УДК 519.688

СЖАТИЕ ТОПОЛОГИЧЕСКИХ СВЯЗЕЙ ТРИАНГУЛЯЦИИ

А. В. Скворцов[1]

Рассматривается задача упаковки топологических связей треугольников в триангуляции. Приводится несколько модификаций алгоритма шелушения треугольников, позволяющих достичь в среднем плотности упаковки порядка 2.12 бит на узел триангуляции.

Ключевые слова: триангуляция Делоне, вычислительная устойчивость, вычислительная геометрия, машинная графика, геоинформационные системы, многоугольники, триангуляционные модели.

Введение. Триангуляция является одной из базовых структур вычислительной геометрии и машинной графики [3]. Она широко используется в геоинформационных системах для моделирования рельефа, системах автоматизированного проектирования для представления различных поверхностей, в различных методах конечных элементов для разбиения объектов на элементарные треугольники.

В связи с продолжающимся быстрым ростом производительности вычислительной техники стремительно растут и объемы обрабатываемых данных, в том числе и размеры триангуляционных моделей. В настоящее время реальные триангуляционные модели рельефа земной поверхности содержат миллионы и миллиарды треугольников. Это порождает серьезные проблемы по хранению, передаче, визуализации и обработке столь огромных моделей данных.

Стандартные триангуляционные модели данных требуют примерно 8 – 16 байт на хранение координат узлов триангуляции и 28 – 72 байта на представление топологических отношений треугольников (отношений смежности узлов, ребер и треугольников) [4]. Видно, что наибольшую долю памяти (до 90 %) занимает топология триангуляции. При представлении триангуляции, содержащей миллионы узлов и треугольников, такие затраты памяти являются слишком расточительными. Одной из проблем, возникающих в связи с этим, является задача упаковки (сжатия) триангуляции для хранения во внешней памяти. При этом требуется получить два алгоритма, один из которых должен выдавать сжатое представление триангуляции для передачи во внешнюю память, а другой должен уметь генерировать исходную триангуляцию по ее сжатому представлению.

Проблема сжатия триангуляции распадается на две основные подзадачи:

— *сжатие узлов*: задача сжатия координат узлов триангуляции и других их числовых характеристик;

— *сжатие топологии* (*топологических связей*): задача сжатия информации, описывающей треугольники как тройки узлов триангуляции, и информации о смежности пар соседних треугольников.

Эти две задачи имеют совершенно разную природу и поэтому решаются разными методами. Задача сжатия узлов обычно решается с помощью комбинации методов с потерей точности (методы квантизации) и точных методов (кодирование энтропии) [5 – 7].

Настоящая работа посвящена второй задаче — задаче сжатия топологии триангуляции.

Один из первых методов упаковки триангуляции заключался в разбиении триангуляции на некоторые *полосы* — последовательности смежных треугольников. Однако такой способ лучше всего подходит для визуализации, т.к. полная топология триангуляции при этом не сохраняется. Другой проблемой здесь является выбор минимального количества полос. В [7] показано, что эта задача является NP-полной.

Среди полноценных алгоритмов сжатия одним из наиболее простых и удобных в применении является *метод шелушения* (*shelling method*) [8]. Он позволяет добиться плотности упаковки топологии до 4.4 бит на узел триангуляции, т.е. сжимает данные примерно в 50 – 140 раз.

Отметим, что количество треугольников в триангуляции примерно в два раза больше числа узлов. Поэтому, выражаясь другими единицами измерения, достигаемая плотность упаковки составляет 2.2 бит на треугольник триангуляции. Мы будем использовать в качестве единицы измерения количество бит на узел.

В настоящей работе предлагается ряд новых алгоритмов, являющихся дальнейшим развитием метода шелушения, позволяющих достичь плотности упаковки порядка 2.12 – 4 бит на узел триангуляции.

---

[1] Томский государственный университет, факультет информатики, пр. Ленина, д. 36, 634050, г. Томск; e-mail: skv@csd.tsu.ru

В дальнейшем в тексте статьи будут приводится полученные автором в экспериментах конкретные величины плотности упаковки и другие экспериментальные величины. Все приводимые величины получены автором на триангуляциях Делоне, содержащих 100 000 узлов, распределенных равномерно в круге.

