Иерархическая дискретная трассировка лучей в октантных деревьях

М. Цыганков
Softgraph International GmbH

Аннотация

Данная статья посвящена алгоритму дискретной трассировки лучей, использующему иерархические структуры данных для представления воксельного пространства. Использование октантных деревьев значительно уменьшает объем памяти, необходимый для хранения трехмерных моделей. Средние значения коэффициента сжатия воксельных моделей находятся в диапазоне от 1:24 до 1:50 для моделей полученных из сканеров объема, например, компьютерных томографов или магнитно-резонансных сканеров. Разработанные методы позволяют визуализировать 256^3 и 512^3 объемы практически в реальном масштабе времени на обычном ПК (Intel Pentium-133). Использование однородного воксельного пространства позволяет компоновать модели из различных источников: биомедицинских сканеров, измерительных устройств, моделей конструктивной объемной геометрии (CSG) и полигональных моделей.

Ключевые слова: Визуализация объемов, дискретная трассировка лучей, октантное дерево.

1. ВВЕДЕНИЕ

В последние годы отмечается значительный рост количества приложений в области визуализации объемов. Этот процесс напоминает революцию компьютерной графики, произошедшую в конце 60-х годов при переходе от векторной 2D к растровой 2D графике. Аналогичная картина царит сейчас и в некоторых областях трехмерной графики, в основном там, где занимаются визуализацией наборов данных изначально организованных как трехмерные массивы значений какого-либо параметра. Чаще всего такая информация приходит с различных сканирующих устройств (напр., медицина) или в результате численного эксперимента (напр., динамика жидких сред).[3, 8]

Такие изменения, главным образом, обусловлены достижениями компьютерных технологий в области аппаратного обеспечения - быстрые процессоры, более объемная, дешевая и быстрая оперативная и внешняя память и т.д. Однако, даже при таком значительном прогрессе визуализация объемов была недоступна для среднестатистических ПК до самого недавнего времени.

Данная статья посвящена одному из методов представления и визуализации воксельного пространства, который может продуктивно использоваться даже на недорогих ПК со средними размерами ОЗУ и внешней памяти.

2. СТРУКТУРЫ ДАННЫХ

Описываемый алгоритм использует октантные деревья для сжатия однородного воксельного пространства. Модель строится из трехмерного массива элементов объема - вокселей (voxels). Все воксели имеют форму куба и одинаковый размер по трем координатным осям. Октантное дерево организует иерархическую структуру группировки вокселей: Весь объект хранится в боксе. Этот бокс делится на восемь равных подбоксов плоскостями, перпендикулярными осям координат. Эти подбоксы, в свою очередь, снова делятся на под-подбоксы и т.д. Этот процесс заканчивается на уровне вокселей - элементарных неделимых единиц объема. При такой схеме кодирования воксельного пространства октантное дерево может представлять кубический объем пространства из $(2^N)^3$ вокселей, где N - глубина дерева.

Каждый воксель внутри модели может быть: либо прозрачным («воздух») либо принадлежать поверхности объекта («кожа»). Воксели «воздуха» - это прозрачные области пространства, они не нужны для представления модели. Структура октантного дерева позволяет избежать хранения «воздушных» подбоксов на каждом уровне иерархии, в результате чего существенным образом уменьшается объем памяти, требуемый для хранения моделей.

Воксели «кожи» содержат информацию о свойствах поверхности в точке - цвет, внешняя нормаль, коэффициент отражения/преломления и т.д. для проведения необходимых вычислений во время построения изображения.

Такая модель данных не позволяет эффективно представлять трехмерные плотностные модели (3D density fields) в необработанном виде. Однако, после предобработки (называемой также *двоичной квантизацией* [7]) визуализировать такие данные (компьютерная томография и магнитно-резонансное сканирование) не представляет большой сложности. В настоящее время разработано множество методов реконструкции изоповерхностей из трехмерных массивов значений плотности или какого-либо другого параметра. [6, 7, 9]

Для того, чтобы хранить информацию только о поверхности объекта каждый непустой подбокс имеет один байт специальной информации (инфобайт). Каждый бит этого байта говорит содержит ли соответствующий ему подбокс воксели поверхности или нет. Только те подбоксы, которые содержат поверхность, будут иметь потомков на нижних уровнях

иерархии. Подбоксы «воздуха» не отслеживаются вниз по дереву.

