Построение полигональных сетей для многомерных графических моделей с высокой детализацией в ГИС

А.Ю. Ледяев

Аннотация. В статье описывается метод работы с полигональными сетями моделируемых объектов в 3D-движке, который обеспечивает достаточно высокую производительность для работы в реальном времени с графическими моделями с высокой детализацией, содержащими многие миллионы граней.

Ключевые слова: географическая информационная система, трехмерные изображения, полигональная сеть, моделирование объектов, параллельные вычисления, кластеры.

Введение

Географические информационные системы (ГИС) за 50 лет своего существования проникли во многие области деятельности человека. Особенно динамично ГИС стали развиваться в двухтысячные годы. Одной из причин стал быстрый рост вычислительных возможностей настольных и мобильных аппаратных систем. Одновременно с развитием систем отображения прикладного трехмерного окружения пользователя (далее называемых графическими движками) используемые в них данные для описания графических моделей приобретают все более и более сложный вид и организуются в огромные многомерные структуры. Возникает вопрос об интерактивном использовании многомерных пространственных данных в реальном времени. Имеющийся уровень развития вычислительной техники не позволяет проводить расчетные операции в реальном времени над пространственными данными в объеме 10Г байт и более.

1. Современные графические движки

На сегодняшний день основными недостатками сетевого программного обеспечения (ПО) являются: нехватка доступных вычислительных мощностей и ресурсов серверного и пользовательского оборудования для обработки существующих потоков данных; недостаточная надежность работы распределенных сетевых систем для работы в режиме 24x7; низкая скорость и целостность передачи больших объемов информации по сетям. Наметилась тенденция выполнения конечной обработки данных на стороне пользователя в клиентских приложениях с расширенной функциональностью, не зависящей от серверов, называемых «толстыми» клиентами. Современные 3D-движки должны быть интерактивными и визуализировать не просто статичные картинки, а давать возможность потоковой симуляции в виртуальной реальности графических моделей или их примитивов в реальном времени. Такая тенденция становиться все более популярной. Наиболее ярким примером сегодня является система NASA Eyes on the Solar System (http://eyes.nasa.gov). Существует потребность разработки метода создания графических моделей, которые можно обрабатывать и визуализировать на карте в ГИС или в любой другой современной автоматизированной системе (АС) реального времени. Таким образом, необходимо оптимизировать методы и алгоритмы работы с многомерными

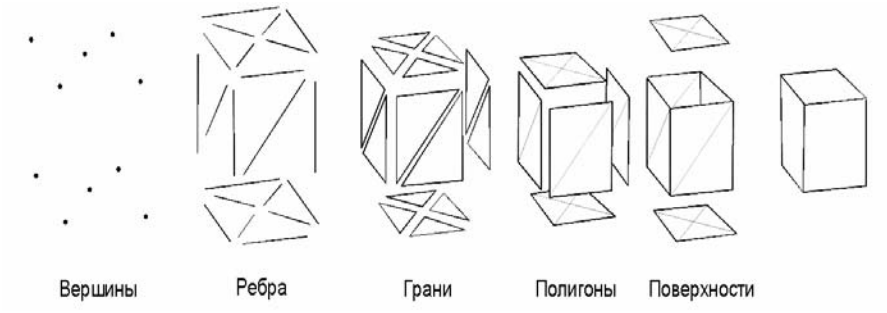

Рис. 1. Типовая сборка объектов

пространственными данными и разработать новый метод построения многомерных графических моделей с высокой детализацией для визуализации с учетом жестких ограничений, накладываемых со стороны пользовательского вычислительного оборудования, например таких, как доступный объем оперативной и видео памяти, скорость обработки данных в центральном и графическом процессорах. Для интерактивного использования в реальном времени необходимо обеспечить достаточную скорость обработки и визуализации используемых данных. Если время обработки данных будет превышать квант времени, необходимый для визуализации отдельного кадра изображения, то будет ощущаться эффект «подтормаживания» ПО системы и у пользователя пропадет ощущение интерактивной работы с системой.

Анализ существующих методов для работы с полигональными сетями [2], используемых в наиболее распространенных программных продуктах Away3D, Blender, OGRE, Unity, и личный опыт работы с ними позволил сделать следующие выводы о способах построения полигональных сетей (Рис. 1).

Графический движок Away3D [4] разработан для многомерной визуализации в Flash-технологии. Для создания 3D-объектов используются четыре базовых примитива: вершина, ребро, треугольная грань, 3D-спрайт (свободно перемещающееся по экрану растровое изображение). Движок не умеет визуализировать вершины, которые являются только математическим описанием концов ребер. Уникальной особенностью данного движка является использование 3D-спрайта в качестве базового примитива. 3D-спрайт представляет собой 2D набор треугольных граней, всегда обращенных к камере. Такой подход не применяется ни в одном другом современном графическом движке. Последовательность сборки примитивов в данном движке: вершины → ребра → грани → объекты.

Графический движок Blender [5] входит в состав одноименного программного пакета создания цифрового 3D контента, являющегося одним из самых популярных на сегодняшний день. Для создания 3D объектов используются четыре базовых примитива: вершина, ребро, треугольная грань, четырехугольная грань. Движок умеет визуализировать любые примитивы, из которых состоит объект. Последовательность сборки примитивов в данном движке: вершины → ребра → грани → объекты.