Кроме того, автором было проведено моделирование работы описываемых алгоритмов на других распределениях и различных реальных данных. Наиболее существенные отклонения плотности упаковки (до 10 %) были отмечены на триангуляциях, не являющихся триангуляциями Делоне. В дальнейшем, чтобы не загромождать текст, такие статистические данные приводить не будем.

1. Метод шелушения. Во время работы алгоритмов шелушения [8] для сжатия и распаковки триангуляции поддерживается некоторый *граничный многоугольник*, охватывающий область обработанных треугольников. Кроме того, имеется очередь *активных ребер* триангуляции, т.е. ребер, входящих в состав граничного многоугольника, но еще не обработанных. При сохранении триангуляции в выходной поток записывается с помощью двух бит один из управляющих кодов: `VERTEX`, `SKIP`, `LEFT` или `RIGHT`. После кода `VERTEX` всегда идут координаты некоторой вершины. Рассмотрим соответствующие алгоритмы этого метода.

Алгоритм упаковки триангуляции методом шелушения.

Шаг 1. Выбирается любой треугольник в триангуляции. В выходной поток посылаются координаты трех образующих узлов этого треугольника. Три его ребра образуют начальный граничный многоугольник и входят в состав очереди активных ребер (рис. 1 (а)).

Шаг 2. Пока очередь активных ребер не пуста, извлекаем из ее начала ребро r и пытаемся увеличить граничный многоугольник за счет треугольника t, смежного с r с внешней стороны от текущей границы.

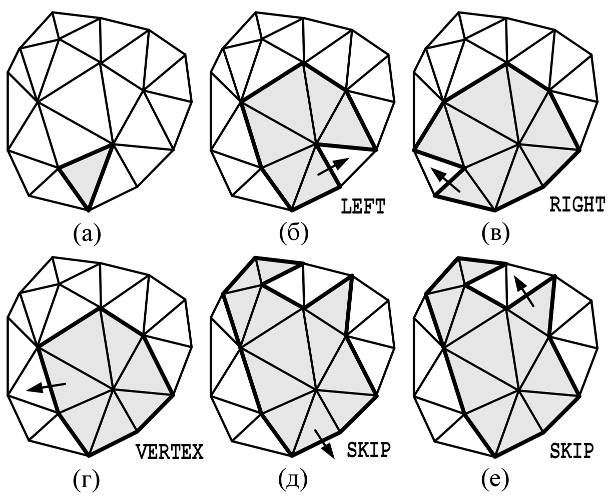

Рис. 1. Упаковка триангуляции методом шелушения: (а) — выбор начального треугольника, (б) — треугольник, замыкаемый налево, (в) — треугольник, замыкаемый направо, (г) — треугольник с новым узлом, (д) — треугольник не существует, (е) — треугольник замыкается некорректно

Шаг 2.1. Если узел n в t напротив ребра r не лежит на граничном многоугольнике, то посылаем в поток код `VERTEX` и координаты узла n, увеличиваем границу и очередь активных ребер (рис. 1 (г)).

Шаг 2.2. Если узел n в t является следующим вдоль границы слева от ребра r, то посылаем код `LEFT` и увеличиваем границу и очередь активных ребер (рис. 1 (б)).

Шаг 2.3. Если узел n в t является следующим вдоль границы справа от ребра r, то посылаем код `RIGHT` и увеличиваем границу и очередь активных ребер (рис. 1 (в)).

Шаг 2.4. Если треугольника t не существует (рис. 1 (д)) или узел n в t не является смежным вдоль границы к ребру r (рис. 1 (е)), то посылаем в поток код `SKIP`. *Конец алгоритма.*

Аналогично построен и алгоритм распаковки.

Алгоритм распаковки триангуляции.

Шаг 1. Из входного потока считываются координаты трех узлов и на них строится треугольник, ребра которого образуют начальный граничный многоугольник и входят в состав очереди активных ребер.

Шаг 2. Пока очередь активных ребер не пуста, извлекаем из ее начала ребро r и пытаемся увеличить граничный многоугольник от ребра r в соответствии со считываемым из потока управляющим кодом.

