УДК 681.3:771.537.442

ПРИМЕНЕНИЕ АЛГОРИТМА ЧАСТИЧНОЙ СОРТИРОВКИ ПО ГЛУБИНЕ К ВИЗУАЛИЗАЦИИ ФУНКЦИЙ ТРЁХ ПЕРЕМЕННЫХ

Гладкий Б.М., Толок А.В.

При построении изображений трёхмерных графических объектов неизбежно возникает проблема отсечения невидимых наблюдателю частей объекта. Известные алгоритмы отсечения работают в основном с объектами, представленными в виде списка ориентированных граней или объединения многогранников, и отсекают невидимые линии или поверхности. Зачастую эти алгоритмы довольно сложны (алгоритм Робертса [1]) и не обладают достаточной универсальностью (алгоритм плавающего горизонта [1]).

Изложенный в данной статье простой алгоритм, названный частичной сортировкой по глубине, изящно решает задачу построения корректного изображения трёхмерного объекта любой сложности.

В основу алгоритма положена идея рекурсивного разбиения выпуклой трёхмерной области на выпуклые подобласти и отображения каждой из них отдельно в порядке приближения к наблюдателю. При этом конкретный вид и способ представления объекта не имеет особого значения. Данный алгоритм может быть применён для построения изображений произвольных трёхмерных объектов, допускающих разбиение на части и отображение отдельной части независимо от остальных.

В данной статье рассматривается конкретная реализация алгоритма для визуализации функций двух и трёх переменных, а также для построения изображений трёхмерных тел, описанных с помощью R-функций.

Строго говоря, частичная сортировка по глубине — это довольно простой принцип рекурсивного разбиения на непересекающиеся области некоторого участка пространства при помощи сечений плоскостями с целью установления между подобластями отношения некоторого предпорядка. Это отношение вводится из интуитивного понятия степени близости частей к наблюдателю при попарном сравнении. Интуитивное представление подкрепляется строгим, математически обоснованным, и в то же время очевидным критерием сортировки.

Внутри этого исследуемого участка пространства могут находиться любые объекты. Далеко не всякий объект допускает разбиение на части произвольным образом и применение к каждой части рекурсивной процедуры разбиения. Как правило, ограничения на детализацию связаны со способом представления объекта. В классической машинной графике большинство алгоритмов отсечения работает с объектами, представленными в виде списка ориентированных граней, объединения выпуклых многогранников и т. п. Алгоритм частичной сортировки по глубине абстрагируется от объекта и исследует участок пространства. По этой причине алгоритм не может быть эффективно применён для визуализации классических объектов машинной графики. Напротив, для построения изображений объектов специального вида, каковыми являются, например, рассматриваемые в данной статье функции трёх переменных и графические объекты, описываемые ими, алгоритм частичной сортировки по глубине подходит как нельзя лучше.

Алгоритм частичной сортировки по глубине принадлежит к разряду алгоритмов, использующих список приоритетов. Алгоритм такого класса называют ещё алгоритмом художника. Процесс построения изображения происходит аналогично тому, как художник рисует картину, переходя от дальних планов к ближним. В результате получается корректное непротиворечивое изображение. В [1] различные алгоритмы отсечения классифицируются по типу системы координат, в которой они работают. Так, алгоритм Робертса работает в пространстве объекта, а, например, алгоритм Варнока — в пространстве изображения. Используя эту терминологию, можно сказать, что алгоритм частичной сортировки по глубине работает в пространстве наблюдателя.

Пусть пространство разделено на две части некоторой плоскостью, и эта плоскость не проходит через точку, в которой находится наблюдатель. Обозначим полупространства, полученные в результате данного сечения, символами H_1 и H_2. Обозначим также символом ν_{21} нормальный вектор плоскости сечения, направленный из полупространства H_2 в полупространство H_1. Возьмём на секущей плоскости произвольную точку и проведём в эту точку вектор, исходящий из точки наблюдателя, обозначив его символом S. Введём для полупространств отношение предшествования по следующему критерию: если косинус угла между векторами S и ν_{21} положителен (неотрицателен), будем считать, что полупространство H_1 предшествует H_2, в противном случае полупространство H_2 будем считать предшествующим H_1.

