УДК 004.925.84

МЕТОД ВИЗУАЛИЗАЦИИ СЛОЖНЫХ ОБЪЕКТОВ, ПРЕДСТАВЛЯЮЩИХ КОМПОЗИЦИЮ ОБЪЁМОВ И ФУНКЦИОНАЛЬНО ЗАДАННЫЕ ПОВЕРХНОСТИ

© **С. И. Вяткин, Б. С. Долговесов**

Институт автоматики и электрометрии СО РАН,
630090, г. Новосибирск, просп. Академика Коптюга, 1
E-mail: sivser@mail.ru

Предложен метод визуализации сложных объектов, представляющих композицию объёмов и функционально заданные поверхности, с применением графических процессоров. В методе используются иерархические $3D$-структуры данных, не требующие параметризации. Описываются объединения сложных полупрозрачных объёмов и поверхностей на основе функций возмущения.

Ключевые слова: функционально заданная поверхность, функции возмущения, теоретико-множественные операции, объёмная текстура, объёмно-ориентированная визуализация, CUDA, графический процессор (GPU).

DOI: 10.15372/AUT20240203
EDN: WUYHOO

Введение. В научной визуализации актуально использование комбинаций скалярных полей и других типов данных. Например, в медицинских приложениях волоконная трактография применяется для получения информации о конфигурации и ориентации нервных путей в нейробиологии с целью лучшего понимания функционирования мозга. Для этого объединяют визуализацию скалярного поля с визуализацией волокон, поскольку они регистрируются относительно друг друга. Применяются алгоритмы рендеринга для вычисления расширенных эффектов объёмного рассеяния и затенения [1, 2].

В [3] формулируется проблема затенения в виде объёмного интеграла. Интеграл решается с использованием линейных выборок для устранения проблем недостаточной дискретизации. Обсуждаются способы включения нормали поверхности в расчёт затенения объёма.

Метод объединения скалярного объёма с векторными полями описан в [4]. Даны представление потоков для интерактивного объёмного рендеринга структуры векторного поля, а также ряд дополнительных компонентов, которые объединяются для создания эффективной визуализации многозначных научных $3D$-данных. Используется подход многослойной визуализации объёма для одновременного отображения объёма вместе с другими значениями данных. Представлены результаты визуализации данных магнитно-резонансной томографии (МРТ) с тензорной оценкой второго порядка, а также имитация $3D$-данных потока жидкости.

Рейкастинг — один из самых распространённых подходов для рендеринга объёма благодаря своей гибкости при генерации изображения высокого качества [5, 6]. Большой объём однородных данных, содержащихся в объёмной модели, очень хорошо распараллеливается при вычислениях с помощью программно-аппаратных шейдеров графического процессора (GPU).

Кроме научной визуализации и медицинских приложений, существуют и другие области применения композиций объёмов и поверхностей, например задачи цифрового изготовления для создания реальных копий виртуальных моделей. Необходимо создавать объекты

с неоднородными атрибутами видимой поверхности, такие как формы, внешняя поверхность которых состоит из областей с различным цветом, непрозрачностью и текстурой. Создание таких объектов в виде единого твёрдого тела требует использования методик изготовления с несколькими атрибутами или материалами и последующей окраски поверхности.

Альтернативой является разложение моделей с многоатрибутными поверхностями на объёмные части с одним атрибутом, соответствующие различным областям поверхности. Затем осуществляется изготовление деталей независимо, используя широкодоступные инструменты для получения одного атрибута, и далее — сборка деталей вместе, чтобы сформировать желаемый объект. Современные высококачественные $3D$-принтеры, такие как Z Corp или HP Jet Fusion, позволяют одновременно наносить материалы с различными цветами и механическими свойствами, однако, они дорогостоящие и могут сочетать только ограниченный набор материалов. Недавнее направление исследований посвящено методу окраски поверхностей изготовленных объектов, включая вычислительную гидрографическую печать [7] и вычислительную термоформовку [8]. Однако эти методы имеют свои недостатки, поэтому в этом направлении ведутся интенсивные исследования.