В сравнении с обычным однородным представлением трехмерного пространства [3, 10], где каждый воксель содержит информацию, а для хранения всего пространства требуются гигантские объемы памяти, октантные деревья являются существенным менее требовательными. Например, для хранения объектов состоящих из 256^3 вокселей требуется до 98 Мбайт: каждый воксель содержит 3 байта для хранения цвета поверхности + 3 байта для хранения нормали + 1/8 байта на тип вокселя («воздух»-«кожа»), итого 6.125 байт/воксель. Прямое перемножение 256^3 на 6.125 дает 98 Мбайт информации. Применение октантных деревьев позволяет сократить этот объем до 1-2.5 Мбайт (см. примеры в табл. 1).

Модель	Разрешение по x,y и z (воксель)			Размер модели (байт)
Dice	128	128	128	533,298
Molecule	451	315	450	2,539,356
Teapot	512	324	258	1,596,509
Mrbrain	190	175	156	1,034,480
3Dhead	199	187	156	1,483,655
Cthead	172	220	223	1,259,417

Таблица 1: Объемы памяти для типичных моделей.

3. АЛГОРИТМ ВИЗУАЛИЗАЦИИ

3.1 Общая идея

Структура октантных деревьев не позволяет напрямую использовать методы отслеживания луча в однородном трехмерном пространстве (типа 3D алгоритма Брезенхема или трехмерного Цифрового Дифференциального Анализатора [1]). Для решения этой задачи были разработаны специальные методы, ориентированные на работу в иерархических структурах данных.

Рис 1. иллюстрирует основной принцип метода: луч отслеживается из точки наблюдения, пересекает плоскость экрана в точке (x_s,y_s) и ударяется в бокс, содержащий дискретную информацию об объекте. Бокс разделен на подбоксы, пронумерованные от 2^0 до 2^7, которые далее рекурсивно делятся на более мелкие подбоксы, формируя октантное дерево. (В данной статье все ячейки пространства, отличные от вокселя именуются «подбоксами» или «боксами» и только элементарные неделимые элементы объема называются вокселями.)

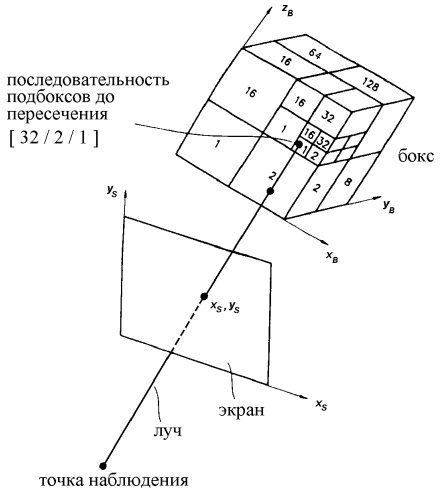

Рисунок 1: Основная идея метода

В основном, алгоритм работает как многие методы обратной трассировки лучей (backward raytracing). Лучи испускаются из точки наблюдения через пиксели на экране и отслеживаются до момента пересечения с объектом. В этой точке вычисляется цвет пикселя или порождаются вторичные отраженные или преломленные лучи, если таковые имеются. Главный вопрос трассировки лучей: «Как найти точку пересечения луча и объекта?». Алгоритм решает эту задачу следующим образом:

- Находит точку пересечения луча и бокса, содержащего объекта (вычисляются точки входа и выхода луча из бокса)
- Используя информацию о точках пересечения, алгоритм отслеживает луч внутри бокса и находит первый пересеченный воксель поверхности по ходу луча
- Используя информацию о свойствах поверхности, содержащуюся в вокселе (цвет, нормаль и т.д.), вычисляет цвет пикселя или испускает вторичные лучи

3.2 Поиск точки пересечения луча и модели

На рис.2 и рис.3 проиллюстрирован метод поиска точки пересечения.

