АЛГОРИТМЫ УПРОЩЕНИЯ ТРИАНГУЛЯЦИОННЫХ ПОВЕРХНОСТЕЙ

Дипломная работа

Томск – 2 0 0 2

Реферат

Отчет на 46 стр., 8 рисунков, 25 таблиц, 26 источников, 6 приложений

ТРИАНГУЛЯЦИЯ, ТРИАНГУЛИРОВАНИЕ ПОВЕРХНОСТЕЙ, УПРОЩЕНИЕ, ОШИБКА АППРОКСИМАЦИИ, ЛОКАЛЬНЫЙ КРИТЕРИЙ ОПТИМАЛЬНОСТИ, ГИС ГрафИн, СВП DELPHI.

(1) Объект исследования – триангуляция поверхностей
(2) Цель исследования – создание эффективных алгоритмов упрощения триангуляции.
(3) Метод исследования – комплексная методика исследований, которая включает использование методов компьютерной графики и вычис-лительной геометрии, а также методов проектирования программного обеспечения.
(4) Реализованы и исследованы алгоритмы упрощения триангуляции удалением вершин, коллапсом ребер, коллапсом треугольников и алгоритм дробления вставкой в суперструктуру.

Оглавление

Введение

Треугольники – самый популярный примитив отображения графической информации. Они поддерживаются всеми графическими библиотеками и аппаратными системами. Таким образом, треугольные меши – пространственные поли - полигоны, очень распространены в компьютерной графике. Сложные модели, с сотнями тысяч граней легко вырабатываются инструментами САПР, ГИС, приложениями моделирования пространственных сцен, симуляторами и т.д.

Вместе с тем, визуализация пространственных сцен – трудоемкая задача. Это связано с непрерывным повышением требований к реалистичности и быстродействию, поэтому для визуализации сложных пространственных сцен применяются дорогостоящие специализированные графические рабочие станции. В связи с ростом вычислительной мощности персональных компьютеров стала возможна разработка на их базе сравнительно дешевых интерактивных систем визуализации, работающих в реальном времени. В системах оснащенных графическим процессором центральный процессор практически не занят вопросами отображения. Образ экрана формирует графический адаптер.

С появлением новых графических процессоров задача повышения эффективности интерактивных систем визуализации пространственных сцен не потеряла своей актуальности. При всех преимуществах применения графических процессоров, для воспроизведения на экранах ПК фотореалистичных изображений в реальном времени, необходимо искать новые методы визуализации. При этом надо учитывать, что полноразмерная модель не нужна для генерации каждого фрейма интерактивной визуализации. Это привело к разработкам методов контролируемого упрощения поверхностных мешей. Эти методы являются базисом для построения уровней детализации поверхности [9,12,17,25,26].

Ниже представлены некоторые методы контролируемого упрощения мешей:

1. Слияние компланарных граней. Компланарные или почти компланарные грани, найденные в меше, сливаются в один полигон, который затем ретриангулируется меньшим количеством граней
2. Контролируемое удаление вершин, ребер, граней. Эти методы итеративно удаляют соответствующие объекты меша, выбранные по некому локальному геометрическому критерию оптимальности и инцидентные с ним грани. Область удаленных треугольников ретриангулируется. Методы удаления вершин и граней могут быть адаптированы до методов коллапса вершин и граней в точку.
3. Перестроение. Вставляются новые вершины в случайные места меша, и затем перемещаются в места максимальной кривизны. Затем вершины исходного меша итеративно удаляются.
4. Кластеризация вершин. Вершины группируются в кластеры, для каждого кластера рассчитывается новая представительная вершина.
5. Упрощение через промежуточное иерархическое представление. Представление промежуточного дерева октантов [8] может быть адаптировано до автоматической генерации упрощенных представлений, потому что дерево октантов может быть очищено на различных уровнях и конвертировано в граничное (упрощенное) представление. Промежуточное иерархическое представление элементов объемного изображения (вокселей), построенное с использованием техники обработки сигналов для контролируемого удаления высокочастотных деталей, вместе с адаптивной подгонкой поверхности представлены в [13, 14].

Из приближения триангуляции Делоне [14] и последовательности удаленных из исходной триангуляции точек можно получить полную иерархию промежуточных триангуляций Делоне простой вставкой точек. При этом топология и иерархия степеней детализации косвенно получаются из триангу-

ляции Делоне, и нет необходимости хранить их прямо [24]. Поэтому перспективным представляется методы удаления вершин и коллапса ребер в триангуляции Делоне как базового метода для построения модели множественного разрешения.

Важным моментом методов удаления является локальный геометрический критерий оптимальности. Ниже представлены некоторые критерии:

1. Ошибка приближения точек. Два кусочно-линейных объекта M_i и M_j являются ε-аппроксимацией друг друга, если каждая точка M_i находится не дальше, чем ε, от некоторой точки M_j и наоборот.
2. Разность угла. Разность инцидентных углов и развернутого угла. Применяется в основном в методах, сохраняющих форму меша. Также существует множество производных критериев [17].
3. Классификация критических точек на основе анализа Гессиана [11]. Применяется в основном в методах, сохраняющих топологию.

Для оценки качества полученной при упрощении триангуляции применяют глобальные геометрические критерии оптимальности:

1. Ошибка приближения основанная на ε-аппроксимации.
2. Ошибка приближения основанная на среднеквадратичном отклонении.
3. Объем разности между исходной и упрощенной триангуляцией.

Наиболее качественным для визуализации является второй критерий. Он и был выбран как глобальный критерий оптимальности. Соответственно были подобраны локальные критерии оптимальности для удаления точки, коллапса ребра и треугольника.

1. Основные понятия и структуры данных для моделирования рельефа

Определение 1. Назовем рельефом скалярное поле над пространством R^2. Аппроксимацию этого поля кусочно-линейной функцией $f(\vec{x}): R^2 \to R$ назовем поверхностью.

Поверхности принято задавать не функцией, а областями линейности, чаще всего выпуклыми, со значениями функции в крайних точках. Это приводит к представлению поверхности в виде набора полигонов.

Практически поверхности, аппроксимирующие реальные рельефы, строят по т.н. опорным точкам - реальным данным, представляющим некоторые измерянные значения скалярного поля. Множество опорных точек не задает единственным образом области линейности поверхности. Задача получения областей линейности по опорным точкам решается задачей триангуляции.

1.1. Триангуляция Делоне

Впервые задача построения триангуляции Делоне была поставлена в 1934 г. в работе советского математика Б.Н.Делоне. Ее трудоёмкость составляет $O(N \log N)$. Существуют алгоритмы, достигающие этой оценки в среднем и худшем случаях. Кроме того, известны алгоритмы, позволяющие в ряде случаев достичь в среднем $O(N)$.

Введём несколько определений [1, 4]:

Определение 2. Триангуляцией называется планарный граф, все внутренние области которого являются треугольникам.

Определение 3. Выпуклой триангуляцией называется такая триангуляция, для которой минимальный многоугольник, охватывающий все треугольники, будет выпуклым. Триангуляция, не являющаяся выпуклой, называется невыпуклой.

Определение 4. Задачей построения триангуляции по заданному набору двумерных точек называется задача соединения заданных точек не пересекающимися отрезками так, чтобы образовалась триангуляция.

Задача построения триангуляции по исходному набору точек является неоднозначной, поэтому возникает вопрос, какая из двух различных триангуляций лучше?

Определение 5. Триангуляция называется оптимальной, если сумма длин всех рёбер минимальна среди всех возможных триангуляций, построенных на тех же исходных точках.

В [3, 4] показано, что задача построения такой триангуляции является NP -полной. Поэтому для большинства реальных задач существующие алгоритмы построения оптимальной триангуляции неприемлемы ввиду слишком высокой трудоёмкости. При необходимости на практике применяют приближенные алгоритмы [3].

Определение 6. Говорят, что триангуляция удовлетворяет условию Делоне, если внутрь окружности, описанной вокруг любого построенного треугольника, не попадает н одна из заданных точек триангуляции.

Определение 7. Триангуляция называется триангуляцией Делоне, если она является выпуклой и удовлетворяет условию Делоне, т.е. условие Делоне выполняется для любого треугольника триангуляции.

Определение 8. Говорят, что пара соседних треугольников триангуляции удовлетворяют условию Делоне, если этому условию удовлетворяет триангуляция, составленная только из этих двух треугольников.

Определение 9. Говорят, что треугольник триангуляции удовлетворяет условию Делоне, если этому условию удовлетворяет триангуляция, составленная только из этого треугольника и трёх его соседей (если они существуют).

Многие алгоритмы построения триангуляции Делоне используют следующую теорему [2, 20]:

Теорема 1: Триангуляцию Делоне можно получить из любой другой триангуляции по той же системе точек, последовательно перестраивая пары соседних треугольников ΔABC и ΔBCD, не удовлетворяющих условию Делоне, в пары треугольников ΔABD и ΔADC (Рис. 1.1). Такая операция перестроения также часто называется флипом.