Пусть теперь пространство разбито на секторы произвольным образом последовательностью сечений, в результате чего получено конечное число многогранных выпуклых подобластей. Для любых двух из них, имеющих общую грань, можно ввести отношение предшествования аналогично тому, как это было сделано для полупространств. В совокупности все попарные сравнения подобластей образуют некоторый предпорядок, на основании которого можно вполне упорядочить полученные секторы пространства (не обязательно единственным образом). В том, что это действительно предпорядок, легко убедиться, проверив выполнение всех аксиом предпорядка. Если изобразить все отношения предшествования подобластей в виде ориентированного графа, то в этом графе не будет замыканий и петель.

Если теперь отображать части пространства именно в этом порядке, то можно быть уверенным, что никакая часть объекта не будет отображена до того, как будут отображены все части, лежащие за ней относительно текущего положения наблюдателя, то есть вновь нарисованные части не будут перекрывать ранее нарисованные, если только они не лежат ближе к наблюдателю.

Следующий шаг — это идея рекурсивного применения процедуры разбиения к полученным на предыдущем этапе подобластям. Регулируя глубину рекурсии, можно строить изображение исследуемого участка пространства с той или иной степенью детализации. Замечательным в этой связи является то обстоятельство, что суммарное количество операций сравнения при сортировке подобластей на всех уровнях рекурсии линейно зависит от количества частей, на которые разбивается участок пространства. Более того, эта оценка быстродействия алгоритма не зависит от конкретного алгоритма сортировки, используемого для упорядочения подобластей на каждом уровне рекурсии. Даже при использовании самого медленного пузырькового алгоритма трудоёмкость сортировки будет эквивалентна простому линейному перечислению всех элементов, на которые разбит участок пространства.

Далее возникает вопрос о выборе оптимального вида исследуемой области. Принимая во внимание рекурсивную природу алгоритма, становится понятно, что части, на которые разбивается исследуемый участок пространства, должны быть подобными исходной области. Отсюда следует, что в качестве исходной области следует использовать параллелепипед (в общем случае косоугольный), и разбивать его на 8 подобных областей. При таком способе разбиения рост количества исследуемых многогранников с увеличением глубины рекурсии будет минимальным.

Итак, фиксируем некоторый параллелепипед в пространстве. Производим последовательно три сечения плоскостями, параллельными граням и проходящими через центр параллелепипеда. Получаем 8 параллелепипедов следующего уровня, подобных исходному. Каждое сечение исходного параллелепипеда даёт возможность ввести отношение предшествования для четырёх пар смежных подобластей, имеющих общую грань, лежащую в плоскости сечения. Все такие отношения, полученные для трёх сечений, в совокупности образуют предпорядок, на основании которого можно отсортировать эти 8 параллелепипедов в порядке приближения к наблюдателю. Всего существует 8! всевозможных перестановок из восьми элементов, однако, оказывается, что имеется $2^3=8$ вариантов упорядочения восьми параллелепипедов, которые полностью исчерпывают все возможные случаи положения объекта относительно наблюдателя. Это значительно облегчает задачу, ведь теперь вместо того, чтобы честно сортировать параллелепипеды на каждом уровне рекурсии, используя матрицу отношения предпорядка, достаточно вычислить двоичный код, определяющий тип положения объекта относительно наблюдателя, и в соответствии с этим кодом выбрать один из восьми вариантов упорядочения. Двоичный код (целое число от 0 до 7) формируется следующим образом: каждая из трёх секущих плоскостей представляется одним двоичным разрядом, принимающим значение 0 для прямого порядка и 1 для инверсного. Для трёх плоскостей существует $2^3=8$ различных значений этого числа.

Данную процедуру рекурсивно применяем к каждому из восьми полученных на предыдущем этапе параллелепипедов в том порядке, в каком они были отсортированы. Если достигнута требуемая глубина рекурсии, отображаем текущий параллелепипед с атрибутами, определяемыми конкретной прикладной задачей.