Целью представленной работы является создание метода визуализации сложных объектов, представляющих композицию объёмов и функционально заданные поверхности.

Функционально заданные поверхности. Для описания поверхностей используются функции отклонения (второго порядка) от базовой квадрики [9]. Функция задаётся алгебраическим неравенством второй степени с тремя неизвестными x, y, z в виде $F(x, y, z) \geq 0$. Рассматриваются поверхности как замкнутые подмножества евклидова пространства, определяемые описывающей функцией $F(x, y, z) \geq 0$, где F — непрерывная вещественная функция; x, y, z — задаваемая координатными переменными точка в E^3. Поверхности строятся с помощью квадрик и представляются композицией базовой квадрики и возмущений:

$$F'(x, y, z) = F(x, y, z) + \sum_{i=1}^{N} f_i R_i(x, y, z), \tag{1}$$

где f_i — формфактор; $R_i(x, y, z)$ — возмущение:

$$R_i(x, y, z) = \begin{cases} Q_i^3(x, y, z), & \text{если} \quad Q_i(x, y, z) \geq 0; \\ 0, & \text{если} \quad Q_i(x, y, z) < 0, \end{cases} \tag{2}$$

где $Q_i(x, y, z)$ — возмущающая квадрика. Для конструирования объектов и их композиций различной сложности используется множество геометрических операций Φ, определяемое математически следующим образом:

$$\Phi_{\text{¡}} : \mathrm{M}^1 + \mathrm{M}^2 + \ldots + \mathrm{M}^{\mathrm{n}} \to \mathrm{M}. \tag{3}$$

Для формирования моделей сложных объектов на базе функций возмущения используются теоретико-множественные операции. Они осуществляются с применением булевых операций объединения и пересечения. Бинарная операция ($n = 2$) (3) объектов G_1 и G_2 означает операцию $G_3 = \Phi_{\text{¡}}(G_1, G_2)$ с определением:

$$f_3 = \psi(f_1(x, y, z), f_2(x, y, z)) \geq 0, \tag{4}$$

где ψ — непрерывная вещественная функция двух переменных.

Предложенный способ описания объектов трёхмерных сцен базовыми поверхностями и функциями возмущения имеет компактное описание, что позволяет уменьшить от десяти

Рис. 1. Слева представлена композиция полупрозрачной функционально заданной поверхности и трёхмерной текстуры, справа — визуализация комбинированной анатомической $3D$-модели (костная структура — объёмные данные, имплантат — функционально заданные поверхности)

до ста раз и выше объём данных в зависимости от конкретных трёхмерных сцен и моделей. Кроме того, при решении описывающей функции в виде неравенства $F(x, y, z) \geq 0$ можно визуализировать не только поверхность, но и внутреннюю структуру объекта (рис. 1).

Трёхмерная текстура. Текстура восьмеричного дерева используется для эффективного хранения информации о цвете без текстурных координат [10]. В этой методике цвет сохраняется в областях, где поверхность пересекает объём. Благодаря этому отсутствуют искажения в изображении и снижаются требования к памяти. Кроме того, текстура восьмеричного дерева оптимально адаптирована для её использования на графических процессорах. Восьмеричное дерево имеет иерархическую структуру данных, хранится в памяти текстур и доступно из программы фрагментов. Корень дерева представляет собой куб. Каждый узел имеет восемь дочерних элементов, либо их нет, если они пустые. Восемь дочерних элементов образуют регулярное подразделение родительского узла размером $2 \times 2 \times 2$. Узел с дочерними элементами является внутренним узлом, без дочерних элементов — называется листом. Восьмеричное дерево окружает $3D$-модель, узлы, внутри которых находится поверхность, уточняются, а пустые узлы отмечаются в виде листьев.