Шаг 2.1. Для кода `VERTEX`: создаем новый узел n и считываем его координаты из потока. На ребре r и узле n создаем новый треугольник, увеличиваем границу и очередь активных ребер.

Шаг 2.2. Для кода `LEFT`: создаем новый треугольник на ребре r и узле, следующим вдоль границы слева от ребра r. Увеличиваем границу и очередь активных ребер.

Шаг 2.3. Для кода `RIGHT`: создаем новый треугольник на ребре r и узле, следующим вдоль границы справа от ребра r. Увеличиваем границу и очередь активных ребер.

Шаг 2.4. Для кода `SKIP`: ничего не делаем. *Конец алгоритма.*

Трудоемкости обоих алгоритмов упаковки и распаковки являются линейными относительно размера триангуляции.

2. Анализ алгоритма шелушения. Отметим, что в методе шелушения информация о топологии и координаты узлов триангуляции сохраняются в одном потоке данных вперемешку. Такой поток данных в дальнейшем практически не сжимается никакими универсальными алгоритмами сжатия (например, алгоритм LZ77 [9], широко используемый в семействах архиваторов Zip и Zlib, на отдельных потоках данных достигает сжатия не более 1 %). Если же координаты узлов сохранять отдельно в новый второй поток, то можно достичь гораздо лучших результатов, т.к. оба полученных потока данных можно сжать затем независимо. Первый поток с кодами топологии алгоритм LZ77 сжимает уже примерно на 25 – 30 %, достигая плотности упаковки порядка трех бит на узел триангуляции.

В дальнейшем в статье мы будем предполагать, что данные упаковываются в два разных потока, соответствующих топологии триангуляции и координатам объектов. Нашей целью будем разработка новых алгоритмов упаковки первого из этих потоков.

Автором была выполнена реализация алгоритма упаковки методом шелушения с последующим тестированием на различных наборах данных. Было измерено количество кодов различного типа, выдаваемых алгоритмом шелушения. В табл. 1 приведена статистика появления кодов различного типа в алгоритме шелушения. Заметим, что код `LEFT` появляется немного чаще, чем код `RIGHT`. Это связано с порядком помещения в очередь активных ребер трех начальных ребер вокруг первого треугольника.

Одним из удобных и широко применяемых универсальных методов сжатия является кодирование Хаффмана [1]. Он обычно применяется для сжатия последовательностей символов. В нашем случае мы имеем четыре разных символа: `VERTEX`, `RIGHT`, `LEFT` и `SKIP`. В нижней строке в табл. 1 после наклонной черты приведены соответствующие им коды Хаффмана. Полученные коды Хаффмана позволяют сжать генерируемую алгоритмом шелушения последовательность при указанной статистике на 6.15 % — примерно до 4.14 бит на узел триангуляции. Отметим однако, что после сжатия Хаффмана алгоритм LZ77 уже не дает никакого эффекта.

Таблица 1

Статистика появления разных кодов в алгоритме шелушения
и соответствующие им коды Хаффмана

Статистика и код Хаффмана для				Сжатие Хаффмана
VERTEX	RIGHT	LEFT	SKIP	
45.1 % / 1	22.1 % / 01	22.7 % / 001	10.1 % / 000	6.15 %

Одним из направлений дальнейшего улучшения алгоритма шелушения может быть изменение внутренних параметров алгоритма для достижения другого соотношения статистики появления различных кодов, а следовательно, и изменения эффективности кода Хаффмана.

Заметим, что довольно большую часть генерируемых кодов составляет код `SKIP`, по сути работающий вхолостую. Снижение частоты его использования позволяет надеяться на дополнительное сжатие в пределах 10 %.

Одной из причин большого количества кодов `SKIP` является неравномерное “шелушение” треугольников. Текущая граница триангуляции в некоторых местах растет быстрее, чем в других. При этом возникают неравномерные выросты из текущей границы с большим количеством треугольников между ними. Дальнейший рост границы с этих выростов становится невозможным, что и приводит к появлению кодов `SKIP` при обработке ребер триангуляции между выростами.

В работе [8] эта проблема была отмечена косвенно. Так, ее авторы отмечают, что ими были опробованы две структуры для списка активных ребер: стек и очередь. При использовании стека выросты становятся просто огромными, и поэтому количество кодов `SKIP` увеличивается очень сильно. В связи с этим в [8] была выбрана структура списка активных ребер в виде очереди.

