category].addWidget(QLabel(cfg["label"]))
0079 |
```

```

0080 |         slider = QSlider(Qt.Horizontal)
0081 |         slider.setMinimum(cfg["min"])
0082 |         slider.setMaximum(cfg["max"])
0083 |         slider.setValue(0)
0084 |         slider.valueChanged.connect(self.update_model)
0085 |
0086 |         self.sliders[cfg["key"]] = slider
0087 |         category_layouts[category].addWidget(slider)
0088 |
0089 |     random_group = QGroupBox("Рандомайзер")
0090 |     random_layout = QVBoxLayout()
0091 |
0092 |     random_layout.addWidget(QLabel("Діапазон"))
0093 |
0094 |     self.random_strength_slider = QSlider(Qt.Horizontal)
0095 |     self.random_strength_slider.setMinimum(1)
0096 |     self.random_strength_slider.setMaximum(100)
0097 |     self.random_strength_slider.setValue(20)
0098 |
0099 |     random_layout.addWidget(self.random_strength_slider)
0100 |
0101 |     random_btn = QPushButton("Рандомізувати параметри")
0102 |     random_btn.clicked.connect(self.randomize_face)
0103 |     random_layout.addWidget(random_btn)
0104 |
0105 |     random_group.setLayout(random_layout)
0106 |     controls.addWidget(random_group)
0107 |
0108 |     save_preset_btn = QPushButton("Зберегти пресет")
0109 |     save_preset_btn.clicked.connect(self.save_preset)
0110 |
0111 |     load_preset_btn = QPushButton("Завантажити пресет")
0112 |     load_preset_btn.clicked.connect(self.load_preset)
0113 |
0114 |     export_btn = QPushButton("Експорт моделі")
0115 |     export_btn.clicked.connect(self.export_glb)
0116 |
0117 |     reset_btn = QPushButton("Скинути налаштування")
0118 |     reset_btn.clicked.connect(self.reset_sliders)
0119 |
0120 |     controls.addWidget(save_preset_btn)
0121 |     controls.addWidget(load_preset_btn)
0122 |     controls.addWidget(export_btn)
0123 |     controls.addWidget(reset_btn)
0124 |     controls.addWidget(self.show_wireframe)
0125 |     controls.addStretch()
0126 |
0127 |     layout = QHBoxLayout()
0128 |     layout.addWidget(self.view, 4)
0129 |     layout.addLayout(controls, 1)
0130 |
0131 |     self.setLayout(layout)
0132 |     self.update_model()
0133 |
0134 |     def update_model(self):
0135 |         values = {}
0136 |
0137 |         for key, slider in self.sliders.items():
0138 |             values[key] = slider.value()
0139 |
0140 |         self.current_vertices, self.faces = self.generator.generate(values)
0141 |
0142 |         self.draw_mesh()
0143 |
0144 |     def draw_mesh(self):
0145 |         if self.mesh_item is not None:
0146 |             try:
0147 |                 self.view.removeItem(self.mesh_item)
0148 |             except ValueError:
0149 |                 pass
0150 |             self.mesh_item = None
0151 |
0152 |         if self.wire_item is not None:
0153 |             try:
0154 |                 self.view.removeItem(self.wire_item)
0155 |             except ValueError:
0156 |                 pass
0157 |             self.wire_item = None
0158 |
0159 |         mesh_data = gl.MeshData(
0160 |             vertexes=self.current_vertices,
0161 |             faces=self.faces
0162 |         )
0163 |
0164 |         self.mesh_item = gl.GLMeshItem(
0165 |             meshdata=mesh_data,
0166 |             smooth=False,
0167 |             shader="shaded",

```

```

0168 |         color=(0.82, 0.82, 0.78, 1.0),
0169 |         drawFaces=True,
0170 |         drawEdges=False
0171 |     )
0172 |
0173 |     self.view.addItem(self.mesh_item)
0174 |
0175 |     if self.show_wireframe.isChecked():
0176 |         edges = set()
0177 |
0178 |         for f in self.faces:
0179 |             a, b, c = f
0180 |             edges.add(tuple(sorted((a, b))))
0181 |             edges.add(tuple(sorted((b, c))))
0182 |             edges.add(tuple(sorted((c, a))))
0183 |
0184 |         lines = []
0185 |
0186 |         for a, b in edges:
0187 |             lines.append(self.current_vertices[a])
0188 |             lines.append(self.current_vertices[b])
0189 |
0190 |         lines = np.array(lines, dtype=np.float32)
0191 |
0192 |         self.wire_item = gl.GLLinePlotItem(
0193 |             pos=lines,
0194 |             mode="lines",
0195 |             color=(0.05, 0.05, 0.05, 0.45),
0196 |             width=1,
0197 |             antialias=True
0198 |         )
0199 |
0200 |         self.view.addItem(self.wire_item)
0201 |
0202 |     def export_glb(self):
0203 |         export_mesh = trimesh.Trimesh(
0204 |             vertices=self.current_vertices,
0205 |             faces=self.faces,
0206 |             process=False
0207 |         )
0208 |
0209 |         export_mesh.export("output/generated_face.glb")
0210 |         print("Exported: output/generated_face.glb")
0211 |
0212 |     def reset_sliders(self):
0213 |         for slider in self.sliders.values():
0214 |             slider.setValue(0)
0215 |
0216 |     def save_preset(self):
0217 |         preset = {}
0218 |
0219 |         for key, slider in self.sliders.items():
0220 |             preset[key] = slider.value()
0221 |
0222 |         file_path, _ = QFileDialog.getSaveFileName(
0223 |             self,
0224 |             "Save Preset",
0225 |             "presets/face_preset.json",
0226 |             "JSON Files (*.json)"
0227 |         )
0228 |
0229 |         if not file_path:
0230 |             return
0231 |
0232 |         if not file_path.endswith(".json"):
0233 |             file_path += ".json"
0234 |
0235 |         with open(file_path, "w", encoding="utf-8") as f:
0236 |             json.dump(preset, f, indent=2)
0237 |
0238 |         print("Preset saved:", file_path)
0239 |
0240 |     def load_preset(self):
0241 |         file_path, _ = QFileDialog.getOpenFileName(
0242 |             self,
0243 |             "Load Preset",
0244 |             "presets/",
0245 |             "JSON Files (*.json)"
0246 |         )
0247 |
0248 |         if not file_path:
0249 |             return
0250 |
0251 |         with open(file_path, "r", encoding="utf-8") as f:
0252 |             preset = json.load(f)
0253 |
0254 |         for key, value in preset.items():
0255 |             if key in self.sliders:

```

```

0256 |         self.sliders[key].setValue(value)
0257 |
0258 |     self.update_model()
0259 |     print("Preset loaded:", file_path)
0260 |
0261 |     def randomize_face(self):
0262 |         strength = self.random_strength_slider.value()
0263 |
0264 |         for cfg in self.sliders_config:
0265 |             key = cfg["key"]
0266 |
0267 |             if key not in self.sliders:
0268 |                 continue
0269 |
0270 |             slider = self.sliders[key]
0271 |
0272 |             local_range = cfg.get("random_range", 20)
0273 |             r = int(local_range * (strength / 100.0))
0274 |             r = max(r, 1)
0275 |
0276 |             slider.setValue(random.randint(-r, r))
0277 |
0278 |         self.update_model()
0279 |
0280 |
0281 | if __name__ == "__main__":
0282 |     app = QApplication(sys.argv)
0283 |     window = FaceGenerator()
0284 |     window.show()
0285 |     sys.exit(app.exec_())

```

Файл config.py