Алгоритм столь же прост, сколь и изящен. Суммарное количество операций сравнения при сортировке параллелепипедов на всех уровнях рекурсии линейно зависит от количества исследуемых параллелепипедов, причём вне зависимости от используемого алгоритма сортировки. Действительно, процедура сортировки восьми параллелепипедов на очередном шаге рекурсии выполняется в общей сложности $(1 + 8 + 8^2 + \ldots + 8^{k-1}) = (8^k-1) / 7$ раз, где k – глубина рекурсии.

Даже при самом медленном пузырьковом методе сортировки количество операций сравнения при упорядочении восьми параллелепипедов постоянно и не зависит от k.

Если же задаться целью совместно упорядочить все 8^k исследуемых параллелепипедов, количество операций сравнения, даже если применить метод быстрой сортировки, будет равно $C \cdot 8^k \cdot k$, не говоря уже о том, что никакой оперативной памяти не хватит для хранения координат центров 8^k

параллелепипедов уже при k = 10. При использовании же частичной сортировки одновременно в стеке находится лишь 8 • k параллелепипедов, что вполне приемлемо.

Рассмотрим метод визуализации значений функции трёх переменных внутри некоторого трёхмерного параллелепипеда. Метод основан на алгоритме частичной сортировки по глубине и моделировании эффекта прозрачности.

Пусть (x_1, x_2, x_3) — координаты неподвижной декартовой системы, жёстко связанной с наблюдателем, (ξ_1, ξ_2, ξ_3) — система координат, в которых задана функция $w = w(\xi_1, \xi_2, \xi_3)$.

Определено преобразование

$$\begin{cases} \xi_1 = \xi_1(x_1, x_2, x_3) \\ \xi_2 = \xi_2(x_1, x_2, x_3) \\ \xi_3 = \xi_3(x_1, x_2, x_3) \end{cases} \quad (1)$$

В простейшем случае преобразование координат (1) сводится лишь к ортогональным преобразованиям движения, то есть повороту и смещению. Могут применяться также общие линейные преобразования: сжатие-растяжение по осям, зеркальный поворот и косоугольные координаты. В общем случае допускается использование криволинейных систем координат.

Будем строить изображение аксонометрической проекции графика сложной функции $w = w(x_1, x_2, x_3)$ на трёхмерное пространство (x_1, x_2, x_3) её аргументов. Для этого воспользуемся алгоритмом частичной сортировки по глубине.

Формальное описание алгоритма:

Если требуемая глубина рекурсии достигнута, изобразить исследуемый параллелепипед с цветовыми атрибутами, вычисленными исходя из значения функции и её градиента в центре параллелепипеда.

Иначе:

Разбить исходный параллелепипед на 8 подобных областей и отсортировать их в порядке приближения к наблюдателю. В указанном порядке применить к каждому из восьми полученных параллелепипедов рекурсивную процедуру исследования параллелепипеда.

Для построения изображения трёхмерного объекта в любой используемой модели освещения необходимо вычисление нормали к поверхности графика функции или к поверхности уровня. Градиент сложной функции $w = w(x_1, x_2, x_3)$ по переменным (x_1, x_2, x_3) в пространстве наблюдателя вычисляется по формуле

$$w'_x = w'_{\xi(x)} \cdot \left(\frac{\partial \xi}{\partial x} \right) \quad (2)$$

В существующей реализации алгоритма значение функции, вычисленное в центре параллелепипеда на самом глубоком уровне рекурсии, преобразуется в цвет по специальному закону. При этом используется весь спектр цветов от самых холодных (чёрный-фиолетовый-синий), соответствующих отрицательным значениям, до самых горячих (жёлтый-красный-белый), соответствующих положительным значениям. Если изображать только положительные или только отрицательные области функций, будет задействована лишь половина спектра (нулевой границе соответствует голубовато-зелёный цвет).

Для создания иллюзии объёма используется простая модель освещения с диффузным отражением и пропусканием. Моделируется одиночный бесконечно удалённый источник белого света. Данная модель подходит для исследования поведения функции внутри рассматриваемой области, но она непригодна для построения реалистических изображений. Изменяя направление источника света, можно выявлять локальные экстремумы или участки наибольшего возрастания и убывания функции.

