УДК 681.518

Н.С. Мирза, А.В. Скворцов, Р.В. Чаднов

ВИЗУАЛИЗАЦИЯ СВЕРХБОЛЬШИХ ПОВЕРХНОСТЕЙ

В данной работе рассматриваются алгоритмы, позволяющие эффективно работать со сверхбольшими поверхностями. В основе разработанных алгоритмов построения и визуализации больших поверхностей лежит особая структура – «мультитриангуляция», которая позволяет получить поверхность с различной степенью детализации. Авторами предлагается модифицировать структуру мультитриангуляции таким образом, чтобы можно было работать со сверхбольшими моделями поверхностей, т.е. моделями данных, которые заведомо не умещаются в оперативной памяти компьютера.

На сегодняшний день очень много графических программных систем используют цифровые модели поверхности. Для задач ГИС, САПР, систем трехмерного моделирования и других графических систем очень актуальной является проблема обработки больших (детализированных) поверхностей. Для визуализации огромных трехмерных моделей с хорошим качеством используются очень трудоемкие вычисления, требующие мощного аппаратного обеспечения, а для задачи визуализации сверхбольших поверхностей в режиме реального времени до сих пор не было найдено решения.

1. ОСНОВНЫЕ ПОНЯТИЯ И ПОСТАНОВКА ЗАДАЧИ

Определение. Поверхностью будем называть однозначную функцию высот от планового положения точек.

Определение. *Сверхбольшой* поверхностью будем называть такую поверхность, исходные данные которой не умещаются в оперативной памяти компьютера.

Для представления поверхностей в 3D-виде, как правило, используют *неравномерную сеть треугольников* (*TIN*), представляющую собой набор треугольников, образующих в проекции на ось *XY триангуляцию* – планарный граф, все конечные грани которого являются треугольниками [1]. Таким образом, *TIN* – это триангуляция, каждому узлу которой поставлена в соответствие его высота (координата *Z*). Для работы с *TIN*, построенной по огромному массиву входных данных, необходимо иметь мощное аппаратное обеспечение или использовать специальные алгоритмы упрощения *TIN* [1]. Но использование алгоритмов упрощения изначально подразумевает необходимость нахождения некоторого компромисса между быстродействием обработки поверхности и качеством получаемых результатов. Данная проблема решается с помощью использования *мультитриангуляции (МТ)*, которую нестрого определяют как особую структуру, представляющую собой набор фрагментов триангуляций, образующих ориентированный граф без циклов. Трехмерные графические модели множественного разрешения, основанные на мультитриангуляции, позволяют производить обработку поверхностей с заданным уровнем детализации. Это значит, что детализация модели (число треугольников в *TIN*) устанавливается согласно некоторому критерию. Таким образом, подобрав подходящий критерий, можно получить модель, скорость обработки и качество которой будут очень высоки.

Для более строгого определения мультитриангуляции нужно ввести ряд терминов.

Определение. Пусть заданы две триангуляции T и T_i, такие, что T_i занимает область меньшую, нежели T. Тогда T и T_i называются *совместимыми* [2], если в T существует подмножество треугольников T', занимающих в точности ту же область, что и T_i, и при этом T' является триангуляцией. В таком случае, T_i называется *фрагментом* (относительно T), а T' – *покрываемой областью* T_i.

Определение. T_i называется *минимально* совместимым с T, если в T_i нет подтриангуляции, совместимой с T.

Определение. *Локальной модификацией* T через фрагмент T_i является операция замены треугольников T' треугольниками T_i и обозначается $T \oplus T_i$.

Определение. Если все T_i совместимы с T, то последовательность преобразований $t = (T_0, ..., T_n)$ называется *совместимой последовательностью триангуляций*.

Определение. В терминах, определенных выше, ***мультитриангуляция*** – направленный граф без циклов, вершинами которого являются элементы совместимой последовательности триангуляций [2].

Определение. *Корнем* графа является полная триангуляция (самая примитивная), а *стоком* – самая примитивная (полная) для МТ, построенной с помощью алгоритмов упрощения (детализации).

Мультитриангуляцию легко представить в виде следующих структур данных:

1. *Список фрагментов*: каждый фрагмент содержит список составляющих его треугольников и список указателей на треугольники покрываемой части.