```
0001 | SLIDERS = [  
0002 |     # =====  
0003 |     # NOSE  
0004 |     # =====  
0005 |     {  
0006 |         "key": "nose_forward",  
0007 |         "label": "Forward / Back",  
0008 |         "category": "Nose",  
0009 |         "group": "Nose",  
0010 |         "type": "move",  
0011 |         "axis": 1,  
0012 |         "strength": -0.001,  
0013 |         "min": -100,  
0014 |         "max": 100,  
0015 |     },  
0016 |     {  
0017 |         "key": "nose_width",  
0018 |         "label": "Width",  
0019 |         "category": "Nose",  
0020 |         "group": "Nose",  
0021 |         "type": "scale",  
0022 |         "axis": 0,  
0023 |         "strength": 1 / 300,  
0024 |         "min": -100,  
0025 |         "max": 100,  
0026 |     },  
0027 |     {  
0028 |         "key": "nose_height",  
0029 |         "label": "Height",  
0030 |         "category": "Nose",  
0031 |         "group": "Nose",  
0032 |         "type": "scale",  
0033 |         "axis": 2,  
0034 |         "strength": 1 / 300,  
0035 |         "min": -100,  
0036 |         "max": 100,  
0037 |     },  
0038 |  
0039 |     # =====  
0040 |     # JAW / CHIN  
0041 |     # =====  
0042 |     {  
0043 |         "key": "jaw_width",  
0044 |         "label": "Jaw Width",  
0045 |         "category": "Jaw / Chin",  
0046 |         "group": "Jaw",  
0047 |         "type": "scale",  
0048 |         "axis": 0,  
0049 |         "strength": 1 / 300,  
0050 |         "min": -100,  
0051 |         "max": 100,  
0052 |     },  
0053 |     {  
0054 |         "key": "chin_forward",  
0055 |         "label": "Chin Forward / Back",  
0056 |         "category": "Jaw / Chin",  
0057 |         "group": "Chin",  
0058 |         "type": "move",  
0059 |         "axis": 1,  
0060 |         "strength": -0.001,  
0061 |         "min": -100,  
0062 |         "max": 100,  
0063 |     },  
0064 |     {  
0065 |         "key": "chin_width",  
0066 |         "label": "Chin Width",  
0067 |         "category": "Jaw / Chin",  
0068 |         "group": "Chin",  
0069 |         "type": "scale",  
0070 |         "axis": 0,  
0071 |         "strength": 1 / 300,  
0072 |         "min": -100,  
0073 |         "max": 100,  
0074 |     },  
0075 |  
0076 |     # =====  
0077 |     # EARS SYMMETRY  
0078 |     # =====  
0079 |     {  
0080 |         "key": "ears_size",  
0081 |         "label": "Both Size",  
0082 |         "category": "Ears Symmetry",  
0083 |         "type": "scale",  
0084 |         "axis": "all",  
0085 |         "strength": 1 / 300,
```

```

0086 |         "min": -100,
0087 |         "max": 100,
0088 |         "targets": [
0089 |             {"group": "Left Ear", "axis": "all", "strength": 1 / 300},
0090 |             {"group": "Right Ear", "axis": "all", "strength": 1 / 300},
0091 |         ],
0092 |     },
0093 |     {
0094 |         "key": "ears_up",
0095 |         "label": "Both Up / Down",
0096 |         "category": "Ears Symmetry",
0097 |         "type": "move",
0098 |         "axis": 2,
0099 |         "strength": 0.001,
0100 |         "min": -100,
0101 |         "max": 100,
0102 |         "targets": [
0103 |             {"group": "Left Ear", "axis": 2, "strength": 0.001},
0104 |             {"group": "Right Ear", "axis": 2, "strength": 0.001},
0105 |         ],
0106 |     },
0107 |     {
0108 |         "key": "ears_forward",
0109 |         "label": "Both Forward / Back",
0110 |         "category": "Ears Symmetry",
0111 |         "type": "move",
0112 |         "axis": 1,
0113 |         "strength": 0.001,
0114 |         "min": -100,
0115 |         "max": 100,
0116 |         "targets": [
0117 |             {"group": "Left Ear", "axis": 1, "strength": 0.001},
0118 |             {"group": "Right Ear", "axis": 1, "strength": 0.001},
0119 |         ],
0120 |     },
0121 |     {
0122 |         "key": "ears_out",
0123 |         "label": "Both Out / In",
0124 |         "category": "Ears Symmetry",
0125 |         "type": "move",
0126 |         "axis": 0,
0127 |         "strength": 0.001,
0128 |         "min": -100,
0129 |         "max": 100,
0130 |         "targets": [
0131 |             {"group": "Left Ear", "axis": 0, "strength": -0.001},
0132 |             {"group": "Right Ear", "axis": 0, "strength": 0.001},
0133 |         ],
0134 |     },
0135 |
0136 |     # =====
0137 |     # LEFT EAR
0138 |     # =====
0139 |     {
0140 |         "key": "ear_l_x",
0141 |         "label": "Left X",
0142 |         "category": "Left Ear",
0143 |         "group": "Left Ear",
0144 |         "type": "move",
0145 |         "axis": 0,
0146 |         "strength": 0.001,
0147 |         "min": -100,
0148 |         "max": 100,
0149 |     },
0150 |     {
0151 |         "key": "ear_l_y",
0152 |         "label": "Left Y",
0153 |         "category": "Left Ear",
0154 |         "group": "Left Ear",
0155 |         "type": "move",
0156 |         "axis": 1,
0157 |         "strength": 0.001,
0158 |         "min": -100,
0159 |         "max": 100,
0160 |     },
0161 |     {
0162 |         "key": "ear_l_z",
0163 |         "label": "Left Z",
0164 |         "category": "Left Ear",
0165 |         "group": "Left Ear",
0166 |         "type": "move",
0167 |         "axis": 2,
0168 |         "strength": 0.001,
0169 |         "min": -100,
0170 |         "max": 100,
0171 |     },
0172 |     {
0173 |         "key": "ear_l_size",

```

```

0174 |         "label": "Left Size",
0175 |         "category": "Left Ear",
0176 |         "group": "Left Ear",
0177 |         "type": "scale",
0178 |         "axis": "all",
0179 |         "strength": 1 / 300,
0180 |         "min": -100,
0181 |         "max": 100,
0182 |     },
0183 |     {
0184 |         "key": "ear_l_rotate",
0185 |         "label": "Left Rotate",
0186 |         "category": "Left Ear",
0187 |         "group": "Left Ear",
0188 |         "type": "rotate",
0189 |         "axis": 2,
0190 |         "strength": 0.01,
0191 |         "min": -100,
0192 |         "max": 100,
0193 |     },
0194 |
0195 |     # =====
0196 |     # RIGHT EAR
0197 |     # =====
0198 |     {
0199 |         "key": "ear_r_x",
0200 |         "label": "Right X",
0201 |         "category": "Right Ear",
0202 |         "group": "Right Ear",
0203 |         "type": "move",
0204 |         "axis": 0,
0205 |         "strength": 0.001,
0206 |         "min": -100,
0207 |         "max": 100,
0208 |     },
0209 |     {
0210 |         "key": "ear_r_y",
0211 |         "label": "Right Y",
0212 |         "category": "Right Ear",
0213 |         "group": "Right Ear",
0214 |         "type": "move",
0215 |         "axis": 1,
0216 |         "strength": 0.001,
0217 |         "min": -100,
0218 |         "max": 100,
0219 |     },
0220 |     {
0221 |         "key": "ear_r_z",
0222 |         "label": "Right Z",
0223 |         "category": "Right Ear",
0224 |         "group": "Right Ear",
0225 |         "type": "move",
0226 |         "axis": 2,
0227 |         "strength": 0.001,
0228 |         "min": -100,
0229 |         "max": 100,
0230 |     },
0231 |     {
0232 |         "key": "ear_r_size",
0233 |         "label": "Right Size",
0234 |         "category": "Right Ear",
0235 |         "group": "Right Ear",
0236 |         "type": "scale",
0237 |         "axis": "all",
0238 |         "strength": 1 / 300,
0239 |         "min": -100,
0240 |         "max": 100,
0241 |     },
0242 |     {
0243 |         "key": "ear_r_rotate",
0244 |         "label": "Right Rotate",
0245 |         "category": "Right Ear",
0246 |         "group": "Right Ear",
0247 |         "type": "rotate",
0248 |         "axis": 2,
0249 |         "strength": 0.01,
0250 |         "min": -100,
0251 |         "max": 100,
0252 |     },
0253 |
0254 |     # =====
0255 |     # LIPS
0256 |     # =====
0257 |     {
0258 |         "key": "upper_lip_forward",
0259 |         "label": "Upper Lip Forward",
0260 |         "category": "Lips",
0261 |         "group": "Upper Lip",

```