Ниже предлагается другой вариант выбора текущего ребра, позволяющий значительно снизить количество кодов SKIP и тем самым дополнительно улучшить сжатие.

3. Шелушение с активным выбором ребра.

3.1. Шелушение с минимальной суммой внешних углов. В предлагаемом *алгоритме шелушения с минимальной суммой углов* необходимо из списка активных ребер выбрать такую сторону AB текущего граничного многоугольника, у которой сумма двух внешних углов $\angle A + \angle B$ является минимальной среди всех активных ребер (рис. 2).

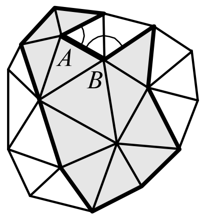

Рис. 2. Выбор активного ребра с минимальной суммой внешних углов

Данный алгоритм в первую очередь выбирает ребра в тех местах границы, где имеются глубокие вогнутости. В результате шелушение становится гораздо более равномерным, а количество кодов SKIP снижается примерно до 0.3 % от общего числа генерируемых кодов.

В табл. 2 представлена статистика появления различных кодов, а также представлены соответствующие им коды Хаффмана. Использование кодирования Хаффмана при указанной статистике кодов позволяет достичь 12.6 % сжатия, т.е. около 3.49 бит на узел триангуляции.

Таблица 2

Статистика появления разных кодов в алгоритме шелушения с минимальной суммой углов и соответствующие им коды Хаффмана

Статистика и код Хаффмана для				Сжатие Хаффмана
VERTEX	RIGHT	LEFT	SKIP	
50 % / 1	24.4 % / 001	25.3 % / 01	0.3 % / 000	12.6 %

Если вместо кодирования Хаффмана использовать алгоритм LZ77, то достигается еще лучшее сжатие — порядка 25 %, т.е. около 3.06 бит на узел триангуляции.

Для реализации процедуры выбора очередного обрабатываемого ребра будем хранить все активные ребра в сортирующем дереве [2]. Тогда при помещении ребра в список активных ребер нам потребуется время $O(\log N)$. Кроме того, при изменении текущей границы необходимо пересчитывать сумму внешних углов двух смежных ребер вокруг изменяемой части границы. Это может привести к перемещению ребер в сортирующем дереве с затратами времени также $O(\log N)$.

Таким образом, суммарная трудоемкость алгоритма шелушения с минимальной суммой углов составляет $O(N \log N)$. При этом заметим, что логарифмическая составляющая этой трудоемкости на практике весьма мала и не сильно увеличивает скорость работы алгоритма по сравнению с исходным.

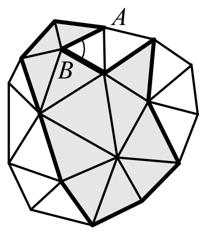

Рис. 3. Выбор активного ребра с минимальным правым внешним углом

3.2. Шелушение с минимальным правым внешним углом. В предлагаемом *алгоритме шелушения с минимальным правым внешним углом* необходимо из списка активных ребер выбрать такую сторону AB текущего граничного многоугольника, у которой внешний $\angle B$ является минимальным среди всех активных ребер (рис. 3).

Данный алгоритм выдает примерно в четыре раза больше кодов `SKIP`, чем предыдущий алгоритм, — примерно 1.2 % от общего числа генерируемых кодов. Но статистика появления различных кодов совершенно иная, в частности, код `LEFT` практически не генерируется (табл. 3). Поэтому здесь получается другой код Хаффмана, позволяющий добиться сжатия при указанной статистике примерно 3.08 бит на узел.

Таблица 3

Статистика появления разных кодов в алгоритме шелушения с минимальным правым углом и соответствующие им коды Хаффмана

Статистика и код Хаффмана для				Сжатие Хаффмана
VERTEX	RIGHT	LEFT	SKIP	
49.6 % / 1	48.5 % / 01	0.7 % / 000	1.2 % / 001	23.85 %

В случае же использования алгоритма LZ77 вместо кодирования Хаффмана достигается еще большее сжатие порядка — 44 %, т.е. около 2.27 бит на узел.

