Лебедев Олег Борисович – Южный федеральный университет; e-mail: oblebedev@sfedu.ru; 347928, г. Таганрог, пер. Некрасовский, 44; тел.: 89085135512; кафедра систем автоматизированного проектирования; к.т.н.; доцент.

Лебедева Елена Михайловна – кафедра системного анализа и телекоммуникаций; магистрант.

Пестов Владислав Андреевич – кафедра вычислительной техники; магистрант.

Lebedev Oleg Borisovich – Southern Federal University; e-mail: oblebedev@sfedu.ru; 44, Nekrasovsky street, Taganrog, 347928, Russia; phone: +79085135512; the department of computer-aided design; cand. of eng. sc.; assistant professor.

Lebedeva Elena Mikhailovna – the department of system analysis and telecommunications; master student.

Pestov Vladislav Andreevich – the department of computer science; master student.

УДК 004.421

Н.И. Витиска, Н.А. Гуляев

МЕТОД ВИЗУАЛИЗАЦИИ ТРЁХМЕРНЫХ СЦЕН И ОБЪЕКТОВ ВОКСЕЛЬНОЙ ГРАФИКИ ДЛЯ СИСТЕМ ИМИТАЦИОННОГО МОДЕЛИРОВАНИЯ

Визуализация является важной составной частью широкого круга задач. Системы компьютерного моделирования – одна из областей, в которых требуется визуализация моделируемых процессов и явлений. Современные средства визуализации справляются с большинством задач, однако потребности более новых и высокотехнологичных систем могут быть гораздо выше и требовать иных методов визуализации. Воксельная графика является одним из таких решений в ряде случаев. Однако сама воксельная графика имеет ряд недостатков и проблем в реализации в различных ситуациях. Для эффективного внедрения воксельной графики требуется решить ряд основных проблем, заключающихся в поиске оптимальных алгоритмов и структур данных. Приводится описание метода организации трёхмерных воксельных сцен для решения задач визуализации в системах технической симуляции и моделирования. Рассмотрены проблемы и недостатки существующих подходов к организации сцен и визуализации при помощи воксельной графики. Основное внимание уделено применению октодерева, широко используемого в настоящее время. Предлагается альтернативой способ организации воксельных данных, ориентированный на специфичный тип сцен, использующихся для визуализации во многих системах моделирования. Предлагаемый способ заключается в разбиении вокселей сцены на классы, разбиении видимых объектов на классы, группировке вокселей по принадлежности к объектам. Описываются структуры данных для организации воксельных данных предлагаемым образом, раздельного хранения и независимой обработки различных групп вокселей. Описываются базовые алгоритмы, необходимые для обработки и визуализации воксельных данных в таком представлении. Рассмотрены преимущества данного метода в ряде случаев. Сделаны выводы о целесообразности использования данного метода в тех или иных случаях.

Трёхмерная графика; компьютерная графика; воксельная графика; разбиение пространства; трёхмерные сцены; организация трёхмерных сцен; структуры данных.

N.I. Vitiska, N.A. Gulyaev

AN APPROACH TO VISUALIZATION OF THREE-DIMENSIONAL SCENES AND OBJECTS VIA VOXEL GRAPHICS FOR SIMULATION SYSTEMS

Visualization is an important part of a wide range of different tasks. Computer modeling is one of the areas, which often require visualization of simulated processes. Modern visualization tools are able to perform visualization in different applications, which covers most cases of use,

but some systems can require different visualization techniques for special cases. Voxel graphics is one of the solutions, applicable in a number of such cases. However, voxel graphics has a significant number of drawbacks and problems in implementation. To perform an effective implementation of voxel visualization, a solution to the basic problems is required. A solution must include optimal algorithms and optimal data structures. This article describes an approach to organization of three-dimensional voxel scenes in engineering, simulation and modeling systems. Typical problems and shortcomings of existing approaches are discussed. A common approach in this area is a usage of octree, which is reviewed in detail. An alternative approach to organization of voxel data, focused on special types of scenes, which are widely used in modern modeling systems, is proposed. The proposed method divides the scene into classes of voxels, dividing visible objects into classes and grouping voxels by their affiliation to each object. Data structures for proposed scene organization, separate storage and handling of various independent groups of voxels is described. Basic algorithms for processing and rendering data in this format are described. Advantages of the proposed method in different cases are discussed.

Three-dimensional graphics; computer graphics; voxel graphics; space partitioning; three-dimensional scenes; organization of three-dimensional scenes; the data structure.

Введение. Современные средства визуализации решают широкий круг задач во многих отраслях техники. Однако всё более совершенствующиеся технологии зачастую требуют новых, а иногда и радикальных решений. Так, традиционная полигональная графика, используемая в подавляющем большинстве случаев, в ряде задач оказывается неоптимальной в применении или вовсе неприменимой в некоторых случаях [1]. Поэтому актуальной становится задача поиска и разработки альтернативных методов визуализации для решения проблем в частных случаях. Одним из таких решений может являться применение воксельной графики вместо полигональной.