```

0262 |         "type": "move",
0263 |         "axis": 1,
0264 |         "strength": -0.001,
0265 |         "min": -100,
0266 |         "max": 100,
0267 |     },
0268 |     {
0269 |         "key": "lower_lip_forward",
0270 |         "label": "Lower Lip Forward",
0271 |         "category": "Lips",
0272 |         "group": "Lower Lip",
0273 |         "type": "move",
0274 |         "axis": 1,
0275 |         "strength": -0.001,
0276 |         "min": -100,
0277 |         "max": 100,
0278 |     },
0279 |
0280 |     # =====
0281 |     # EYES / EYEBROWS
0282 |     # =====
0283 |     {
0284 |         "key": "eyes_up",
0285 |         "label": "Both Eyes Up / Down",
0286 |         "category": "Eyes",
0287 |         "type": "move",
0288 |         "axis": 2,
0289 |         "strength": 0.001,
0290 |         "min": -100,
0291 |         "max": 100,
0292 |         "targets": [
0293 |             {"group": "Eye Left", "axis": 2, "strength": 0.001},
0294 |             {"group": "Eye Right", "axis": 2, "strength": 0.001},
0295 |         ],
0296 |     },
0297 |     {
0298 |         "key": "eyebrows_up",
0299 |         "label": "Both Eyebrows Up / Down",
0300 |         "category": "Eyebrows",
0301 |         "type": "move",
0302 |         "axis": 2,
0303 |         "strength": 0.001,
0304 |         "min": -100,
0305 |         "max": 100,
0306 |         "targets": [
0307 |             {"group": "Eyebrow Left", "axis": 2, "strength": 0.001},
0308 |             {"group": "Eyebrow Right", "axis": 2, "strength": 0.001},
0309 |         ],
0310 |     },
0311 | ]

```

Файл generate_procedural_head.py

```
0001 | from core.procedural_head import save_procedural_profile
0002 |
0003 | save_procedural_profile(
0004 |     output_dir="models/procedural_head_v3"
0005 | )
```

Файл core/model_profile.py

```
0001 | import json
0002 | import numpy as np
0003 | from pathlib import Path
0004 |
0005 |
0006 | class ModelProfile:
0007 |
0008 |     def __init__(self, profile_dir):
0009 |
0010 |         self.profile_dir = Path(profile_dir)
0011 |
0012 |         self.mesh_path = self.profile_dir / "mesh.npz"
0013 |         self.groups_path = self.profile_dir / "vertex_groups.json"
0014 |         self.sliders_path = self.profile_dir / "sliders.json"
0015 |
0016 |         self.load()
0017 |
0018 |     def load(self):
0019 |
0020 |         data = np.load(self.mesh_path)
0021 |
0022 |         self.vertices = data["vertices"]
0023 |         self.faces = data["faces"]
0024 |
0025 |         with open(self.groups_path, "r", encoding="utf-8") as f:
0026 |             self.vertex_groups = json.load(f)
0027 |
0028 |         with open(self.sliders_path, "r", encoding="utf-8") as f:
0029 |             self.sliders = json.load(f)
```

Файл core/face_generator.py

```
0001 | from core.deform import apply_deformation
0002 |
0003 | class FaceMeshGenerator:
0004 |     def __init__(self, base_vertices, faces, vertex_groups, sliders_config):
0005 |         self.base_vertices = base_vertices
0006 |         self.faces = faces
0007 |         self.vertex_groups = vertex_groups
0008 |         self.sliders_config = sliders_config
0009 |
0010 |     def generate(self, values):
0011 |         vertices = self.base_vertices.copy()
0012 |
0013 |         for cfg in self.sliders_config:
0014 |             key = cfg["key"]
0015 |             value = values.get(key, 0)
0016 |
0017 |             if value == 0:
0018 |                 continue
0019 |
0020 |             targets = cfg.get("targets")
0021 |
0022 |             if targets is None:
0023 |                 targets = [{"group": cfg["group"]}]]
0024 |
0025 |             for target in targets:
0026 |                 group_name = target["group"]
0027 |
0028 |                 if group_name not in self.vertex_groups:
0029 |                     continue
0030 |
0031 |                 indices = self.vertex_groups[group_name]
0032 |
0033 |                 apply_deformation(
0034 |                     vertices,
0035 |                     indices,
0036 |                     cfg,
0037 |                     value,
0038 |                     target
0039 |                 )
0040 |
0041 |         return vertices, self.faces
```

Файл core/deform.py

```
0001 | import numpy as np
0002 |
0003 |
0004 | def apply_move(vertices, indices, axis, amount):
0005 |     vertices[indices, axis] += amount
0006 |
0007 |
0008 | def apply_scale(vertices, indices, axis, amount):
0009 |     scale = 1.0 + amount
0010 |
0011 |     if axis == "all":
0012 |         center = vertices[indices].mean(axis=0)
0013 |         vertices[indices] = center + (vertices[indices] - center) * scale
0014 |     else:
0015 |         center = vertices[indices, axis].mean()
0016 |         vertices[indices, axis] = center + (vertices[indices, axis] - center) * scale
0017 |
0018 |
0019 | def apply_rotate(vertices, indices, axis, amount):
0020 |     center = vertices[indices].mean(axis=0)
0021 |     verts = vertices[indices] - center
0022 |
0023 |     c = np.cos(amount)
0024 |     s = np.sin(amount)
0025 |
0026 |     if axis == 0:
0027 |         rot = np.array([[1, 0, 0], [0, c, -s], [0, s, c]])
0028 |     elif axis == 1:
0029 |         rot = np.array([[c, 0, s], [0, 1, 0], [-s, 0, c]])
0030 |     else:
0031 |         rot = np.array([[c, -s, 0], [s, c, 0], [0, 0, 1]])
0032 |
0033 |     vertices[indices] = verts @ rot.T + center
0034 |
0035 |
0036 | def apply_deformation(vertices, indices, cfg, value, target=None):
0037 |     axis = target.get("axis", cfg.get("axis")) if target else cfg.get("axis")
0038 |     strength = target.get("strength", cfg.get("strength", 0.001)) if target else cfg.get("strength", 0.001)
0039 |
0040 |     amount = value * strength
0041 |
0042 |     if cfg["type"] == "move":
0043 |         apply_move(vertices, indices, axis, amount)
0044 |     elif cfg["type"] == "scale":
0045 |         apply_scale(vertices, indices, axis, amount)
0046 |     elif cfg["type"] == "rotate":
0047 |         apply_rotate(vertices, indices, axis, amount)
```

Файл core/procedural_head.py

```
0001 | import json
0002 | import os
0003 | import numpy as np
0004 |
0005 |
0006 | # =====
0007 | # Math helpers
0008 | # =====
0009 |
0010 | def smoothstep(edge0, edge1, x):
0011 |     t = np.clip((x - edge0) / (edge1 - edge0), 0.0, 1.0)
0012 |     return t * t * (3.0 - 2.0 * t)
0013 |
0014 |
0015 | def gaussian(x, center, width):
0016 |     return np.exp(-((x - center) / width) ** 2)
0017 |
0018 |
0019 | def gaussian2(x, z, cx, cz, wx, wz):
0020 |     return np.exp(-(((x - cx) / wx) ** 2 + ((z - cz) / wz) ** 2))
0021 |
0022 |
0023 | def add_group(groups, name, index):
0024 |     if name not in groups:
0025 |         groups[name] = []
0026 |     groups[name].append(index)
0027 |
0028 |
0029 | # =====
0030 | # Primitive meshes
0031 | # =====
0032 |
0033 | def add_uv_sphere(vertices, faces, center, scale, rings=12, segments=18, group_name=None, groups=None):
0034 |     start = len(vertices)
0035 |
0036 |     for i in range(rings + 1):
0037 |         tv = i / rings
0038 |         theta = np.pi * tv
0039 |
0040 |         for j in range(segments):
0041 |             u = j / segments
0042 |             phi = 2 * np.pi * u
0043 |
0044 |             x = center[0] + np.sin(theta) * np.cos(phi) * scale[0]
0045 |             y = center[1] + np.sin(theta) * np.sin(phi) * scale[1]
0046 |             z = center[2] + np.cos(theta) * scale[2]
0047 |
0048 |             vertices.append([x, y, z])
0049 |
0050 |             if group_name and groups is not None:
0051 |                 add_group(groups, group_name, len(vertices) - 1)
0052 |
0053 |         for i in range(rings):
0054 |             for j in range(segments):
0055 |                 a = start + i * segments + j
0056 |                 b = start + i * segments + (j + 1) % segments
0057 |                 c = start + (i + 1) * segments + j
0058 |                 d = start + (i + 1) * segments + (j + 1) % segments
0059 |
0060 |                 faces.append([a, c, b])
0061 |                 faces.append([b, c, d])
0062 |
0063 |     return list(range(start, len(vertices)))
0064 |
0065 |
0066 | def add_ear(vertices, faces, groups, name, side):
0067 |     """
0068 |     side:
0069 |         -1 = left
0070 |         1 = right
0071 |     """
0072 |
0073 |     center_x = side * 0.43
0074 |
0075 |     ear_indices = add_uv_sphere(
0076 |         vertices,
0077 |         faces,
0078 |         center=(center_x, 0.015, 0.055),
0079 |         scale=(0.050, 0.030, 0.155),
0080 |         rings=16,
0081 |         segments=22,
0082 |         group_name=name,
0083 |         groups=groups
0084 |     )
0085 |
```