2. С*писок треугольников*: каждый треугольник содержит ссылки на три образующих его узла, ссылки на фрагмент, где он содержится (верхний фрагмент) и на фрагмент, который содержит его в покрываемой части (нижний фрагмент).

3. *Список вершин*: каждая вершина содержит необходимый набор геометрических данных.

На базе МТ разработаны алгоритмы, позволяющие строить триангуляцию переменного разрешения произвольных поверхностей [3]. Реализация одного из таких алгоритмов существует в DirectX 8.0 API (Progressive Mesh) и поэтому имеет широкое распространение в графических системах, работающих с помощью DirectX.

Однако все разработанные алгоритмы работают, если вся структура МТ находится в ОЗУ компьютера. Но часто для решения практически важных задач необходимо отобразить поверхность, большую, чем та, что умещается в оперативной памяти.

Таким образом, необходимо разработать алгоритм, работающий для больших объемов входных данных и позволяющий быстро отображать поверхность на экране без видимых погрешностей в качестве.

2. ПОСТРОЕНИЕ СВЕРХБОЛЬШОЙ МУЛЬТИТРИАНГУЛЯЦИИ

Основная идея построения структуры МТ для работы с большими объемами входных данных заключается в том, что в памяти можно держать только те фрагменты, которые вероятнее всего будут использованы, т.е. те, которые войдут в результирующий список треугольников для вывода на экран. Очевидно, что такими фрагментами являются те, что потенциально могут удовлетворять заданному критерию.

Поэтому структура МТ должна позволять в реальном режиме времени загружать нужные фрагменты и выгружать ненужные.

Применяя алгоритмы построения МТ [4], структура МТ получается таковой, что при детализации одного треугольника может возникнуть необходимость детализировать еще некоторый ряд произвольных треугольников. Это значит, что для работы алгоритма выбора результирующего списка треугольников необходимым условием будет наличие в памяти всей структуры МТ. Поэтому для работы алгоритма на больших объемах входных данных структура графа МТ должна быть изменена. Чтобы уменьшить взаимозависимости треугольников, нужно попытаться упорядочить граф МТ.

Авторами предлагается вариант упорядочения, при котором граф МТ должен состоять из нескольких ***независимых*** подграфов, что позволит оперативно работать с каждым из них. Для этого предлагается следующий алгоритм построения МТ:

Алгоритм построения МТ

Шаг 1. Все множество исходных точек триангуляции разбивается на части, каждая из которых представляет собой некоторый прямоугольник.

Шаг 2. Внутри каждого прямоугольника осуществляется построение МТ алгоритмом, описанным в [4].

Шаг 3. Оставшаяся триангуляция упрощается до самого грубого фрагмента.

Конец алгоритма.

Трудоемкость данного алгоритма – $O(N)$ и получается она следующим образом: за время $O(N)$ можно разбить все точки триангуляции на части, затем в каждой части происходит упрощение триангуляции за время $O(N_i)$, где N_i – число удаляемых вершин, так как $\sum_i N_i \le N$, то в итоге трудоемкость составляет $O(N)$, где N – число вершин в триангуляции.

Таким образом, на выходе алгоритма получится структура МТ, схематично изображенная на рис. 1.

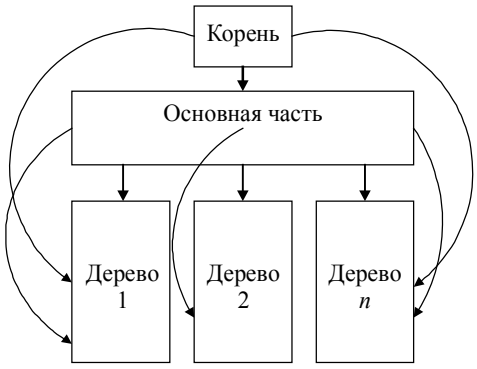


Рис. 1. Структура мультитриангуляции

Как видно из рисунка, получается несколько частей: «деревья МТ» и основная часть (по сути, *дерево 0*), которые можно загружать или выгружать независимо друг от друга. Следует отметить, что термин «дерево» используется в данном случае из соображения удобства и не подразумевает наличия свойств, определенных в теории графов.

