МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ

«ХАРКІВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ»

КАФЕДРА КОМП'ЮТЕРНОЇ МАТЕМАТИКИ І АНАЛІЗУ ДАНИХ

ЗВІТ

З ПЕРЕДДИПЛОМНОЇ ПРАКТИКИ

Виконала: Яресько К.О.  
ст. гр. КН-52а

Керівник практики: Галуза О.А.

доц. каф. КМАД

Харків — 2018

ВСТУП

Переддипломна практика проводилася в товаристві з обмеженою відповідальністю "Клауд Воркс". Практика проводилася в період з 5 лютого по 15 квітня 2018 року.

Керівником практики була поставлена задача: ознайомитись з методами спрощення тріангуляційної моделі 3D оболонок та розробити відповідне програмне забезпечення.

Для реалізації методів спрощення тріангуляційної моделі був використаний язик програмування С++. Інтерфейс програми було створено за допомогою язика програмування С# та технології WPF.

# ТРІАНГУЛЯЦІЙНА МОДЕЛЬ ТА МЕТОДИ ЇЇ СПРОЩЕННЯ

Існують два основні підходи до вирішення завдання спрощення тріангуляційних моделей: інтерактивний і глобальний. Ітеративний підхід дозволяє послідовно спростити модель шляхом застосування деякого локального оператора. Глобальний, в свою чергу, застосовується до вихідної моделі в цілому. Ітеративний підхід, як правило, використовується більш широко, так як з його допомогою зручніше контролювати результат, ступінь спрощення сітки і т.п. Наприклад, використовуючи такий алгоритм, користувач може встановлювати точну кількість граней результуючої моделі або ж вибирати в якості критерію зупинки певне порогове значення помилки. Виходячи з цього, ми будемо розглядати далі ітеративні алгоритми. Між собою їх можна розділити на кілька категорій, так як кожен з алгоритмів може бути використаний для своєї специфічної задачі. Однією з таких категорій є підтримування спрощення немноговидів.

## 1.1 Немноговиди

Тіла, що представляють собою многовиди, можна інакше описати як тіла із замкнутим об'ємом. Порушеннями умови многовиду є, наприклад, такі явища як: перетин двох поверхонь в одній точці, перетин двох поверхонь уздовж відкритої або замкнутої кривої, два замкнутих обсягу із загальною межею, ребром або вершиною, ребро, що виступає з точки на поверхні. Приклади многовидів і немноговидів представлені на рис. 1.1.

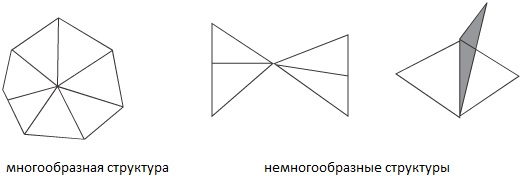


Рисунок 1.1 - Порівняння многовидів і немноговидів

Розглянемо докладніше відмінність між многовидами і немноговидами. У многовидів кожна точка на поверхні є двовимірною. Іншими словами, хоча поверхня існує в тривимірному просторі, з топологічної точки зору вона є плоскою, якщо розглядати досить малу її ділянку в околиці будь-якої заданої точки. У моделі, яка не є многовидом, околиця деякої точки на поверхні не повинна бути плоскою. Точка може бути перетином двох і більше топологічно плоских поверхонь або плоскій поверхні і одновимірної кривої.

## Алгоритми спрощення многовидів

Хоппе та ін. [1] запропонували алгоритм, мета якого полягає в тому, щоб отримати таку сітку, яка б найкращим чином відповідала множині точок, розташованій по поверхні, і при цьому містила якомога меншу кількість вершин. Така множина точок складається з початкових вершин, що становлять поверхню, а так само додаткових, які розміщуються на кордонах і рівномірно по всій поверхні. Для того, щоб досягти мети, необхідно знайти такий симпліціальний комплекс K і таку множину вершин V, що визначають сітку M = (K, V), які мінімізують наступну енергетичну функцію:

E(K, V) = Edist(K, V) + Erep(K) + Espring(K, V).