```

0086 |     # inner ear small raised oval
0087 |     inner_name = name + " Inner"
0088 |
0089 |     add_uv_sphere(
0090 |         vertices,
0091 |         faces,
0092 |         center=(side * 0.438, -0.012, 0.052),
0093 |         scale=(0.024, 0.014, 0.080),
0094 |         rings=10,
0095 |         segments=14,
0096 |         group_name=inner_name,
0097 |         groups=groups
0098 |     )
0099 |
0100 |     return ear_indices
0101 |
0102 |
0103 | # =====
0104 | # Main procedural head
0105 | # =====
0106 |
0107 | def generate_detailed_head(rings=84, segments=112):
0108 |     vertices = []
0109 |     faces = []
0110 |
0111 |     groups = {
0112 |         "Head": [],
0113 |         "Cranium": [],
0114 |         "Forehead": [],
0115 |         "Nose": [],
0116 |         "Nose Bridge": [],
0117 |         "Nose Tip": [],
0118 |         "Jaw": [],
0119 |         "Chin": [],
0120 |         "Mouth": [],
0121 |         "Upper Lip": [],
0122 |         "Lower Lip": [],
0123 |         "Eye Left": [],
0124 |         "Eye Right": [],
0125 |         "Eyebrow Left": [],
0126 |         "Eyebrow Right": [],
0127 |         "Cheek Left": [],
0128 |         "Cheek Right": [],
0129 |         "Neck": []
0130 |     }
0131 |
0132 |     # Main continuous head surface
0133 |     for i in range(rings + 1):
0134 |         tv = i / rings
0135 |
0136 |         # z from top to bottom
0137 |         z_norm = 1.0 - 2.0 * tv
0138 |
0139 |         # vertical coordinate
0140 |         z = z_norm * 0.64
0141 |
0142 |         # profile radius by height
0143 |         # Wider cranium, narrower jaw/chin
0144 |         cranium = smoothstep(-0.10, 0.45, z)
0145 |         jaw_zone = smoothstep(-0.15, -0.55, -z)
0146 |
0147 |         base_rx = 0.34 + 0.080 * cranium - 0.090 * jaw_zone
0148 |         base_ry = 0.27 + 0.045 * cranium - 0.055 * jaw_zone
0149 |
0150 |         # narrow top and bottom
0151 |         vertical_taper = np.sqrt(max(0.0, 1.0 - (z / 0.66) ** 2))
0152 |         rx = base_rx * vertical_taper
0153 |         ry = base_ry * vertical_taper
0154 |
0155 |         # make chin less pointy than sphere
0156 |         if z < -0.48:
0157 |             rx *= 0.78
0158 |             ry *= 0.82
0159 |
0160 |         for j in range(segments):
0161 |             u = j / segments
0162 |             phi = 2.0 * np.pi * u
0163 |
0164 |             # coordinate convention:
0165 |             # x = width
0166 |             # y = depth; negative y = face/front
0167 |             # z = height
0168 |             sx = np.cos(phi)
0169 |             sy = np.sin(phi)
0170 |
0171 |             x = sx * rx
0172 |             y = sy * ry
0173 |

```

```

0174 |         # front/back indicators
0175 |         front = smoothstep(0.12, 0.95, -sy)
0176 |         back = smoothstep(0.12, 0.95, sy)
0177 |
0178 |         # Flatten front of face
0179 |         y += front * 0.030
0180 |
0181 |         # Back of skull protrudes
0182 |         y += back * 0.055 * smoothstep(-0.20, 0.45, z)
0183 |
0184 |         # Temple narrowing
0185 |         temple = gaussian(z, 0.18, 0.18) * front
0186 |         x *= 1.0 - temple * 0.055
0187 |
0188 |         # Jaw angles
0189 |         jaw_angle_l = gaussian2(x, z, -0.245, -0.335, 0.085, 0.120) * front
0190 |         jaw_angle_r = gaussian2(x, z, 0.245, -0.335, 0.085, 0.120) * front
0191 |         x += (-0.018 * jaw_angle_l) + (0.018 * jaw_angle_r)
0192 |         y -= (jaw_angle_l + jaw_angle_r) * 0.025
0193 |
0194 |         # Chin
0195 |         chin = gaussian2(x, z, 0.0, -0.455, 0.155, 0.100) * front
0196 |         y -= chin * 0.065
0197 |         x *= 1.0 - chin * 0.045
0198 |
0199 |         # Cheekbones
0200 |         cheek_l = gaussian2(x, z, -0.185, 0.020, 0.105, 0.155) * front
0201 |         cheek_r = gaussian2(x, z, 0.185, 0.020, 0.105, 0.155) * front
0202 |         y -= (cheek_l + cheek_r) * 0.050
0203 |
0204 |         # Under cheek soft hollow
0205 |         cheek_hollow = (
0206 |             gaussian2(x, z, -0.155, -0.145, 0.120, 0.130)
0207 |             + gaussian2(x, z, 0.155, -0.145, 0.120, 0.130)
0208 |         ) * front
0209 |         y += cheek_hollow * 0.025
0210 |
0211 |         # Eye sockets
0212 |         eye_l = gaussian2(x, z, -0.130, 0.205, 0.073, 0.060) * front
0213 |         eye_r = gaussian2(x, z, 0.130, 0.205, 0.073, 0.060) * front
0214 |         y += (eye_l + eye_r) * 0.075
0215 |
0216 |         # Brow ridge
0217 |         brow_l = gaussian2(x, z, -0.130, 0.285, 0.095, 0.045) * front
0218 |         brow_r = gaussian2(x, z, 0.130, 0.285, 0.095, 0.045) * front
0219 |         y -= (brow_l + brow_r) * 0.040
0220 |
0221 |         # Nose bridge
0222 |         nose_bridge = gaussian2(x, z, 0.0, 0.185, 0.055, 0.135) * front
0223 |         y -= nose_bridge * 0.070
0224 |
0225 |         # Nose body
0226 |         nose_body = gaussian2(x, z, 0.0, 0.055, 0.078, 0.130) * front
0227 |         y -= nose_body * 0.105
0228 |
0229 |         # Nose tip
0230 |         nose_tip = gaussian2(x, z, 0.0, -0.020, 0.090, 0.070) * front
0231 |         y -= nose_tip * 0.090
0232 |
0233 |         # Nostrils / wing shape
0234 |         nostril_l = gaussian2(x, z, -0.055, -0.070, 0.035, 0.030) * front
0235 |         nostril_r = gaussian2(x, z, 0.055, -0.070, 0.035, 0.030) * front
0236 |         y += (nostril_l + nostril_r) * 0.035
0237 |
0238 |         # Philtrum
0239 |         philtrum = gaussian2(x, z, 0.0, -0.125, 0.045, 0.055) * front
0240 |         y += philtrum * 0.018
0241 |
0242 |         # Lips
0243 |         upper_lip = gaussian2(x, z, 0.0, -0.175, 0.170, 0.032) * front
0244 |         lower_lip = gaussian2(x, z, 0.0, -0.225, 0.155, 0.042) * front
0245 |         mouth_cut = gaussian2(x, z, 0.0, -0.200, 0.185, 0.018) * front
0246 |
0247 |         y -= upper_lip * 0.036
0248 |         y -= lower_lip * 0.040
0249 |         y += mouth_cut * 0.045
0250 |
0251 |         # Corners of mouth slightly pushed inward
0252 |         corner_l = gaussian2(x, z, -0.175, -0.200, 0.035, 0.035) * front
0253 |         corner_r = gaussian2(x, z, 0.175, -0.200, 0.035, 0.035) * front
0254 |         y += (corner_l + corner_r) * 0.020
0255 |
0256 |         # Neck connection at bottom/back
0257 |         neck_influence = smoothstep(0.48, 0.63, -z) * (0.5 + 0.5 * back)
0258 |         x *= 1.0 - neck_influence * 0.25
0259 |         y += neck_influence * 0.060
0260 |
0261 |         idx = len(vertices)

```