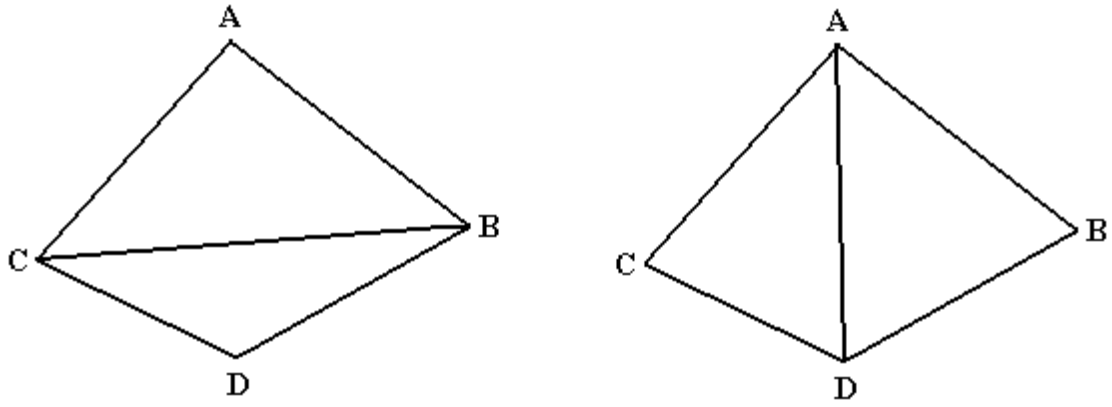

Рис. 1.1. Перестроение треугольников, не удовлетворяющих условию Делоне

Данная теорема позволяет строить триангуляцию Делоне последовательно, вначале строя некоторую триангуляцию, а потом последовательно улучшая её в смысле условия Делоне. При проверке условия Делоне для пар соседних треугольников можно использовать непосредственно определение, но иногда используются другие способы, основанные на следующих теоремах [2, 19, 21]:

Теорема 2. Триангуляция Делоне обладает максимальной суммой минимальных углов всех своих треугольников среди всех возможных триангуляций.

Теорема 3. Триангуляция Делоне обладает минимальной суммой радиусов окружностей, описанных около треугольников, среди всех возможных триангуляций.

В данных теоремах фигурирует некая суммарная характеристика всей триангуляции (сумма минимальных углов или сумма радиусов), оптимизируя которую, можно получить триангуляцию Делоне.

Как показано в [5] наименьших затрат по математическим операциям в среднем требует проверка условия Делоне с заранее вычисленной окружностью, но применять ее следует лишь в алгоритмах, в которых перестроение треугольников происходит редко. Поэтому была использована модифицированная проверка суммы противоположных углов.

1.2. Структуры данных для построения триангуляции

Основными геометрическими примитивами трехмерного пространства являются точка, отрезок, треугольник, тетраэдр. Для моделирования поверхностей и визуализации актуальными являются первые три, которые называют соответственно вершиной, ребром, гранью поверхности. Так как триангуляция была ориентирована на упрощение удалением и коллапсом всех примитивов, то был выбран способ явного задания вершин, ребер и треугольников.

При удалении вершины необходимо знать набор инцидентных вершине граней, собственно которые и будут удалены, а область, покрываемая удаленными треугольниками - ретриангулирована. Хранение всех инцидентных вершине треугольников является неэффективным по памяти и может быть выгодным лишь при упорядочивании списка обходом по часовой стрелке. Вместе с тем поддержание такого упорядочивания при вставке и удалении вершин является неэффективным по трудоемкости. Поэтому список инцидентных треугольников формируется динамически при удалении вершины. Для этого надо знать хотя бы один инцидентный вершине треугольник, остальные легко получить обходом по общим сторонам треугольников.

Автоматически получается область удаленных треугольников списком вершин, упорядоченных по обходу часовой стрелки. Эту область в дальнейшем будем называть "дырой". Для поддержания структуры модели поверхно-

сти вместе с дырой создается кромка, которая кроме списка вершин дыры содержит ссылку на ребра, образующие кромку.

Принимая во внимание приведенные выше соображения, предложена следующая структура. Вершина задается координатами, ссылкой на один из инцидентных треугольников. Ребро задается ссылками на начальную и конечную вершины, левый и правый треугольник. Треугольник задается ссылками на вершины в порядке обхода против часовой стрелки, ребра и треугольники, противоположные вершинам (Рис. 1.2).

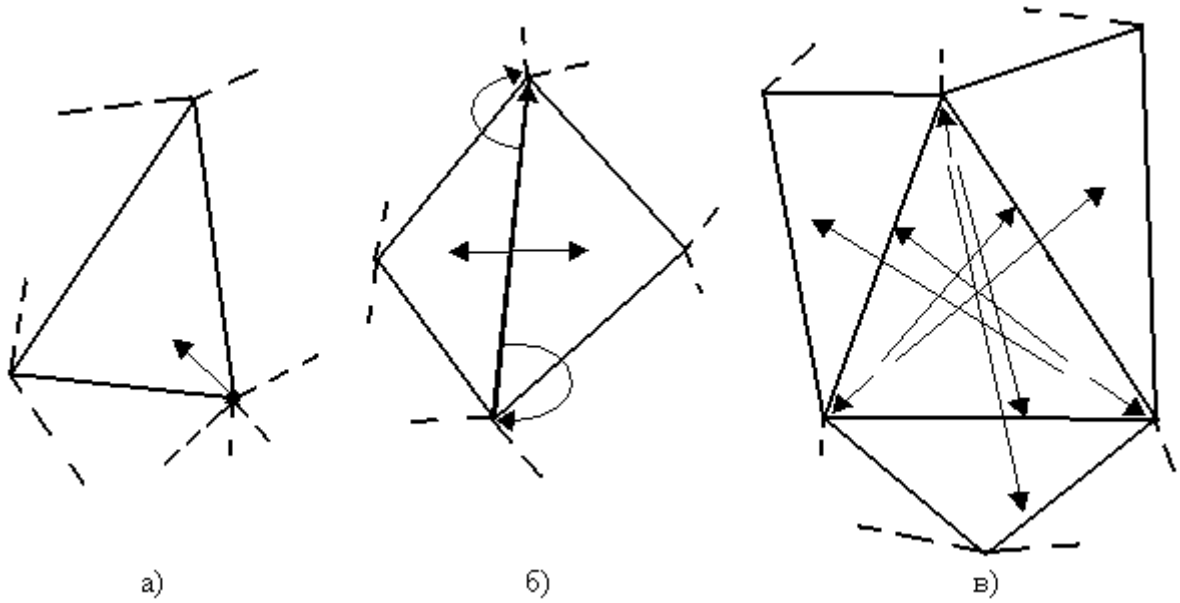

Рис. 1.2. Структура данных представления триангуляции (а – вершины, б – ребра, в – треугольника).

Таким образом при 8-байтовом представлении координат и 4-байтовом представлении указателей требуется: для задания вершины – 28 байт, ребра – 16 байт, треугольника – 36 байт, для триангуляции в среднем – $136N$, где N – количество опорных точек.

Задача коллапса ребра и треугольника сводится к задаче удаления двух и трех вершин соответственно с последующей ретриангуляцией и вставкой новой вершины. При этом, несмотря на увеличение количества обрабатываемых примитивов в среднем на 1,2 для коллапса ребра и на 1.5 для коллапса треугольника, значительного увеличения времени работы алгоритма не про-

изошло. Связано это, скорее всего с тем, что время, освобожденное уменьшением количества обрабатываемых примитивов, используется на дополнительные проверки и поддержание внутренних структур в корректном состоянии.

1.3. Базовый алгоритм удаления вершины

Алгоритм удаления вершины и последующая ретриангуляция полученной в результате удаления вершины *дыры* является основным шагом итеративных алгоритмов упрощения триангуляций.

Заметим, что область удаленных треугольников не всегда совпадает с дырой. Если удаляется вершина на границе триангуляции, то дыра получается отсечением области удаляемых треугольников выпуклой оболочкой неудаленных вершин (Рис. 1.3).

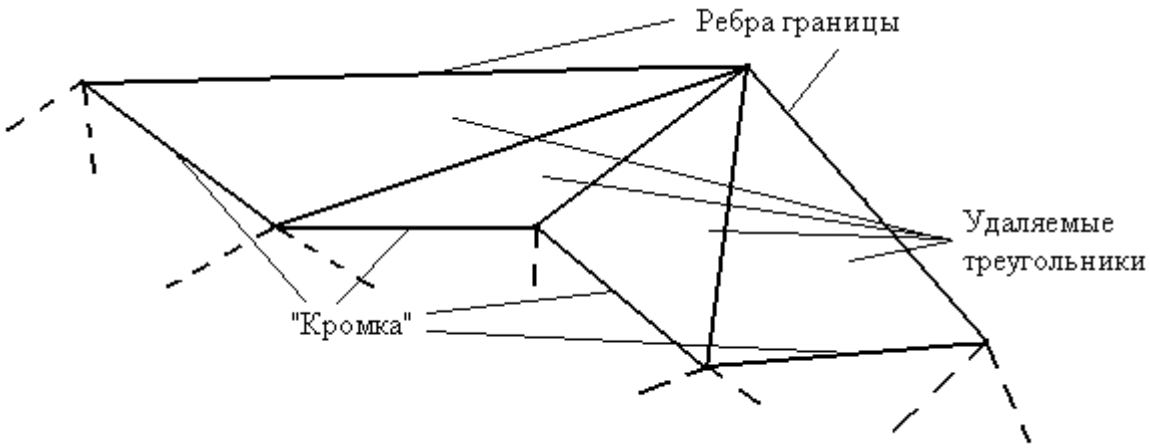

Рис. 1.3 Дыра при удалении вершины на границе триангуляции

Дыра получается последовательным обходом треугольников против часовой стрелки от первого инцидентного, ссылка на который хранится вместе с вершиной. При выходе за пределы триангуляции происходит достраивание дыры обходом по часовой стрелки от первого инцидентного треугольника. Описывается дыра списком вершин и ребер удаляемых треугольников, противоположных удаляемой вершине. Список организован так, что вершина

из списка является общей для двух соседних ребер и ребро состоит из двух соседних вершин. Ретриангуляция дыры может быть произведена различными методами [15]. Здесь используется метод заштриховки области треугольниками: для последовательных трех вершин проверяется условие поворота по часовой стрелке и, при выполнении условия, пересечения с другими возможными треугольниками. Если пересечений нет и вершины расположены по часовой стрелке, вставляется новый треугольник. Из списка удаляются вторая вершина и ее соседние ребра, на их место вставляется новое ребро, соединяющее первую и третью вершину триады, образовавшего треугольник.