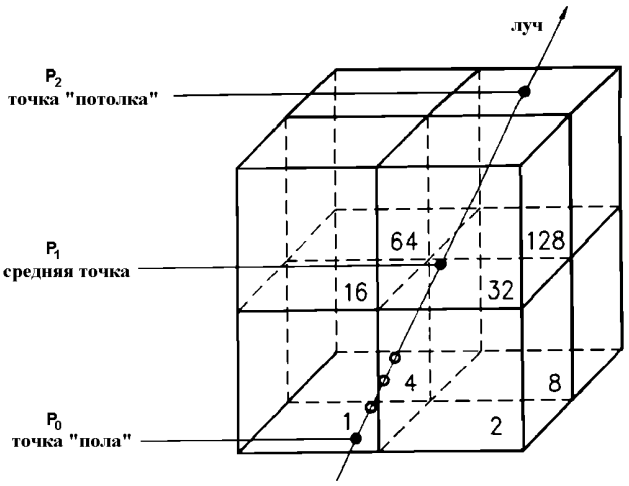

Рисунок 2: Поиск вокселя пересечения

Луч входит в бокс в *точке «пола» P_0*, проходит через *среднюю точку P_1* и покидает бокс в *точке «потолка» P_2*. Важно, чтобы эти три точки располагались на трех параллельных плоскостях, две из которых - плоскости, содержащие противоположные грани бокса, а третья - делит бокс на две равных половины. Совершенно очевидно, что точки P_0 и P_2 не обязательно будут принадлежать граням бокса, в то время как *средняя точка P_1* обязательно будет находиться внутри бокса.

Всего может быть три варианта выбора для плоскостей, содержащих *точки «пола»* и «*потолка*»: плоскости могут быть перпендикулярны одной из трех координатных осей *X, Y* или *Z*. Пусть точка на луче описывается в системе координат бокса следующим уравнением:

$$P = S_0 + t \cdot (dx,dy,dz) \qquad (1)$$

где: S_0 - начальная точка луча
(dx, dy, dz) - вектор направления луча
t - скалярный параметр

Тогда *осью главного направления луча (ОГНЛ)* называется ось, составляющая которой в векторе направления луча преобладает над другими. То есть:

$$ОГНЛ = \begin{cases} X, dx = \max(dx, dy, dz) \\ Y, dy = \max(dx, dy, dz) \\ Z, dz = \max(dx, dy, dz) \end{cases}$$

Методика выбора главного направления совпадает с использованной в целочисленном алгоритме Брезенхема [1] и служит тем же целям: избежать появления разрывов во время дискретизации прямой луча.

Плоскости «пола» и *«потолка»* выбираются перпендикулярными оси главного направления луча.