```

0262 |         vertices.append([x, y, z])
0263 |
0264 |         groups["Head"].append(idx)
0265 |
0266 |         if z > 0.25:
0267 |             groups["Cranium"].append(idx)
0268 |
0269 |         if front > 0.45 and z > 0.245:
0270 |             groups["Forehead"].append(idx)
0271 |
0272 |         if front > 0.45 and abs(x) < 0.105 and -0.095 < z < 0.265:
0273 |             groups["Nose"].append(idx)
0274 |
0275 |         if nose_bridge > 0.25:
0276 |             groups["Nose Bridge"].append(idx)
0277 |
0278 |         if nose_tip > 0.25:
0279 |             groups["Nose Tip"].append(idx)
0280 |
0281 |         if front > 0.35 and z < -0.270 and abs(x) > 0.105:
0282 |             groups["Jaw"].append(idx)
0283 |
0284 |         if chin > 0.25:
0285 |             groups["Chin"].append(idx)
0286 |
0287 |         if upper_lip > 0.25:
0288 |             groups["Upper Lip"].append(idx)
0289 |             groups["Mouth"].append(idx)
0290 |
0291 |         if lower_lip > 0.25:
0292 |             groups["Lower Lip"].append(idx)
0293 |             groups["Mouth"].append(idx)
0294 |
0295 |         if eye_l > 0.25:
0296 |             groups["Eye Left"].append(idx)
0297 |
0298 |         if eye_r > 0.25:
0299 |             groups["Eye Right"].append(idx)
0300 |
0301 |         if brow_l > 0.25:
0302 |             groups["Eyebrow Left"].append(idx)
0303 |
0304 |         if brow_r > 0.25:
0305 |             groups["Eyebrow Right"].append(idx)
0306 |
0307 |         if cheek_l > 0.25:
0308 |             groups["Cheek Left"].append(idx)
0309 |
0310 |         if cheek_r > 0.25:
0311 |             groups["Cheek Right"].append(idx)
0312 |
0313 |         if z < -0.48:
0314 |             groups["Neck"].append(idx)
0315 |
0316 |         # faces
0317 |         for i in range(rings):
0318 |             for j in range(segments):
0319 |                 a = i * segments + j
0320 |                 b = i * segments + (j + 1) % segments
0321 |                 c = (i + 1) * segments + j
0322 |                 d = (i + 1) * segments + (j + 1) % segments
0323 |
0324 |                 faces.append([a, c, b])
0325 |                 faces.append([b, c, d])
0326 |
0327 |         return vertices, faces, groups
0328 |
0329 |
0330 | # =====
0331 | # Sliders config for this generated model
0332 | # =====
0333 |
0334 | def create_default_sliders():
0335 |     return [
0336 |         {
0337 |             "key": "head_width",
0338 |             "label": "Head Width",
0339 |             "category": "Head",
0340 |             "group": "Head",
0341 |             "type": "scale",
0342 |             "axis": 0,
0343 |             "strength": 0.0015,
0344 |             "min": -100,
0345 |             "max": 100,
0346 |             "random_range": 18
0347 |         },
0348 |         {
0349 |             "key": "head_height",

```

```

0350 |         "label": "Head Height",
0351 |         "category": "Head",
0352 |         "group": "Head",
0353 |         "type": "scale",
0354 |         "axis": 2,
0355 |         "strength": 0.0015,
0356 |         "min": -100,
0357 |         "max": 100,
0358 |         "random_range": 18
0359 |     },
0360 |     {
0361 |         "key": "cranium_size",
0362 |         "label": "Cranium Size",
0363 |         "category": "Head",
0364 |         "group": "Cranium",
0365 |         "type": "scale",
0366 |         "axis": "all",
0367 |         "strength": 0.0015,
0368 |         "min": -100,
0369 |         "max": 100,
0370 |         "random_range": 14
0371 |     },
0372 |
0373 |     {
0374 |         "key": "nose_forward",
0375 |         "label": "Nose Forward",
0376 |         "category": "Nose",
0377 |         "group": "Nose",
0378 |         "type": "move",
0379 |         "axis": 1,
0380 |         "strength": -0.0007,
0381 |         "min": -100,
0382 |         "max": 100,
0383 |         "random_range": 18
0384 |     },
0385 |     {
0386 |         "key": "nose_width",
0387 |         "label": "Nose Width",
0388 |         "category": "Nose",
0389 |         "group": "Nose",
0390 |         "type": "scale",
0391 |         "axis": 0,
0392 |         "strength": 0.0018,
0393 |         "min": -100,
0394 |         "max": 100,
0395 |         "random_range": 18
0396 |     },
0397 |     {
0398 |         "key": "nose_tip",
0399 |         "label": "Nose Tip",
0400 |         "category": "Nose",
0401 |         "group": "Nose Tip",
0402 |         "type": "move",
0403 |         "axis": 1,
0404 |         "strength": -0.0008,
0405 |         "min": -100,
0406 |         "max": 100,
0407 |         "random_range": 14
0408 |     },
0409 |
0410 |     {
0411 |         "key": "jaw_width",
0412 |         "label": "Jaw Width",
0413 |         "category": "Jaw / Chin",
0414 |         "group": "Jaw",
0415 |         "type": "scale",
0416 |         "axis": 0,
0417 |         "strength": 0.0018,
0418 |         "min": -100,
0419 |         "max": 100,
0420 |         "random_range": 18
0421 |     },
0422 |     {
0423 |         "key": "chin_forward",
0424 |         "label": "Chin Forward",
0425 |         "category": "Jaw / Chin",
0426 |         "group": "Chin",
0427 |         "type": "move",
0428 |         "axis": 1,
0429 |         "strength": -0.0008,
0430 |         "min": -100,
0431 |         "max": 100,
0432 |         "random_range": 14
0433 |     },
0434 |     {
0435 |         "key": "chin_width",
0436 |         "label": "Chin Width",
0437 |         "category": "Jaw / Chin",

```

```

0438 |         "group": "Chin",
0439 |         "type": "scale",
0440 |         "axis": 0,
0441 |         "strength": 0.0015,
0442 |         "min": -100,
0443 |         "max": 100,
0444 |         "random_range": 12
0445 |     },
0446 |
0447 |     {
0448 |         "key": "mouth_width",
0449 |         "label": "Mouth Width",
0450 |         "category": "Mouth",
0451 |         "group": "Mouth",
0452 |         "type": "scale",
0453 |         "axis": 0,
0454 |         "strength": 0.0015,
0455 |         "min": -100,
0456 |         "max": 100,
0457 |         "random_range": 14
0458 |     },
0459 |     {
0460 |         "key": "upper_lip_forward",
0461 |         "label": "Upper Lip Forward",
0462 |         "category": "Mouth",
0463 |         "group": "Upper Lip",
0464 |         "type": "move",
0465 |         "axis": 1,
0466 |         "strength": -0.0006,
0467 |         "min": -100,
0468 |         "max": 100,
0469 |         "random_range": 10
0470 |     },
0471 |     {
0472 |         "key": "lower_lip_forward",
0473 |         "label": "Lower Lip Forward",
0474 |         "category": "Mouth",
0475 |         "group": "Lower Lip",
0476 |         "type": "move",
0477 |         "axis": 1,
0478 |         "strength": -0.0006,
0479 |         "min": -100,
0480 |         "max": 100,
0481 |         "random_range": 10
0482 |     },
0483 |
0484 |     {
0485 |         "key": "eyes_up",
0486 |         "label": "Eyes Up / Down",
0487 |         "category": "Eyes",
0488 |         "type": "move",
0489 |         "axis": 2,
0490 |         "strength": 0.0006,
0491 |         "min": -100,
0492 |         "max": 100,
0493 |         "random_range": 10,
0494 |         "targets": [
0495 |             {"group": "Eye Left", "axis": 2, "strength": 0.0006},
0496 |             {"group": "Eye Right", "axis": 2, "strength": 0.0006}
0497 |         ]
0498 |     },
0499 |     {
0500 |         "key": "eyes_width",
0501 |         "label": "Eyes Width",
0502 |         "category": "Eyes",
0503 |         "type": "scale",
0504 |         "axis": 0,
0505 |         "strength": 0.0015,
0506 |         "min": -100,
0507 |         "max": 100,
0508 |         "random_range": 10,
0509 |         "targets": [
0510 |             {"group": "Eye Left", "axis": 0, "strength": 0.0015},
0511 |             {"group": "Eye Right", "axis": 0, "strength": 0.0015}
0512 |         ]
0513 |     },
0514 |
0515 |     {
0516 |         "key": "brows_up",
0517 |         "label": "Brows Up / Down",
0518 |         "category": "Eyebrows",
0519 |         "type": "move",
0520 |         "axis": 2,
0521 |         "strength": 0.0006,
0522 |         "min": -100,
0523 |         "max": 100,
0524 |         "random_range": 10,
0525 |         "targets": [

```