Розглянемо складові цієї функції докладніше. Edist(K, V) являє собою суму квадратів відстаней до сітки від точок X:

Варто відзначити, що спочатку точки X лежать на поверхні сітки, проте в ході ітерацій алгоритму і зміни топології, сітка може відхилятися від цих точок на деяку відстань.

Erep(K), в свою чергу, це штрафна складова, яка збільшується, якщо вершин стає більше, і описується наступною формулою:

Erep(K) = crepm,

де m - кількість вершин;

crep - параметр, який визначається користувачем і контролюючий компроміс між ступенем спрощення сітки і точністю геометричної відповідності.

відповідає за розтягнення ребер щодо їх первісного стану і визначається за формулою:

де k - коефіцієнт пружності, встановлений заздалегідь.

Скорочення сітки відбувається ітеративно, шляхом виконання допустимих операцій на її ребрах: видалення ребра, поворот ребра (зміна на ребро, що з'єднує дві протилежні вершини), додавання нового ребра шляхом розщеплення поточного (додавання вершини в середину ребра і з'єднання її з рештою вершинами всередині поточного трикутника ). Вибір допустимих дій проводиться за допомогою процесу оптимізації енергетичної функції. На кожному кроці видаляється елемент, видалення якого дає найменше збільшення енергетичної функції. Варто також відзначити, що при видаленні ребра, нова вершина поміщається або в одну з координат вершин, розташованих на кінцях цього ребра, або посередині.

Пізніше Хоппе [2] встановив, що насправді для ефективного спрощення сіток достатньо всього однієї операції: видалення ребра. Також пропонується модифікувати енергетичну функцію наступним чином:

E(M) = Edist(M) + Espring(M) + Escalar(M) + Edisc(M).

Edist(M) визначається аналогічних попередньому алгоритму чином.

Для визначення Espring(M) внесли деякі зміни. Пропонується не ставити коефіцієнт пружності як константу, а міняти його в результаті виконання завдання. Таким чином, Espring(M) постійно зменшується і адаптується при спрощення сітки.

Escalar(M) вимірює точність скалярних атрибутів сітки (наприклад, таким атрибутом може бути колір), а Edisc(M) - геометричну точність кривих розриву сітки. Введення в функцію таких складових замість Erep(K), дозволяє уникнути неоднозначного параметра crep, а також дозволяє вказувати явне кількість граней, які необхідно отримати в оптимізованої сітці. Також вони дають можливість зберегти не тільки геометрію сітки, а й її зовнішній вигляд.

З недоліків даного підходу можна виділити наступні. Розглянемо ситуацію видалення ребра. Як було сказано раніше, нову вершину пропонувалося помістити або в одну з координат вершин, що належать ребру, або посередині. Можна припустити, що це не найефективніший рішення, і можна було б замість цього розглянути функцію помилки, пов'язану з операцією стиснення, і спробувати мінімізувати її значення в порівнянні з простором можливих місць розміщення вершин. Також, не дивлячись на дуже хороші результати, які дає алгоритм, обчислення оптимальної цільової позиції для даного стиснення є нелінійної проблемою і на практиці досить неефективно, а також складно в реалізації. Крім того, формулювання такої функції помилки істотно обмежує її застосування щодо різноманіття сітки.

Іншим підходом до спрощення різноманіття сіток є реконструювання поверхні, яке включає в себе визначення нового набору вершин, що проектуються на поверхню вихідної сітки, а потім використання цих вершин для формування нових граней поверхні. Алгоритм реконструювання поверхні, представлений Турком [3], проектує новий набір вершин на поверхню сітки, як правило, за певним шаблоном, хоча кривизна поверхні може бути врахована. Точність цього підходу сильно залежить від методу, обраного для проектування нового набору вершин на поверхню.

Алгоритм спрощення, заснований на векселях, був представлений Хі і ін. [4]. Він сегментує модель в сітку на основі векселів, а потім застосовує фільтр нижніх частот для усунення векселем з найменшими перекриваються обсягами з вихідної сіткою. Потім алгоритм використовує алгоритм маршових кубів [5] для генерації нової сітки на основі векселів що були залишені. Через підходу фільтрації, заснованого на обсязі, вексельного спрощення погано працює на моделях з гострими краями і квадратними кутами.