Определение. Мультитриангуляцию, имеющую структуру, согласно рис. 1, будем называть *разделенной*.

Теперь рассмотрим вопрос о том, на какие прямоугольники необходимо разделить исходное множество точек, так как это существенно влияет на скорость и корректность работы алгоритма. Число прямоугольников определяет число деревьев в МТ, если это число слишком мало, то возможно появление областей с недостаточной детализацией, а, если слишком велико – объем памяти, занимаемый МТ, будет необоснованно большим.

Определение. Если в разделенной МТ число вершин во всех деревьях и в основной части одинаково, то такую МТ будем называть *сбалансированной*.

При использовании сбалансированной структуры МТ можно добиться хорошего соотношения между качеством визуализации и объемом занимаемой памяти МТ.

Таким образом, число деревьев для сбалансированной МТ, построенной на множестве из N узлов, можно вычислить из следующего соотношения:

$$TreeCount \cdot EdgePointCount(N, TreeCount) = \\ = \frac{N}{TreeCount},$$

где *EdgePointCount* – это среднее число узлов триангуляции, лежащих на границе области построения деревьев, которые будут использованы при построении основной части МТ.

Согласно ряду экспериментальных исследований, функция *EdgePointCount* имеет вид

$$EdgePointCount = c \cdot \sqrt[3]{\frac{N}{TreeCount}},$$

где c – константа, зависящая от вида распределения исходных точек.

Таким образом, формула для расчета числа деревьев для сбалансированной МТ будет иметь вид

$$TreeCount = \left\lceil \sqrt[5]{\frac{N^2}{c^3}} \right\rceil.$$

Зная число деревьев в МТ, разбиение на прямоугольники производится с помощью следующего алгоритма.

Алгоритм разбиения исходного множества точек.

Входные данные:

1. *Nodes* – множество вершин для разделения на части с числом вершин *Count* и ограничивающим прямоугольником *Bounds*.
2. N – число узлов в исходной триангуляции.
3. *TreeCount* – требуемое число деревьев в МТ.

Выходные данные: прямоугольники с числом узлов, примерно равным $\frac{N}{TreeCount}$.

Структура алгоритма:

Шаг 1. Если

$$\left|Count - \frac{N}{TreeCount}\right| \le \left|\frac{Count}{2} - \frac{N}{TreeCount}\right|,$$

то прямоугольник *Bounds* является прямоугольником результирующего разбиения.

Шаг 2. Находим геометрический центр множества точек на входе.

Шаг 3. По геометрическому центру происходит разделение исходного прямоугольника *Bounds* на два (по самой длинной стороне).

Шаг 4. Для полученных прямоугольников рекурсивно выполняем алгоритм, начиная с *Шага 1*.

Конец алгоритма.

Трудоемкость данного алгоритма равна $O(N \log N)$, так как нахождение геометрического центра и разбиения осуществляется за линейное время, а затем задача делится на две с размерностью в два раза меньше, чем исходная.

Таким образом, полная трудоемкость алгоритма построения сбалансированной МТ равна $O(N \log N)$.

3. ОПРЕДЕЛЕНИЕ ЧАСТИ МТ, ПОДЛЕЖАЩЕЙ ВИЗУАЛИЗАЦИИ

Чтобы была возможность выгружать фрагменты, когда они не нужны, и загружать, когда они нужны, необходимо сохранять структуру МТ на жесткий диск, а затем загружать ее. При работе со сверхбольшими МТ необходимо загружать структуру МТ частично: лишь те части, что вероятнее всего будут удовлетворять критерию выбора треугольников. Например, деревья, которые находятся вблизи наблюдателя, нужно загружать с максимальной степенью детализации, и чем дальше дерево от наблюдателя, тем ниже степень его детализации.

Чтобы определить, с какой степенью детализации следует загружать каждое из деревьев МТ, необходимо воспользоваться эффективным способом пространственного поиска объектов на плоскости, таким, как, например, *R*-дерево [5].

При загрузке структуры МТ с диска необходимо построить *R*-дерево. В качестве его элементов будет выступать структура, состоящая из номера дерева и указателя на открытый файл, содержащий дерево МТ. Таким образом, после построения *R*-дерева целесообразно загрузить лишь основную часть МТ. Остальные деревья будут загружаться в соответствии со следующим алгоритмом:

Алгоритм загрузки деревьев МТ.