Процесс заштриховки считается завершенным, если кромка состоит из одного ребра либо любая последовательная триада вершин кромки расположены против часовой стрелки (Рис. 1.4).

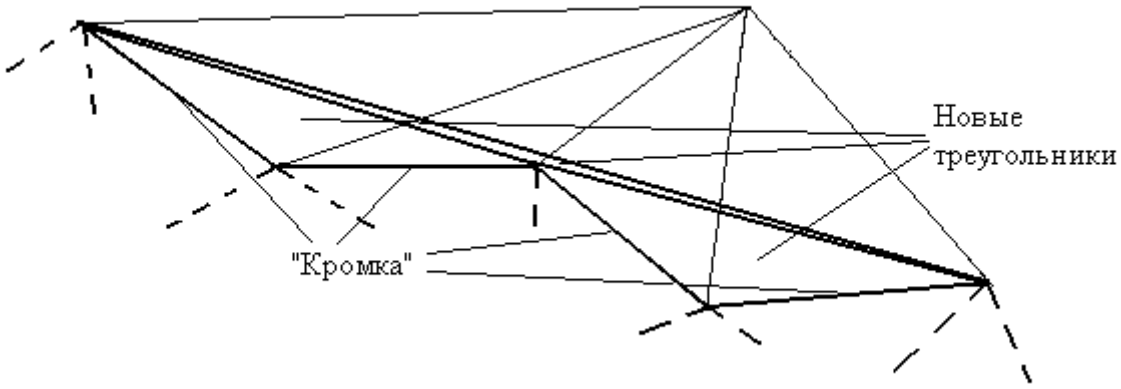

Рис. 1.4. Ретриангуляция "дыры" до проверок условия Делоне.

Трудоемкость алгоритма выделения дыры и ее ретриангуляции можно принять равной в среднем константе по следующим соображениям: количество инцидентных вершине треугольников в среднем равно шести, следовательно операция удаления вершины и ретриангуляция, зависящие от количества инцидентных вершине треугольников, не зависят от количества вершин в триангуляции и количества удаляемых вершин.

При покрытии дыры новыми треугольниками может быть нарушено условие Делоне для новых треугольников, поэтому необходимо после ретри-

ангуляции проверка условия Делоне для всех новых треугольников и, при необходимости, перестроение. Перестроенные треугольники вновь проверяются на условие Делоне. Таким образом, могут быть перестроены в худшем все треугольники. Однако в среднем происходит перестроение 3 треугольников на одно удаление.

Как было рассмотрено в [5], наиболее эффективным методом триангуляции Делоне является последовательная вставка опорных вершин с динамическим кэшем для направленного поиска треугольников. Использование данного метода для получения начальной триангуляции определяет тип получающихся дыр: вместе с невыпуклыми замкнутыми областями возможны незамкнутые кривые, которые, тем не менее, могут быть ретриангулированы до выпуклости всей триангуляции. Метод заштриховки эффективен для ретриангуляции "дыр" обоих типов.

2. Алгоритмы упрощения триангуляций

Существуют две стратегии алгоритмов упрощения. Алгоритмы стратегия «снизу-вверх» действуют следующим образом: на предварительном этапе рассчитывается потенциальная ошибка удаления/коллапса примитивов с образованием очереди, затем итеративно удаляют или коллапсируют примитивы с одновременным пересчетом потенциальных ошибок и перестроением очереди. Алгоритм работает до тех пор, пока не будет достигнуто требуемое разрешение.

Основной вклад в трудоемкость алгоритмов упрощения, использующих данную стратегию, является перестроение очереди, т.к. удаление/коллапс примитивов (1.3.) и пересчет потенциальной ошибки в силу локальности критерия можно принять за константу, а пересчет очереди зависит от размера очереди и ее представления. Для очереди важнейшими критериями качества является доступность к элементу очереди с минимальным значением за константу и минимизация времени на перестроение очереди при внесении или удалении элемента. Наиболее отвечающим этим требованиям является представление очереди в виде пирамиды.

Стратегия «сверху-вниз» основана на дроблении некой аппроксимации рельефа, состоящей из одного и более треугольников, добавляя в нее точки до тех пор, пока не будет достигнуто требуемое разрешение.

Основной вклад в трудоемкость алгоритмов упрощения, использующих данную стратегию, является перераспределение точек по треугольникам и перестроение очереди на вставку.

2.1. Локальный критерий оптимальности

Локальный критерий оптимальности должен обладать по определению следующими свойствами: быть локальным, аппроксимировать некий глобальный критерий, т.е. давать некую потенциальную ошибку изменения гло-

бального критерия. Класс значений, который может принимать критерий, должен быть шкалой сравнений.

2.1.1. Локальный критерий оптимальности удаления точки

Очевидно, что для обеспечения минимальной ошибки в первую очередь из триангуляции Делоне следует удалять вершины, которые лежат в одной плоскости со своими смежными треугольниками. Для определения стоимости удаления каждой вершины v_i предлагается использовать среднюю плоскость P_i^{avg}, проведенную через точку v_i. Нормаль n_i^{avg} средней плоскости P_i^{avg} вычисляется при инициализации как нормированная сумма нормалей n_j смежных к вершине v_i треугольников t_j в полном разрешении:

$$n_i^{avg} = \sum_{j=0}^{m-1} n_j \Big/ \left\| \sum_{j=0}^{m-1} n_j \right\|, \qquad (1)$$

где m - количество смежных треугольников t_j. Для новых точек, которые образуются при коллапсе ребер и треугольников, нормаль средней плоскости вычисляется как нормированная сумма нормалей точек, составляющих ребро и треугольник. Скаляр d_i^{avg} средней плоскости P_i^{avg} вычисляется как скалярное произведение нормали n_i^{avg} на вектор v_i. Уравнение средней плоскости P_i^{avg} проходящей через точку v_i:

$$P_i^{avg} = (n_i^{avg}, -(n_i^{avg} v_i)), \qquad (2)$$

Стоимость удаления (СУ) любой вершины v_i определена как средняя сумма квадратов расстояний от смежных вершине v_j текущей триангуляции до средней плоскости P_i^{avg}:

$$СУ(v_i) = \left[\sum_{j=0}^{m-1} (n_i^{avg} v_j + d_i^{avg})^2 \right] \Big/ m \qquad (3)$$

Для вычисления стоимости удаления каждой вершины v_i, как минимум, необходимо хранить нормаль полного разрешения n_i^{avg}. Скаляр d_i^{avg} можно либо хранить в памяти, либо пересчитывать каждый раз, когда нужно считать стоимость удаления точки v_i.

Внешний контур рельефа желательно сохранить, поэтому предложен способ искусственного повышения стоимости удаления граничных вершин. Для граничных вершин средняя нормаль корректируется. Для этого вначале вычисляется граничная вершина и затем, используя информацию о смежных треугольниках, находятся все граничные вершины v_j^g, j=0..N-1, N - число граничных вершин. Затем, для каждой граничной вершины v_j^g к её вычисленной нормали n_j^{avg} добавляется корректирующий вектор. Полученная сумма снова нормируется:

$$n_j^{avg} = \left\| n_j^{avg} + BW * \left\| n_j^{avg} \times (v_{j+1}^g - v_{j-1}^g) \right\| \right\|, \qquad (4)$$

Такой способ коррекции нормалей граничных вершин позволяет повысить стоимость удаления граничных вершин по сравнения с обычными вершинами. Степень влияния коррекции нормали на ранги вершин задаётся весовым коэффициентом защиты границ BW. Параметр BW является параметром тонкой настройки алгоритма и по умолчанию ему прописывается оптимальное значение, выбранное по нескольким экспериментам с типовыми наборами точек.

2.1.2. Локальный критерий оптимальности коллапса ребер

Для коллапса ребер неприменим описанный выше критерий, т.к. появляется новая вершина, наличие которой может изменить геометрию триангуляции так, что удаленные вершины будут находиться на гораздо меньшем расстоянии от новых треугольников, чем средние плоскости удаленных вершин от смежных им вершин.

Поэтому предложен следующий локальный критерий оптимальности: СУ ребра, соединяющее вершины v_0 и v_1

$$СУ(r(v_0, v_1)) = (n_0^{avg} v_1 + d_0^{avg})^2 + (n_1^{avg} v_0 + d_1^{avg})^2, \quad (5)$$

который является приближенной суммой квадратов расстояний от удаленных вершин до триангуляции, полученной в результате удаления ребра. Для уменьшения количества вычислений в качестве приближения триангуляции

используется плоскость, проходящая не через среднюю точку ребра, а через противоположную. При этом расстояние увеличивается в два раза, следовательно, СУ уменьшают в 4 раза. Но так как значения критерия должны быть шкалой порядков, то коэффициент 4 можно опустить.