Воксельная графика представляет объекты при помощи атомарных элементов, расположенных в узлах сетки – вокселей. Каждый воксель может содержать некоторое количество данных – от цвета и прозрачности, аналогично двухмерным пикселям, – до специализированных данных, например, информации о типе материала, плотности, упругости [2]. Помимо этого, воксельная графика, в отличие от полигональной, позволяет визуализировать высоко детализированные объекты без использования дополнительных средств, визуализировать внутреннюю структуру объектов, позволяет производить модификации объектов на уровне вокселей [3]. Возможность визуализации объектов совершенно разной природы и легкость преобразования делают воксельную графику выгодной альтернативой полигональной графики во многих направлениях и приложениях. Особенно это касается областей применения компьютерного моделирования, где требуется визуализация моделируемых процессов.

Постановка проблемы. Однако для современных аппаратных и программных систем достаточно проблематично производить визуализацию комплексных сцен при помощи воксельной графики из-за ряда причин. Основной и главной причиной является большой размер сетки, и соответственно, большие объёмы воксельных данных, присутствующих в этой сетке. Процесс обработки и визуализации таких сеток зачастую требует в десятки или сотни раз больше вычислительных ресурсов, чем могут предоставить современные аппаратные системы [4]. Поэтому широкое применение воксельной графики для систем моделирования затруднено, в первую очередь, из-за этой проблемы.

Существуют различные методы повышения эффективности воксельной графики, но наиболее популярным и эффективным способом как повышения скорости обработки, так и сокращения размеров воксельных данных, является применение иерархической структуры – разреженного октодерева [5]. Так или иначе данная структура является основой многих известных методов, ориентированных на повышение скорости обработки [6]. Или методов, ориентированных на сокраще-

ние объёмов данных [7]. Однако универсальность таких структур хоть и высока, но не бесконечна. Многим современным программным системам требуются более специализированные подходы к работе с воксельными данными [8]. Работа с комплексными сценами, содержащими большое количество объектов, сценами, содержащими изменяющиеся объекты, сценами, имеющими большие разрешения сетки, требует новых методов организации и обработки воксельных данных.

Предлагаемое решение. Для решения проблем визуализации при помощи воксельной графики в системах моделирования предлагается специализированный метод – метод, ориентированный на решение проблем, связанных с объёмами данных и скоростью их обработки в ряде случаев. Конкретно, предлагается метод, ориентированный на определённые типы сцен, использующиеся во многих программных системах симуляции и моделирования. В большинстве таких сцен присутствует ярко выраженная неоднородность содержимого: в сцене присутствуют различные подвижные объекты – например, разнообразные механизмы и аппараты, а также статичные объекты – какие-либо помещения, поверхности, препятствия [9–11]. При визуализации такой сцены при помощи воксельной графики, можно определить следующее. Во-первых, неоднородность заключается в принадлежности различных групп вокселей различным объектам – можно явно разделить к какому объекту относится тот или иной воксель. Во-вторых, неоднородность существует среди самих этих объектов – можно явно выделить, например, объекты изменяющиеся и объекты неизменные. Таким образом, можно определить два варианта неоднородности: разделение вокселей на классы – по принадлежности к тому или иному объекту и разделение объектов на классы – объекты изменяющиеся и неизменные. Очевидно, что сцены такого типа требуют индивидуального подхода и существующие методы, заключающиеся в применении октодерева, разделяющего всю сцену всего на два класса вокселей будут неэффективны.

Допустим наличие некоторой сцены $V = \{v_1, \dots, v_n\}$ и наличие в ней определённого количества вокселей v. Исходя из такого свойства вокселей, как однозначное состояние – пустой или непустой, в сцене можно определить два непересекающихся подмножества, P – множество непустых вокселей и Z – множество пустых вокселей: $P \cup Z = V$, в таком случае, если $v \in V$, то $v \in P \oplus v \in Z$, что говорит об однозначном разделении.

Для оптимизации обработки и хранения все воксели (или как можно больше вокселей) из множества Z требуется пропускать, так как они не несут полезной информации. Октодерево даёт такую возможность путём прекращения ветвления в областях, заполненных пустыми вокселями. Однако это оптимально только тогда, когда в сцене присутствует всего два множества вокселей. Например, при обработке сцен, состоящих из одного объекта или из неизменяемых объектов. Таким образом, если сцена V содержит не просто некоторое количество вокселей, а несколько различаемых объектов, состоящих из непустых вокселей, $P_1..P_n$, которые являются подмножествами P, $\cup P_n = P$, тогда $\forall v \in V, v \in P_1 \oplus \dots \oplus v \in P_n \oplus v \in Z$.