Коен та ін. [6] представили інший підхід до спрощення різноманітною сітки з використанням двох зміщених копій кожної поверхні, званих спрощують огинають. Ці зміщення використовувалися для забезпечення того, щоб спрощена поверхню залишалася в обсязі, утвореному огинають. У той час як спрощують огинають можуть гарантувати високу ступінь точності, з огляду на відстань зсуву, що використовується для огинають, відстань також служить нижньою межею для алгоритму і запобігає радикальне спрощення сітки.

Таким чином, було запропоновано багато підходів щодо спрощення різноманітних моделей. Кожен підхід має свої сильні і слабкі сторони з точки зору точності спрощених сіток, здатності визначати ступінь помилки, часу виконання і складності реалізації.

## Алгоритми спрощення немноговидів

Незважаючи на хороший результат, який показують алгоритми, що працюють тільки з різноманітними моделями, існує досить велика кількість прикладних задач, в яких використовуються немноговиди. Розглянемо найбільш відомі алгоритми, які були розроблені в цій сфері.

## Vertex Decimation

Одним з перших алгоритмів спрощення сітки, здатних обробляти немноговиди, був представлений Шроудером та ін. [7]. Алгоритм проріджування працює, роблячи кілька проходів по всіх вершинах моделі. Під час кожного проходу для кожної вершини алгоритм визначає, чи можна видалити вершину без порушення топології локальної околиці граней. Якщо результуюча поверхню буде перебувати на заданому користувачем відстані від початкової сітки, алгоритм видаляє вершину і всі пов'язані з нею межі. Отримане отвір в сітці заповнюється шляхом локальної тріангуляції. Процес видалення вершин повторюється з можливим корегуванням критерію видалення, поки не буде досягнута умова виходу. Таким умовою, наприклад, може бути відсоткове зменшення вихідної сітки.

Представлений алгоритм децимації здатний обробляти немноговиди в разі, якщо він не буде видаляти немноговидні вершини під час ітерацій проріджування. Однак це правило накладає обмеження знизу для алгоритму проріджування, так як він не може створити низькополігональну модель з меншою кількістю вершин, ніж число немноговидних вершин. Крім того, визначена користувачем відстань істотно впливає як на нижню межу, так і на точність алгоритму. Також мінус такого підходу в тому, що спрощення застосовується рівномірно до всього набору даних, що призводить до розмиття даних при нерівномірній структурі сітки.

## Vertex Clustering

Іншим підходом до спрощення немноговидних сіток є кластеризація вершин. Алгоритм кластеризації вершин, представлений Россіньяком і Боррелем [8], привласнює значення важливості кожної вершині в залежності від розміру її суміжних граней і їх кривизни. Потім алгоритм застосовує сегментацію поверхні: модель поміщається в якийсь паралелепіпед, що складається з осередків, що представляє собою 3D сітку. Залежно від розміру осередку, в кожну клітинку може потрапити як одна, так і декілька вершин. У разі потрапляння кількох, розраховується, в якому місці цього осередку буде розташована нова точка, яка замінить собою старі. Після того, як в кожному осередку залишилося максимум по одній точці, тріангуляційна модель перераховується.

Мінус такого підходу в тому, що, не дивлячись на швидкість роботи і свою узагальненість, складно контролювати якість одержуваної моделі, і вона часто виявляється досить низькою. Також, досить проблематично поставити кількість бажаних граней, які повинні бути отримані на виході. Крім того, результат може залежати від положення моделі щодо паралелепіпеда.

## Hierarchical dynamic simplification (HDS)

Іншим підходом до спрощення, заснованим на кластеризації вершин, є алгоритм ієрархічного динамічного спрощення, авторами якого є Люебке і Еріксон [9]. Алгоритм HDS представляє всю сітку як дерево вершин, яка представляє собою ієрархію, що складається з кластерів вершин. Вузли дерева можуть бути згорнуті в їх батьківські вузли, щоб зменшити загальну кількість граней або розгорнуті в дочірні вузли для повторного отримання вихідної сітки. Оскільки дерево вершин не вимагає вершинної зв'язності, алгоритм HDS підтримує спрощення немноговидів. Однак точність результату, що отримується за допомогою даного алгоритму, як правило виявляється нижчою в порівнянні з результатами інших підходів.