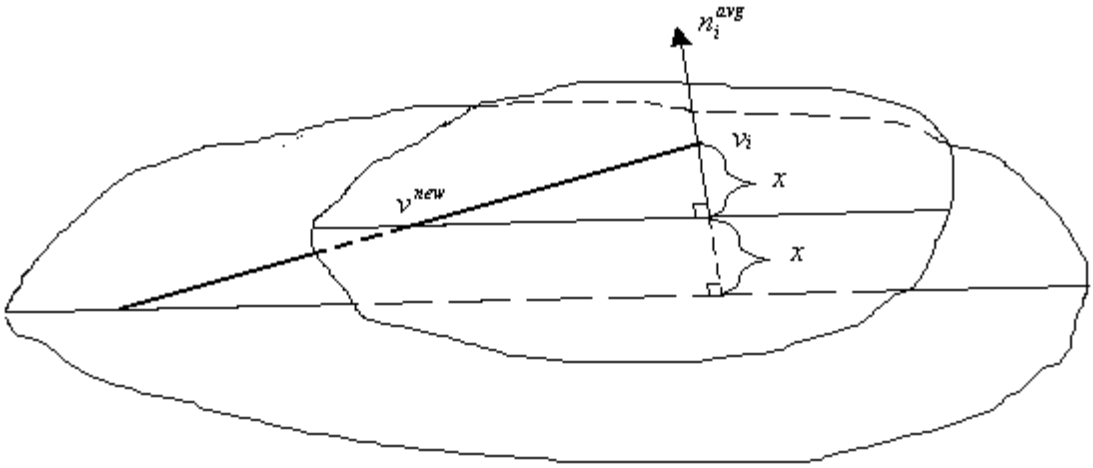

Рис. 2.1. СУ ребра: расстояние до плоскостей, проходящих через середину и вторую вершину ребра.

2.1.3. Локальный критерий оптимальности коллапса треугольника

Подобно (5) предложена СУ треугольника, вершины которого точки v_0, v_1 и v_2:

$$СУ(t(v_0, v_1, v_2)) = \sum_{i=0}^{2} (n_i^{avg} v^{new} + d_i^{avg})^2 , \qquad (6)$$

где v^{new} - новая точка, находящаяся в центре треугольника. Расчет непосредственно по средней точке позволяет сократить количество вычислений в почти два раза.

2.2. Алгоритмы стратегии «сверху-вниз»

2.2.1. Удаление вершин

Для использования описанного в 2.1.1 локального критерия оптимальности в структуру вершины введен вектор нормали средней плоскости и зна-

чение локального критерия. Для эффективного удаления и перестроения очереди введен индекс элемента в пирамиде. Это привело к увеличению памяти для представления одной вершины с 28 до 64 байт, а триангуляции в среднем с $136N$ до $172N$ байт.

На первом этапе алгоритма рассчитываются нормали средних плоскостей всех опорных точек триангуляции, строится очередь на удаление. На втором этапе итеративно удаляется первая вершина в очереди, пересчитываются стоимости удаления смежных вершин, и перестраивается очередь на удаление до тех пор, пока не будет достигнуто требуемое разрешение.

Трудоемкость предварительного шага алгоритма расчета потенциальных ошибок и создание очереди на удаление в среднем:

$$T(N) = O\left(\sum_{i=1}^{N} \log i\right) = O(\log N!) = O\left(\log(\sqrt{2\pi N} \cdot N^N e^{-N+\theta/12})\right) = \\ = O((N-1)\log N) \qquad (7)$$

где $T_d(i) = O(\log(i))$ - трудоемкость вставки в пирамиду размера i, $0 < \theta < 1$ по формуле Стирлинга.

Трудоемкость алгоритма выделения дыры и ее ретриангуляции можно принять равной в среднем константе по следующим соображениям: количество инцидентных вершине треугольников в среднем равно шести, следовательно операция удаления вершины и ретриангуляция, зависящие от количества инцидентных вершине треугольников, не зависят от количества вершин в триангуляции и количества удаляемых вершин.

Трудоемкость пересчета очереди зависит от трудоемкости вставки и удаления из пирамиды и изменением ранга вершин, потенциальная ошибка которых изменяется. В среднем эта величина есть $O(\log N)$ (2.2.4).

Итого, трудоемкость упрощения триангуляции удалением вершин $O(m \log N)$, где m – количество удаляемых объектов, N – количество опорных точек.

Для тестирования трудоемкости кода были проведены следующие тесты:

1. Для случайного равномерного распределения в параллелепипеде серия тестов удаления 10%, 25%, 50%, 90%, 99.9% объектов от общего числа вершин. Число вершин изменялось от 10000 до 50000. Результаты тестов приведены в табл. А-1.
2. Для случайного распределения на полусфере серия тестов удаления 10%, 25%, 50%, 90%, 99.9% объектов от общего числа вершин. Число вершин изменялось от 10000 до 50000. Результаты тестов приведены в табл. А-2.
3. Для случайного нормального распределения со случайной равномерной высотой серия тестов удаления 10%, 25%, 50%, 90%, 99.9% объектов от общего числа вершин. Число вершин изменялось от 10000 до 50000. Результаты тестов приведены в табл. А-3.
4. Для регулярного квадратного размещения со случайной равномерной высотой серия тестов удаления 10%, 25%, 50%, 90%, 99.9% объектов от общего числа вершин. Число вершин изменялось от 10000 до 50000. Результаты тестов приведены в табл. А-4.
5. Для регулярного треугольного размещения со случайной равномерной высотой серия тестов удаления 10%, 25%, 50%, 90%, 99.9% объектов от общего числа вершин. Число вершин изменялось от 10000 до 50000. Результаты тестов приведены в табл. А-5.

Тесты были проведены на ПК с процессором Pentium Celeron 500МГц и 128Мб оперативной памяти.

Для обработки результатов применялся тест Стьюдента с доверительной вероятностью 95%. Время в таблицах приведено в мсек.

2.2.2. Коллапс ребер

Для использования описанного в 2.1.2 локального критерия оптимальности в структуру вершины введен вектор нормали средней плоскости, в структуру ребра введено значение локального критерия. Для эффективного

удаления и перестроения очереди введен в структуру ребра индекс элемента в пирамиде. Это привело к увеличению памяти для представления одной вершины с 28 до 52 байт, ребра с 16 до 28 байт, а триангуляции – в среднем с $136\,N$ до $196\,N$ байт.

На первом этапе алгоритма рассчитываются нормали средних плоскостей всех опорных точек триангуляции, строится очередь на коллапс. На втором этапе итеративно удаляется пара вершин первого ребра в очереди, вставляется вершина посередине ребра, нормаль средней плоскости которой считается как нормированная сумма нормалей средних плоскостей удаленных вершин, пересчитываются стоимости удаления новых и перестроенных для выполнения условия Делоне ребер, и перестраивается очередь на коллапс до тех пор, пока не будет достигнуто требуемое разрешение.

Трудоемкость предварительного шага алгоритма расчета потенциальных ошибок и создание очереди на удаление в среднем аналогична формуле (7).

Трудоемкость алгоритма выделения дыры и ее ретриангуляции можно принять равной в среднем константе по соображениям, изложенным в 2.2.Вставка вершины в триангуляцию Делоне есть константа [5].

Трудоемкость пересчета очереди зависит от трудоемкости вставки и удаления из пирамиды и изменением ранга вершин, потенциальная ошибка которых изменяется. В среднем эта величина есть $O(\log N)$ (см. 2.2.4).

Итого, трудоемкость упрощения триангуляции коллапсом ребер есть $O(m \log N)$, где m – количество удаляемых объектов, N – количество опорных точек.

Для тестирования трудоемкости кода были проведены следующие тесты:

1. Для случайного равномерного распределения в параллелепипеде серия тестов коллапса 10%, 25%, 50%, 90%, 99.9% ребер от общего числа вершин. Число вершин изменялось от 10000 до 50000. Результаты тестов приведены в табл. В-1.

2. Для случайного распределения на полусфере серия тестов коллапса 10%, 25%, 50%, 90%, 99.9% ребер от общего числа вершин. Число вершин изменялось от 10000 до 50000. Результаты тестов приведены в табл. В-2.
3. Для случайного нормального распределения со случайной равномерной высотой серия тестов коллапса 10%, 25%, 50%, 90%, 99.9% ребер от общего числа вершин. Число вершин изменялось от 10000 до 50000. Результаты тестов приведены в табл. В-3.
4. Для регулярного квадратного размещения со случайной равномерной высотой серия коллапса 10%, 25%, 50%, 90%, 99.9% ребер от общего числа вершин. Число вершин изменялось от 10000 до 50000. Результаты тестов приведены в табл. В-4.
5. Для регулярного треугольного размещения со случайной равномерной высотой серия тестов коллапса 10%, 25%, 50%, 90%, 99.9% ребер от общего числа вершин. Число вершин изменялось от 10000 до 50000. Результаты тестов приведены в табл. В-5.

Тесты были проведены на ПК с процессором Pentium Celeron 500МГц и 128Мб оперативной памяти.

Для обработки результатов применялся тест Стьюдента с доверительной вероятностью 95%.

2.2.3. Коллапс треугольников

Для использования описанного в 2.1.3 локального критерия оптимальности в структуру вершины введен вектор нормали средней плоскости, в структуру треугольника введено значение локального критерия. Для эффективного удаления и перестроения очереди введен в структуру треугольника индекс элемента в пирамиде. Это привело к увеличению памяти для представления одной вершины с 28 до 52 байт, треугольника с 36 до 48 байт, а триангуляции – в среднем с $136\,N$ до $184\,N$ байт.

На первом этапе алгоритма рассчитываются нормали средних плоскостей всех опорных точек триангуляции, строится очередь на коллапс. На втором этапе итеративно удаляется тройка вершин первого треугольника в оче-

реди, вставляется вершина посередине треугольника, нормаль средней плоскости которой считается как нормированная сумма нормалей средних плоскостей удаленных вершин, пересчитываются стоимости удаления новых и перестроенных для выполнения условия Делоне треугольников, и перестраивается очередь на коллапс до тех пор, пока не будет достигнуто требуемое разрешение.

Трудоемкость предварительного шага алгоритма расчета потенциальных ошибок и создание очереди на удаление в среднем аналогична формуле (7).

Трудоемкость алгоритма выделения дыры и ее ретриангуляции можно принять равной в среднем константе по соображениям, изложенным в 2.2. Вставка вершины в триангуляцию Делоне есть константа [5].