3.3. Шелушение с 4-ситуационным кодированием Хаффмана. Посмотрим, какие конфигурации ребер могут быть на границе вокруг очередного активного ребра AB. В зависимости от того, превышают ли внешние $\angle A$ и $\angle B$ текущего граничного многоугольника угол π, возможны четыре ситуации, приведенные на рис. 4. Темным цветом показаны треугольники, находящиеся внутри текущего граничного многоугольника.

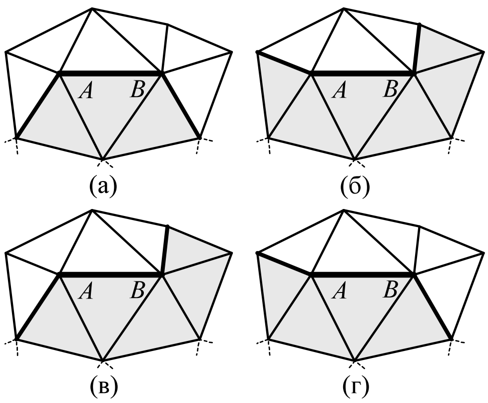

Рис. 4. Различные четыре ситуации на границе около активного ребра

Заметим, в некоторых из указанных конфигураций появление отдельных кодов невозможно вообще никогда. Например, в ситуации на рис. 4 (а) коды `LEFT` и `RIGHT` невозможны, т.к. иначе будет построен вывернутый треугольник.

Идея предлагаемого *алгоритма шелушения с 4-ситуационным кодированием Хаффмана* заключается в том, чтобы сгенерировать различные коды Хаффмана для разных ситуаций на активном ребре.

В табл. 4 приведены условия образования и статистика появления различных ситуаций при выборе активного ребра по минимуму суммы внешних углов.

В табл. 5 приведена статистика генерации различных кодов в разных ситуациях.

Таким образом, в результате применения различного кодирования Хаффмана отдельно для четырех ситуаций по сравнению с исходным алгоритмом шелушения с минимальной суммой внешних углов достигается сжатие порядка 31.1 %, т.е. около 2.76 бит на узел триангуляции.

Аналогично алгоритму шелушения с минимальной суммой внешних углов можно также использовать алгоритм с выбором минимального правого угла, однако в этом случае ситуация на рис. 4 (в) становится доминирующей, и поэтому большой выгоды от разделения на ситуации мы не получаем.

Теперь обратим внимание на то, что код `SKIP` появляется, как правило, в ситуации, изображенной на рис. 4 (а). Во всех остальных случаях он появляется очень редко. В проведенных автором экспериментах в

Таблица 4

Условия образования и статистика появления четырех ситуаций на границе в алгоритме шелушения с минимальной суммой углов

Ситуация как на	Условие на $\angle A$	Условие на $\angle B$	Статистика появления
рис. 4 (а)	$\angle A \geqslant \pi$	$\angle B \geqslant \pi$	0.4 %
рис. 4 (б)	$\angle A < \pi$	$\angle B < \pi$	27.8 %
рис. 4 (в)	$\angle A \geqslant \pi$	$\angle B < \pi$	35.3 %
рис. 4 (г)	$\angle A < \pi$	$\angle B \geqslant \pi$	36.5 %

Таблица 5

Статистика генерации разных кодов в алгоритме шелушения с минимальной суммой углов в различных ситуациях, соответствующие им коды и уровень сжатия Хаффмана

Ситуация как на	Статистика и код Хаффмана для				Сжатие Хаффмана
	`VERTEX`	`RIGHT`	`LEFT`	`SKIP`	
рис. 4 (а)	7.9 % / 0	—	—	92.1 % / 0	50 %
рис. 4 (б)	77.6 % / 1	11.4 % / 01	11 % / 001	0 % / 000	33.5 %
рис. 4 (в)	40 % / 01	60 % / 1	—	0 % / 00	30 %
рис. 4 (г)	39.1 % / 01	—	60.1 % / 1	0 % / 00	30 %

ситуациях, представленных на рис. 4 (б)–(г), код `SKIP` появлялся с частотой не более пяти случаев на 10 000 узлов триангуляции. В связи с этим представляется разумным в этих ситуациях коды `SKIP` сохранять в новом отдельном потоке данных, просто указав отдельно порядковые номера возникновения кода `SKIP` внутри последовательности основного потока кодирования топологии. Тогда в основном потоке не нужно их сохранять, а поэтому можно и перестроить код Хаффмана.