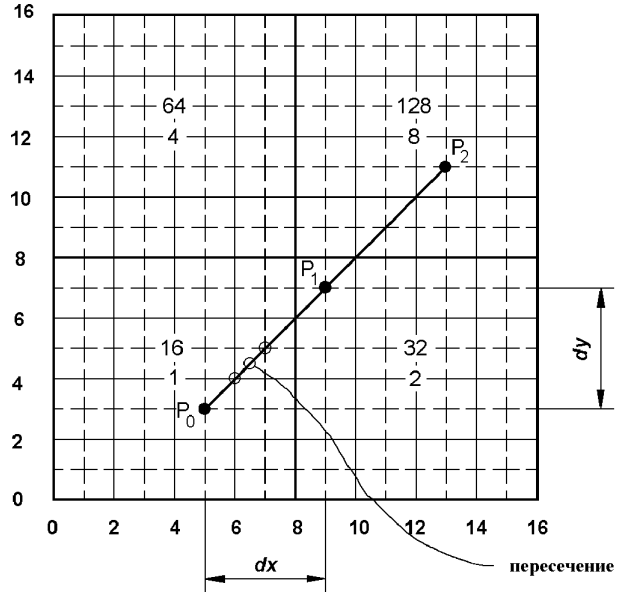

Рисунок 3: Вид бокса в проекции на плоскость *xy*

На рис.3 можно видеть проекцию бокса на плоскость, перпендикулярную оси главного направления луча. Информация о подбоксах, пересеченных лучом, может быть достаточно просто вычислена на основании двумерных координат точек P_0, P_1 и P_2 в системе координат *плоскостей «пола»*, «*потолка»* и *средней* плоскости.

Процесс определения списка задетых лучом подбоксов иллюстрируется на рис.4 для точки P_1 с координатами *x=9* (в двоичной системе - 1001) и *y=7* (0111). Каждый бит представляет соответствует одному уровню иерархии. Например, для уровня глубины *D = 1* в координатах точки P_1 мы имеем 1 (первый бит слева из *x*) и 0 (из *y*). По этим битам и информации о *z-координате* точек мы можем определить те подбоксы, которым принадлежат *точки «пола»* и *«потолка»*, и, пользуясь этим, определить путь луча внутри бокса.

Точки P_0 и P_2 находятся в подбоксах $2^{(00)}=1$ и $2^{4+(11)}=128$ соответственно, а точка P_1 принадлежит одновременно двум подбоксам - $2^{(01)}=2$ и $2^{4+(01)}=32$ ибо она находится точно посередине бокса, то есть между двумя «этажами».

Путь луча, таким образом, будет представлен последовательностью подбоксов 1/2/32/128. В байтовом представлении - 10100011. Инфобайт из текущего узла дерева (например, 00110101) показывает, что подбоксы 1, 4, 16 и 32 содержат поверхность. Логическая операция *И* этих двух байтов (пути луча и инфобайта) дает информацию о всех непустых подбоксах, которые пересечены лучом. В нашем примере - подбоксы 1 и 32 (10100011 *И* 00110101 = 0100001). Теперь нужно рассмотреть все пробитые подбоксы в порядке прохождения луча. В принятой системе нумерации подбоксов нам достаточно просто

выбирать ненулевые биты из результирующего байта слева направо.

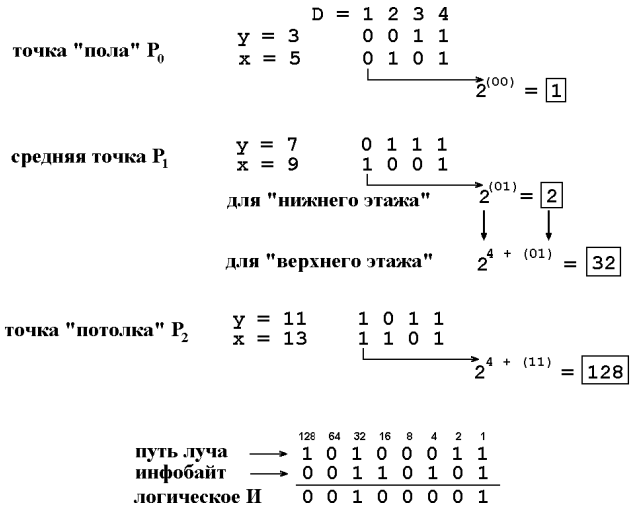

Рисунок 4: Двоичные вычисления при поиске пересеченных подбоксов

Для ускорения процесса вычислений в программной реализации применена look-up таблица, на вход которой подается 14-битовый индекс (8-битовый инфобайт текущего узла дерева и 6 битов из составляющих координат). На выходе из таблицы имеем список подбоксов, которые содержат поверхность и пересечены лучом.

3.3 Избежание «туннельного» эффекта

Изложенный алгоритм может порождать последовательности подбоксов, примыкающих друг к другу только ребрами (например, 1/16/128). В этом случае луч может «попасть в туннель» в поверхности, воксели которой расположены в диагональном порядке (см. рис. 5).