Трудоемкость пересчета очереди зависит от трудоемкости вставки и удаления из пирамиды и изменением ранга вершин, потенциальная ошибка которых изменяется. В среднем эта величина есть $O(\log N)$ (2.2.4).

Итого, трудоемкость упрощения триангуляции коллапсом треугольника есть $O(m \log N)$, где m – количество удаляемых объектов, N – количество опорных точек.

Для тестирования трудоемкости кода были проведены следующие тесты:

1. Для случайного равномерного распределения в параллелепипеде серия тестов коллапса 5%, 12.5%, 25%, 45%, 49.95% треугольников от общего числа вершин. Число вершин изменялось от 10000 до 50000. Результаты тестов приведены в табл. С-1.
2. Для случайного распределения на полусфере серия тестов коллапса 5%, 12.5%, 25%, 45%, 49.95% треугольников от общего числа вершин. Число вершин изменялось от 10000 до 50000. Результаты тестов приведены в табл. С-2.
3. Для случайного нормального распределения со случайной равномерной высотой серия тестов коллапса 5%, 12.5%, 25%, 45%,

49.95% треугольников от общего числа вершин. Число вершин изменялось от 10000 до 50000. Результаты тестов приведены в табл. С-3.

4. Для регулярного квадратного размещения со случайной равномерной высотой серия тестов коллапса 5%, 12.5%, 25%, 45%, 49.95% треугольников от общего числа вершин. Число вершин изменялось от 10000 до 50000. Результаты тестов приведены в табл. С-4.
5. Для регулярного треугольного размещения со случайной равномерной высотой серия тестов коллапса 5%, 12.5%, 25%, 45%, 49.95% треугольников от общего числа вершин. Число вершин изменялось от 10000 до 50000. Результаты тестов приведены в табл. С-5.

Тесты были проведены на ПК с процессором Pentium Celeron 500МГц и 128Мб оперативной памяти.

Для обработки результатов применялся тест Стьюдента с доверительной вероятностью 95%.

2.2.4. Сокращение всех примитивов

Так как описанные в 2.1 критерии представляют потенциальную ошибку одного глобального критерия, то был предложен алгоритм упрощения триангуляции, который использует описанные выше локальные критерии для выбора примитива для сокращения.

Вместо одной очереди на сокращение было использовано три очереди: для вершин, ребер и треугольников отдельно. Сделано это было по следующим соображениям:

Трудоемкость вставки элемента в пирамиду есть $O(\log m)$, где m – размер пирамиды, перемещения – в среднем $O(\log(m-1))$, удаления – в среднем $O((\log m)/2)$. Следовательно, общая трудоемкость перестроения очереди есть $O(a \log m)$, где a – количество элементов в пирамиде, требующих перестановки, вставки или удаления. Предположим, что в промежуточной триангуляции m_1 вершин, m_2 ребер и m_3 треугольников, и что на следующем шаге

итерации требуется вычислить стоимость удаления или удалить a_1 вершин, a_2 ребер и a_3 треугольников. Тогда, если очередь для сокращения будет представлена одной пирамидой, то трудоемкость перестроения будет

$$O\left[(a_1 + a_2 + a_3)\log(m_1 + m_2 + m_3)\right] = O\left[\log(m_1 + m_2 + m_3)^{a_1+a_2+a_3}\right]. \qquad (8)$$

Если очередь будет представлена в виде трех пирамид, то трудоемкость перестроения будет

$$O\left[a_1 \log m_1 + a_2 \log m_2 + a_3 \log m_3\right] = O\left[\log(m_1^{a_1} m_2^{a_2} m_3^{a_3})\right]. \qquad (9)$$

Очевидно, что (8) больше (9) для a_i и m_i больших 1.

Для использования описанных в 2.1 критериев в структуру представления вершины были введена нормаль средних плоскостей, в структуры примитивов – значение стоимости сокращения. Для эффективного удаления и перестроения очередей введен в структуру примитивов индекс элемента в пирамиде. Это привело к увеличению объема требуемой памяти для представления вершины с 28 до 64 байт, ребра – с 16 до 28 байт, треугольника с 36 до 48 байт, триангуляции – в среднем с $136N$ до $230N$ байт.

На первом этапе алгоритма рассчитываются нормали средних плоскостей всех опорных точек триангуляции, строится очереди на коллапс. На втором этапе итеративно сокращается примитив, СУ которого меньше (для этого сравниваются СУ первых элементов трех очередей), пересчитываются стоимости удаления примитивов, и перестраивается очереди до тех пор, пока не будет достигнуто требуемое разрешение.

Трудоемкость предварительного шага алгоритма расчета потенциальных ошибок и создание очереди на удаление в среднем аналогична формуле (7).

Трудоемкость алгоритма выделения дыры и ее ретриангуляции можно принять равной в среднем константе по следующим соображениям: количество инцидентных вершине треугольников в среднем равно шести, следовательно операция удаления вершины и ретриангуляция, зависящие от количе-

ства инцидентных вершине треугольников, не зависят от количества вершин в триангуляции и количества удаляемых вершин.

Трудоемкость пересчета очереди зависит от трудоемкости вставки и удаления из пирамиды и изменением ранга вершин, потенциальная ошибка которых изменяется. В среднем эта величина есть $O(\log N)$.

Итого, трудоемкость упрощения триангуляции сокращением всех примитивов есть $O(m \log N)$, где m – количество удаляемых объектов, N – количество опорных точек.

Для тестирования трудоемкости кода были проведены следующие тесты:

1. Для случайного равномерного распределения в параллелепипеде серия тестов сокращения до триангуляции с 90%, 75%, 50%, 10%, 0.1% вершин от общего числа вершин. Число вершин изменялось от 10000 до 50000. Результаты тестов приведены в табл. D-1.
2. Для случайного распределения на полусфере серия тестов сокращения до триангуляции с 90%, 75%, 50%, 10%, 0.1% вершин от общего числа вершин. Число вершин изменялось от 10000 до 50000. Результаты тестов приведены в табл. C-2.
3. Для случайного нормального распределения со случайной равномерной высотой серия тестов сокращения до триангуляции с 90%, 75%, 50%, 10%, 0.1% вершин от общего числа вершин. Число вершин изменялось от 10000 до 50000. Результаты тестов приведены в табл. D-3.
4. Для регулярного квадратного размещения со случайной равномерной высотой серия тестов сокращения до триангуляции с 90%, 75%, 50%, 10%, 0.1% вершин от общего числа вершин. Число вершин изменялось от 10000 до 50000. Результаты тестов приведены в табл. D-4.
5. Для регулярного треугольного размещения со случайной равномерной высотой серия тестов сокращения до триангуляции с 90%, 75%, 50%,

10%, 0.1% вершин от общего числа вершин. Число вершин изменялось от 10000 до 50000. Результаты тестов приведены в табл. D-5.

Тесты были проведены на ПК с процессором Pentium Celeron 500МГц и 128Мб оперативной памяти.

Для обработки результатов применялся тест Стьюдента с доверительной вероятностью 95%.

2.3. Алгоритм стратегии «сверху-вниз»

Алгоритмы, представленные выше используют стратегию «снизу-вверх», т.е. последовательно модифицируя триангуляцию полного разрешения, уменьшают количество узлов.

Алгоритмы, использующие стратегию «сверху-вниз», начинают с простой аппроксимирующей модели, состоящую из треугольников, покрывающих исходную триангуляцию. Затем в триангуляцию добавляют последовательно новые точки до тех пор, пока не будет достигнуто требуемое разрешение.

Определение 10. *Суперструктурой* множества опорных точек триангуляции назовем такую выпуклую триангуляцию, треугольники которой покрывают все опорные точки.

Алгоритм ранжирования итеративной вставкой в суперструктуру действуют следующим образом. На начальном этапе все опорные точки разделяются по покрывающим их треугольникам суперструктуры, для них рассчитываются стоимости вставки в покрывающий треугольник, для каждого треугольника вносятся в очередь опорная точка, покрываемая этим треугольником, с наибольшей ошибкой вставки. Затем итеративно вставляется вершина с наибольшей ошибкой с одновременным перераспределением вершин по новым треугольникам, пересчетом ошибок вставки и перестроением очереди. Перестроение очереди происходит следующим образом: из очереди удаляется вставленная вершина, для каждого перестроенного треугольника в очередь

вносится вершина, покрываемая этим треугольником, с наибольшей ошибкой вставки.

Для такого алгоритма был предложен очевидный локальный критерий оптимальности, а именно: расстояние от точки до триангуляции. При этом практически использовать его сложно: надо рассчитать расстояние до плоскости покрывающего треугольника, если перпендикуляр на плоскость выходит за треугольник, расстояние до ближайшего ребра треугольника. Поэтому практически было предложено использовать следующее упрощение этого критерия оптимальности: расстояние от точки до ее проекции на плоскость покрывающего треугольника по оси аппликат.

Для использования описанного критерия в структуру представления треугольника были введены: нормаль плоскости, список покрываемых вершин, вершина с максимальной потенциальной ошибкой, покрываемая этим треугольником. Для эффективного удаления и перестроения очередей введен в структуру представления вершины индекс в пирамиде. Это привело к увеличению объема требуемой памяти для представления вершины с 28 до 36 байт, треугольника с 36 до 78+4n_i байт, где n_i - количество покрываемых вершин, триангуляции – в среднем с 92N до 180N байт.

Самый значительный вклад в трудоемкость алгоритма вносит процедура перераспределения вершин по изменившимся треугольникам. Для уменьшения времени работы этой процедуры все вершины отсортированы по одной из координат (далее предполагаем сортировку по оси абсцисс), и при внесении в список треугольника порядок сохраняется.