```

0526 |         {"group": "Eyebrow Left", "axis": 2, "strength": 0.0006},
0527 |         {"group": "Eyebrow Right", "axis": 2, "strength": 0.0006}
0528 |     ]
0529 | },
0530 |
0531 | {
0532 |     "key": "cheeks_width",
0533 |     "label": "Cheeks Width",
0534 |     "category": "Cheeks",
0535 |     "type": "scale",
0536 |     "axis": 0,
0537 |     "strength": 0.0014,
0538 |     "min": -100,
0539 |     "max": 100,
0540 |     "random_range": 14,
0541 |     "targets": [
0542 |         {"group": "Cheek Left", "axis": 0, "strength": 0.0014},
0543 |         {"group": "Cheek Right", "axis": 0, "strength": 0.0014}
0544 |     ]
0545 | },
0546 |
0547 | {
0548 |     "key": "ears_size",
0549 |     "label": "Ears Size",
0550 |     "category": "Ears",
0551 |     "type": "scale",
0552 |     "axis": "all",
0553 |     "strength": 0.0015,
0554 |     "min": -100,
0555 |     "max": 100,
0556 |     "random_range": 12,
0557 |     "targets": [
0558 |         {"group": "Left Ear", "axis": "all", "strength": 0.0015},
0559 |         {"group": "Right Ear", "axis": "all", "strength": 0.0015}
0560 |     ]
0561 | },
0562 | {
0563 |     "key": "ears_out",
0564 |     "label": "Ears Out / In",
0565 |     "category": "Ears",
0566 |     "type": "move",
0567 |     "axis": 0,
0568 |     "strength": 0.0006,
0569 |     "min": -100,
0570 |     "max": 100,
0571 |     "random_range": 10,
0572 |     "targets": [
0573 |         {"group": "Left Ear", "axis": 0, "strength": -0.0006},
0574 |         {"group": "Right Ear", "axis": 0, "strength": 0.0006}
0575 |     ]
0576 | }
0577 | ]
0578 |
0579 |
0580 | # =====
0581 | # Save profile
0582 | # =====
0583 |
0584 | def save_procedural_profile(output_dir="models/procedural_head_v3"):
0585 |     os.makedirs(output_dir, exist_ok=True)
0586 |
0587 |     vertices, faces, groups = generate_detailed_head()
0588 |
0589 |     # Add ears
0590 |     add_ear(vertices, faces, groups, "Left Ear", side=-1)
0591 |     add_ear(vertices, faces, groups, "Right Ear", side=1)
0592 |
0593 |     # Add eyeballs as separate visible objects/groups
0594 |     add_uv_sphere(
0595 |         vertices,
0596 |         faces,
0597 |         center=(-0.130, -0.300, 0.205),
0598 |         scale=(0.043, 0.024, 0.031),
0599 |         rings=10,
0600 |         segments=14,
0601 |         group_name="Eye Ball Left",
0602 |         groups=groups
0603 |     )
0604 |
0605 |     add_uv_sphere(
0606 |         vertices,
0607 |         faces,
0608 |         center=(0.130, -0.300, 0.205),
0609 |         scale=(0.043, 0.024, 0.031),
0610 |         rings=10,
0611 |         segments=14,
0612 |         group_name="Eye Ball Right",
0613 |         groups=groups

```

```

0614 |     )
0615 |
0616 |     vertices = np.array(vertices, dtype=np.float32)
0617 |     faces = np.array(faces, dtype=np.int32)
0618 |     sliders = create_default_sliders()
0619 |
0620 |     np.savez(
0621 |         os.path.join(output_dir, "mesh.npz"),
0622 |         vertices=vertices,
0623 |         faces=faces
0624 |     )
0625 |
0626 |     with open(os.path.join(output_dir, "vertex_groups.json"), "w", encoding="utf-8") as f:
0627 |         json.dump(groups, f, indent=2)
0628 |
0629 |     with open(os.path.join(output_dir, "sliders.json"), "w", encoding="utf-8") as f:
0630 |         json.dump(sliders, f, indent=2)
0631 |
0632 |     print("Generated procedural profile:", output_dir)
0633 |     print("vertices:", vertices.shape)
0634 |     print("faces:", faces.shape)
0635 |
0636 |     for name, indices in groups.items():
0637 |         print(name, len(indices))

```

Файл blender/export_mesh_data.py

```
0001 | import bpy
0002 | import bmesh
0003 | import json
0004 | import os
0005 | import numpy as np
0006 |
0007 | OBJ_NAME = "Male Average Head"
0008 |
0009 | OUT_DIR = r"C:\Users\Bohdan\Desktop\face generator\assets"
0010 | os.makedirs(OUT_DIR, exist_ok=True)
0011 |
0012 | obj = bpy.data.objects[OBJ_NAME]
0013 |
0014 | # Берем исходный mesh
0015 | source_mesh = obj.data
0016 |
0017 | # Копируем mesh, чтобы не портить оригинал в Blender
0018 | temp_mesh = source_mesh.copy()
0019 |
0020 | # Триангулируем копию через bmesh
0021 | bm = bmesh.new()
0022 | bm.from_mesh(temp_mesh)
0023 |
0024 | bmesh.ops.triangulate(
0025 |     bm,
0026 |     faces=bm.faces[:]
0027 | )
0028 |
0029 | bm.to_mesh(temp_mesh)
0030 | bm.free()
0031 |
0032 | # Экспорт вершин
0033 | vertices = np.array(
0034 |     [v.co[:] for v in temp_mesh.vertices],
0035 |     dtype=np.float32
0036 | )
0037 |
0038 | # Экспорт треугольников
0039 | faces = np.array(
0040 |     [list(poly.vertices) for poly in temp_mesh.polygons],
0041 |     dtype=np.int32
0042 | )
0043 |
0044 | # Экспорт vertex groups
0045 | groups_data = {}
0046 |
0047 | for group in obj.vertex_groups:
0048 |     groups_data[group.name] = []
0049 |
0050 | for vert in source_mesh.vertices:
0051 |     for g in vert.groups:
0052 |         group_name = obj.vertex_groups[g.group].name
0053 |         groups_data[group_name].append(vert.index)
0054 |
0055 | # Сохраняем mesh
0056 | np.savez(
0057 |     os.path.join(OUT_DIR, "mesh.npz"),
0058 |     vertices=vertices,
0059 |     faces=faces
0060 | )
0061 |
0062 | # Сохраняем группы
0063 | with open(os.path.join(OUT_DIR, "vertex_groups.json"), "w", encoding="utf-8") as f:
0064 |     json.dump(groups_data, f, indent=2)
0065 |
0066 | print("Exported mesh:")
0067 | print("vertices:", vertices.shape)
0068 | print("faces:", faces.shape)
0069 |
0070 | print("Exported groups:")
0071 | for name, indices in groups_data.items():
0072 |     print(name, len(indices))
0073 |
0074 | # Удаляем временный mesh из Blender data
0075 | bpy.data.meshes.remove(temp_mesh)
```

Файл blender/export_groups.py

```
0001 | import bpy
0002 | import json
0003 | import os
0004 |
0005 | OBJ_NAME = "Male Average Head"
0006 | OUT_PATH = r"C:\Users\Bohdan\Desktop\face generator\assets\vertex_groups.json"
0007 |
0008 | obj = bpy.data.objects[OBJ_NAME]
0009 |
0010 | groups_data = {}
0011 |
0012 | for group in obj.vertex_groups:
0013 |     groups_data[group.name] = []
0014 |
0015 | for vert in obj.data.vertices:
0016 |     for g in vert.groups:
0017 |         group_name = obj.vertex_groups[g.group].name
0018 |         groups_data[group_name].append(vert.index)
0019 |
0020 | os.makedirs(os.path.dirname(OUT_PATH), exist_ok=True)
0021 |
0022 | with open(OUT_PATH, "w", encoding="utf-8") as f:
0023 |     json.dump(groups_data, f, indent=2)
0024 |
0025 | print("Exported vertex groups:")
0026 | for name, indices in groups_data.items():
0027 |     print(name, len(indices))
0028 |
0029 | print("Saved to:", OUT_PATH)
```

Файл models/male_average/sliders.json