Следует помнить, что алгоритм строит изображение графика функции в четырёхмерном пространстве, поэтому и направление источника света представляется четырёхмерным вектором.

После того, как в исследуемой точке вычислена интенсивность цвета, отвечающая значению функции и её производной, она преобразуется с учётом «прозрачности» малого элемента пространства в данной точке. Результирующая интенсивность каждого пикселя рассчитывается непосредственно в пространстве изображения по простому закону:

$$I_{result} = I_{background} \cdot p + I_{foreground} \cdot (1-p) \quad (3)$$

Здесь $I_{background}$ — интенсивность цвета фона, $I_{foreground}$ — вычисленное значение интенсивности в точке, p — коэффициент прозрачности.

Коэффициент прозрачности вычисляется по эмпирической формуле:

$$p = p_0^T \tag{4}$$

где p_0 — так называемый удельный коэффициент прозрачности, определяющий степень ослабления света на отрезке единичной длины, T — показатель толщины малого элемента пространства, для которого рассчитывается коэффициент. Параметр T определяется формулой

$$T=a/2^n \tag{5}$$

где n — глубина рекурсии, a — некоторая постоянная.

В свою очередь, удельный коэффициент прозрачности рассчитывается по эмпирической формуле

$$p_0 = 1 - e^{-\lambda \cdot \|\text{grad } w\|} \tag{6}$$

где λ — некоторый произвольный неотрицательный параметр, зависящий от свойств функции. При этом вблизи точек локального экстремума функции (в которых её градиент равен нулю) прозрачность будет минимальной, чем больше модуль производной функции (участки наибольшего возрастания и убывания), тем прозрачность выше.

Изменяя значения параметра λ, можно регулировать «глубину» просвечивания исследуемого участка пространства. Например, если в окрестности точки экстремума функция достаточно медленно возрастает или убывает, можно увеличить параметр λ, чтобы увидеть, как ведёт себя функция в непосредственной близости от этой точки. Обычно параметр λ приходится подбирать методом проб и ошибок. На практике установлено, что, например, при глубине рекурсии 7 он лежит в пределах 10^{-7} – 10^{-3} в зависимости от конкретной функции. Пример изображения представлен на рисунке 1:

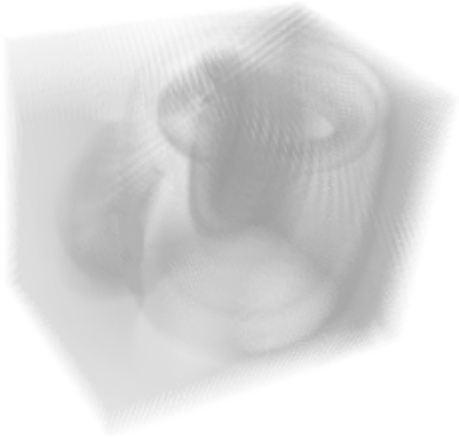


Рис. 1. Изображение функции трёх переменных при глубине рекурсии 7.

Как частный случай применения алгоритма частичной сортировки по глубине может быть рассмотрен изящный алгоритм построения изображения поверхности графика функции двух переменных в трёхмерном пространстве.

Рассмотрим функцию $z=f(x,y)$, заданную на некотором двумерном промежутке (прямоугольнике). Пусть (x,y,z) — координаты пространства объекта.

Поверхность графика функции может быть ориентирована произвольным образом относительно точки наблюдателя, при этом прямоугольник параметров отображается в прямоугольный участок плоскости, в которую переходит плоскость XY при переходе от координат объекта к координатам наблюдателя.

В качестве исходной области пространства, к которой мы будем применять рекурсивную процедуру алгоритма частичной сортировки по глубине, возьмём бесконечный параллелепипед, сечением которого плоскостью XY является прямоугольник области определения функции, а по оси Z он неограничен. Разобьём исходный прямоугольник на 4 подобных ему прямоугольника. Это разбиение индуцирует разбиение нашего бесконечного параллелепипеда на 4 подобные области двумя взаимно перпендикулярными плоскостями, проходящими через его бесконечную ось. Каждая из этих плоскостей устанавливает отношение предшествования для двух пар смежных подобластей. В совокупности эти отношения образуют предпорядок. Отсортировав 4 бесконечных параллелепипеда в соответствии с установленным предпорядком, в этом же порядке рекурсивно применим к ним аналогичную процедуру разбиения. Так поступаем до тех пор, пока не будет достигнута требуемая глубина рекурсии.