Для реализации восьмеричного дерева на графическом процессоре определён способ хранения структуры в памяти текстур и получения доступа к структуре из фрагментарной программы. Для этого используются указатели (индексы внутри текстуры) соединения узлов дерева. Каждый внутренний узел содержит массив указателей на его дочерние узлы. Дочерним элементом может быть другой внутренний узел или лист. Лист содержит только поле данных. Индексы кодируются значениями RGB, а содержимое листьев хранится непосредственно в виде значений RGB в массиве указателей родительского узла. В методе применяется алгоритм поиска текстур. Значения данных и индексы дочерних элементов хранятся в виде троек RGB. Альфа-канал используется в качестве флага для определения содержимого ячейки ($\alpha = 1$ указывает на данные, $\alpha = 0{,}5$ — на индекс, $\alpha = 0$ — на пустую ячейку). После сохранения дерева в памяти текстур необходимо получить к нему доступ из фрагментарной программы. Как и в случае со стандартной $3D$-текстурой дерево определяет текстуру внутри единичного куба. Поиск по дереву начинается с корня, последовательно посещаются узлы, содержащие точку, в которой хранятся значения дерева, пока не будет достигнут лист. Поиск заканчивается при достижении листа. На рис. 2 показаны трёхмерная текстура восьмеричного дерева (средний объект композиции слева) и объёмное облако (справа).

Визуализация композиции объёмов и функционально заданных поверхностей. Процесс растрирования разбивается на два этапа и распределяется между центральным процессором (ЦПУ) и графическим. Центральный процессор выполняет деление объектного пространства по четверичному дереву до клетки размером 8×8. Преимущество

Рис. 2. Слева представлено объединение функционально заданных поверхностей и трёхмерной текстуры (текстуры восьмеричного дерева — средний объект композиции), справа — функционально заданный объект в объёмном облаке

такого подхода заключается в том, что можно отбросить на ранней стадии большие части куба, в которых нет заданного объекта.

Поиск точек функционально заданной поверхности и её внутренней части (при необходимости) производится в пространстве внутри куба от -1 до $+1$ по каждой координате так, что центр куба соответствует началу координат. На первом шаге рекурсии куб разбивается на четыре бруска, спроецированных на экранную плоскость [11]. Для каждого нового бруска выполняется тест на пересечение с функционально заданной поверхностью. Если пересечение имеет место, то брусок подвергается следующему уровню рекурсии. Бруски, не пересекающиеся с поверхностью, дальнейшему погружению в рекурсию не подлежат, что соответствует исключению из рассмотрения квадратных областей экрана, на которые данные бруски (следовательно, и поверхность объекта) не отображаются. На шаге деления доходим до брусков, отображаемых в клетку размером 8×8 пикселей изображения. Примитивами промежуточного описания являются фрагменты пересечения геометрических объектов с клетками. Фрагментами называются части поверхностей (1) и объёмов, принадлежащих данной клетке.

Далее все оставшиеся вычисления выполняются на графическом процессоре. Используется большое количество вычислительных процессоров, одновременно будет происходить проверка сразу нескольких лучей. А в большей части графических процессоров, поддерживающих CUDA, находится не менее 128 скалярных вычислительных ядер.

Второй этап вычислений — обработка списка объектов, определение видимости и цвета пикселей. Наблюдатель смотрит вдоль оси Z, необходимо получить проекцию сцены на плоскость XY. Проекция должна представлять собой конечный набор значений. Через плоскость XY куба проходят лучи, и каждому лучу соответствует пиксель на изображении. Лучи ограничены передней и задней гранями куба. В процессе поиска точки пересечения луча и поверхности каждый луч делится вдоль оси Z, образуя набор вокселей. Таким образом, получим функцию плотности вдоль луча, которая зависит от одной переменной. Задача будет состоять в нахождении первой точки, в которой функция обращается в ноль. Найдя такую точку для каждого луча, можно вычислить координату Z. Далее в каждом пикселе вычисляется нормаль. Имея все координаты и нормали в каждом пикселе, можно использовать модель локального освещения. В итоге получится изображение гладкого объекта с учётом освещения. На рис. 3 и 4 показаны алгоритмы бинарного деления луча и квадрик. Таким образом, алгоритм обходит объём по четверичному дереву (в экранном пространстве — функция ЦПУ), листьями которого являются двоичные под-

Рис. 3. Алгоритм бинарного деления квадрики

деревья (соответствуют лучам, направленным вглубь экрана, — функция графического процессора).