## Quadric Error Metrics (QEM)

Можливо, найбільш поширеним підходом до спрощення немноговидних сіток є використання квадратичної метрики помилки (QEM), вперше представленої Гарланд і Хекбертом [10]. QEM вершини являє собою матрицю розміром 4 × 4, яка визначається як сума квадратів відстаней від вершини до площин суміжних граней. Коли вершина зливається з іншою вершиною в межах заданого користувачем порога відстані, значення помилки може бути обчислено як сума матриць QEM об'єднаних вершин, яка стає QEM нової вершини. При об'єднанні вершин алгоритм зберігає відсортовану чергу пріоритетів всіх пар вершин-кандидатів на основі їх об'єднаного QEM. Алгоритм видаляє пару вершин з найменшою помилкою з верхньої частини черги, об'єднує вершини і потім оновлює значення помилок тих пар вершин, які були з'єднані з даною. Позиція нової вершини обчислюється на основі рішення задачі мінімізації функції помилки.

Даний підхід дозволяє отримати сітки зі збереженням досить високої точності, навіть при радикальній ступеня спрощення. Його відмінною рисою є те, що він дозволяє спрощувати незв'язні моделі, а також дозволяє працювати з немноговидами. Крім того, як вже було сказано, в даному алгоритмі обчислюється найбільш оптимальне положення нових вершин, що не було враховано більшістю інших авторів.

Одним з обмежень початкового алгоритму QEM є те, що він має тенденцію до великих відхилень на граничних ребрах через зменшення числа суміжних граней, що призводить до виникнення великої кількості пропусків на кордонах (рис. 1.2).

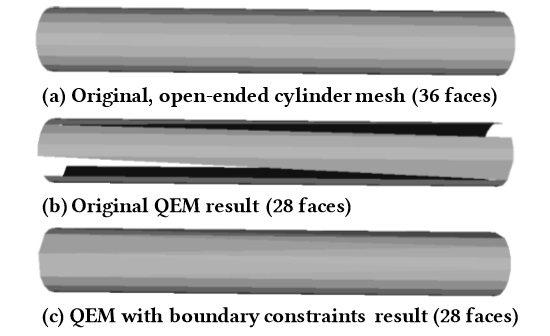


Рисунок 1.2 – Приклад нездатності алгоритму QEM зберегти важливі граничні ребра

Щоб вирішити цю проблему і забезпечити збереження кордонів, Гарланд і Хекберт [10] визначили площину граничного обмеження, що проходить через кожне граничне ребро. Для кожної такої площини пропонується обчислити матрицю QEM, яка множиться на досить великий постійний ваговий коефіцієнт, і додати її до матриць QEM ребрових вершин. Різні дослідники також запропонували зважувати площину граничного обмеження як квадрат довжини кромки. Проте, всі ці підходи призводять до того, що граничні ребра стають більш пріоритетними, ніж всі інші, що призводить в результаті до спрощених сіток з отворами на їх поверхнях (рис. 1.3).

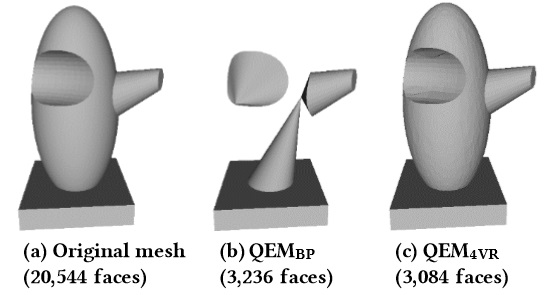


Рисунок 1.3 – Приклад нездатності алгоритму QEMBP зберігати необхідні для поверхні ребра

Іншим обмеженням вихідного алгоритму QEM є те, що він не враховує властивості поверхні, такі як нормалі, кольору і текстур координати. Гарланд і Хекберт [11] узагальнили оригінальний алгоритм QEM, щоб також обробляти властивості поверхні. Для обробки нормалей або квітів вихідна матриця помилок 4 x 4 може бути розширена до матриці 6 x 6, яка містить помилку нормалі вершини (abc) або її кольору (rgb). Аналогічно, матриця помилок 5 × 5 може використовуватися для фіксації помилки текстурних координат вершини (st). Хоппе [12] також представив узагальнену QEM, яка займає менше місця для зберігання, ніж у Гарланда і Хекберта [11]. З моменту впровадження QEM багато дослідники вивчили, як використовувати його для різних додатків. Однак більшість варіацій QEM підходять тільки для конкретних типів сіток (наприклад, для різноманітних сіток з межами, різноманітних сіток без кордонів, немноговидних сіток і т. Д.), А при використанні для інших типів сіток дають менш бажані результати.