Если же необходимая глубина рекурсивного разбиения достигнута, делим очередной прямоугольник на 4 треугольника, как показано на рисунке 2.

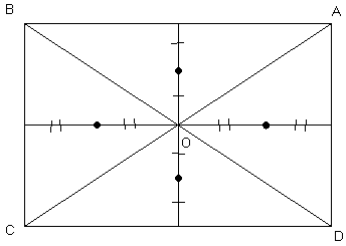


Рис. 2. Схема разбиения элементарного прямоугольника на нижнем уровне рекурсии.

Это будут треугольники OAB, OBC, OCD и ODA. Вычислив значения функции в точках O,A,B,C,D и произведя для каждой из точек преобразование вектора ***(x,y,f(x,y))*** из пространства объекта в пространство наблюдателя, получим четыре треугольные грани многогранной поверхности, аппроксимирующей поверхность графика исследуемой функции, причём вершины граней являются общими точками для обеих поверхностей. Чтобы корректно отобразить эти четыре микрограни, к ним также необходимо применить сортировку по глубине. Секущими в данном случае будут две плоскости, проходящие через прямые AC и BD, а также через бесконечную ось параллелепипеда, сечением которого плоскостью ***XY*** является прямоугольник ABCD.

В случае двух секущих плоскостей всего существует $2^2=4$ возможных положения объекта относительно наблюдателя. Это значит, что достаточно иметь четыре варианта упорядочивания подобластей, получаемых из исходной области на очередном уровне рекурсии, чтобы исчерпать все возможные случаи.

Нормаль к микрограни вычисляется как векторное произведение векторов, коллинеарных рёбрам треугольной грани и исходящих из общей вершины четырёх прилегающих друг к другу граней.

Например, если векторы OA и OB из касательного пространства объекта отобразить на касательное пространство наблюдателя и там вычислить их векторное произведение, получим нормаль к микрограни OAB в пространстве наблюдателя.

Значения функции, вычисленные в точках, отмеченных на рисунке 2 внутри треугольников, можно использовать для генерации соответствующего этим значениям цвета из некоторой палитры. Можно также изобразить поверхность в монохромном варианте.

Из сказанного видно, что модификация алгоритма частичной сортировки по глубине в применении к построению изображения поверхности графика функции двух переменных имеет более высокий показатель быстродействия, чем в случае функции трёх переменных. Если в оригинальном варианте алгоритма на каждом следующем уровне рекурсии количество исследуемых многогранников увеличивалось в 8 раз, то теперь лишь в 4. Фактически, в данном случае задача из трёхмерной превращается в двумерную. Это даёт возможность увеличить глубину рекурсии и довести видимые размеры отображаемых малых элементов поверхности до сравнимых с размерами пикселя. Пример изображения представлен на рисунке 3:

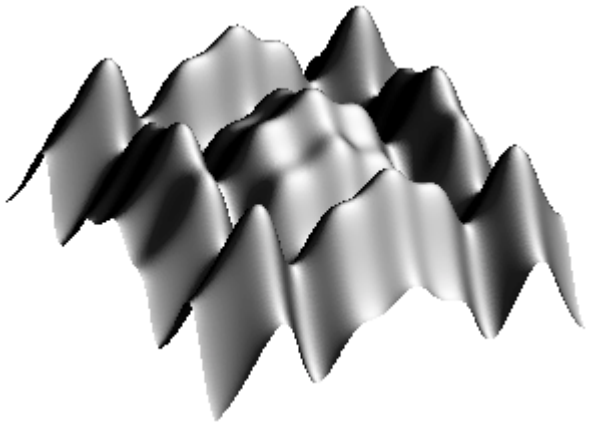


Рис. 3. Изображение поверхности графика функции двух переменных при глубине рекурсии 9.