В этом случае очевидно, что классические реализации октодерева здесь не подходят. Так как в данном случае воксели распределяются либо в объектах, либо в множестве пустых вокселей, в то время, как разреженное октодерево предполагает только наличие двух вариантов распределения, как будто бы сцена состоит только из одного объекта и пустых вокселей. В случае, если требуется, например, исключить из расчётов не только воксели пустого множества, но и воксели, ассоциируемые с некоторым объектом, простое октодерево не поможет.

Предлагаемый способ организации содержимого сцены, предполагает неравномерное разделение ограничивающими объёмами в сочетании с каким-либо вариантом разбиения пространства внутри объёма. Например, применение внутри

ограничивающего объёма одной из эффективных модификаций статичного октодерева, организованного при помощи ориентированного ациклического графа [12], либо обычного воксельного массива. Ограничивающие объёмы имеют приложение во многих задачах трёхмерной полигональной графики. Фактически ограничивающие объёмы заключают в себя некоторое количество объектов, находящихся в некотором топологическом отношении [13]. В воксельной графике применение ограничивающих объёмов ещё более выгодно, так как пространство дискретно, а множества – конечны. Таким образом, для ограничивающего объёма K, применяемого для объекта P_i: $\forall v \in K,\ v \in P_i \bigoplus v \in Z$, когда $K \subset V$. При этом, $v \in P_i \wedge v \notin P_1 \wedge .. \wedge v \notin P_{i-1} \wedge v \notin P_{i+1} \wedge .. \wedge v \notin P_n$ при $\forall v \in K$ выполняется всегда. В любой узел сцены и его ограничивающий объём попадают как непустые воксели, принадлежащие объекту, так и "побочные" пустые воксели, однако пустые воксели вне всех узлов будут успешно удалены. В то же время, все воксели объекта будут полностью включены в объём, при этом, воксели других объектов гарантированно в него не попадут.

Для осуществления непосредственного разбиения сцены и формирования узлов предлагается использовать подход, аналогичный разбиению при помощи октодерева. Сформулировать идею можно в двух принципах: первый принцип – пропуск пустых вокселей, аналогичный пропуску при использовании октодерева. Второй принцип – оптимальная организация изменяющихся объектов, с чем при использовании октодерева возникают проблемы. Если рассматривать сам процесс формирования октодерева [14], то основная идея заключается в рекурсивном разбиении пространства на восемь равных частей, которые заключаются в узлы (рис. 1).

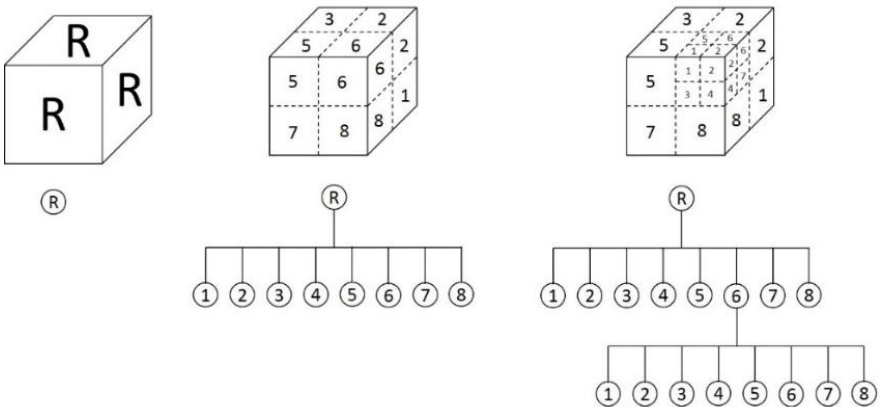

Рис. 1. Разбиение при помощи октодерева

Очевидно, что изменяющиеся воксели изменяющегося объекта не выгодно хранить в структуре такого типа. Так как после каждого изменения объекта требуется изменение самого октодерева, что в некоторых случаях может сводиться к полной перестройке этого дерева. Более того, воксели, принадлежащие одному объекту могут оказаться в разных узлах, поэтому логическая целостность объектов может нарушиться.

Главным отличием предлагаемого метода от применения октодерева для организации воксельных данных является разделение всех вокселей сцены по признаку принадлежности к тому или иному объекту, а разделение объектов – по способу обработки – объекты изменяющиеся и объекты неизменные (рис. 2). Каждый узел ассоциируется с конкретным объектом, представленным на сцене, а не с частью пространства. Все воксели, представляющие один объект, размещаются в

одном узле. Каждому узлу ставится в соответствие некоторый блок данных, фактически – некоторая часть всей воксельной сетки сцены. Таким образом, производится переход от большой однородной сетки к некоторому количеству малых сеток. Благодаря тому, что пространство в воксельной графике дискретно, воксели, не несущие полезной информации (пустые воксели), хранить не требуется.