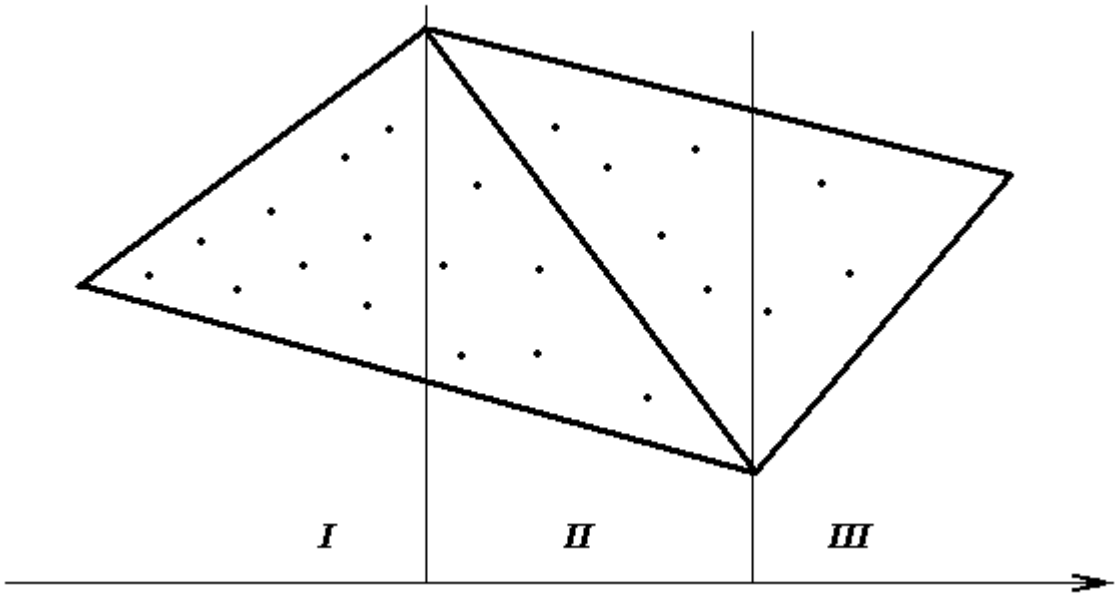

Рис. 2.2. Области распределения точек по треугольникам (I – область до вершины разделяющего ребра с наименьшей координатой, П – область от вершины разделяющего ребра с наименьшей координатой до вершины разделяющего ребра с наибольшей координатой, Ш – область от вершины разделяющего ребра с наибольшей координатой).

Это позволяет при флипе распределять вершины следующим образом: из двух списков выбирается вершина с наименьшим значением по выбранной координате, пока не достигли конца обоих списков. Если абсцисса вершины не достигла наименьшей из абсцисс вершин нового ребра, вершина вносится в один из треугольников, расположенный в «наименьшей» полуплоскости от разделяющего ребра (

Рис. 2.2).

Затем, пока абсцисса вершины не достигла вершины разделяющего ребра с наибольшей абсциссой, вершина вносится в один из треугольников, согласно положению относительно разделяющего ребра. Оставшиеся верши-

ны вносятся в треугольник, расположенный в «наибольшей» полуплоскости от разделяющего ребра.

Заметим, что области I и Ш могут быть вырожденными, т.е. значения всех ординат лежат между значениями ординат разделяющего ребра. В данном случае это в среднем на половину сократит количество проверок.

Также сортировка по ординате позволяет снизить время вычислений при вставке вершины (Рис. 2.3).

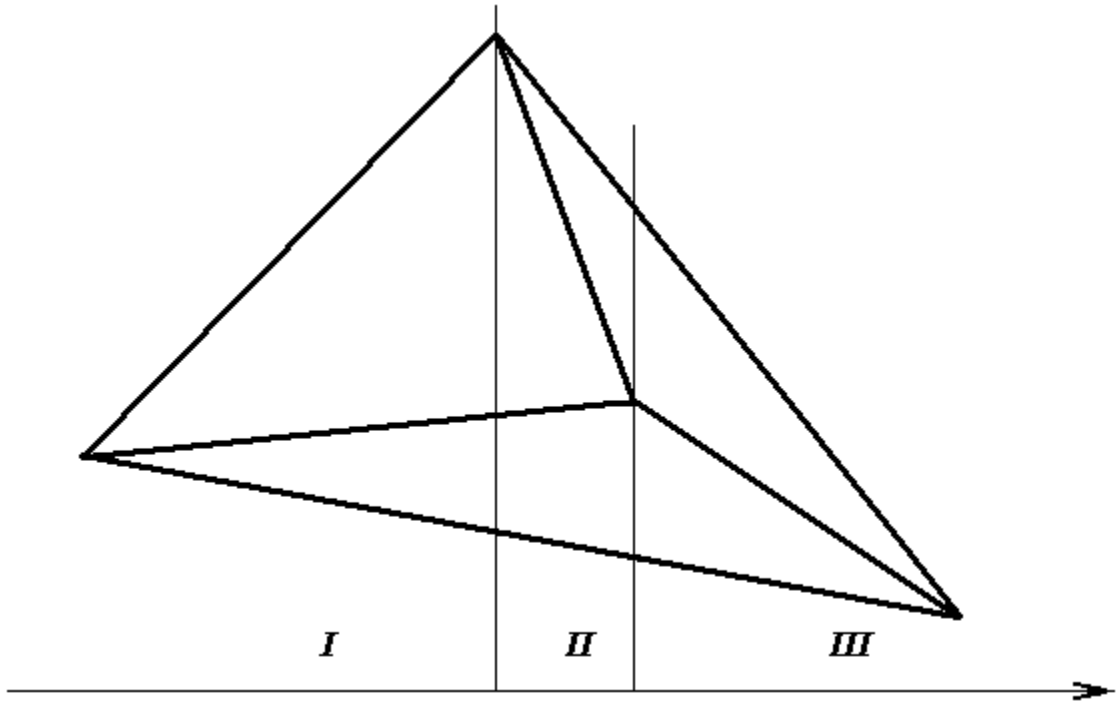

Рис. 2.3. Области распределения точек по треугольникам

Для первой и третьей области проверяется положение относительно одной прямой, для второй проверяется положение вершины относительно одной (верхней) прямой, при нахождении внизу относительно верхней прямой, проверяется положение относительно второй (нижней) прямой. Общее количество проверок уменьшается в среднем на треть.

Трудоемкость построения суперструктуры зависит от ее вида: вершины треугольника, квадрата, множество точек, распределенных по ребрам много-

угольника, выпуклая оболочка. Как показано в [4] выбор супеурструктуры не может увеличить или уменьшить скорость работы алгоритма более, чем на 10%, поэтому была выбрана суперструктура в виде прямоугольника, покрывающий все опорные точки. Трудоемкость задания такой суперструктуры $O(N)$, если неизвестны минимальные и максимальные значения координат, и константа, если известны.

Трудоемкость вставки вершины есть константа[7].

Трудоемкость пересчета вершин по треугольникам и расчет ошибок вставки зависит линейно от количества вершин в перестраиваемых треугольниках. В среднем эта величина $O\big((N-m)/m\big)$есть, где $N-$ количество опорных точек, $m-$ количество вершин в промежуточной триангуляции, если $m < \frac{N}{3}$, и $O(1)$, если $m \geq \frac{N}{3}$.

Трудоемкость перестроения очереди есть $O(\log L_m)$, где L_m- размер очереди после вставки $m-1$ первых вершин. В среднем эта величина возрастает от O(1) до $\frac{N}{3}$ при возрастании m от 1 до $\frac{N}{3}$, и убывает от $\frac{N}{3}$ до 0 при возрастании m от $\frac{N}{3}$ до N.

Итого трудоемкость алгоритма дробления в среднем есть:

$$T(m,N) = O\big((N+m)\log N\big) \qquad (10)$$

Для тестирования трудоемкости кода были проведены следующие тесты:

1. Для случайного равномерного распределения в параллелепипеде серия тестов вставки 10%,25%, 50%, 90%, 99.9% вершин от общего числа опорных точек. Число вершин изменялось от 10000 до 50000. Результаты тестов приведены в табл. Е-1.
2. Для случайного распределения на полусфере серия тестов вставки 10%, 25%, 50%, 90%, 99.9% вершин от общего числа опорных точек.

Число вершин изменялось от 10000 до 50000. Результаты тестов приведены в табл. Е-2.

3. Для случайного нормального распределения со случайной равномерной высотой серия тестов вставки 10%, 25%, 50%, 90%, 99.9% вершин от общего числа опорных точек. Число вершин изменялось от 10000 до 50000. Результаты тестов приведены в табл. Е-3.
4. Для регулярного квадратного размещения со случайной равномерной высотой серия тестов вставки 10%, 25%, 50%, 90%, 99.9% вершин от общего числа опорных точек. Число вершин изменялось от 10000 до 50000. Результаты тестов приведены в табл. Е-4.
5. Для регулярного треугольного размещения со случайной равномерной высотой серия тестов вставки 10%, 25%, 50%, 90%, 99.9% вершин от общего числа опорных точек. Число вершин изменялось от 1000 до 100000. Результаты тестов приведены в табл. Е-5.

Тесты были проведены на ПК с процессором Pentium Celeron 500МГц и 128Мб оперативной памяти.

Для обработки результатов применялся тест Стьюдента с доверительной вероятностью 95%.

3. Сравнение алгоритмов упрощения триангуляции

3.1. Оценка времени работы

Рассматривая алгоритмы упрощения триангуляции как алгоритмы ранжирования вершин, можно сравнить алгоритмы стратегии «сверху-вниз» и «снизу-вверх» при упрощении до треугольника и при построении триангуляции полного разрешения.