В заключение рассмотрим оригинальный алгоритм построения изображений трёхмерных тел, описываемых R-функциями. Подробное изложение авторской концепции R-функций можно найти в работе [2].

Принимается следующее допущение: область положительных значений R-функции считать сплошным однородным телом с одинаковыми оптическими характеристиками по всей толщине. Очевидно, что для отображения сплошного тела достаточно отобразить его границу. При этом задача из трёхмерной фактически превращается в двумерную, что существенно повышает быстродействие и позволяет на порядок увеличить глубину рекурсии. Если в описанном выше алгоритме визуализации функции трёх переменных количество исследуемых точек при увеличении глубины рекурсии растёт со скоростью геометрической прогрессии со знаменателем 8, то в данном случае — лишь со знаменателем 4.

На первом этапе алгоритма строится покрытие поверхности нуля R-функции кубическими элементами. Для этого используется дерево кубических кластеров. Каждый кластер покрывает участок границы области, в совокупности кластеры дерева образуют покрытие границы объекта параллелепипедами. Порядок обхода дерева при отображении определяется динамически текущим положением наблюдателя и объекта, кластеры сортируются в порядке приближения к наблюдателю (в соответствии с описанным выше алгоритмом частичной сортировки по глубине). При этом сохраняется первоначальная кубическая рекурсивная схема исследования части пространства, ограниченной исходным параллелепипедом.

Рекурсивная процедура построения дерева кубических кластеров:

Если достигнута требуемая глубина рекурсии, вычислить значения функции в углах параллелепипеда. **Если** среди этих значений встречаются как положительные, так и отрицательные, или есть хотя бы одно нулевое значение, добавить текущий параллелепипед к дереву кластеров в соответствии с его порядковым номером. **Иначе** пропустить его.

Иначе:

Разбить исходный параллелепипед на 8 подобных областей и присвоить каждой из них порядковый номер (от 1 до 8). Применив к каждому из восьми полученных параллелепипедов процедуру построения дерева, выяснить, есть ли среди них непустые кластеры (то есть такие, которые содержат в себе также непустые дочерние кластеры). **Если** таковые есть, сделать их дочерними текущего кластера и добавить текущий кластер к дереву в соответствии с его порядковым номером. **Иначе** не добавлять текущий кластер к дереву.

В корневом узле дерева находится наибольший параллелепипед, охватывающий весь объект.

Процедура обхода дерева:

Если текущий кластер не имеет дочерних кластеров, отобразить ограничиваемый им малый участок поверхности (он может быть аппроксимирован многоугольником, проходящим через центр кубического элемента) с цветовыми атрибутами, вычисленными в рамках используемой модели освещения.

Иначе:

Отсортировать дочерние кластеры в порядке приближения к наблюдателю и отобразить их в этом порядке.

Из сказанного видно, что принцип частичной сортировки по глубине прекрасно работает и в случае кластерной модели поверхности.

Введение в рассмотрение дерева кубических кластеров сразу же на порядок понижает размерность задачи и значительно сокращает время отображения объекта. Если при изображении сплошного тела количество исследуемых параллелепипедов растёт примерно как 8^k, то при отображении кластерного покрытия поверхности — как 4^k, где k – глубина рекурсии.

Теперь работа с объектом организуется следующим образом:

Фиксируется положение главного кластера сцены относительно исследуемого тела.

Строится дерево кластеров.

После того как дерево построено (это может быть весьма длительный процесс), его можно сохранить в файле, из которого оно может быть восстановлено в любое время.

Запись и считывание дерева из файла происходит очень быстро, поскольку эти действия не связаны с какими бы то ни было вычислениями. Дерево хранит лишь структурную информацию — порядковый номер кластера и ссылки на дочерние кластеры. Вся информация, необходимая при отображении дерева, хранится в объекте, который владеет деревом кластеров.

Как уже упоминалось, каждый элемент дерева кубических кластеров содержит минимум информации: свой порядковый номер в группе из восьми кластеров, указатель на дочерние кластеры и указатель на следующий кластер в цепочке, связывающей параллелепипеды одного уровня рекурсии. В сумме это занимает 9 байт. При записи же дерева на диск объём используемой памяти сокращается втрое — каждый кластер на диске занимает всего 3 байта. Таким образом, структурная информация о покрытии поверхности хранится в максимально компактном виде без ущерба для быстродействия, причём без использования файлов прямого доступа.