В табл. 6 приведены модифицированные коды Хаффмана и новые уровни сжатия. Таким образом, в результате применения 4-ситуационного кодирования Хаффмана и индивидуального кодирования кодов `SKIP` по сравнению с исходным алгоритмом шелушения с минимальной суммой внешних углов достигается сжатие порядка 46.9 %, т.е. около 2.12 бит на узел триангуляции (в том числе с учетом затрат на хранение номеров кодов `SKIP` в отдельном потоке).

Таблица 6

Статистика появления разных кодов в алгоритме шелушения с минимальной суммой углов в различных ситуациях и соответствующие им коды и уровень сжатия Хаффмана при кодировании кодов `SKIP` в отдельном потоке

Ситуация как на	Статистика и код Хаффмана для				Сжатие Хаффмана
	`VERTEX`	`RIGHT`	`LEFT`	`SKIP`	
рис. 4 (а)	7.9 % / 0	—	—	92.1 % / 1	50 %
рис. 4 (б)	77.6 % / 1	11.4 % / 01	11 % / 00	—	38.8 %
рис. 4 (в)	40 % / 0	60 % / 1	—	—	50 %
рис. 4 (г)	39.1 % / 0	—	60.1 % / 1	—	50 %

3.4. Шелушение с 6-ситуационным кодированием Хаффмана. Аналогично можно рассмотреть другие варианты ситуаций. Например, на рис. 5 представлены шесть ситуаций. Первые две их них (рис. 5 (а)–(б)) аналогичны ситуациям на рис. 4 (а)–(б). Ситуации на рис. 4 (в)–(г) мы разделяем на два подслучая, в зависимости от того, меньше ли внешний угол, чем $\pi/2$, или больше; тем самым, получаем четыре новые ситуации (рис. 5 (в)–(е)).

В предлагаемом *алгоритме шелушения с 6-ситуационным кодированием Хаффмана* используются приведенные шесть ситуаций для генерации различных кодов Хаффмана на активном ребре.

В табл. 7 приведены условия образования и статистика появления различных ситуаций при выборе активного ребра по минимуму суммы внешних углов.

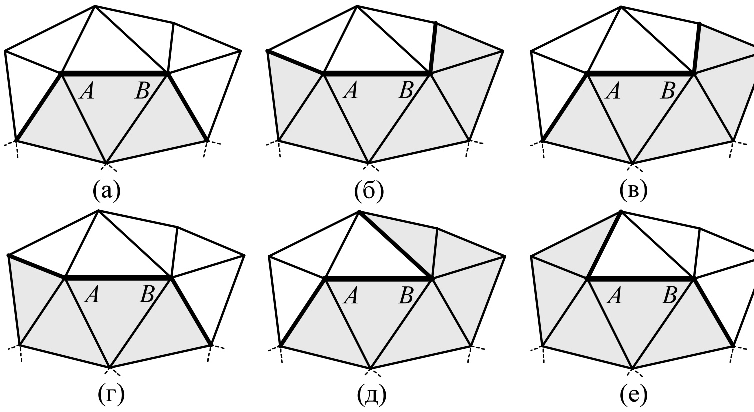

Рис. 5. Различные шесть ситуаций на границе около активного ребра

Таблица 7

Условия образования и статистика появления шести ситуаций на границе в алгоритме шелушения с минимальной суммой углов

Ситуация как на	Условие на $\angle A$	Условие на $\angle B$	Статистика появления
рис. 5 (а)	$\angle A \geqslant \pi$	$\angle B \geqslant \pi$	0.4 %
рис. 5 (б)	$\angle A < \pi$	$\angle B < \pi$	27.8 %
рис. 5 (в)	$\angle A \geqslant \pi$	$\angle B \in [\pi/2, \pi)$	11.2 %
рис. 5 (г)	$\angle A \in [\pi/2, \pi)$	$\angle B \geqslant \pi$	11.3 %
рис. 5 (д)	$\angle A \geqslant \pi$	$\angle B < \pi/2$	24.1 %
рис. 5 (е)	$\angle A < \pi/2$	$\angle B \geqslant \pi$	25.2 %