Графический движок OGRE [7] является исключением из общего правила построения полигональных сетей. Движок использует внутренний XML-формат описания объектов, не совместимый с общепринятыми форматами описания полигональных сетей. Движок не умеет визуализировать вершины и ребра, в связи с чем абсолютно не пригоден для применения в научных системах визуализации. Движок OGRE работает только с наборами треугольников. Любой объект в движке представляется бинарным массивом, в котором хранятся координаты вершин, нормали, цвета и т.п. В описании объекта присутствует набор указателей для выбора параметров примитивов из массива при построении треугольника. Для данного графического движка существует ряд сторонних конвертеров, не входящих в его состав. С их помощью любое общепринятое описание объекта разбивается на набор треугольников и сохраняется во внутреннем формате представления полигональных сетей.

Графический движок Unity [6] является наиболее популярным и входит в состав одноименного игрового движка. Для создания 3D объектов

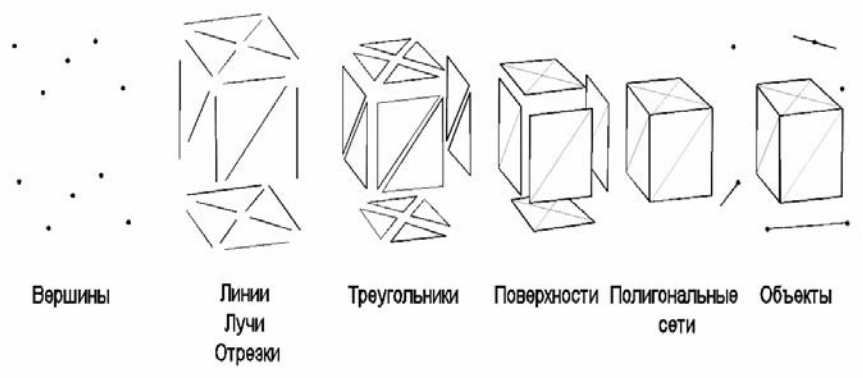

Рис. 2. Модифицированная сборка объектов

используются базовые примитивы: вершина, ребро, треугольная грань, 2D полигоны, полигональная сеть. При импорте любого 3D объекта движок разбивает полигоны на треугольные грани. Полигоны в Unity могут быть только плоскими. Движок умеет визуализировать любые примитивы, из которых состоит объект. Последовательность сборки примитивов в данном движке: вершины → ребра → треугольные грани → полигоны → полигональные сети → объекты. Обход примитивов в графических движках Away3D, Blender и Unity осуществляется с использованием списков граней и вершин.

Для оптимизации работы любой вычислительной системы необходимо внимательно подходить к вопросу организации структур данных и оптимизации обрабатывающих их алгоритмов [1, 9, 10]. В попытке решить проблему нехватки вычислительных ресурсов при работе с массивами данных и построении многомерных графических моделей в реальном времени необходимо разработать метод хранения полигональных сетей многомерных объектов и сопутствующий набор алгоритмов обхода, которые учитывают специфику очень высокой детализации объектов в ГИС, необходимость визуализировать отдельные базовые примитивы и недостаток вычислительных ресурсов.

2. Альтернативный метод построения полигональных сетей многомерных объектов

Для представления полигональных сетей используются разнообразные способы хранения вершин, ребер и граней моделируемых объектов. Например, наиболее популярными представлениями полигональных сетей являются: список вершин, список граней, крылатое представление, полуреберное представление, представление через таблицы углов. Каждое из представлений имеет как свои преимущества, так и недостатки. Все представления структур данных полигональных сетей отличаются возможным набором операций, производительностью выполняемых над ними операций, требованиями к объему памяти, простотой реализации. С учетом аппаратных особенностей современного графического оборудования, простоту выполнения операций в вычислительной геометрии и требований к набору операций при визуализации в ГИС, гораздо легче работать с треугольниками, чем с многоугольниками. Любой многоугольник легко разбивается на треугольники.

На основании рассмотренных отличий существующих графических движков можно обобщить типичный принцип построения полигональной сети в 3D объекте в виде последовательности (Рис. 1): вершины → ребра → треугольные грани → полигоны → полигональные сети (поверхности) → объекты.

Принципиальным отличием предлагаемого метода от аналогов является расширенный набор базовых примитивов и модифицированная последовательность сборки графических примитивов в объект (Рис. 2). В качестве базовых примитивов используются вершины, линии, лучи, отрезки и треугольники. Все они являются независимыми друг от друга, но с помощью геометрических преобразований могут быть получены друг из друга. Первый этап сборки объектов начинается с треугольников. Треугольники собираются в поверхности, которые в свою очередь собираются в полигональные