После нахождения пересечения с оболочкой сканируется её внутренняя часть, заполненная $3D$-текстурой. Объёмные данные хранятся в виде трёхмерной текстуры [12]. На рис. 5 и 6 показаны объединения $3D$-текстур и функционально заданных поверхностей.

Результаты работы. Для реализации была использована Compute Unified Device Architecture (CUDA) от NVIDIA. CUDA — это модель параллельного программирования вместе с набором программных средств, которая позволяет реализовывать программы на языке C для исполнения на графическом процессоре. Сама архитектура основана на потоковых мультипроцессорах (Streaming Multiprocessors).

Вычисление занимает несколько десятков миллисекунд на процессорах Intel Core2 CPU E8400 3.0 GHz и GPU 470 GTX в зависимости от сложности сцен. При реализации учитывалось влияние скорости работы с памятью. Максимально применяются регистры, потому что это самый быстрый вид доступной памяти. При работе с трёхмерной текстурой были использованы компромиссы между производительностью, эффективностью хранения и качеством рендеринга. Предлагаемый метод проиллюстрирован несколькими тестовыми примерами. На рис. 1 показана полупрозрачная функционально заданная поверхность, заполненная трёхмерной текстурой, на рис. 2 — объединение двух функционально заданных поверхностей и объёмной текстуры восьмеричного дерева. Объёмная текстура дана в средней части композиции. На рис. 5 изображено объединение функционально заданных поверхностей и полупрозрачного скалярного объёма с разрешением 256^3. Для описания этих функциональных поверхностей потребовалось бы порядка 200 тыс. треугольников