В табл. 8 приведена статистика появления различных кодов в разных ситуациях рис. 5, а также соответствующие коды Хаффмана. Таким образом, в результате применения кодирования Хаффмана отдельно для шести ситуаций по сравнению с исходным алгоритмом шелушения с минимальной суммой внешних углов достигается сжатие порядка 37.9 %, т.е. около 2.54 бит на узел триангуляции.

Таблица 8

Статистика появления разных кодов в алгоритме шелушения с минимальной суммой углов в различных ситуациях, соответствующие им коды и уровень сжатия Хаффмана

Ситуация как на	Статистика и код Хаффмана для				Сжатие Хаффмана
	`VERTEX`	`RIGHT`	`LEFT`	`SKIP`	
рис. 5 (а)	7.9 % / 0	—	—	92.1 % / 1	50 %
рис. 5 (б)	77.6 % / 1	11.4 % / 01	11 % / 001	0 % / 000	33.5 %
рис. 5 (в)	74.4 % / 1	25.6 % / 01	—	0 % / 00	37.2 %
рис. 5 (г)	73.2 % / 1	—	26.8 % / 01	0 % / 00	36.6 %
рис. 5 (д)	24.1 % / 01	75.9 % / 1	—	0% / 00	37.5 %
рис. 5 (е)	23.9 % / 01	—	76.1 % / 1	0 % /00	38.1 %

Так же как и в 4-ситуационном кодировании, обратим внимание на то, что код `SKIP` появляется, как правило, в ситуации, изображенной на рис. 5 (а). Во всех остальных случаях он появляется очень редко (не более пяти случаев на 10 000 узлов триангуляции). В связи с этим сохраним коды `SKIP` в новом отдельном потоке, указав отдельно порядковые номера возникновения кода `SKIP` внутри последовательности основного потока кодирования топологии.

В табл. 9 приведены модифицированные коды Хаффмана и новые уровни сжатия. Отметим, что почти все ситуации требуют для своего кодирования только один бит, кроме ситуации, изображенной на рис. 5 (б). Та же самая картина наблюдалась и в 4-ситуационном кодировании, когда для кодирования нужен только один бит, кроме ситуации, представленной на рис. 4 (б). Однако ситуации на рис. 5 (б) и на рис. 4 (б) одинаковы; поэтому вероятности их появления одинаковы, а следовательно, и уровень сжатия будет таким же. Поэтому в результате применения 6-ситуационного кодирования Хаффмана и индивидуального кодирования кодов `SKIP` по сравнению с исходным алгоритмом шелушения с минимальной суммой внешних углов достигается сжатие порядка 46.9 %, т.е. около 2.12 бит на узел триангуляции (в том числе с учетом затрат на хранение номеров кодов `SKIP` в отдельном потоке).

Таблица 9

Статистика появления разных кодов в алгоритме шелушения с минимальной суммой углов в различных ситуациях, соответствующие им коды Хаффмана при кодировании кодов `SKIP` в отдельном потоке и достигаемый уровень сжатия Хаффмана

Ситуация как на	Статистика и код Хаффмана для				Сжатие Хаффмана
	`VERTEX`	`RIGHT`	`LEFT`	`SKIP`	
рис. 5 (а)	7.9 % / 0	—	—	92.1 % / 1	50 %
рис. 5 (б)	77.6 % / 1	11.4 % / 01	11 % / 00	—	38.8 %
рис. 5 (в)	74.4 % / 1	25.6 % / 0	—	—	50 %
рис. 5 (г)	73.2 % / 1	—	26.8 % / 0	—	50 %
рис. 5 (д)	24.1 % / 0	75.9 % / 1	—	—	50 %
рис. 5 (е)	23.9 % / 0	—	76.1 % / 1	—	50 %

4. Заключение. В табл. 10 приведены плотности сжатия и трудоемкости работы всех рассмотренных в настоящей статье алгоритмов. В целом следует выделить алгоритм шелушения с минимальным суммой внешних углов с ситуационным кодированием Хаффмана и отдельным сжатием кодов `SKIP`.