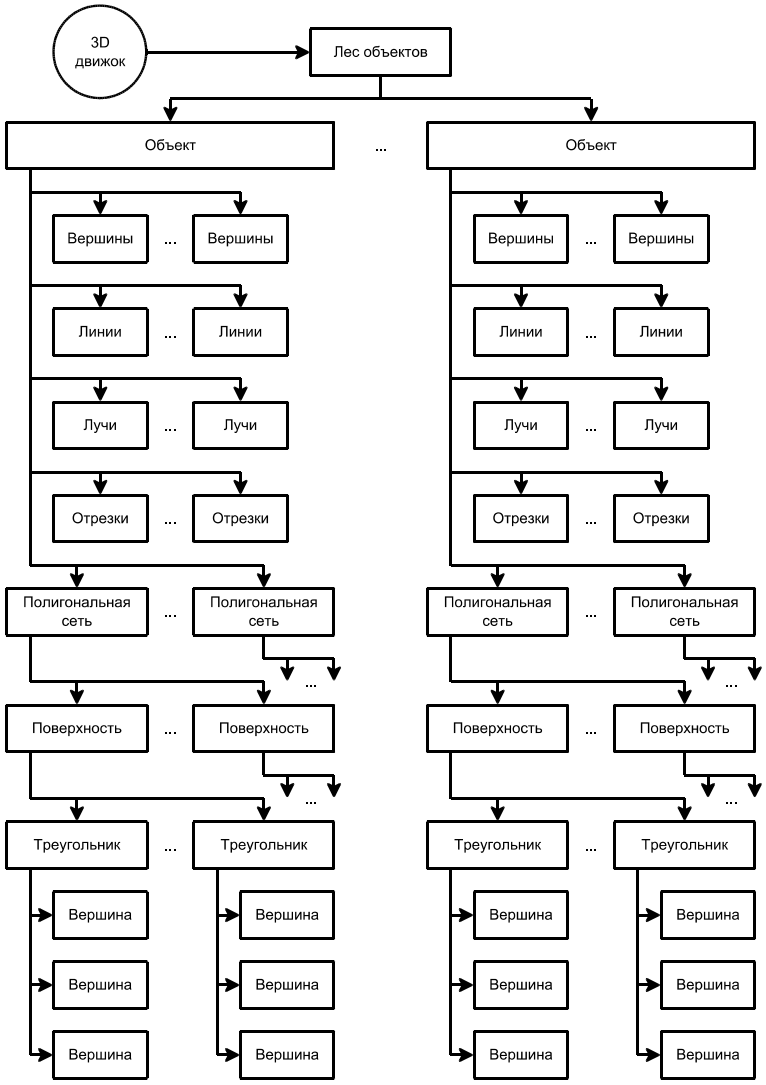

Рис. 3. Структура данных в многообъектном 3D движке

сети для удобства выполнения групповых операций. Объект в системе образуется из произвольного количества полигональных сетей, линий, лучей и отрезков. Такой метод сборки 3D объектов предоставляет гораздо более богатые возможности для моделирования сложных объектов в ГИС за счет смешанного использования полигональных сетей и дополнительных базовых примитивов. Например, благодаря такому методу группировки геометрических примитивов легко решается задача отображения поверхности рельефа трехмерными изолиниями уровня высоты.

Объектов с высокой детализацией в виртуальной сцене может быть очень много. Каждый объект может содержать несколько миллионов примитивов. Для оптимизации соотношения скорости доступа и потребляемой оперативной памяти логично выстроить все объекты и примитивы в системе в виде направленного иерархического дерева (Рис. 3). Использование такого дерева позволяет легко и быстро осуществлять поиск и доступ к необходимым объектам и примитивам. Также появляется возможность хранить не все элементы дерева в оперативной памяти, а только те, которые используются в текущий момент. Неиспользуемые объекты и примитивы можно перенести во временный кэш (буфер памяти) в хранилище пользовательской системы.

С точки зрения 3D движка удобно рассматривать данную структуру в виде множества

деревьев, которые собираются в корне, принадлежащем самому ядру 3D движка. Такой подход позволяет обрабатывать примитивы моделей объектов независимо друг от друга, выполнять асинхронные вычисления над данными каждого объекта, хранить их в распределенной аппаратной среде (например, в оперативной памяти, локальном хранилище, сетевом хранилище и другие), выполнять различные оптимизационные приемы в процессе визуализации.

Для доступа к какому-либо примитиву ядро 3D движка последовательно выполняет обход древовидной структуры в соответствии с заданными направлениями. Например, для доступа к нужной вершине необходимо выполнить следующую последовательность иерархических обращений: объект → полигональная сеть → поверхность → треугольник → вершина.

При осуществлении низкоуровневой визуализации направленная древовидная структура не является оптимальной для аппаратного графического обеспечения из-за необходимости постоянных накладных преобразований. Поэтому у каждого объекта в дереве существует свой временный кэш бинарного вида и набор двоичных указателей для этого кэша, которые строятся при первом обходе примитивов объекта ядром 3D движка. Кэш строится таким образом, чтобы напрямую обрабатываться графическим оборудованием. Если объект не подвергается конфигурационным изменениям, то временный кэш не перестраивается на протяжении всей жизни объекта. Также, если известно, что объект не будет меняться и использоваться для каких-либо других операций, то его можно выгрузить из оперативной памяти для экономии ресурсов, оставив только бинарный кэш. При этом базовые геометрические операции над объектом в виртуальном пространстве (например, поворот, параллельный перенос, масштабирование и другие), не вызывающие изменений в базовой геометрической конфигурации объекта, не приводят к необходимости перестройки временного кэша. Такой подход дает огромный прирост производительности в случае со статическими моделями объектов. В ГИС таких объектов большинство.

В современных системах необходимо визуализировать объекты с высокой детализацией, содержащие огромное количество граней. Количество граней в одном объекте может доходить до нескольких миллионов. Для аппаратной визуализации нужны простые и компактные структуры. В предлагаемом методе используется объектное представление, логически похожее на представление в виде списка граней. Представление полигональных сетей реализовано в виде трехуровневой объектной модели, в которой логическое и структурное описание примитивов полигональных сетей выстроено в виде направленного дерева и обеспечивает оптимальный баланс между производительностью и требованиями к памяти. Программная реализация созданного представления полигональных сетей показана на Рис. 4. Сплошными стрелками обозначается наследование родительского класса, пунктирными - использование другого класса, штрихпунктирными стрелочками - трансляция объектов.