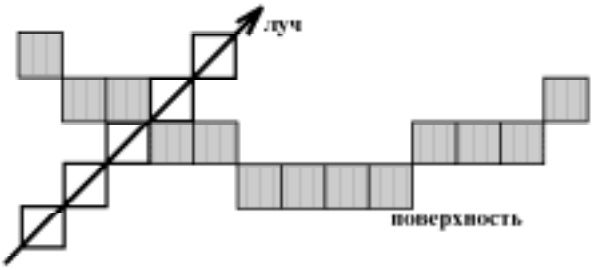

Рисунок 5: Проявление «туннельного» эффекта

Для избежания проявления описанного эффекта алгоритм вычисления пути луча должен быть изменен таким образом, чтобы все воксели на пути луча примыкали друг к другу гранями. В нашем примере в путь луча должны быть добавлены воксели 32 и 64. Разумеется, луч не может *одновременно* пересечь эти воксели, но отслеживание луча но более низких уровнях иерархии позволит выбрать из них самый подходящий.

По терминологии введенной в [3, 10], такой луч является *6-связным*: любые два соседних вокселя имеют общую грань. Ягель и Кауфман показали, что туннельный эффект отсутствует при трассировке *6-связными* лучами *26-связных* поверхностей (когда любые два соседних вокселя поверхности имеют по крайней мере общую вершину). Применение *26-связных* поверхностей позволяет значительно сократить количество вокселей, требуемых для представления моделей, хотя это несколько замедляет процесс отслеживания лучей.

3.4 Положение объекта в пространстве

Так как информация о топологии поверхности приобщена к боксу, все аффинные преобразования модели в составе сцены затрагивают только координаты и ориентацию бокса, оставляя нетронутыми воксельные поверхности внутри него. Для отслеживания луча в боксе должны быть определены только *точка «пола»* P_0 и «*потолка*» P_2 (см. рис. 2). Для этого необходимо, пользуясь правилами геометрии, вычислить точки пересечения прямой (1) и *плоскостей «пола»* и «*потолка»*, преобразовав их затем в систему координат бокса.

Эти вычисления требуют больших затрат процессорного времени, которые превосходят время необходимое на отслеживание луча внутри бокса. Для ускорения этого процесса применена следующая идея: так как входной параметр (например, x_0) для трассировки луча внутри бокса вычисляется индивидуально для каждого пикселя на экране, можно заменить громоздкие векторные вычисления интерполяцией ($x0=f(x_S,y_S)$, где x_S и y_S - координаты пикселя, через который испускается луч). Эта трехмерная функция может быть интерполирована простым линейным функционалом для случая параллельной проекции. Более того, при интерполяции линейной функции каждое следующее значение может быть вычислено на основе предыдущего, путем добавления постоянного приращения.

Таким образом, для параллельной проекции можно организовать вычисления координат *точек «пола»* и «*потолка»* для каждого пикселя при помощи четырех операции сложения с плавающей точкой. Также реализована интерполяция, использующая только целочисленную арифметику (сложение и сдвиг). Такой подход позволяет в полной мере использовать возможности архитектур с усеченным набором инструкций (RISC) или заказные прикладные ИС (ASIC).

Также весьма просто решается задача визуализации при помещении точки наблюдения внутрь модели: достаточно лишь ограничить минимальную координату *z* во время отслеживания луча (см. рис. 10).