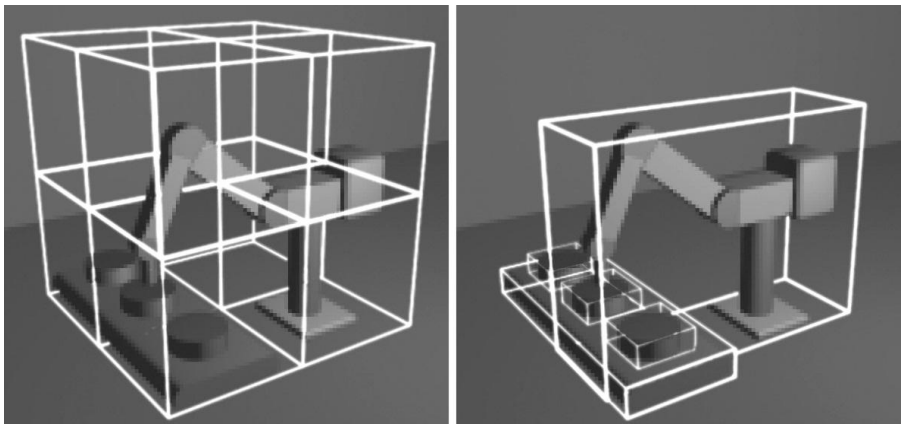

Рис. 2. Пример первого уровня разбиения сцены при помощи классического октодерева (слева) и при помощи предлагаемого способа (справа)

От иерархии узлов и ограничивающих объёмов на данном этапе предлагается отказаться и располагать все узлы и их объёмы на одном уровне (рис. 3). Это обусловлено в большей степени тем, что каждый объём уже заключает в себя отдельный самостоятельный объект, даже если логически он может являться частью другого объекта. Тем более, что внутри этого объёма данные могут быть организованы произвольным образом. Во-вторых, при изменении содержимого, потребуется произвести вышеописанный пересчёт ограничивающего объёма, в зависимости от количества вокселей, эта задача может потребовать существенных затрат.

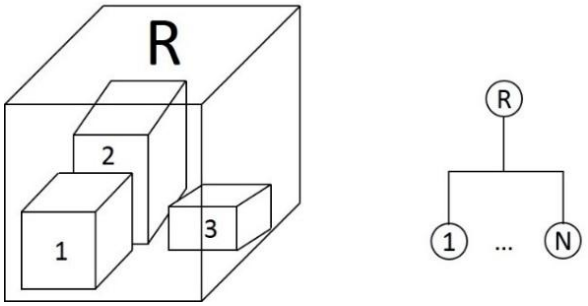

Рис. 3. Разбиение сцены предлагаемым способом

Применение ограничивающих объёмов в данном случае сводится к определению границ каждого объекта и, соответственно, блока воксельных данных каждого узла. Наиболее подходящим типом ограничивающего объёма для дискретного воксельного пространства является параллельный осям ограничивающий параллелепипед (AABB), [15]. Такой ограничивающий объём требует всего трёх координат для обозначения начальной точки и трёх значений, обозначающих длины его рёбер по осям (рис 4).

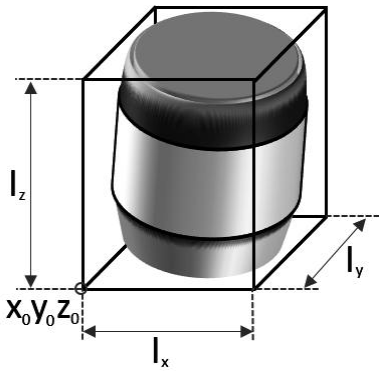

Рис. 4. Ограничивающий объём для каждого объекта

В памяти ЭВМ такая структура ограничивающего объёма будет иметь компактный вид, более того, все значения будут целочисленными (рис. 5).

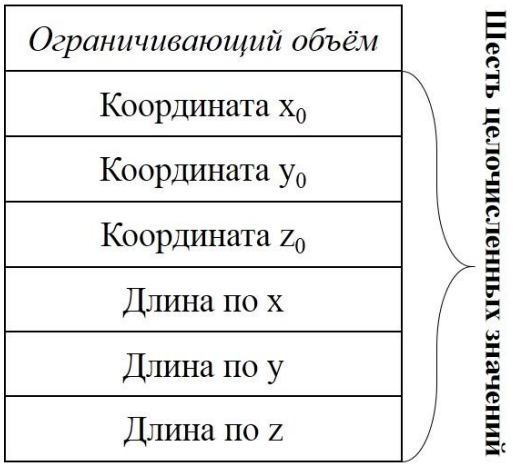

Рис. 5. Структура ограничивающего объёма в памяти ЭВМ

Определение значений ограничивающего объёма заключается в нахождении в блоке данных минимальных и максимальных значений каждых координат вокселей. Такая процедура должна выполняться один раз во время сегментации всей сцены и каждый раз после изменения координат (превышение существующих границ) вокселей внутри этого объёма. Процедуру сегментации сцены предлагается производить на основании расположения воксельных данных в памяти. В рассматриваемом классе сцен каждый самостоятельный является объект является независимым, хранится либо в отдельном файле, либо в отдельной области памяти, перед тем, как загружается в сцену. Поэтому для формирования каждого узла требуется только раздельная загрузка воксельных данных из соответствующего файла или области данных. После этого, каждый узел будет содержать свой блок воксельных данных.