Первый уровень является интерфейсным. Он описывает основные свойства геометрических примитивов и выполняемые над ними базовые операции.

Второй уровень реализует моделируемые объекты с точки зрения вычислительной геометрии. Объекты этого уровня наследуются от объектов первого уровня. В нем реализованы геометрические представления базовых примитивов, моделируемых объектов, а также вычислительные операции над ними. От интерфейсного описания геометрического примитива наследуются четыре вида базовых примитивов системы: точка, линия, треугольник и полилиния. Линия представляет собой примитив, на который накладываются граничные условия. В зависимости от наложения одного или двух граничных условий, соответственно получается луч или отрезок. Полилиния является отдельно стоящим примитивом, потому что, по своей сути, это та же линия, но она имеет множество точек и непрерывна. Наследником геометрической точки является точка пересечения с лучом, которая вычисляется при нахождении пересечения с каким-либо примитивом объекта при трассировке луча. Треугольники собираются в поверхности. Набор поверхностей образует полигональную сеть. Объект может состоять из нескольких полигональных сетей, линий, лучей и отрезков.

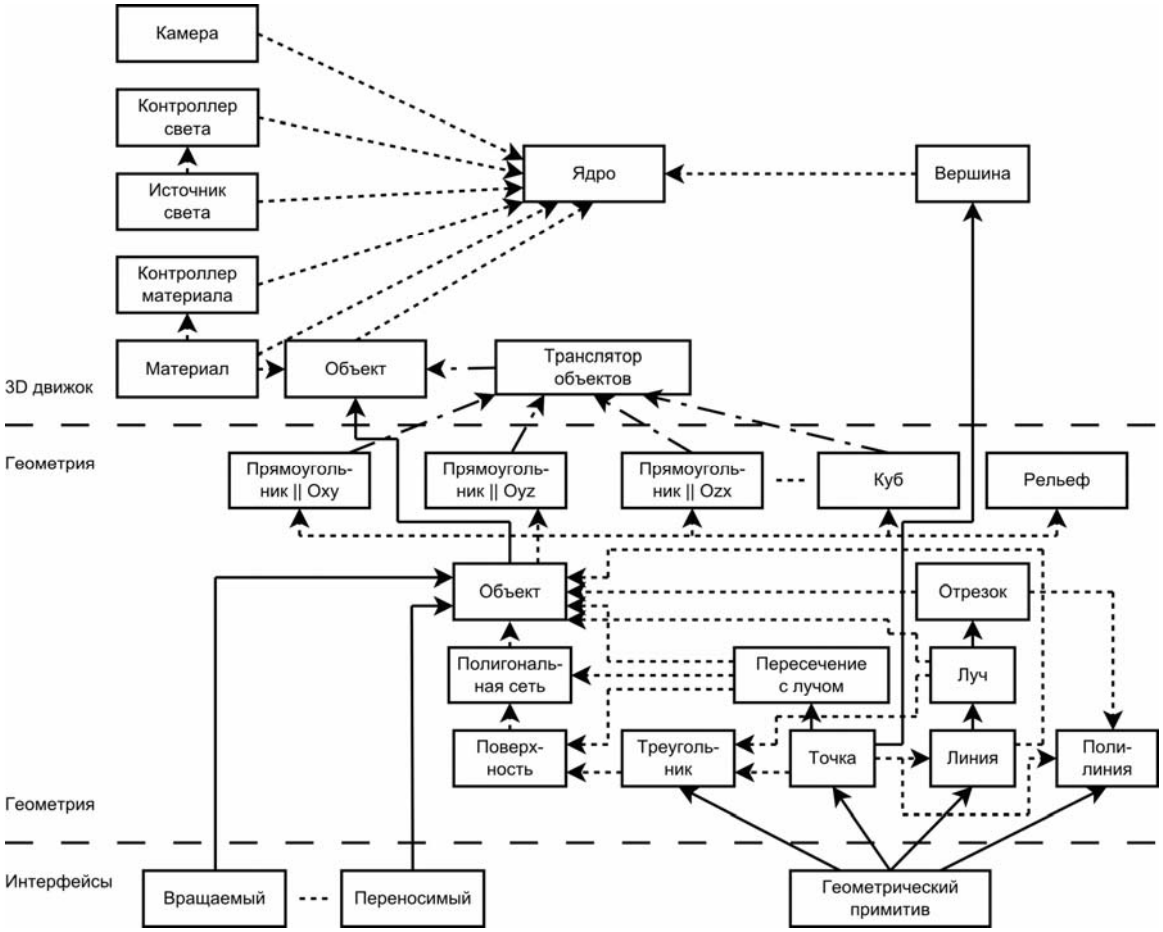

Рис.4. Представление полигональных сетей

Третий уровень предназначен для выполнения низкоуровневой визуализации. Он использует трансляционный модуль стыковки объектов второго и третьего уровня. Трансляция обеспечивается с помощью специального объекта вершины. Она наследуется от объекта геометрической точки второго уровня. Обычная точка представляет собой набор из координат X, Y, Z. Вершины хранят еще дополнительные атрибуты: цвет, нормаль, координаты текстур и другие. Так как вершина является базовым примитивом объектов третьего уровня, то она обеспечивает возможность визуализации примитивов и объектов, сохраняя все вычислительные возможности второго уровня. В соответствии с представлением полигональных сетей все объекты проходят конвейерную обработку в три этапа: подготовка геометрических данных, сборка примитивов и объектов, подготовка к визуализации. После прохождения объектами этих этапов они передаются в ядро 3D движка. Ядро осуществляет визуализацию объектов в интерактивном режиме.