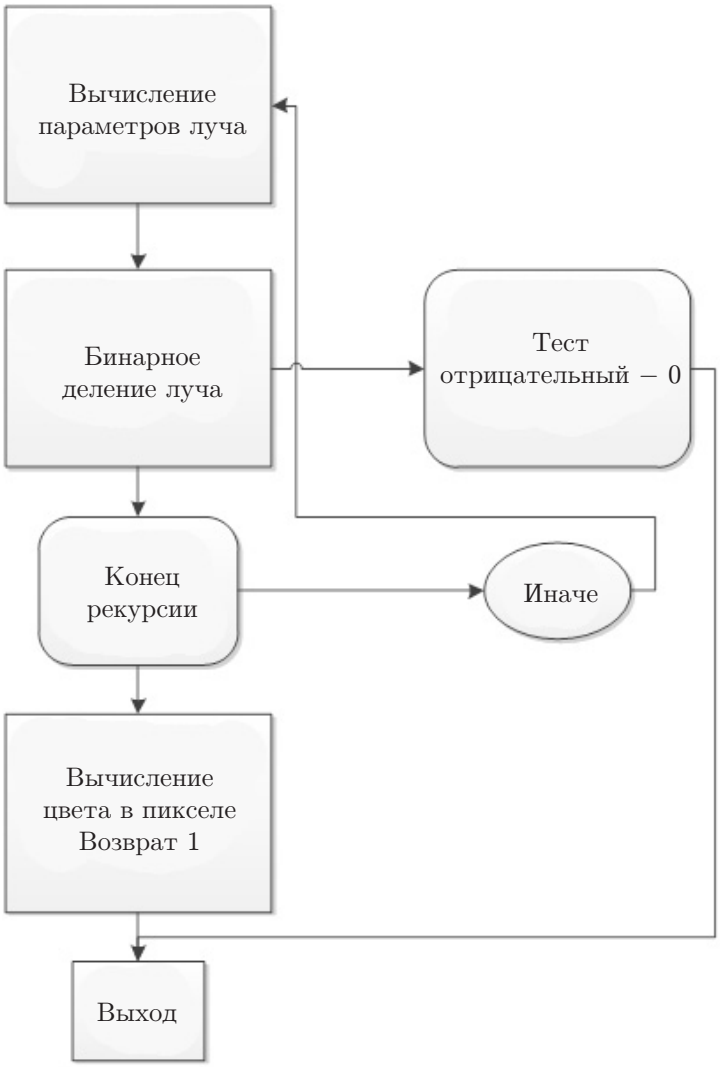

Рис. 4. Алгоритм бинарного деления луча

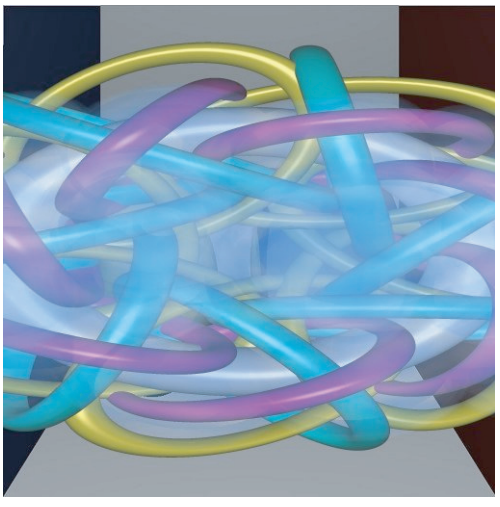

Рис. 5. Объединение полупрозрачного объёма и функционально заданных поверхностей

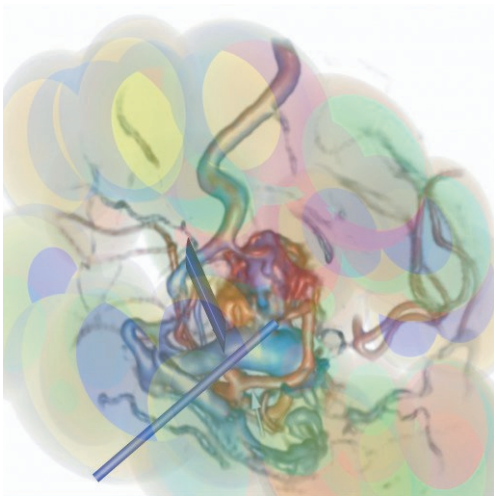

Рис. 6. Композиция полупрозрачной трёхмерной текстуры и функционально заданных поверхностей

в полигональном задании. На рис. 6 показаны кровеносные сосуды головного мозга, два хирургических инструмента и полупрозрачный объём МРТ (разрешение 256^3), визуализированные с помощью предлагаемого метода, который имеет ряд преимуществ в сравнении с известными методами. С помощью данного метода можно обрабатывать не только функционально заданные поверхности, но и полупрозрачные объёмы. Благодаря этому можно визуализировать сложные модели. Использование функционально заданных поверхностей позволяет сократить базу данных сцен и уменьшить время рендеринга. Метод прост в реализации и не имеет проблем с надёжностью или вырождением поверхностей. Предварительное применение метода для разных тестовых образцов показывает хорошее приближение с ограниченной ошибкой на границах пересечений объёмов и функционально заданных поверхностей.

Заключение. Разработан новый подход к визуализации сложных объектов, представляющих композицию объёмов и функционально заданные поверхности с применением графических процессоров. Метод, включающий стадию растеризации, реализован на CUDA, что позволяет полностью контролировать иерархию памяти, в частности, доступ к высокой пропускной способности и низкому времени ожидания распределённой памяти. Обеспечивается интерактивная частота кадров при отображении объёмов и поверхностей. Вычисление занимает несколько десятков миллисекунд на процессорах Intel Core2 CPU E8400 3.0 GHz и GPU 470 GTX.