Рассмотрим базовый алгоритм расчёта (рис 6) ограничивающего объёма для блока воксельных данных (Бвд) соответствующего узла (Узл), имеющего ненулевое количество непустых вокселей V и ограничивающего объёма этого узла (Узл.ГО):

```
Процедура Расчёт_Граничного_Объёма(Узл)
1   Vмин.X ← мин знач V.x из Узл.Бвд
2   Vмин.Y ← мин знач V.y из Узл.Бвд
3   Vмин.Z ← мин знач V.z из Узл.Бвд
4   Vмакс.X ← макс знач V.x из Узл.Бвд
5   Vмакс.Y ← макс знач V.y из Узл.Бвд
6   Vмакс.Z ← макс знач V.z из Узл.Бвд
7   Узл.ГО.Коорд ← Vмин
8   Узл.ГО.Длин ← Vмакс - Vмин
```

Рис. 6. Процедура расчёта граничного объёма

Расчёт ограничивающего объёма (рис 7) может потребоваться после модификации группы вокселей (Вксл) в каком-либо узле (Узл), поэтому требуется проверить, необходимо ли пересчитать объём для этого узла:

```
Процедура Проверка_Необходимости_Пересчёта(Вксл,Узл)
1  Vмин.x ← мин знач V.x из Вксл
2  Vмин.y ← мин знач V.y из Вксл
3  Vмин.z ← мин знач V.z из Вксл
4  Vмакс.x ← макс знач V.x из Вксл
5  Vмакс.y ← макс знач V.y из Вксл
6  Vмакс.z ← макс знач V.z из Вксл
7  Если (Vмин.x < Узл.Го.Коорд.X) ИЛИ
8       (Vмакс.x > Узл.Го.Коорд.X+ Узл.ГО.Длин.X) ИЛИ
9       (Vмин.y < Узл.Го.Коорд.Y) ИЛИ
10      (Vмакс.y > Узл.Го.Коорд.Y+ Узл.ГО.Длин.Y) ИЛИ
11      (Vмин.z < Узл.Го.Коорд.Z) ИЛИ
12      (Vмакс.z > Узл.Го.Коорд.Z+ Узл.ГО.Длин.Z)
13              Расчёт_Граничного_Объёма(Узл)
```

Рис. 7. Процедура проверки необходимости пересчёта

Стоит отметить, что параллельное смещение всех вокселей объёма должно осуществляться путём изменения значения начальной координаты ограничивающего объёма, так как координаты вокселей внутри узла – относительные. Также возможен другой подход к проверке необходимости пересчёта – проверка координат каждого вокселя может осуществляться непосредственно в процедуре, которая производит изменение. Такой подход был бы более оптимальным при большом количестве изменённых вокселей.

Для организации сцены в памяти ЭВМ предлагается следующая структура данных (рис. 8), состоящая из следующих элементов:

1. Корневой элемент сцены – структура, содержащая указатели на все узловые элементы сцены;
2. Узловые элементы сцены – структуры, содержащие указатель на блок данных памяти, где хранятся воксельные данные, а также содержащие ограничивающий объём и значения, определяющие тип представляемого объекта и тип структуры данных, используемой для организации хранения воксельных данных;
3. Блок данных – область памяти, где располагается часть сетки (все воксели объекта), представленная либо октодеревом, либо обычным массивом.