3. Обработка пространственных данных в прототипе распределенной ГИС

Для проверки предлагаемого метода построения полигональных сетей разработан прототип распределенной сетевой ГИС, имеющий 3D движок для трехмерной визуализации пространственных данных. Общая схема потоков данных в разработанном прототипе распределенной сетевой ГИС приведена на Рис. 5. Рассмотрим схему последовательно слева направо в порядке прохождения данных по различным этапам конвейерной обработки в системе. Сейчас имеется много разных форматов и стандартов хранения пространственных данных.

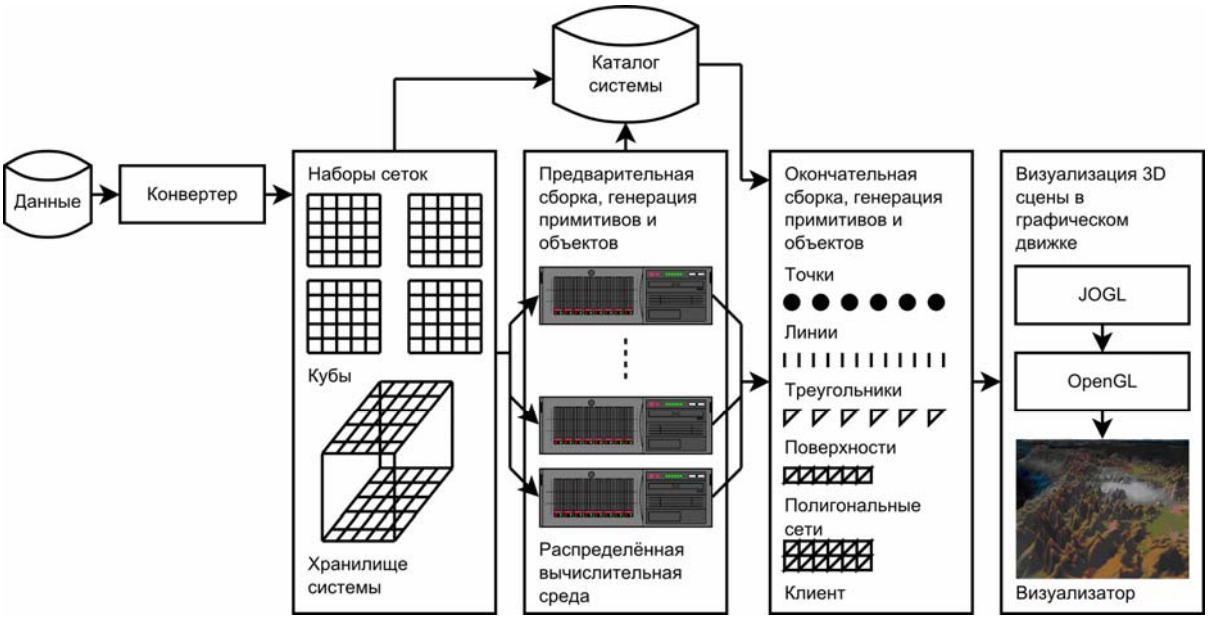

Рис. 5. Общая схема потоков данных в прототипе распределенной сетевой ГИС

Обусловлено это тем, что на ранних этапах развития разработчики ГИС создавали собственные форматы под свои задачи. Для усвоения их системой необходимо выполнить преобразование во внутренний формат с помощью конвертера. Конвертер создан с использованием общепринятой в ГИС библиотеки GDAL, а специфичные форматы конвертируются собственными алгоритмами.

После выполнения необходимых преобразований пространственные данные поступают в хранилище прототипа системы, где и остаются до тех пор, пока они нужны пользователям и системе. В зависимости от конфигурации системы хранилище может быть централизованным с множеством вычислителей в распределенной вычислительной среде или может быть рассредоточено непосредственно по вычислителям. Пространственные данные в хранилище приведены во внутренний оптимизированный бинарный формат системы.

При поступлении пространственных данных в хранилище, они регистрируются в общем каталоге прототипа системы как исходные. Далее пространственные данные поступают на обработку в распределенную вычислительную систему. На этом этапе пространственные данные проходят предварительную обработку и осуществляется предварительная сборка графических моделей. После выполнения преобразований все результаты также регистрируются в общем каталоге прототипа системы. Обычно результаты предварительной обработки хранятся в самой распределенной вычислительной среде. Пользователь, осуществляя запрос данных, использует каталог, в котором прописано где они хранятся и как к ним получить доступ. Выбрав необходимые пользователю данные, прототип системы загружает нужные наборы данных или графические модели из распределенной вычислительной среды.

Последние два этапа представляют собой клиентское ПО, которое исполняется на стороне пользователя. На первом этапе производится окончательная сборка графических моделей, т.е. генерируются все необходимые для визуализации примитивы, выполняются геометрические преобразования. Второй этап является презентационным. На данном этапе осуществляется конечная визуализация подготовленных ранее графических моделей с использованием 3D -движка на основе программного интерфейса OpenGL и набора трансляционных библиотек JOGL. В итоге пользователь на выходе получает трехмерную картину.