Бахірат і ін. [12] запропонували алгоритм, спеціально розроблений для віртуальної реальності (VR). Даний алгоритм дозволяє якісно обробляти сітки з великою кількістю граничних областей, а також підтримує можливість спрощення немноговидних поверхонь, що є ключовою вимогою при роботі з VR. Крім того, при розробці алгоритму були усунені недоліки оцінки квадратичної помилки (QEM). Також, розроблений алгоритм QEM4VR дозволяє зберегти ключові властивості поверхні, такі як нормалі, текстурні координати, кольори і матеріали.

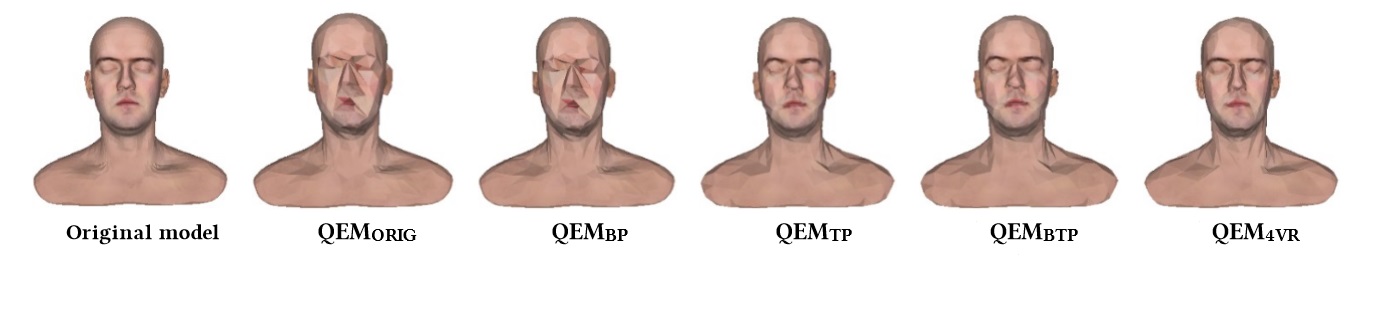
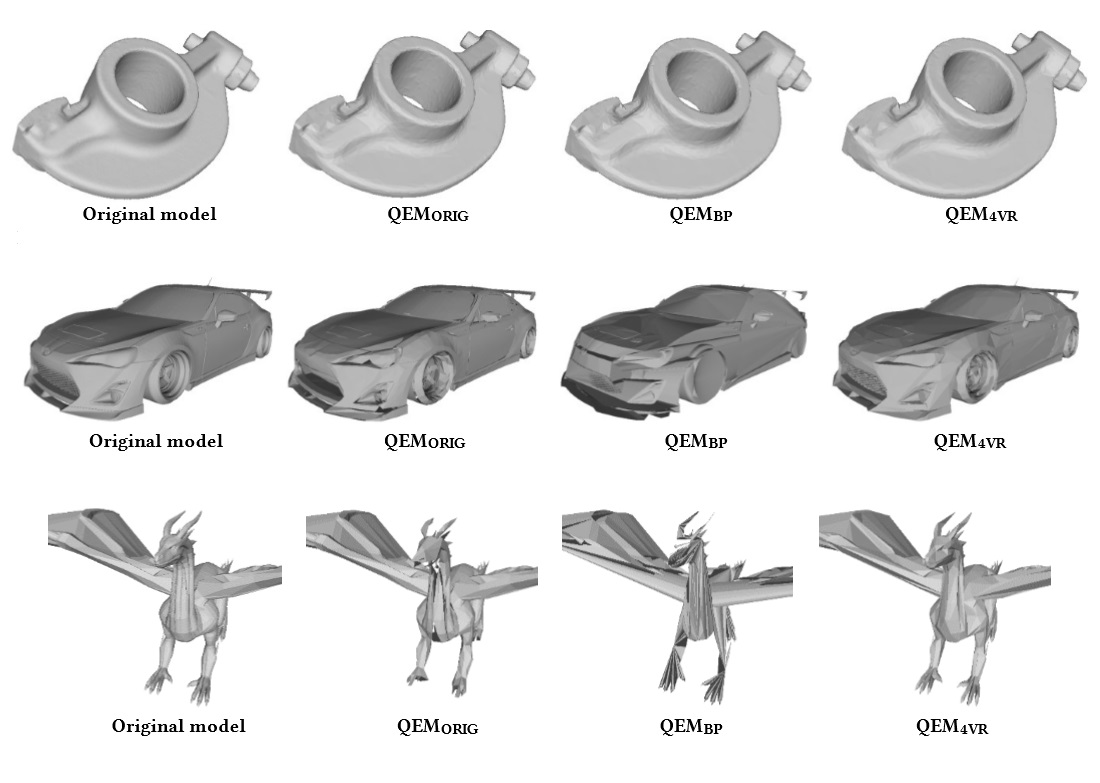
Розглянемо докладніше підхід збереження кордонів на основі кривизни. Спочатку визначають граничні ребра при ініціалізації. Для різноманітних і немноговидних поверхонь, якщо ребро належить двом або більшій кількості трикутників, воно вважається внутрішнім. Якщо ребро належить тільки одному трикутнику, воно розглядається як граничне, а відповідні кінцеві точки розглядаються як граничні вершини. Для початку розглянемо випадок різноманітних поверхонь. Кожна гранична вершина буде мати рівно дві сусідні граничні вершини. Розглянемо граничну вершину 𝑣1 і її сусідні граничні вершини 𝑣2 і 𝑣3. Обчислимо кривизну кордону в вершині 𝑣1 як:

де 𝑘 - кривизна граничної кривої в v\_1.

Для нменоговидних поверхонь припущення про точність наявності двох сусідніх граничних вершин може бути неправильним. Отже, кривизна, розрахована з урахуванням будь-випадкової пари сусідніх граничних вершин, призведе до неправильного судження про вершині. Щоб обробляти такі випадки, пропонується відзначати немноговидні граничні вершини як складні вершини і не спрощувати їх, привласнюючи їм великі ваги. Аналогічно оригінальним алгоритмом QEM, пропонується обчислити площину граничного обмеження і відповідну квадріку аналогічним чином. Обчислення квадрік для площин граничних обмежень дозволяє зберігати кордону без зміни функціональності вихідного алгоритму. Потім слід додати квадріку площини граничних обмежень до обох кінців граничного краю. Замість того, щоб множити її на постійний ваговий коефіцієнт, слід множити квадріку на значення кривизни в кожній кінцевій точці незалежно, а потім додавати її до початкових квадрік відповідних вершин. Ця схема зважування на основі кривизни для збереження кордонів привласнює більшу вагу граничним вершин з високою кривизною. Граничним вершин, які мають меншу кривизну, присвоюється меншу вагу. Це дозволяє уникнути зміщення алгоритму в сторону граничних вершин.

Іншим ключовим внеском алгоритму QEM4VR є його здатність обробляти кілька властивостей поверхні, такі як нормалі, текстурні координати, кольори і матеріали. Деякі попередні варіанти QEM [11, 12] також дозволяли зберігати властивості поверхні, однак ці підходи розглядали тільки той випадок, коли вся модель порівнюється з одним текстурованим матеріалом, так що між вершинами і координатами текстури існує взаємно однозначна відповідність. Але на практиці кілька текстурних матеріалів можуть бути пов'язані з однією 3D-моделлю. Крім того, декільком значенням з однієї текстури також може бути присвоєна задана вершина. Отже, модель може складатися з вершин з декількома наборами текстурних координат. Для обробки цих випадків було запропоновано створити нову структуру даних для кожної вершини, яка містить в собі інформацію про сусідніх гранях і відповідних текстурних координатах, пов'язаних з даною вершиною. Ці структури оновлюються під час видалення ребер. Також було розширено підхід збереження кордонів, щоб зберігати ребра, які утворюють кордону між декількома текстурами. Ребра, пов'язані з декількома текстурами, являють собою матеріальні кордону. Якщо модель має підмножина граней, які не мають інформації про властивості поверхні, їх значення встановлюються рівними нулю. Це гарантує, що порядок видалення відповідних ребер грунтується виключно на геометричній помилку, а не на властивостях поверхні, яких немає. Щоб визначити межі матеріалу, використовується вищезгадана структура, яка містить список граней в околиці. Якщо будь-які дві грані в цьому списку мають різний матеріальний індекс, можна вважати, що даній вершині присвоєні два різних матеріалу. Отже, її можна ідентифікувати як матеріальну граничну вершину. Крім того, якщо будь-яка вершина пов'язана з декількома текстурними значеннями, то вона вважається критичною вершиною. Потім застосовується визначений користувачем ваговий коефіцієнт 𝑊𝑡 до квадрик, пов'язаних з вершинами на матеріальних кордонах. Таким чином, матеріальні кордону залишаються незмінними, а матеріальні показники належним чином зберігаються.