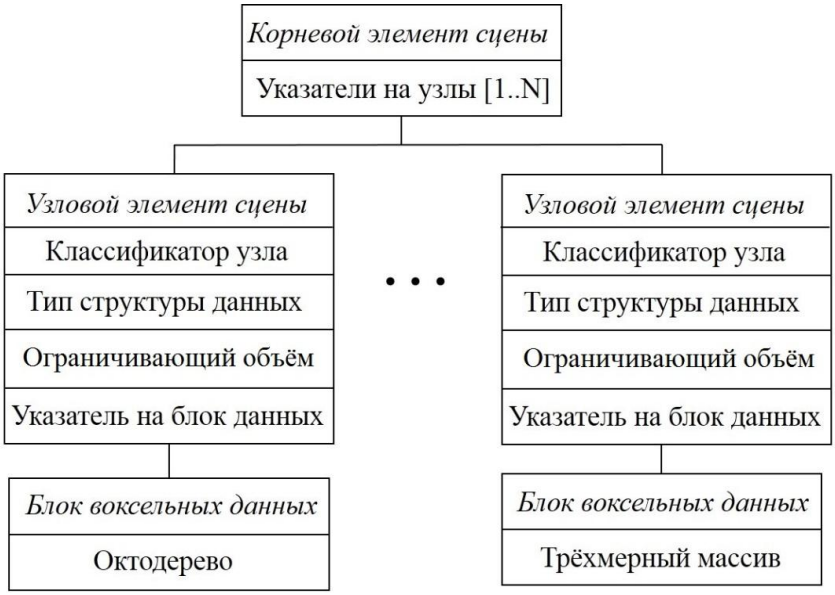

Рис. 8. Организация на уровне данных

В каждый такой узел сцены в будущем возможно будет добавлять различную дополнительную информацию, такую, как, например, палитру или уровень детализации для октодерева, что позволит дополнительно сократить объём данных, либо повысить производительность визуализации.

Выполнение трассировки лучей на такой структуре данных несколько изменится. Обход массива или обход октодерева, используемый в большинстве методов прямого рендеринга, таких, как метод сплэттинга (бросания снежков) [16, 17] и его модификаций [18, 19], дополнится определением пересекаемого ограничивающего объёма узла, а затем обходом только блока данных закреплённым за ним. Другие методы прямого рендеринга (методы “от изображения”) в данном случае будут менее эффективными, так как обход данных внутри блока будет прерываться, либо будет требовать дополнительных (повторных) вычислений для отбрасывания лучей, не попадающих в текущий ограничивающий объём.

Для первоначального определения тех ограничивающих объёмов, через которые проходит луч, предлагается воспользоваться одним из известных методов, например, методом склонов, предложенным в [20]. А последующий обход данных внутри каждого узла должен выполняться в соответствии с внутренней организацией данных. Если в узле содержится массив – должен выполняться обход массива, если октодерево – обход октодерева, например, используя актуальный в настоящее время метод обхода, рассмотренный в [21]. Благодаря этому, преимущества обоих способов сохраняются.

Базовый (общий) алгоритм трассировки луча для сцены с таким способом организации будет иметь ряд отличительных особенностей. Во-первых, в нём учитывается, требуется ли определять пересечение очередного узла по его классу, таким образом реализуется разделение объектов на классы. Во-вторых, рассматриваются только узлы, что реализует разделение вокселей на классы по принадлежности к объекту. В-третьих, структуры, содержащиеся в узлах обрабатываются поразному. Процедура трассировки луча (Луч) общего назначения (рис. 9) для всех узлов (Узл) сцены (КорнЭлем) имеет вид:

```
Процудура Трассировка_Луча(Луч)
1 Для всех узлов Узл из КорнЭлем.Узлы
2       Если Узл.Класс = нкласс
3             Если Пересечение(Луч, Узл.ГО)
4                   Если Узл.ТипСтрДанн = массив
5                         ПоискВМассиве(Узл.Бвд)
6                   Иначе
7                         ПоискВОктодереве(Узл.Бвд)
```

Рис. 9. Базовая процедура трассировки луча

В целом, данный подход подразумевает использование такого алгоритма для широкого круга задач – от трассировки лучей при визуализации до проведения операций с единичными вокселями. Очевидно, что для той или иной задачи данный алгоритм необходимо модифицировать соответственно преследуемой цели. Если необходима обработка всей группы вокселей, то необходимо производить перебор узлов только после обработки узла целиком, если же необходимо обрабатывать разные воксели из разных узлов, то, возможно, имеет смысл помещать их в некоторый буфер при малом количестве.

Выводы. При использовании рассмотренного подхода достигается следующее:

1. Исходная сетка высокого разрешения преобразуется на некоторое количество малых сеток (узлов).
2. Пустые воксели, не попавшие ни в один узел, не хранятся и не обрабатываются.
3. Воксели в разных узлах независимы, хранятся и обрабатываются поразному.
4. При модификации вокселей не требуется реорганизации всей сцены, требуется только повторный пересчёт ограничивающего объёма.
5. Трассировщик лучей получает возможность обрабатывать только ограниченные группы вокселей, распределённые по принадлежности к конкретному объекту.
6. Закреплённые за узлам группы вокселей могут быть оперативно изъяты из обработки, что позволит сократить затраты на обработку и визуализацию;
7. Достигается возможность комбинирования различных способов хранения и обработки воксельных данных.