Любые интерактивные запросы пользователя обрабатываются на последних двух этапах конвейера за квант времени, необходимый для

визуализации одного кадра изображения. Соответственно, необходимо оптимальное распределение стадий обработки по этапам конвейера. Все предварительные этапы должны обработать и подготовить пространственные данные и графические модели таким образом, чтобы их окончательная обработка и сборка на стороне клиента заняла минимально возможное время.

Рассмотрим более подробно внутренний формат хранения пространственных данных. Внутренний формат типизирован для разработанного алгоритма построения полигональных сетей и представляет собой пространственные данные в виде многомерных кубических и пирамидальных структур. Все точечные цифровые данные скалярных полей хранятся в двумерном матричном виде в терминах прототипа системы в виде сеточных поверхностей. Поверхность является атомарной единицей пространственных данных. Трехмерные или объемные пространственные данные состоят из набора идентичных сеток с различной высотной составляющей, которые образуют куб данных. Анимированный куб - это связанный набор кубов данных, например, по времени. Многомерный куб представляет собой связанный набор анимированных кубов, например, по общей модели ситуации, но с разными параметрами развития. С точки зрения моделирования, пространственная модель реализуется поверхностями и кубами, пространственно-временная модель - анимированными кубами, модель параллельных измерений - многомерными кубами (Рис. 6).

Для оптимизации обработки поверхности разбиваются по площади на множество прямоугольных участков. Каждый такой участок называется «черепицей». В ГИС такой метод известен как парадигма «черепиц» [8]. Для определения размеров черепицы используется специальный алгоритм подбора размеров кратных степени двойки, который учитывает аппаратные особенности современного оборудования. Развитие идеи черепиц применимо к многомерным данным. Предлагается каждый куб данных разбить по объему на множество составляющих кубов. Каждый такой куб в терминах системы называется «кирпичиком», а метод обозначим как парадигма «кирпичиков». На основе парадигмы черепиц и кирпичиков ПО прототипа ГИС получает возможность использовать многопоточные технологии для асинхронной передачи и обработки отдельных частей виртуального пространства. Причем не обязательно передавать и обрабатывать все части набора данных, если они не нужны пользователю. Этот подход дает возможность симуляции сложных виртуальных миров в реальном времени на современном аппаратном обеспечении.

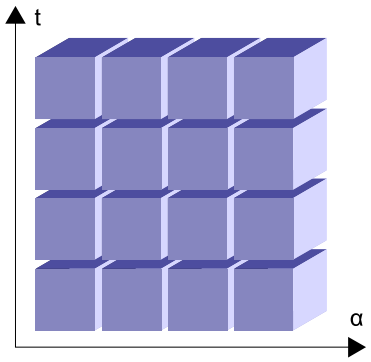

Рис. 6. Многомерный куб

Используемые пространственные данные в системе могут иметь различную плотность на карте. Очень плотные пространственные данные необходимо прореживать, так как нет смысла показывать их в максимальной детализации из-за ограничений разрешающей способности современных устройств вывода и жесткого лимита доступных вычислительных ресурсов. Различные наборы прореженных и исходных пространственных данных образуют уровни детализации. Каждая поверхность может иметь несколько уровней детализации (Рис. 7), будучи представлена различным количеством черепиц с разным пространственным разрешением. Нужный уровень детализации черепицы выбирается в зависимости от положения наблюдателя в виртуальном пространстве и расстояния до отображаемой черепицы. Визуализация в высоком качестве с максимальной детализацией осуществляется непосредственно вокруг пользователя. Соответственно, чем дальше черепица находится от наблюдателя, тем ниже уровень ее детализации. Наиболее удаленные черепицы отображаются в виде «декорации» для обеспечения целостности визуализации. Если пользователь осуществляет перемещение по карте, то автоматически загружаются и перестраиваются недостающие

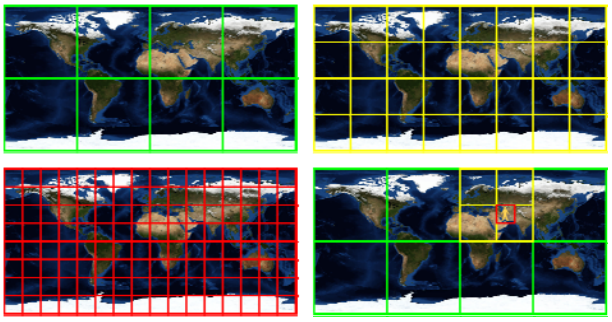

Рис.7. Уровни детализации

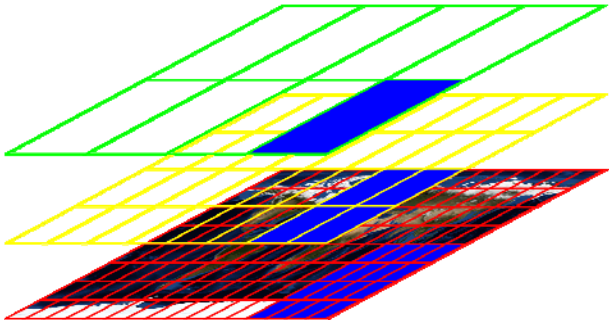

Рис. 8. Пирамидальные структуры

черепицы с нужным уровнем детализации. Пользователь все время находится в зоне с максимальной детализацией, при этом нет полной нагрузки на вычислительную систему, так как нет необходимости обрабатывать весь исходный массив данных в реальном времени. Черепицы, построенные и расположенные в порядке возрастания уровня детализации, образуют пирамидальные структуры, схематично показанные на Рис. 8. Такие пирамидальные структуры очень удобно хранить и легко обходить программно.