4 ГЕНЕРАЦИЯ ВОКСЕЛЬНЫХ МОДЕЛЕЙ

Воксельные модели могут быть построены на основе данных из следующих источников:

- Конструктивная объемная геометрия (CSG) см. рис. 7
- Данные со сканеров объема (CT, MR и т.д.) см. рис. 8 - 10
- Полигональные модели (системы 3d-моделирования, CAD/CAM и т.д.) см. рис. 6

Разработаны и реализованы специальные алгоритмы манипуляций и построения октантных деревьев из воксельных пространств. Можно компоновать полигональные или воксельные модели с CSG-примитивами. Все логические операции конструктивной объемной геометрии проводятся в воксельном пространстве. Для ускорения и уменьшения требуемых объемов ОЗУ при работе с воксельными наборами данных использованы механизмы *своппинга*, *кеширования* и *компрессии информации «на лету»*. Например, при построении типичных моделей размером 512^3 вокселей необходимо примерно 190 Мбайт для хранения информации о модели. Компрессированные данные требуют уже только 11 Мбайт в ОЗУ или на жестком диске. Применение указанных выше методик сделало возможным построение достаточно больших моделей (до 1024^3) на обычном ПК с ОЗУ объемом 8 Мбайт и работающем даже под управлением MS-DOS (то есть в отсутствие штатного менеджера виртуальной памяти).

Нормаль к поверхности и цвет хранится в каждом вокселе, это позволяет наносить на поверхности цветовую текстуру (см. рис. 9 - 10). Направление нормали определяется в процессе вокселизации. Метод, используемый для вычисления нормали, зависит от природы вокселизуемой поверхности:

Воксели и нормали к аналитическим поверхностям (CSG-примитивы, полигоны) могут быть получены непосредственно. Применение look-up таблиц или метода конечных разностей для интерполяции значений позволяет значительно ускорить этот процесс.

Неаналитические поверхности, порожденным из данных от измерительных устройств (CT, MR и т.д.), могут быть построены при помощи одного из методов *двоичной сегментации* объема: контекстного, градиентного, биквадратной интерполяции поверхности и т.д [2, 5-7] В описываемой системе применена модификация градиентного метода, которая для оценки направления нормали использует значения соседних вокселей в окрестности 5x5x5 а для двоичной сегментации. При помощи фильтрации в трехмерном воксельном пространстве все поверхности приводятся к *6-связному* виду.

5 ЗАКЛЮЧЕНИЕ

5.1 Основные достоинства дискретной трассировки лучей

В сравнении с традиционными полигональными методиками представления поверхностей, дискретная трассировка лучей имеет следующие преимущества:

- Количество информации и скорость визуализации почти не зависят от сложности объектов
- Нет необходимости в специальных алгоритмах нанесения текстур, так как каждый воксель имеет собственный цвет
- Высокая точность определения нормалей, так как они сопоставлены каждому вокселю
- Высокая точность определения точек пересечения
- Есть созможность непосредственной визуализации наборов данных с измерительных устройств без преобразования в сложные полигональные модели
- Простота реализации алгоритмов определения пересечения двух объектов (например, для CAD/CAM систем)

5.2 Достоинства предложенного алгоритма в сравнении с обычными методиками дискретной трассировки лучей

Изложенный в статье алгоритм обладает следующими дополнительными преимуществами по сравнению с базовыми методами дискретной трассировки лучей:

- Требуется значительно меньшее количество данных для представления моделей, так как хранится и обрабатывается только информация о поверхности: увеличение разрешения модели в два раза по каждой из координатных осей приводит к росту объема данных в модели всего в 4 раза (в противоположность однородной схеме представления пространства, где увеличение разрешения в два раза приводит к восьмикратному росту объема информации).
- Более высокая скорость: типичные медицинские объекты полученные методом компьютерной томографии (разрешение воксельного пространства – 256^3, разрешение изображения - 320^2; см. рис. 8-10) визуализируются за 0.3-0.9 секунд/кадр на ПК с процессором Пентиум с тактовой частотой 133 МГц (разброс в длительности интервалов обусловлен различием в углах наблюдения моделей). Такая скорость визуализации достигнута благодаря следующим свойствам алгоритма:

- Главным образом, вычисления проводятся с использованием целочисленных операций типа сдвига, сложения, логического «И» и т.д.
- Наиболее длительные операции были ускорены с использованием look-up таблиц или интерполяции конечными разностями
- Использование иерархических структур данных позволяет сократить количество операций при поиске точки пересечения луча и объекта
- Использована когерентность цветов соседних пикселей, что позволяет значительно сократить количество лучей, которые необходимо отследить для получения конечного изображения

Описанный в статье метод визуализации комбинирует преимущества дискретного представления моделей в виде воксельных пространств и широко применяемую в системах трассировки лучей методику неоднородного разбиения пространства. Использование октантных деревьев позволяет на порядок сократить объемы памяти, требуемые для хранения моделей. Применение в алгоритме главным образом целочисленных операций существенно упрощает реализацию на аппаратном уровне.