Заключение. Существующие проблемы воксельной графики могут быть решены не только за счёт увеличения вычислительной мощности и аппаратной поддержки, но и за счёт внедрения эффективных методов обработки и визуализации в частных случаях. Такие методы особенно важны в ряде быстро развивающихся отраслей, в которых визуализация является второстепенной задачей, поэтому не может использовать все ресурсы ЭВМ. Предложенный способ визуализации трёхмерных воксельных сцен может быть внедрён в различные области науки и техники, где необходимо компактное представление и эффективная визуализация специфических типов сцен, содержащих объекты в воксельном представлении. Достоинствами предложенного метода можно назвать, во-первых, перевод визуализации однородного массива вокселей в визуализацию организованных групп вокселей. Во-вторых, разделение вокселей и разделение самих объектов, позволяющее вести работу с каждым классом объектов отдельно, высвобождая некоторое количества памяти и вычислительных мощностей. В-третьих, логичное, приближенное к натуральному, представление сцены и видимых объектов необходимое для большинства современных задач. Основное применение данный метод может найти в области симуляционного и модельного программного обеспечения, где для этого существуют все предпосылки.

БИБЛИОГРАФИЧЕСКИЙ СПИСОК

1. *Purgathofer W., Tobler R.* Current Trends in Computer Graphics // Buletinul Institutului Politehnic din Iasi. – 2010. – Vol. 56, no. 2. – P. 9-24.
2. *Gibson S.* Beyond Volume Rendering: Visualization, Haptic Exploration, and Physical Modeling of Voxel-based Objects // Visualization in Scientific Computing. – 1995. – Vol. 32, no. 1. – P. 10-24.
3. *Funkhouser T.* Solid Modeling // Computer Graphics Fall, Princeton University. – 2000. – P. 1-22.
4. *Elvins T.* A survey of algorithms for volume visualization // SIGGRAPH Computer Graphics. – 1992. – Vol. 26, no.3. – P. 194-201.
5. *Meagher D.* Octree encoding: a new technique for the representation, manipulation and display of arbitrary 3-D objects by computer. – Rensselaer Polytechnic Institute Report IPL-TR-80-111, 1980.
6. *Eyiyurekli M. Breen D.* Data structures for interactive high resolution level-set surface editing // Graphics Interface (G.I.). – 2011. – Vol. 37, no. 1. – P. 95-102.
7. *Komma P., Fischer J., Duffner F.* Lossless Volume Data Compression Schemes // Simulation and Visualization. – 2007. – Vol. 18. – P. 169-182
8. *Knoll A.* A Survey of Octree Volume Rendering Methods // 1st IRTG Workshop. – 2006. – P. 8-19.
9. *Hoffmann C., Popescu V., Kilic S.* Integrating modeling, simulation, and visualization // Computers in Science & Engineering. – 2004. – Vol. 6, no. 1. – P. 52-60.
10. *Miller A., Allen P., Santos V.* From robotic hands to human hands: a visualization and simulation engine for grasping research // Industrial Robot: An International Journal. – 2005. – Vol. 32, no. 1. – P. 55-63.
11. *Frits H., Walsum T.* Fluid flow visualization // Computer Graphics: Systems and Applications. – 1993. – Vol. 1. – P. 1-40.
12. *Kämpe V., Sintorn E., Assarsson U.* High Resolution Sparse Voxel DAGs // SIGGRAPH Computer Graphics. – 2013. – Vol. 32, no. 4. – P. 39-45.
13. *Rubin S., Whitted T.* A 3-Dimensional Representation for Fast Rendering of Complex Scenes // Computer Graphics (proc. SIGGRAPH '80). – 1980. – Vol. 14, no. 3. – P. 110-116.
14. *Laine S., Karras T.* Efficien2t Sparse Voxel Octrees // Transactions on Visualization & Computer Graphics. – 2011. – Vol. 17, no. 8. – P. 1048-1059.
15. *Jansen F.* Data Structures for Ray Tracing // Data Structures for Raster Graphics. – 1986. – P. 57-73.
16. *Westover L.* Interactive volume rendering // Volume visualization (proc. '89 Chapel Hill Workshop). – 1989. – P. 9-16.
17. *Westover L.* Footprint Evaluation for Volume Rendering // Computer Graphics. – 1990. – Vol. 24, no. 4. – P. 367-376.
18. *Mueller K., Crawfis R.* Eliminating Popping Artifacts in Sheet Buffer-Based Splatting // Computer Graphics and Visualization. – 1998. – Vol. 15, no 3. – P. 239-246.
19. *Laur D., Hanrahan P.* Hierarchical Splatting: A Progressive Refinement Algorithm for Volume Rendering // Computer Graphics (proc. SIGGRAPH '91). – 1991. – Vol. 25, no. 2. – P. 285-288.
20. *Eisemann M., Grosch T., Müller S.* Fast Ray Axis-Aligned Bounding Box Overlap Tests using Ray Slopes // Journal of Graphics Tools (JGT). – 2007. – Vol. 12. – P. 35-46.
21. *Frisken S., Perry R.* Simple and efficient traversal methods for quadtrees and octrees // Journal of Graphics Tools. – 2002. – Vol. 7. – P. 202-215.