Таблица 10

Статистика появления разных кодов в алгоритме шелушения с минимальной суммой углов в различных ситуациях и соответствующие им коды Хаффмана

Название алгоритма	Плотность сжатия, бит на узел триангуляции	Трудоемкость сжатия
Шелушение [8]	4.41	$O(N)$
Шелушение + упаковка LZ77	3.57	$O(N)$
Шелушение + кодирование Хаффмана	4.14	$O(N)$
Шелушение с минимальной суммой углов	4.00	$O(N \log N)$
Шелушение с минимальной суммой углов + упаковка LZ77	3.06	$O(N \log N)$
Шелушение с минимальной суммой углов кодирование Хаффмана	3.49	$O(N \log N)$
Шелушение с минимальной суммой углов + 4-ситуационный код Хаффмана	2.76	$O(N \log N)$
Шелушение с минимальной суммой углов + 4-ситуационный код Хаффмана + отдельное сжатие `SKIP` для ситуаций на рис. 4 (б)–(г)	**2.12**	$O(N \log N)$
Шелушение с минимальной суммой углов + 6-ситуационный код Хаффмана	2.54	$O(N \log N)$
Шелушение с минимальной суммой углов + 6-ситуационный код Хаффмана + отдельное сжатие `SKIP` для ситуаций на рис. 5 (б)–(е)	**2.12**	$O(N \log N)$
Шелушение с минимальным правым углом	4.04	$O(N \log N)$
Шелушение с минимальным правым углом + упаковка LZ77	2.27	$O(N \log N)$
Шелушение с минимальным правым углом + кодирование Хаффмана	3.08	$O(N \log N)$

В настоящей работе предложены новые подходы к упаковке топологии триангуляции. По сравнению с другими ранее известными алгоритмами, предложенные алгоритмы шелушения с активным выбором очередного обрабатываемого ребра позволяют достигнуть более чем в два раза большей степени сжатия. Полученные алгоритмы имеют трудоемкость работы $O(N \log N)$, реально показывая на практике высокую скорость выполнения.

Однако предложенный подход обладает и некоторым недостатком. Так, совместно с предложенными алгоритмами теперь нельзя использовать распространенные алгоритмы сжатия координат узлов триангуляции с потерями, т.к. для правильной работы алгоритма распаковки необходимо иметь в точности те же координаты, что и при упаковке. Незначительные изменения координат при упаковке потенциально могут привести к другому порядку выбора активных ребер во время распаковки и, как следствие, к разрушению триангуляции. Поэтому для сжатия координат необходимо разработать новые методы, учитывающие специфику представленных в работе алгоритмов.

СПИСОК ЛИТЕРАТУРЫ

1. *Кнут Д.* Искусство программирования для ЭВМ. Т. 2. Получисленные алгоритмы. М.: Мир, 1977.
2. *Кнут Д.* Искусство программирования для ЭВМ. Т. 3. Сортировка и поиск. М.: Мир, 1978.
3. *Препарата Ф., Шеймос М.* Вычислительная геометрия: Введение. М.: Мир, 1989.
4. *Скворцов А.В.* Обзор алгоритмов построения триангуляции Делоне // Вычислительные методы и программирование. 2002. **3**, № 1. 18–43 (http://num-meth.srcc.msu.su).
5. *Chow M.M.* Optimized geometry compression for real-time rendering // IEEE Visualization Proceedings. Phoenix. 1997. 347–354.
6. *Deering M.* Geometry compression // Computer Graphics. Proceedings of ACM SIGGRAPH. Los Angeles. 1995. 13–20.
7. *Evans F., Skiena S., Varshney A.* Optimizing triangle strips for fast rendering // IEEE Visualization Proceedings. San Francisco. 1996. 319–326.
8. *De Floriani L., Magillo P., Puppo E.* Compressing triangulated irregular networks // Geoinformatica. 2000. **1**, N 4. 67–88.
9. *Ziv J., Lempel A.* A universal algorithm for sequential data compression // IEEE Transactions on Information Theory. 1977. **23**, N 3. 337–343.

Поступила в редакцию
30.03.2002