Элементы точечных цифровых данных в системе реализуют объекты, хранящие пространственные данные в геоцентрической системе с высотной координатой относительно среднего уровня моря. Они транслируются системой в «пригодный» для аппаратного обеспечения вид примитивов многомерной полигональной сетки. Следует отметить, что полигональную сеть заранее построенного объекта можно загрузить напрямую с сервера. Для этого она должна храниться в открытом формате Kronos COLLADA [3], который используется для обмена данными между 3D приложениями. Текстуры для созданных объектов визуализации могут браться как с внутреннего сервера данных системы, так и с картографических серверов, использующих стандарты OGC.

4. Результаты

Предложенные методы были успешно протестированы на массивах пространственных данных NOAA Earth System Research Laboratory NCEP/NCAR Reanalysis и British Oceanographic Data Centre GEBCO и доказали свою жизнеспособность в сетевой среде. Пример построения полигональной сети объекта рельефа из одноминутного массива GEBCO 2008 онлайн в интерактивном режиме показан на Рис. 9.

Использование многопоточной предварительной обработки данных в распределенной вычислительной сети увеличивает производительность и уменьшает время обслуживания пользователя по сравнению с типичной работой графического движка только в вычислительной системе пользователя. По результатам проведенных исследований (таблица ниже) скорость построения тестовой графической модели в распределенной вычислительной сети на основе массива данных объемом 9,64 Гбайт увеличилась в более чем 10 раз по сравнению с обычным способом построения графической модели в один пользовательский поток на машине пользователя, как в большинстве современных 3D движков. Для тестирования использовалась сеть машин, состоящая из двух одинаковых серверов начального уровня на базе процессоров Q6600, имеющих массивы RAID 0+1 из шести дисков, и ноутбука на базе процессора Core i7-740QM с одним жестким диском. Выполнение некоторого количества потоков на одной серверной машине производится дольше, чем выполнение того же количества потоков, распределенных по двум серверным машинам. Этот факт полностью доказывает, что при таком объеме данных, обрабатываемых несколькими потоками, узким местом в системе становиться дисковая подсистема сервера, а не процессор. Поэтому применение нескольких вычислительных систем позволяет значительно снизить нагрузку на дисковые массивы и повысить производительность распределенной вычислительной системы в целом. Универсальность метода позволяет использовать в качестве вычислительной среды разнородные аппаратные вычислительные системы.

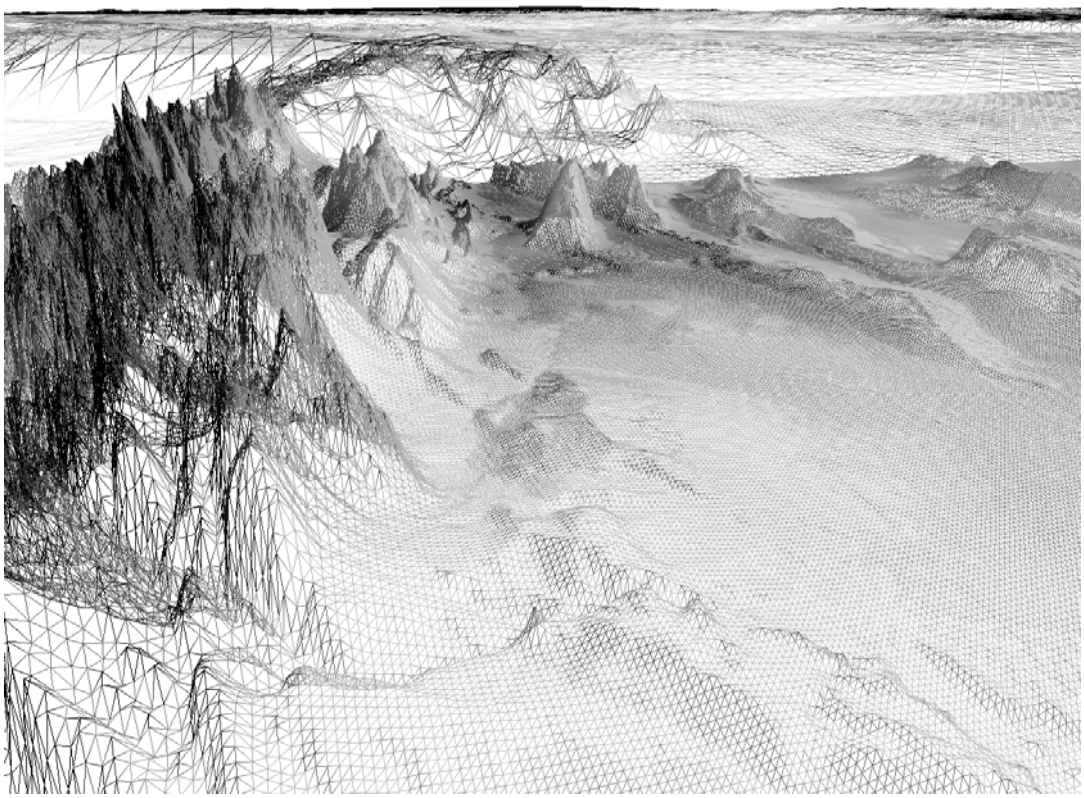

Рис. 9. Модель виртуальной планеты

Результаты тестов скорости построения графической модели в распределенной вычислительной сети