Одно и то же дерево кластеров может быть использовано для построения изображений исследуемого тела при любом его расположении относительно наблюдателя. Однако при этом должно сохраняться то положение главного кластера сцены относительно объекта, которое было зафиксировано при построении дерева. На практике это означает, что если, например, необходимо повернуть тело в пространстве, то следует повернуть также и главный кластер сцены, причём вокруг той же оси и на тот же угол, что и сам объект.

Для моделирования отражающей поверхности реализована локальная модель освещения Кука-Торренса-Спэрроу, детально описанная в [1]. В этой наиболее реалистической и теоретически обоснованной из всех существующих модели учитываются диффузная и зеркальная компоненты отражения от множества сферических источников конечного размера различной интенсивности, находящихся на конечном расстоянии от освещаемого объекта, а также рассеянный свет. Сделана попытка расширить данную модель на случай диффузного и направленного пропускания света. Это дало возможность реализовать эффект прозрачности аналогично тому, как это описано в разделе, посвящённом визуализации функций трёх переменных. Локальный коэффициент прозрачности малого участка поверхности рассчитывается по следующей эмпирической формуле:

$$p = A \bullet \cos (\operatorname{grad} w , \hat{S})^{1/f} \qquad (7)$$

где ***grad w*** — вектор нормали к поверхности, вычисленный в пространстве наблюдателя, $\hat{S}$ — единичный вектор, направленный из точки наблюдателя к центру кубического элемента, содержащего участок границы, $f>0$ — так называемый дифференциальный фактор, $A \in [0,1]$ — максимально допустимое значение коэффициента прозрачности. Эта формула получена из следующих соображений: прозрачность малого участка поверхности максимальна (и равна A), если взгляд наблюдателя нормально падает на этот участок; с увеличением угла расхождения между вектором наблюдения и нормалью к поверхности прозрачность уменьшается и достигает минимального значения (нуля), когда взгляд наблюдателя скользит по поверхности. Примеры изображений непрозрачной и квазипрозрачной поверхности представлены на рисунках:

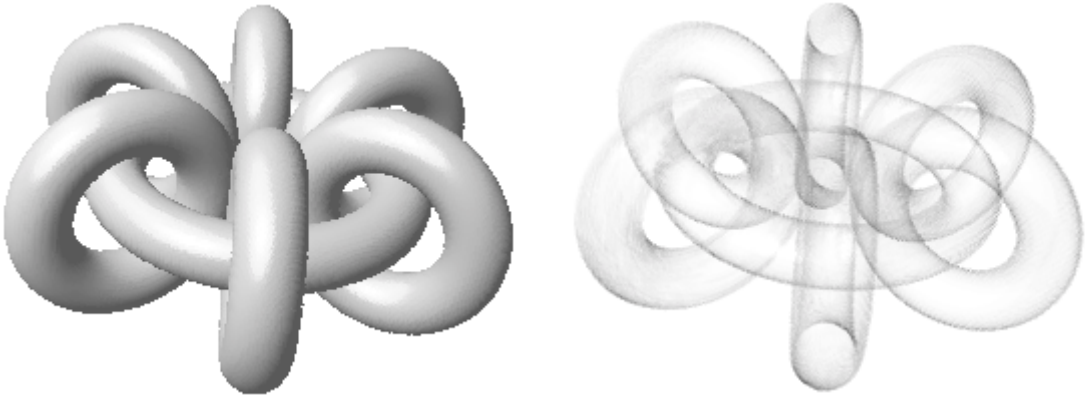


Рис. 4. Изображение поверхности нуля R-функции при глубине рекурсии 8.

ЛИТЕРАТУРА

1. Дэвид Ф. Роджерс. Алгоритмические основы машинной графики. Пер. с англ. — М.: Мир, 1980. — 476 с.

2. Рвачёв В. Л., Слесаренко А. П. Алгебра логики и интегральные преобразования в краевых задачах. - К.: Наукова думка, 1978. — 243 с.