Метод актуален для широкого спектра применения объёмного рендеринга сложных сцен, включая композиции объёмов и поверхностей для медицинских приложений, цифрового изготовления моделей и т. д. Как показано в работе, метод является простым в реализации и позволяет использовать его для многих задач.

Финансирование. Работа выполнена при поддержке Министерства науки и высшего образования РФ в рамках выполнения работ по Государственному заданию № 1023032300221-9-1.2.1 в ИАиЭ СО РАН.

СПИСОК ЛИТЕРАТУРЫ

1. **Kniss J., Premoze S., Hansen C. et al.** A model for volume lighting and modeling // IEEE Trans. Visualization and Comp. Graph. 2003. **9**, Iss. 2. P. 150–162. DOI: 10.1109/TVCG.2003.1196003.

2. **Solteszova V., Patel D., Bruckner S., Viola I.** A multidirectional occlusion shading model for direct volume rendering // Comput. Graph. Forum. 2010. **29**, Iss. 3. P. 883–891. DOI: 10.1111/j.1467-8659.2009.01695.x.
3. **Loos B. J., Sloan P. P.** Volumetric obscurance // Proc. of the ACM SIGGRAPH Symp. Interactive 3D Graphics and Games. Washington, USA, 19–21 Feb., 2010. P. 151–156. DOI: 10.1145/1730804.1730829.
4. **Wenger A., Keefe D. F., Zhang S., Laidlaw D. H.** Interactive volume rendering of thin thread structures within multivalued scientific data sets // IEEE Trans. Visualization and Comp. Graph. 2004. **10**, Iss. 6. P. 664–672. DOI: 10.1109/TVCG.2004.46.
5. **Kruger J., Westermann R.** Acceleration Techniques for GPU-based Volume Rendering // Proc. of the IEEE Visualization (VIS 2003). Seattle, USA, 19–24 Oct., 2003. DOI: 10.1109/VISUAL.2003.1250384.
6. **Stegmaier S., Strengert M., Klein T., Ertl T.** A Simple and Flexible Volume Rendering Framework for Graphics-Hardware-based Ray Casting // Proc. of the Fourth Int. Workshop on Volume Graphics. N.-Y., USA, 20–21 June, 2005. DOI: 10.1109/VG.2005.194114.
7. **Panozzo D., Diamanti O., Paris S. et al.** Texture mapping real-world objects with hydrographics // Comput. Graph. Forum. 2015. **34**, Iss. 5. P. 65–75. DOI: 10.1111/cgf.12697.
8. **Schüller C., Panozzo D., Grundhofer A. et al.** Computational thermoforming // ACM Trans. Graph. 2016. **35**, Iss. 4. DOI: 10.1145/2897824.2925914.
9. **Вяткин С. И., Долговесов Б. С.** Физически корректная визуализация функционально заданных объектов // Автометрия. 2022. **58**, № 3. С. 98–105. DOI: 10.15372/AUT20220311.
10. **Benson D., Davis J.** Octree Textures // Proc. of the 29th Annual Conf. Computer Graphics and Interactive Techniques (SIGGRAPH 2002). San Antonio, USA, 23–26 July, 2002. DOI: 10.1145/566570.566652.
11. **Вяткин С. И.** Метод рекурсивного поиска элементов изображения функционально заданных поверхностей // Автометрия. 2017. **53**, № 3. С. 53–57. DOI: 10.15372/AUT20170307.
12. **Вяткин С. И., Долговесов Б. С.** Метод визуализации мультиобъёмных данных и функционально заданных поверхностей с применением графических процессоров // Автометрия. 2021. **57**, № 2. С. 32–40. DOI: 10.15372/AUT20210204.

Поступила в редакцию 16.01.2024
После доработки 13.02.2024
Принята к публикации 13.02.2024