Алгоритм дробления суперсруктуры показал меньшее время работы в среднем на всех наборах исходных данных. Алгоритм сокращения всех примитивов работает в среднем медленнее алгоритма коллапса ребер. Алгоритм коллапса ребер работает в среднем медленнее алгоритма коллапса треугольников. Алгоритм коллапса треугольников работает в среднем медленнее алгоритма удаления вершин.

3.2. Оценка качества аппроксимации

Для оценки качества полученной в результате работы алгоритма триангуляции применялся глобальный критерий: максимальное отклонение опорных точек по высоте.

Алгоритм сокращения всех примитивов показал в среднем лучше качество аппроксимации. Алгоритм дробления суперструктуры показал почти одинаковое качество с алгоритмом удаления вершин. Время работы алгоритма дробления суперструктуры в среднем почти равно времени работы алгоритма удаления вершин при построении триангуляции, содержащей 65% вершин от количества опорных точек.

4. Руководство программиста

Алгоритм триангуляции и описание структур данных для работы алгоритма разработан в НПО «СибГеоИнформатика» как часть ГИС «Графин». Структуры данных и алгоритмы упрощения триангуляций стратегии «снизу-вверх» представлены в файле **trnsN.pas**. Для работы алгоритмов введены следующие описания структур данных и методы:

```
Узел триангуляции
TTrnNode = packed object
 private
   FT: pTriangle;      ссылка на инцидентный треугольник
   FPE: PEType;        потенциальная ошибка удаления
   FNormal: TDoublePoint3D;    нормаль средней плоскости
   FIndex: integer;    индекс в очереди на удаление

Ребро триангуляции
TTrnRib = class
 private
   FIndex: Integer;    индекс в очереди на коллапс
   FPE: PEType;        потенциальная ошибка коллапса

Треугольник триангуляции.
TTriangle = packed object
 private
   FIndex: integer;    индекс в очереди на коллапс
   FPE: PEType;        потенциальная ошибка коллапса
   FNormal: TDoublePoint3D;    нормаль плоскости треугольника

Триангуляция
TTriangulation = class
 private
   FPrepareTime: Cardinal;     время подготовительного этапа упрощения
   FSimplifyTime: Cardinal;    время основного этапа упрощения
```

```
    FLastTime: Cardinal;        время окончательного этапа упрощения
  public
    function Simplify1(ntd: integer): PEType;   упрощение удалением вершин
    function Simplify2(ntd: integer): PEType;   упрощение  коллапсом ребер
    function Simplify3(ntd: integer): PEType;   упрощение  коллапсом треугольников
    function Simplify_(ntd: integer): PEType;   упрощение сокращением всех примитивов
    property PrepareTime: Cardinal read FPrepareTime;   доступ к переменной  (чтение)
    property SimplifyTime: Cardinal read FSimplifyTime; доступ к переменной  (чтение)
    property LastTime: Cardinal read FLastTime; доступ к переменной  (чтение)
```

Структуры данных и алгоритм упрощения триангуляций стратегии «сверху-вниз» представлены в файле **trns0.pas**. Для работы алгоритма введены следующие описания структур данных и методы:

```
Треугольник триангуляции.
TTriangle = packed object
private
    MP: pTrnNode;       ссылка на точку с максимальной ошибкой вставки
    FIndex: integer;    индекс в очереди точки с максимальной ошибкой вставки
    InNodes: TList;     список указателей на покрываемые треугольником точки
    FNormal: TDoublePoint3D;    нормаль плоскости треугольника

Триангуляция
TTriangulation = class
  public
    function Build(ntd: integer): PEType;      построение триангуляции заданного раз-
                                               решения дроблением суперструктуры
```

Заключение

В ходе выполнения дипломной работы:

1. Разработаны и реализованы алгоритмы упрощения триангуляции удалением вершин, коллапсом ребер, коллапсом треугольников и сокращением всех примитивов в триангуляции Делоне с сохранением условия Делоне, а также алгоритм дробления суперструктуры до триангуляции Делоне заданного разрешения.
2. Разработаны алгоритмы ранжирования на основе алгоритма упрощения триангуляции и вставки вершин в суперструктуру.
3. На основании имитационного моделирования работы алгоритма получены статистические оценки трудоемкости в среднем на различных наборах исходных данных.

Области применения:

1. Моделирование рельефов по опорным точкам
2. Визуализация сложных поверхностей

Дальнейшие направления:

1. Реализация триангуляции множественного разрешения для интерактивной визуализации рельефа с динамическим изменением параметров наблюдения в реальном времени.

Список используемой литературы

1. Делоне Б.Н. О пустоте сферы // Изв. АН СССР, ОМЕН, 1993, №4, с.793-800
2. Ильман В.М. Экстремальные свойства триангуляции Делоне // Алгоритмы и программы, ВИЭМС. Вып. 10(88).– М., 1985, с. 57-66.
3. Костюк Ю.Л., Фукс А.Л. Приближенное вычисление оптимальной триангуляции // Геоинформатика. Теория и практика. Вып. 1. Томск: Изд-во Том. ун-та, 1998, с.61-66.
4. Препарата Ф., Шеймос М. Вычислительная геометрия: Введение / Пер. с англ. – М.:Мир, 1989. – 478 с.
5. Скворцов А.В., Триангуляция Делоне и ее применение. Изд-во Том. ун-та, 2002.– 130 с.
6. Скворцов А.В., Костюк Ю.Л., Применение триангуляции для решения задач вычислительной геометрии // Геоинформатика. Теория и практика. Вып. 1. Томск: Издательство Том. ун-та, 1998, с.127-138.
7. Скворцов А.В., Костюк Ю.Л., Эффективные алгоритмы построения триангуляции Делоне // Геоинформатика. Теория и практика. Вып. 1. Томск: Издательство Том. ун-та, 1998, с.22–47.
8. Andujar C., Ayala D., Brunet P., Joan-Arinyo R. and Sole J., Automatic generation of multiresolution boundary representations. // Computer Graphics Forum (Eurographics ’96 Proc.), 1996, Vol. 15, No. 3, pp.87–96.
9. Ciampalini A., Cignoni P., Montani C. and Scopigno, R., Multiresolution decimation based on global error. The Visual Computer, June, 1997, Vol. 13, No. 5, pp.228–246.
10. Cohen J., Varshney A., Manocha D., Turk G., Weber H., Agarwal P., Brooks F. and Wright W., Simplification envelops // Computer Graphics Proc., Annual Conf. Series (Siggraph ’96). – ACM Press, August 6-8, 1996, pp.119–128.
11. Chandrajit L.B. and Danilel R.S., Topology preserving data simplification with error bounds // Comput. & Graphics, 1998, Vol. 22, No. 1, pp.3–12.

12. Funkhouser T.A. and Sequin C. H., Adaptive display algorithm for interactive frame rates during visualization of complex environment // Computer Graphics Proc.. Annual Conf. Series (Siggraph ’93). – ACM Press, 1993, pp.247–254.
13. He T., Hong L., Kaufman A., Varshey A., Wang S. Voxel-based object simplification // IEEE Visualization ’95 Proceedings. IEEE. – Comp. Soc. Press, 1996, pp.296–303.
14. He T., Hong L., Varshey A. and Wang S., Controlled topology simplification // IEEE Visualization & Computer Graphics, 1996, Vol. 2(2), pp.171–183.
15. Klein R. and Krammer J., Multiresolution representations for surface meshes // Proceeding of Spring Conference on Computer Graphics held in Budmerice, Slovakia, 05–08 June1997, ed. W. Straber, 1997.
16. Klein R., Liebich G. and Straber W., Mesh reduction with error control // Proceedings of Visualization ’96, ed. R Yagel and G.Neilson. 1996, pp.311–318.
17. Klein R., Multiresolution representation for surfaces meshes based on vertex decimation method // Comput. &Graphics, 1998, Vol. 22, No. 1, pp.13–26.
18. Leon J.C. and Veron P., Shape preserving plyhedral simplification with bound error // Comput. & Graphis, 1998, Vol. 22, No. 5, pp.565–585.
19. Lawson C. Software for C^1 surface interpolation // Mathematical Software III.– NY: Academic Press, 1977, pp.161-194.
20. Lawson C. Transforming triangulations // Discrete Mathematics, No. 3, North-Holland Publishing Company, 1972, pp. 365-372.
21. Lee D. Proximity and reachability in the plane. – Tech. Rep. No. R-831, Coordinated Sci. Lab., Univ. of Illinois at Urbana, IL, 1978.
22. Lingas A. The Greedy and Delaunay triangulaton are not bad… // Lect. Notes Comp. Sc., 1983, Vol. 21, No. 4, pp. 270-284.
23. Manacher G., Zobrist A. Neither Greedy nor Delaunay triangulation of planar point set approximates the optimal triangulation // Inf. Prof. Let., Vol. 9, No. 1, 1977, pp. 31-34.
24. Sibson R. Locally equiangular triangulatoins // Computer Journal, Vol. 21, No. 3, 1978, pp.243-245.

25. The Virtual Reality Modeling Language Specification – version 2.0. August,1996.
26. Werneck J., The Inventor mentor: programming Object-oriented 3D graphics with Open Inventor. – Addison Wesley, 1994.

Приложение А. Время работы алгоритма удаления вершин

Табл. А-1. Время работы на случайном равномерном распределении в параллелепипеде в мсек.

Количество удалений	Количество опорных точек				
	10000	20000	30000	40000	50000
10%	360	721	1082	1462	2516
25%	641	1282	1933	2624	3305
50%	1071	2294	3505	4707	5929
90%	1933	4116	6209	8292	10455
99,9%	2223	4456	6740	9113	11286

Табл. А-2. Время работы на случайном распределении на полусфере в мсек.

Количество удалений	Количество опорных точек				
	10000	20000	30000	40000	50000
10%	298	619	946	1257	1585
25%	492	1010	1538	2066	2604
50%	829	1703	2594	3492	4405
90%	1374	2837	4316	5841	7374
99,9%	1492	3087	4713	6356	8026

Табл. А-3. Время работы на случайном нормальном распределении в мсек.