Определим функцию *RequestedLOD(TreeNumber: integer)*, которая возвращает запрашиваемый (требуемый) уровень детализации дерева с номером *TreeNumber* в зависимости от критерия визуализации. Запрашиваемый уровень детализации может меняться от нуля до единицы, где ноль соответствует полной выгрузке дерева из памяти, а единица – полной загрузке.

Структуры данных:

1. *R*-дерево, построенное на этапе загрузки МТ с диска.

2. Прямоугольник *InterestZone*, представляющий собой зону интереса, в которой уровень загрузки деревьев МТ должен быть отличен от нуля.

3. Разделенная мультитриангуляция.

Шаг 1. При помощи *R*-дерева производится поиск всех деревьев МТ, попавших в зону интереса.

Шаг 2. Все деревья, загруженные в память, но не попавшие в зону интереса, полностью выгружаются из памяти.

Шаг 3. При помощи функции *RequestLOD* для всех деревьев, попавших в зону интереса, устанавливается запрашиваемый уровень детализации в соответствии с которым дерево загружается в память.

Конец алгоритма.

Что касается преимущества данного алгоритма перед простым линейным просмотром всех деревьев, то оно заключается в том, что для больших поверхностей зона интереса очень невелика, таким образом, в каждый момент времени обрабатывается лишь небольшое число деревьев.

4. Загрузка и выгрузка фрагментов МТ по требованию.

Определив для каждого из деревьев МТ уровень детализации, нужно научиться загружать дерево в соответствии с ним. Далее авторами предлагается следующий алгоритм загрузки дерева с заданным уровнем детализации:

4. АЛГОРИТМ ЗАГРУЗКИ ДЕРЕВА С ЗАДАННЫМ УРОВНЕМ ДЕТАЛИЗАЦИИ

Шаг 1. Если действительный уровень детализации равен требуемому, то алгоритм заканчивает работу. Иначе переход на *Шаг 2*.

Шаг 2. Если действительный уровень детализации больше требуемого, то переход на *Шаг 3*, иначе на *Шаг 4*.

Шаг 3. Если *Число загруженных фрагментов* дерева больше, чем *Порог загрузки* × *Запрашиваемый уровень детализации* × *Максимальное число фрагментов* в дереве, то всем ссылкам на нижний фрагмент треугольников из покрываемой части лишних фрагментов присваивается указатель на сток МТ. Удаляем все лишние фрагменты и освобождаем память.