```
0001 | [
0002 |   {
0003 |     "key": "nose_forward",
0004 |     "label": "Вперед / назад",
0005 |     "category": "Ніс",
0006 |     "group": "Ніс",
0007 |     "type": "move",
0008 |     "axis": 1,
0009 |     "strength": -0.001,
0010 |     "min": -100,
0011 |     "max": 100
0012 |   },
0013 |   {
0014 |     "key": "nose_width",
0015 |     "label": "Ширину",
0016 |     "category": "Ніс",
0017 |     "group": "Ніс",
0018 |     "type": "scale",
0019 |     "axis": 0,
0020 |     "strength": 0.0033333333,
0021 |     "min": -100,
0022 |     "max": 100
0023 |   },
0024 |   {
0025 |     "key": "nose_height",
0026 |     "label": "Висота",
0027 |     "category": "Ніс",
0028 |     "group": "Ніс",
0029 |     "type": "scale",
0030 |     "axis": 2,
0031 |     "strength": 0.0033333333,
0032 |     "min": -100,
0033 |     "max": 100
0034 |   },
0035 |   {
0036 |     "key": "jaw_width",
0037 |     "label": "Ширину щелепи",
0038 |     "category": "Щелепа / підборіддя",
0039 |     "group": "Щелепа",
0040 |     "type": "scale",
0041 |     "axis": 0,
0042 |     "strength": 0.0033333333,
0043 |     "min": -100,
0044 |     "max": 100
0045 |   },
0046 |   {
0047 |     "key": "chin_forward",
0048 |     "label": "Підборіддя вперед / назад",
0049 |     "category": "Щелепа / підборіддя",
0050 |     "group": "Підборіддя",
0051 |     "type": "move",
0052 |     "axis": 1,
0053 |     "strength": -0.001,
0054 |     "min": -100,
0055 |     "max": 100
0056 |   },
0057 |   {
0058 |     "key": "chin_width",
0059 |     "label": "Ширину підборіддя",
0060 |     "category": "Щелепа / підборіддя",
0061 |     "group": "Підборіддя",
0062 |     "type": "scale",
0063 |     "axis": 0,
0064 |     "strength": 0.0033333333,
0065 |     "min": -100,
0066 |     "max": 100
0067 |   },
0068 |   {
0069 |     "key": "ears_size",
0070 |     "label": "Загальний розмір",
0071 |     "category": "Симетрія вух",
0072 |     "type": "scale",
0073 |     "axis": "all",
0074 |     "strength": 0.0033333333,
0075 |     "min": -100,
0076 |     "max": 100,
0077 |     "targets": [
0078 |       {
0079 |         "group": "Ліве вухо",
0080 |         "axis": "all",
0081 |         "strength": 0.0033333333
0082 |       },
0083 |     ],
0084 |   },
0085 | ]
```

```

0086 |         "group": "Праве вухо",
0087 |         "axis": "all",
0088 |         "strength": 0.0033333333
0089 |     }
0090 | ]
0091 | },
0092 |
0093 | {
0094 |     "key": "ear_l_x",
0095 |     "label": "Ліве X",
0096 |     "category": "Ліве вухо",
0097 |     "group": "Ліве вухо",
0098 |     "type": "move",
0099 |     "axis": 0,
0100 |     "strength": 0.001,
0101 |     "min": -100,
0102 |     "max": 100
0103 | },
0104 | {
0105 |     "key": "ear_l_y",
0106 |     "label": "Ліве Y",
0107 |     "category": "Ліве вухо",
0108 |     "group": "Ліве вухо",
0109 |     "type": "move",
0110 |     "axis": 1,
0111 |     "strength": 0.001,
0112 |     "min": -100,
0113 |     "max": 100
0114 | },
0115 | {
0116 |     "key": "ear_l_z",
0117 |     "label": "Ліве Z",
0118 |     "category": "Ліве вухо",
0119 |     "group": "Ліве вухо",
0120 |     "type": "move",
0121 |     "axis": 2,
0122 |     "strength": 0.001,
0123 |     "min": -100,
0124 |     "max": 100
0125 | },
0126 |
0127 | {
0128 |     "key": "ear_r_x",
0129 |     "label": "Праве X",
0130 |     "category": "Праве вухо",
0131 |     "group": "Праве вухо",
0132 |     "type": "move",
0133 |     "axis": 0,
0134 |     "strength": 0.001,
0135 |     "min": -100,
0136 |     "max": 100
0137 | },
0138 | {
0139 |     "key": "ear_r_y",
0140 |     "label": "Праве Y",
0141 |     "category": "Праве вухо",
0142 |     "group": "Праве вухо",
0143 |     "type": "move",
0144 |     "axis": 1,
0145 |     "strength": 0.001,
0146 |     "min": -100,
0147 |     "max": 100
0148 | },
0149 | {
0150 |     "key": "ear_r_z",
0151 |     "label": "Праве Z",
0152 |     "category": "Праве вухо",
0153 |     "group": "Праве вухо",
0154 |     "type": "move",
0155 |     "axis": 2,
0156 |     "strength": 0.001,
0157 |     "min": -100,
0158 |     "max": 100
0159 | }
0160 | ]

```

Файл models/procedural_head/sliders.json

```
0001 | [
0002 |   {
0003 |     "key": "head_width",
0004 |     "label": "Head Width",
0005 |     "category": "Head",
0006 |     "group": "Head",
0007 |     "type": "scale",
0008 |     "axis": 0,
0009 |     "strength": 0.0033333333,
0010 |     "min": -100,
0011 |     "max": 100
0012 |   },
0013 |   {
0014 |     "key": "head_height",
0015 |     "label": "Head Height",
0016 |     "category": "Head",
0017 |     "group": "Head",
0018 |     "type": "scale",
0019 |     "axis": 2,
0020 |     "strength": 0.0033333333,
0021 |     "min": -100,
0022 |     "max": 100
0023 |   },
0024 |   {
0025 |     "key": "nose_forward",
0026 |     "label": "Nose Forward",
0027 |     "category": "Nose",
0028 |     "group": "Nose",
0029 |     "type": "move",
0030 |     "axis": 1,
0031 |     "strength": -0.001,
0032 |     "min": -100,
0033 |     "max": 100
0034 |   },
0035 |   {
0036 |     "key": "nose_width",
0037 |     "label": "Nose Width",
0038 |     "category": "Nose",
0039 |     "group": "Nose",
0040 |     "type": "scale",
0041 |     "axis": 0,
0042 |     "strength": 0.0033333333,
0043 |     "min": -100,
0044 |     "max": 100
0045 |   },
0046 |   {
0047 |     "key": "ears_size",
0048 |     "label": "Both Ears Size",
0049 |     "category": "Ears",
0050 |     "type": "scale",
0051 |     "axis": "all",
0052 |     "strength": 0.0033333333,
0053 |     "min": -100,
0054 |     "max": 100,
0055 |     "targets": [
0056 |       {
0057 |         "group": "Left Ear",
0058 |         "axis": "all",
0059 |         "strength": 0.0033333333
0060 |       },
0061 |       {
0062 |         "group": "Right Ear",
0063 |         "axis": "all",
0064 |         "strength": 0.0033333333
0065 |       }
0066 |     ]
0067 |   },
0068 |   {
0069 |     "key": "eyes_size",
0070 |     "label": "Both Eyes Size",
0071 |     "category": "Eyes",
0072 |     "type": "scale",
0073 |     "axis": "all",
0074 |     "strength": 0.0033333333,
0075 |     "min": -100,
0076 |     "max": 100,
0077 |     "targets": [
0078 |       {
0079 |         "group": "Eye Left",
0080 |         "axis": "all",
0081 |         "strength": 0.0033333333
0082 |       },
0083 |       {
0084 |         "group": "Eye Right",
0085 |         "axis": "all",
```