В заключение хочется выразить благодарность за ценные идеи и неоценимую помощь в работе профессору П. Крумхауеру из Берлинского Технического Университета и доктору Х. Райху из Европейского Института Информатики.

Литература

[1] Д. Роджерс, «Алгоритмические основы машинной графики», М.:Мир, 1989, стр. 50-63

[2] D. Cohen and A. Kaufman, "Scan-Conversion Algorithms for Linear and Quadratic Objects", Volume Visualization, A. Kaufman, ed., IEEE Computer Society Press, Los Alamitos, CA, USA, 1990, pp. 280-301

[3] A. Kaufman, D. Cohen and R. Yagel, "Volume Graphics", IEEE Computer, Vol. 26, 7, July 1993, pp.51-64

[4] M. Levoy, "Display of Surfaces from Volume Data", IEEE Computer Graphics & Applications, Vol.8, 5, May 1988, pp. 29-37

[5] T. Mueller and R. Yagel, "Efficient Rasterization of Implicit Functions", OSU-CISRC-11/95-TR49, Department of Computer Science, The Ohio State University, November, 1995

[6] K. Mueller, R. Yagel, and J.F. Cornhill, "Accelerating the Anti-aliased Algebraic Reconstruction Technique (ART) by Table-Based Voxel Backward Projection", Proceedings of IEEE EBMS 17th annual Conference, EBMS 95, Montreal, Canada, 1995, pp. 497.

[7] U. Tiede et al, "Investigation of 3D-Rendering Algorithms", IEEE Computer Graphics & Applications, Vol. 10, 3, March 1990, pp. 41-53

[8] R. Yagel, "Towards Real Time Volume Rendering", Proceedings of GraphiCon'96, Vol 1., St.Petersburg, Russia, July 1996, pp. 230-241

[9] R. Yagel, D. Cohen and A. Kaufman, "Normal Estimation in 3D Discrete Space", The Visual Computer, 8, 5-6, June 1992, pp. 278-291.

[10] R. Yagel, D. Cohen and A. Kaufman, "Discrete Ray Tracing", IEEE Computer Graphics & Applications, Vol. 12, 9, September 1992, pp. 19-28

Сведения об авторе

Цыганков Михаил Арнольдович

Руководитель отдела исследований и разработок
Российского отделения Softgraph Intl. GmbH.
Член IEEE Computer Society.

Адрес: Россия, 199053, Санкт-Петербург, п/я 703.

Email: Mike@Softgraph.com

Телефоны: +7-812-3279900
+7-812-5877331
факс: +7-812-3279865

Hierarchical Discrete Ray Tracing in Octrees

Abstract

This paper describes a discrete ray tracing algorithm which uses hierarchical data structures (octrees) for 3D uniform voxel space representation. Usage of octrees dramatically reduces memory amount needed to store 3D models. Average compression ratio achieved is in range between 1:24 up to 1:50 for real-world binary segmented models (CT and MRI data sets). Ray casting methods applied made it possible to render 256^3 and 512^3 volumes nearly in real-time on standard PCs (Intel Pentium-133). Usage of uniformly partitioned voxel space for representing in octree allows to compose input models from many sources like biomedical scanners, measuring devices, constructive solid geometry (CSG) modeling and polygonal models.