Оборудование	Количество потоков обработки данных	Время обработки данных, с
Q6600	1 поток	357,286438
Q6600	2 потока	183,161621
Q6600	3 потока	123,907616
Q6600	4 потока	96,805588
2xQ6600	1+1 поток	180,238113
2xQ6600	2+1 поток	119,155624
2xQ6600	3+1 поток	91,169777
2xQ6600	4+1 поток	77,451500
2xQ6600	2+2 поток	89,096031
2xQ6600	2+3 поток	74,057602
2xQ6600	2+4 поток	65,546280
2xQ6600	3+4 поток	58,761829
2xQ6600	4+4 поток	51,125565
Core i7-740QM	1 поток	336,170349
Core i7-740QM	2 потока	198,379547
Core i7-740QM	3 потока	147,181717
Core i7-740QM	4 потока	119,063606
Core i7-740QM	5 потоков	104,036804
Core i7-740QM	6 потоков	97,562744
Core i7-740QM	7 потоков	93,545044
Core i7-740QM	8 потоков	89,128853
2xQ6600 + Core i7-740QM	4+4+1 поток	48,237354
2xQ6600 + Core i7-740QM	4+4+2 поток	44,575657
2xQ6600 + Core i7-740QM	4+4+3 поток	41,633251
2xQ6600 + Core i7-740QM	4+4+4 поток	39,319942
2xQ6600 + Core i7-740QM	4+4+5 поток	38,134674
2xQ6600 + Core i7-740QM	4+4+6 поток	36,267429
2xQ6600 + Core i7-740QM	4+4+7 поток	35,846401
2xQ6600 + Core i7-740QM	4+4+8 поток	34,007725

Заключение

Разработан метод построения полигональных сетей многомерных объектов с высокой детализацией, принцип организации многомерных пространственных данных в многомерные кубические и пирамидальные структуры с использованием уровней детализации, методы черепиц и кирпичиков для работы с поверхностями и кубами пространственных данных.

Метод представления полигональных сетей моделируемых 3D объектов имеет ряд преимуществ перед аналогами: обеспечивается возможность интерактивной работы с массивами данных в сети за счет использования поэтапной конвейерной обработки данных; увеличивается скорость обработки многомерных структур данных за счет асинхронных вычислений в распределенной вычислительной сети и многопоточных вычислений на пользовательской системе; полигональная сеть заранее построенного объекта может загружаться напрямую с сервера; предложенная реализация облегчает выполнение ряда математических и пространственных преобразований с полигональными сетками на «лету», например, для построения изолиний и изоповерхностей.

Перспективными шагами для развития предложенного метода являются: реализация системы многоуровневого кэширования, реализация расчетных алгоритмов на специализированных вычислительных языках, создание ускоренного трассировщика лучей. Учитывая универсальность разработанного метода построения полигональных сетей, область его применения не ограничивается ГИС. Он может использоваться в 3D движках при симуляции виртуальной реальности, где требуется интерактивная работа с графическими моделями, имеющими высокую детализацию.

Литература

1. Никулин Е.А. Компьютерная геометрия и алгоритмы машинной графики. Учебное пособие: Книга. Санкт-Петербург: БХВ-Петербург, 2003. - 560 с.
2. Полигональная сетка: Статья. 15 февраля 2013 / Википедия, 2013. Электронный ресурс. URL: http://ru.wikipedia.org/wiki/Mesh_(компьютерная_графика) (дата обращения: 27 февраля 2013).
3. Barnes M., Finch E.L. COLLADA - Digital Asset Schema: Спецификация. Clearlake Park, California, U.S.A.: The Khronos Group Inc., Sony Computer Entertainment Inc., 2008. - 532 с.
4. Casperson M. Away3D 3.6 Essentials: United Kingdom: Packt Publishing, 2011. - 400 с.
5. Flavell L. Beginning Blender: United States of America: Apress, 2010. - 448 с.
6. Goldstone W. Unity Game Development Essentials: United Kingdom: Packt Publishing, 2009. - 316 с.
7. Kerger F. OGRE 3D 1.7 Beginner's Guide: United Kingdom: Packt Publishing, 2010. - 300 с.
8. Maso J., Pomakis K., Julia N. OpenGIS Web Map Tile Service Implementation Standard: Спецификация. Wayland, USA: Open Geospatial Consortium Inc., 2010. – 128 с.
9. Smith C. On Vertex-Vertex Systems and Their Use in Geometric and Biological Modelling: University of Calgary, Calgary, Alberta, Canada: 2006. - 216 с.
10. Tobler R.F., Maierhofer S. A Mesh Data Structure for Rendering and Subdivision: WSCG 2006 Short Papers Proceedings. 31 января - 2 февраля 2006. Plzen, Czech Republic: UNION Agency - Science Press, 2006. - С. 157-162.

Ледяев Александр Юрьевич. Программист Всероссийского научно-исследовательского института гидрометеорологической информации – мирового центра данных. Окончил Национальный исследовательский ядерный университет «МИФИ» (бывший Обнинский институт атомной энергетики) в 2010 году. Автор трех научных работ. Область научных интересов: ГИС- технологии, компьютерная графика, распределенные вычисления. E-mail: ledalex@inbox.ru