Далі розглянемо порівняння роботи алгоритмів QEM на прикладах.



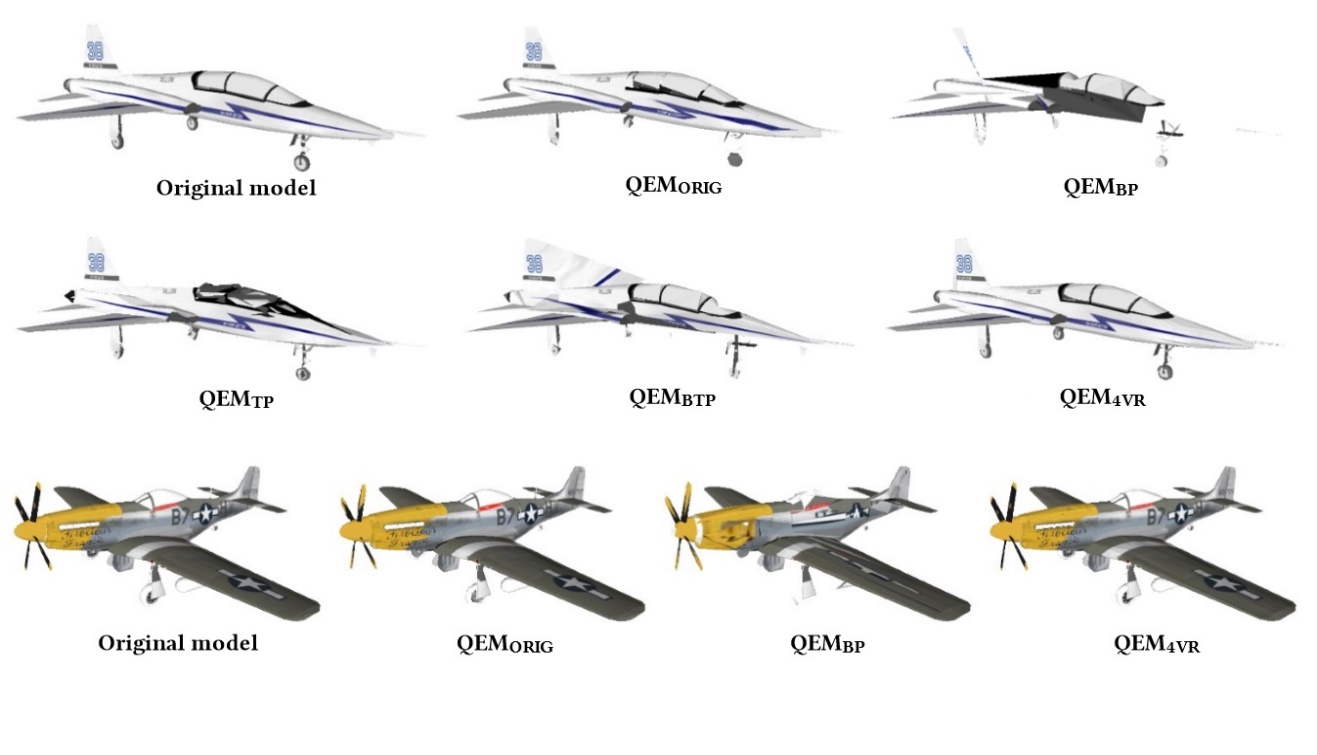


Рисунок 1.4 – Порівняння результатів спрощення 3D моделей різними QEM алгоритмами