REFERENCES

1. *Purgathofer W., Tobler R.* Current Trends in Computer Graphics, *Buletinul Institutului Politehnic din Iasi*, 2010, Vol. 56, No. 2, pp. 9-24.
2. *Gibson S.* Beyond Volume Rendering: Visualization, Haptic Exploration, and Physical Modeling of Voxel-based Objects, *Visualization in Scientific Computing*, 1995, Vol. 32, No. 1, pp. 10-24.
3. *Funkhouser T.* Solid Modeling, *Computer Graphics Fall, Princeton University*, 2000, pp. 1-22.
4. *Elvins T.* A survey of algorithms for volume visualization, *SIGGRAPH Computer Graphics*, 1992, Vol. 26, No. 3, pp. 194-201.

5. *Meagher D.* Octree encoding: a new technique for the representation, manipulation and display of arbitrary 3-D objects by computer. – Rensselaer Polytechnic Institute Report IPL-TR-80-111, 1980.
6. *Eyiyurekli M. Breen D.* Data structures for interactive high resolution level-set surface editing, *Graphics Interface (G.I.),* 2011, Vol. 37, No. 1, pp. 95-102.
7. *Komma P., Fischer J., Duffner F.* Lossless Volume Data Compression Schemes, *Simulation and Visualization,* 2007, Vol. 18, pp. 169-182
8. *Knoll A.* A Survey of Octree Volume Rendering Methods, *1st IRTG Workshop,* 2006, pp. 8-19.
9. *Hoffmann C., Popescu V., Kilic S.* Integrating modeling, simulation, and visualization, *Computers in Science & Engineering,* 2004, Vol. 6, No. 1, pp. 52-60.
10. *Miller A., Allen P., Santos V.* From robotic hands to human hands: a visualization and simulation engine for grasping research, *Industrial Robot: An International Journal,* 2005, Vol. 32, No. 1, pp. 55-63.
11. *Frits H., Walsum T.* Fluid flow visualization, *Computer Graphics: Systems and Applications,* 1993, Vol. 1, pp. 1-40.
12. *Kämpe V., Sintorn E., Assarsson U.* High Resolution Sparse Voxel DAGs, *SIGGRAPH Computer Graphics,* 2013, Vol. 32, No. 4, pp. 39-45.
13. *Rubin S., Whitted T.* A 3-Dimensional Representation for Fast Rendering of Complex Scenes, *Computer Graphics (proc. SIGGRAPH '80),* 1980, Vol. 14, No. 3, pp. 110-116.
14. *Laine S., Karras T.* Efficien2t Sparse Voxel Octrees, *Transactions on Visualization & Computer Graphics,* 2011, Vol. 17, No. 8, pp. 1048-1059.
15. *Jansen F.* Data Structures for Ray Tracing, *Data Structures for Raster Graphics,* 1986, pp. 57-73.
16. *Westover L.* Interactive volume rendering, *Volume visualization (proc. '89 Chapel Hill Workshop),* 1989, pp. 9-16.
17. *Westover L.* Footprint Evaluation for Volume Rendering, *Computer Graphics,* 1990, Vol. 24, No. 4, pp. 367-376.
18. *Mueller K., Crawfis R.* Eliminating Popping Artifacts in Sheet Buffer-Based Splatting, *Computer Graphics and Visualization,* 1998, Vol. 15, No. 3, pp. 239-246.
19. *Laur D., Hanrahan P.* Hierarchical Splatting: A Progressive Refinement Algorithm for Volume Rendering, *Computer Graphics (proc. SIGGRAPH '91),* 1991, Vol. 25, No. 2, pp. 285-288.
20. *Eisemann M., Grosch T., Müller S.* Fast Ray Axis-Aligned Bounding Box Overlap Tests using Ray Slopes, *Journal of Graphics Tools (JGT),* 2007, Vol. 12, pp. 35-46.
21. *Frisken S., Perry R.* Simple and efficient traversal methods for quadtrees and octrees, *Journal of Graphics Tools,* 2002, Vol. 7, pp. 202-215.

Статью рекомендовал к опубликованию д.т.н., профессор О.Б. Макаревич.

Витиска Николай Иванович – Таганрогский институт им. А.П. Чехова (филиал) Ростовского государственного экономического университета (РИНХ); e-mail: vit614294@rambler.ru; 347936, г. Таганрог, ул. Инициативная, 48; кафедра информатики; д.т.н.; профессор.

Гуляев Никита Андреевич – e-mail: m.yo.da@yandex.ru; кафедра ИСиПИ; аспирант.

Vitiska Nikolay Ivanovich – Rostov State University of Economics; e-mail: vit614294@rambler.ru; 48, Initsiativnaya street, Taganrog, 347936, Russia; the department of informational systems; dr. of eng. sc.; professor.

Gulyaev Nikita Andreevich – e-mail: m.yo.da@yandex.ru; the department of information systems and applied informatics; postgraduate student.