Keywords: Volume visualization, discrete ray tracing, octree

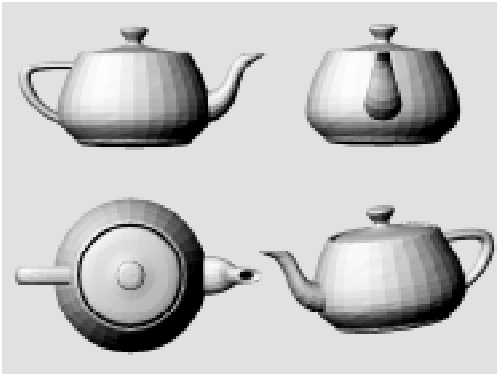

Рисунок 6: Вокселизованная полигональная модель (1024^3 вокселей)

Рисунок 7: CSG модель (256^3)

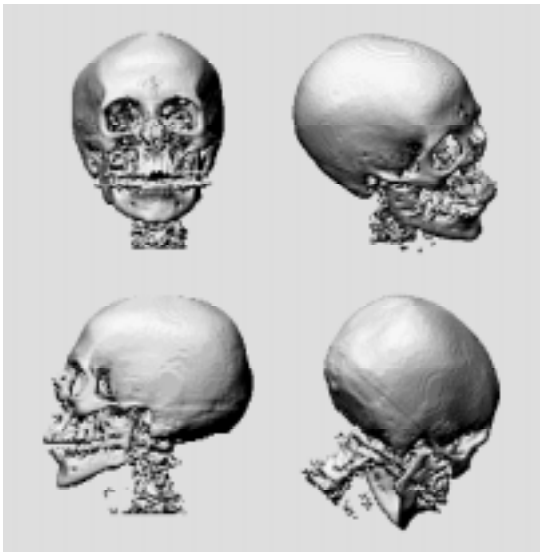

Рисунок 8: Модель черепа, построенная из срезов компьютерной томограммы (256^3 вокселей)

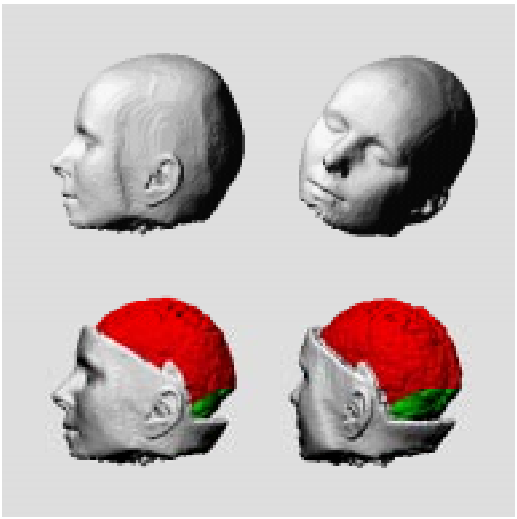

Рисунок 9: Модели, построенные из магнитно-резонансных томограмм (256^3 вокселей)

Рисунок 10: Модели с нанесенными текстурами и вид изнутри.

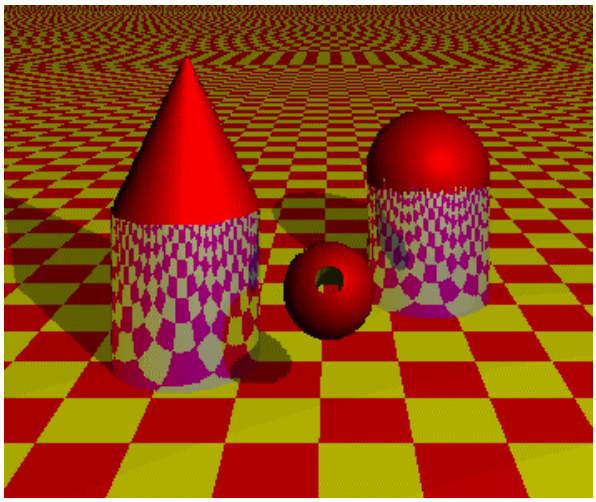

Рисунок 11: Сцена, состоящая из трех CSG объектов с эффектами отражения и тенями.