Количество удалений	Количество опорных точек				
	10000	20000	30000	40000	50000
10%	317	645	985	1320	1656
25%	558	1138	1722	2311	2910
50%	986	2014	3051	4096	5157
90%	1716	3503	5303	7126	8949
99,9%	1883	3860	5850	7865	9874

Табл. А-4. Время работы на регулярном квадратном размещении в мсек.

Количество удалений	Количество опорных точек				
	10000	20000	30000	40000	50000
10%	341	701	1011	1392	1753
25%	611	1272	1842	2564	3224
50%	1052	2153	3165	4406	5157
90%	1793	3615	5408	7440	9343
99,9%	1953	3966	5949	8211	10224

Табл. А-5. Время работы на регулярном треугольном размещении в мсек.

Количество удалений	Количество опорных точек				
	10000	20000	30000	40000	50000
10%	321	671	1021	1382	1733
25%	561	1202	1832	2464	3114
50%	941	2043	3105	4178	5238
90%	1602	3445	5297	7010	8833
99,9%	1722	3805	5828	7782	9794

Приложение В. Время работы алгоритма коллапса ребер

Табл. В-1. Время работы на случайном равномерном распределении в параллелепипеде в мсек.

Количество удалений	Количество опорных точек				
	10000	20000	30000	40000	50000
10%	532	1080	1636	2190	2767
25%	1053	2133	3242	4340	5513
50%	1938	3946	5950	8017	11446
90%	3363	6838	10365	13943	28221
99,9%	3712	7543	11455	15397	46907

Табл. В-2. Время работы на случайном распределении на полусфере в мсек.

Количество удалений	Количество опорных точек				
	10000	20000	30000	40000	50000
10%	489	990	1513	2027	2557
25%	937	1911	2903	3887	4898
50%	1699	3463	5268	7062	8885
90%	2918	5976	9113	12242	16136
99,9%	3189	6529	9966	13418	26367

Табл. В-3. Время работы на случайном нормальном распределении в мсек.

Количество удалений	Количество опорных точек				
	10000	20000	30000	40000	50000
10%	543	1097	1660	2232	2817
25%	1067	2168	3288	4401	5535
50%	1966	3999	6043	8103	10192
90%	3401	6932	10487	14085	22242
99,9%	3746	7648	11581	15523	25682

Табл. В-4. Время работы на регулярном квадратном размещении в мсек.

Количество удалений	Количество опорных точек				
	10000	20000	30000	40000	50000
10%	481	962	1472	1973	2493
25%	960	1948	2986	3294	5043
50%	1783	3635	5548	5548	9413
90%	3096	6337	9681	9191	16966
99,9%	3412	6996	10698	10066	18318

Табл. В-5. Время работы на регулярном треугольном размещении в мсек.

Количество удалений	Количество опорных точек				
	10000	20000	30000	40000	50000
10%	471	942	1422	1913	2404
25%	840	1707	2568	3456	4334
50%	1472	3015	4507	6089	7641
90%	2481	5110	7635	10344	13357
99,9%	2724	5622	8405	11367	14380

Приложение С. Время работы алгоритма коллапса треугольников

Табл. С-1. Время работы на случайном равномерном распределении в параллелепипеде в мсек.

Количество удалений	Количество опорных точек				
	10000	20000	30000	40000	50000
10%	551	1041	1563	2103	2623
25%	927	1843	2783	3720	4638
50%	1572	3214	4847	6479	8091
90%	2602	5410	8177	10938	14059
99,9%	2849	5946	8996	12010	15127

Табл. С-2. Время работы на случайном распределении на полусфере в мсек.

Количество удалений	Количество опорных точек				
	10000	20000	30000	40000	50000
10%	471	971	1462	1962	2473
25%	806	1659	2522	3375	4241
50%	1382	2834	4316	5788	7270
90%	2301	4717	7210	9686	12506
99,9%	2522	5177	7922	10623	13443

Табл. С-3. Время работы на случайном нормальном распределении в мсек.

Количество удалений	Количество опорных точек				
	10000	20000	30000	40000	50000
10%	511	1042	1563	2083	2614
25%	902	1837	2775	3707	4644
50%	1572	3195	4827	6480	8122
90%	2642	5371	8136	10960	14134
99,9%	2899	5902	8951	12037	15210

Табл. С-4. Время работы на регулярном квадратном размещении в мсек.

Количество удалений	Количество опорных точек				
	10000	20000	30000	40000	50000
10%	451	902	1362	1822	2293
25%	801	1626	2474	3298	4139
50%	1402	2864	4356	5818	7301
90%	2361	4847	7392	9890	12767
99,9%	2592	5331	8139	10868	13745

Табл. С-5. Время работы на регулярном треугольном размещении в мсек.

Количество удалений	Количество опорных точек				
	10000	20000	30000	40000	50000
10%	461	952	1432	1913	2403
25%	837	1710	2596	3441	4337
50%	1482	3005	4567	6049	7651
90%	2512	5080	7746	10263	13379
99,9%	2759	5586	8528	11276	14404

Приложение D. Время работы алгоритма сокращения всех примитивов

Табл. D-1. Время работы на случайном равномерном распределении в параллелепипеде в мсек.

Количество удалений	Количество опорных точек				
	10000	20000	30000	40000	50000
10%	691	1402	2103	2844	3595
25%	1174	2381	3587	4883	6112
50%	2003	4056	6099	8362	10425
90%	3326	6738	10151	13984	17880
99,9%	3644	7393	11148	15336	19214

Табл. D-2. Время работы на случайном распределении на полусфере в мсек.

Количество удалений	Количество опорных точек				
	10000	20000	30000	40000	50000
10%	721	1482	2244	3014	3795
25%	1204	2465	3735	5053	6342
50%	2033	4146	6259	8532	10705
90%	3356	6838	10330	14154	18247
99,9%	3674	7496	11332	15506	19597

Табл. D-3. Время работы на случайном нормальном распределении в мсек.

Количество удалений	Количество опорных точек				
	10000	20000	30000	40000	50000
10%	681	1412	2153	2874	3605
25%	1160	2403	3641	4894	6141
50%	1983	4096	6159	8342	10485
90%	3296	6809	10221	13913	17995
99,9%	3612	7471	11221	15253	19338

Табл. D-4. Время работы на регулярном квадратном размещении в мсек.

Количество удалений	Количество опорных точек				
	10000	20000	30000	40000	50000
10%	641	1332	2023	2724	3425
25%	1095	2286	3503	4667	5872
50%	1873	3916	6008	7982	10065
90%	3115	6528	10048	13339	17313
99,9%	3414	7165	11043	14627	18609

Табл. D-5. Время работы на регулярном треугольном размещении в мсек.

Количество удалений	Количество опорных точек				
	10000	20000	30000	40000	50000
10%	661	1332	2013	2734	3435
25%	1140	2315	3490	4735	5941
50%	1962	3996	5989	8151	10235
90%	3274	6688	10020	13670	17657
99,9%	3589	7346	11013	14997	18985

Приложение Е. Время работы алгоритма сокращения всех примитивов

Табл. Е-1. Время работы на случайном равномерном распределении в параллелепипеде в мсек.

Количество удалений	Количество опорных точек				
	10000	20000	30000	40000	50000
10%	390	901	1382	1852	2674
25%	521	1182	2093	2503	3415
50%	721	1572	2414	3505	4236
90%	1202	2113	3184	4496	5568
99,9%	1332	2274	3515	5039	5818

Табл. Е-2. Время работы на случайном распределении на полусфере в мсек.

Количество удалений	Количество опорных точек				
	10000	20000	30000	40000	50000
10%	9531	25460	44576	65390	84012
25%	12220	31109	55561	78634	99117
50%	15836	38468	66717	91835	117432
90%	21702	47012	80921	108735	140939
99,9%	21795	48777	84582	113484	145068

Табл. Е-3. Время работы на случайном нормальном распределении в мсек.

Количество удалений	Количество опорных точек				
	10000	20000	30000	40000	50000
10%	371	881	1873	2594	2544
25%	521	1192	2503	3455	3365
50%	731	1612	3364	3666	4426
90%	971	2113	4386	4516	5768
99,9%	1022	2203	4627	4717	6049

Табл. Е-4. Время работы на регулярном квадратном размещении в мсек.

Количество удалений	Количество опорных точек				
	10000	20000	30000	40000	50000
10%	661	1962	3385	5117	6970
25%	791	2229	3845	5709	7712
50%	942	2525	4357	6389	8572
90%	1101	2814	4856	7120	9553
99,9%	1122	2854	4924	7220	9687

Табл. Е-5. Время работы на регулярном треугольном размещении в мсек.

Количество удалений	Количество опорных точек				
	10000	20000	30000	40000	50000
10%	531	1473	2494	3836	5027
25%	671	1772	2944	4446	5818
50%	851	2173	3536	5258	6829
90%	1062	2593	4156	6149	7991
99,9%	1101	2664	4266	6279	8172

Приложение F. Список файлов

Дискета 1 содержит файлы trnsN.pas, heaps.pas, mainN.pas.

Дискета 2 содержит файлы trns0.pas, heaps.pas, main0.pas.

Файл trnsN.pas содержат описание класса TTriangulation для алгоритмов стратегии “снизу-вверх”, файл trns0.pas содержат описание класса TTriangulation для алгоритма стратегии “сверху-вниз” файл heaps.pas содержит описание класса TLocalHeap для быстрого выделения памяти, mainN.pas и main0.pas предоставляют оконный интерфейс для тестирования алгоритмов стратегий «снизу-вверх» и «сверху-вниз» соответсвенно.