На рис. 1.4 представлені 3D моделі різних типів: деталь і машина - многовидні моделі з межами, дракон - немноговид, людина - многовидна модель з накладеною текстурою без використання кордонів, реактивний літак - текстурна многовидна модель з межами, літак - текстурная немноговидна модель з кордонами. Як ми можемо бачити, деякі попередні алгоритми дають непогані результати для спрощення моделей певних типів, однак в інших ситуаціях показують себе гірше. У свою чергу, алгоритм QEM4VR показує хороші результати для всіх типів представлених моделей.

ВИСНОВКИ

Переддипломна практика була направлена на ознайомлення з різновидами тріангуляційних поверхонь, з методами їх спрощення та на реалізацію найбільш ефективного алгоритму.

В ході проходження практики були виконані наступні завдання:

1. ознайомлення з тріангуляційними поверхнями та їх різновидами;
2. ознайомлення з методами спрощення тріангуляційних поверхонь, їх аналіз та порівняння між собою;
3. реалізація найбільш ефективного методу спрощення тріангуляційних поверхонь.

Завдання, що були поставлені керівником практики, виконані в повному обсязі.

# СПИСОК ЛІТЕРАТУРИ

1. H. Hoppe, T. DeRose, T. Duchamp, J. McDonald, and W. Stuetzle, “Mesh optimization,” in ACM Annual Conference on Computer Graphics and Interactive Techniques (SIGGRAPH), 1993, pp. 19-26.
2. H. Hoppe, “Progressive meshes,” in ACM Annual Conference on Computer Graphics and Interactive Techniques (SIGGRAPH), pp. 99-108, 1996.
3. G. Turk, “Re-tiling polygonal surfaces,” ACM SIGGRAPH Computer Graphics, vol. 26, no. 2, pp. 55-64, 1992.
4. T. He, L. Hong, A. Kaufman, A. Varshney, and S. Wang, “Voxel based object simplification,” in IEEE Conference on Visualization, 1995, pp. 296-303.
5. W. E. Lorensen, and H. E. Cline, “Marching cubes: A high resolution 3D surface construction algorithm,” ACM SiGGRAPH Computer Graphics, vol. 21, no. 4, pp. 163-169, 1987.
6. J. Cohen, A. Varshney, D. Manocha, G. Turk, H. Weber, P. Agarwal, F. Brooks, and W. Wright, “Simplification envelopes,” in ACM Annual Conference on Computer Graphics and Interactive Techniques (SIGGRAPH), 1996, pp. 119-128.
7. W. J. Schroeder, J. A. Zarge, and W. E. Lorensen, “Decimation of triangle meshes,” in ACM SIGGRAPH Computer Graphics, vol. 26, no. 2, pp. 65-70, 1992.
8. J. Rossignac, and P. Borrel, “Multi-resolution 3D approximations for rendering complex scenes,” in Modeling in Computer Graphics, 1993, pp. 455-465.
9. D. Luebke, and C. Erikson, “View-dependent simplification of arbitrary polygonal environments,” in ACM Annual Conference on Computer Graphics and Interactive Techniques (SIGGRAPH), 1997, pp. 199-208.
10. M. Garland, and P. S. Heckbert, “Surface simplification using quadric error metrics,” in ACM Annual Conference on Computer Graphics and Interactive Techniques (SIGGRAPH), 1997, pp. 209-216.
11. M. Garland, and P. S. Heckbert, “Simplifying surfaces with color and texture using quadric error metrics,” in IEEE Conference on Visualization, 1998, pp. 263-269.
12. H. Hoppe, “New quadric metric for simplifiying meshes with appearance attributes,” in IEEE Visualization, 1999, pp. 59-66.

K. Bahirat, C. Lai, R. P. McMahan, and B. Prabhakaran, “A Boundary and Texture Preserving Mesh Simplification Algorithm for Virtual Reality”, in ACM Multimedia Systems Conference on Multimedia Systems, 2017, pp. 50-61.