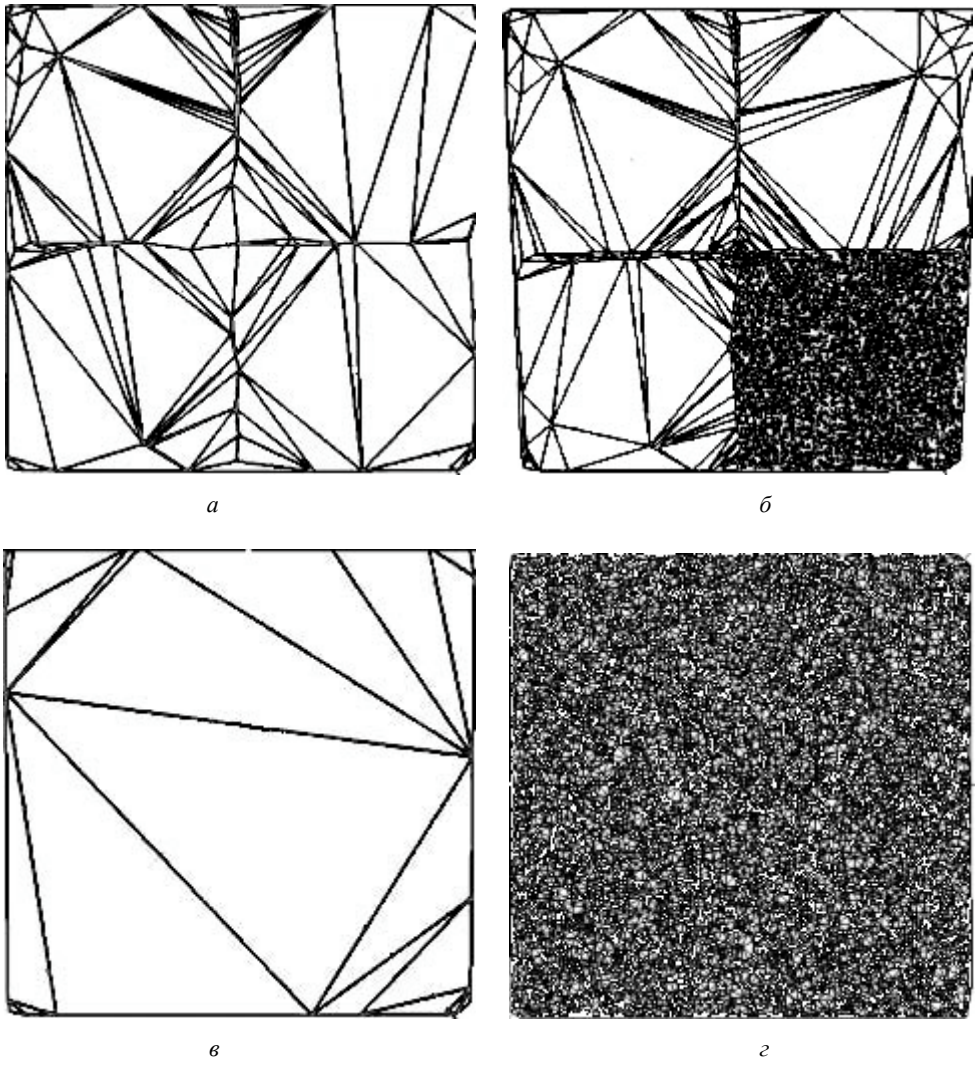
Шаг 4. Если *Число загруженных фрагментов* дерева меньше, чем *Порог загрузки* × *Запрашиваемый уровень детализации* × *Максимальное число фрагментов* в дереве, то производится выделение памяти под нужные фрагменты, после чего они загружаются из файла.

Конец алгоритма.

В данном алгоритме используется *Порог загрузки*, необходимый для повышения скорости работы. Выигрыш получается из минимизации обращений к жесткому диску.

5. ЭКСПЕРИМЕНТАЛЬНОЕ МОДЕЛИРОВАНИЕ

Пример экспериментального моделирования приведен на рис. 2 (на котором изображена разделенная МТ с различной степенью загрузки деревьев).



а *б*

в *г*

Рис. 2. Разделенная МТ с различной степенью загрузки деревьев: *а* – МТ, в которой загружена только основная часть; *б* – МТ, в которой загружена основная часть и одно дерево; *в* – корень МТ; *г* – сток МТ

ЗАКЛЮЧЕНИЕ

На основе структуры мультитриангуляции авторами впервые предлагаются алгоритмы построения и визуализации сверхбольших моделей поверхностей на основе мультитриангуляции, позволяющие интерактивно работать с моделями, существенно превосходящими объем доступной оперативной памяти. При экспериментальном моделировании работы разработанных алгоритмов получены оценки трудоемкости алгоритмов и оценки используемой оперативной и вторичной (дисковой) памяти, которые показали высокую скорость визуализации поверхности при приемлемом объеме занимаемой оперативной памяти.

ЛИТЕРАТУРА

1. *Скворцов А.В.* Триангуляция Делоне и ее применение. Томск: Изд-во Том. ун-та, 2002. 128 с.
2. *Puppo E.* Variable resolution triangulations // Computational Geometry. 1998. V. 11. P. 219 – 238.
3. *Hoppe H.* Progressive Meshes // Computer Graphics. 1996. P. 99 – 108.
4. *De Floriani L., Magillo P., Puppo E.* Building and traversing a surface at variable resolution // Proc. Conf. On Visualization '97. 1997. P. 18 – 24.
5. *Скворцов А.В.* Глобальные алгоритмы R-деревьев // Геоинформатика: Теория и практика. Вып. 1. Томск: Изд-во Том. ун-та, 1998. С. 67 – 83.

Статья представлена кафедрой теоретических основ информатики факультета информатики Томского государственного университета, поступила в научную редакцию «Информатика» 18 мая 2005 г.