```
0086 |         "strength": 0.0033333333
0087 |     }
0088 | ]
0089 | },
0090 | {
0091 |     "key": "mouth_width",
0092 |     "label": "Mouth Width",
0093 |     "category": "Mouth",
0094 |     "group": "Mouth",
0095 |     "type": "scale",
0096 |     "axis": 0,
0097 |     "strength": 0.0033333333,
0098 |     "min": -100,
0099 |     "max": 100
0100 | }
0101 | ]
```

Мета роботи

Мета роботи: розробити програмний засіб для генерації та редагування 3D-моделей обличчя з можливістю зміни параметрів моделі через графічний інтерфейс.

Об'єкт дослідження: процес створення та редагування тривимірних моделей обличчя.

Предмет дослідження: методи програмної обробки 3D-сітки, зміни груп вершин та параметричного редагування форми обличчя.



Навчально-науковий інститут атомної та теплової енергетики
Кафедра інженерії програмного забезпечення в енергетиці

РОЗРОБКА ПРОГРАМНИХ ЗАСОБІВ ГЕНЕРАЦІЇ 3D-МОДЕЛЕЙ ОБЛИЧЧЯ

виконав: студент групи ТВ-21 Цвігун Богдан Володимирович

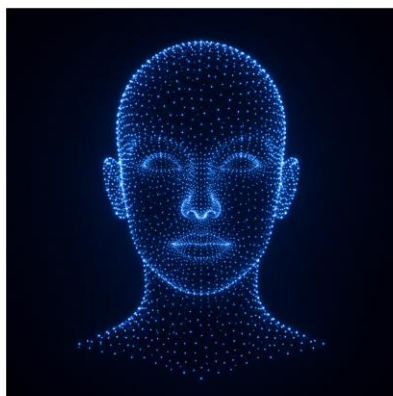
керівник: доцент кафедри ІПЗЕ Залевська Ольга Валеріївна, кандидат технічних наук,
доцент

2026

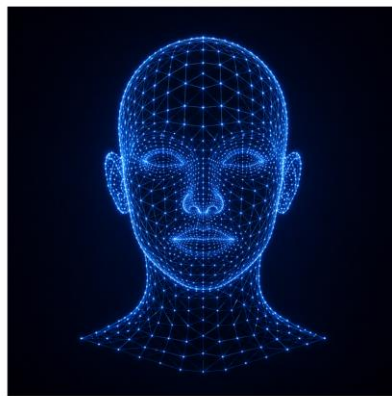
Актуальність теми



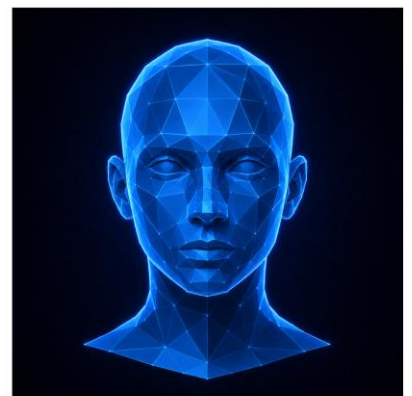
Аналіз предметної області



Точки у 3D просторі

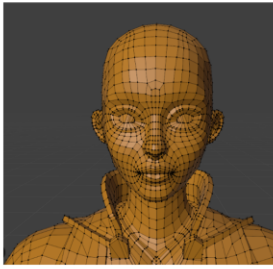


Ребра



Фігури(Полігони)

Наявні рішення



Ручне моделювання(Blender)

- + Створення моделі будь-якої якості та складності
- + Зміна моделей необмежена
- Великі затрати часу та ресурсів на створення моделі
- Дефіцит кваліфікованих кадрів



Готові плагіни(MetaHuman)

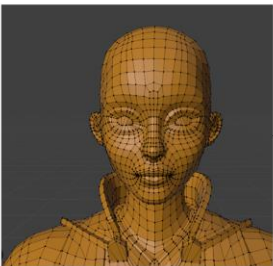
- + Швидкість створення
- Проблеми з експортом
- Діапазон редагування
- Складність у створенні великої кількості моделей
- Проблеми з вихідними форматами файлів



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

4

Завдання



Ручне моделювання

- Необмеженість редагування
- Результат у .glb



Гібридна система



Плагіни

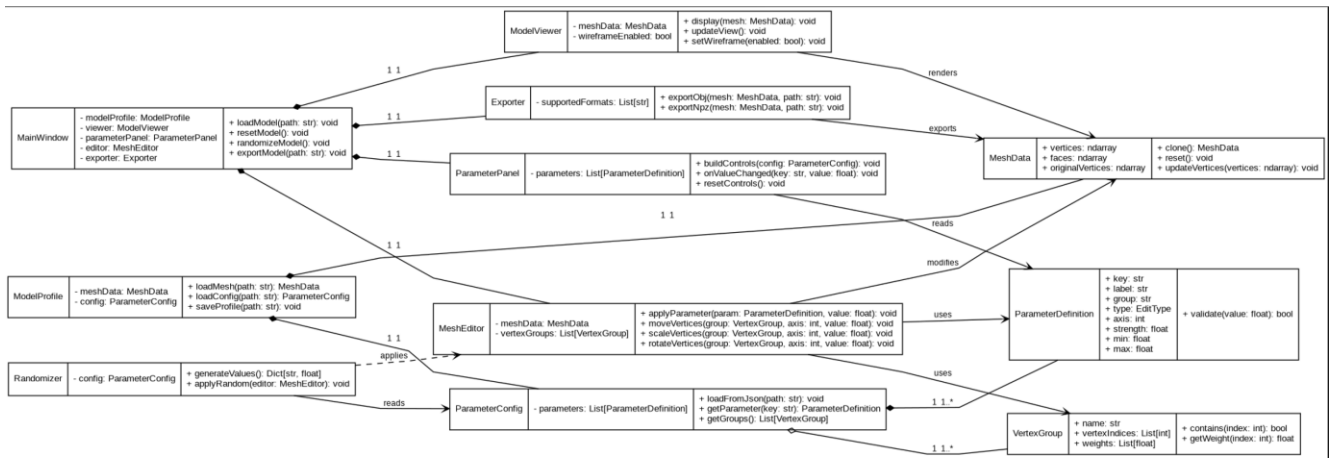
- Легкість використання
- Швидкість створення моделі



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

5

Діаграма класів

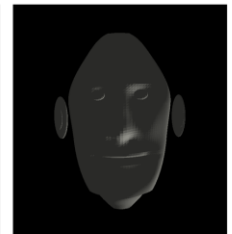
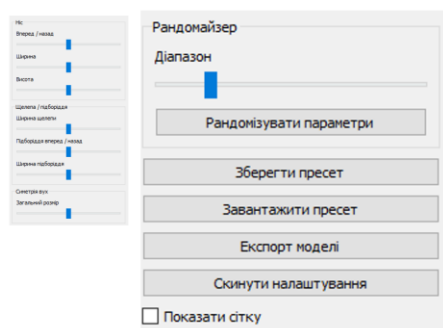
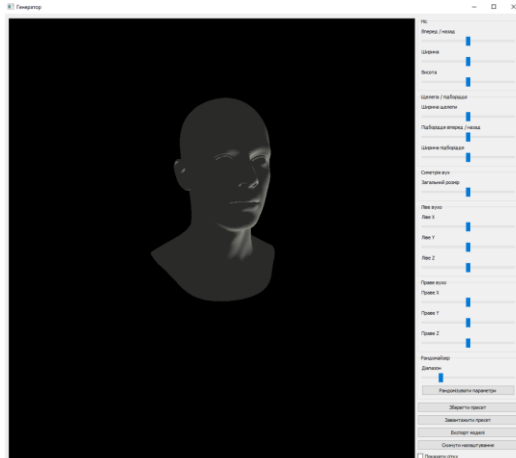


Національний технічний університет України
«Київський політехнічний інститут імені Ігоря Сікорського»

6

Реалізація системи

Інтерфейс користувача



"C:\Program Files\Blender Foundation\Blender 3.4\blender.exe" --background "base_head.blend" --python "script.py"



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

6

Тестування системи

Методи тестування

Для перевірки роботи системи було використано модульне, інтеграційне та системне тестування.

Результати тестування

Базова модель: **12 608 вершин, 24 820 полігонів**

Протестовано: **18 параметрів редагування**

Модульні перевірки: **35 тестів, 33 успішні**

Інтеграційні сценарії: **12 перевірок**

Середній час оновлення моделі: **до 1 с**

Критичних помилок під час тестування не виявлено



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

7

Висновки

Основні результати роботи

У результаті роботи було розроблено програмний застосунок для генерації 3D-моделі обличчя людини

Досягнення мети та виконання завдань

- + Необмеженість редагування
- + Результат у .glb
- + Легкість використання
- + Швидкість створення моделі



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

9



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

National Technical University of Ukraine
"Igor Sikorsky Kyiv Polytechnic Institute"