АЛГОРИТМ ВИЗУАЛИЗАЦИИ ПОЛУПРОЗРАЧНЫХ 3D-МОДЕЛЕЙ С ПРИМЕНЕНИЕМ ПРИНЦИПОВ OIT И ОТЛОЖЕННОГО ОСВЕЩЕНИЯ

Сакун И.Ю., Красковский П.Н.

Белорусский государственный университет информатики и радиоэлектроники
г. Минск, Республика Беларусь

Хмелев А.Г. – докт. экон. наук, доц.

Для реалистичной визуализации трёхмерных моделей используются различные параметры для описания характера взаимодействия света с поверхностью объекта. Существует множество материалов, которые не только отражают падающий на них свет, но и полностью или частично пропускают свет через себя (вода, стекло, пластик и т.д.). В данной работе предлагается алгоритм визуализации трёхмерных моделей, обладающих свойством прозрачности.

Сложностью визуализации полупрозрачных моделей является то, что для получения правильного результата необходимо рисовать фрагменты объекта в строго определённом порядке. На рисунке 1 представлен пример неправильного порядка смешивания цветов, при котором некоторые дальние треугольники нарисованы поверх ближних. Простейшим решением данной проблемы является предварительная сортировка объектов или примитивов в порядке удаления от виртуальной камеры. Такой подход решает проблему рисования простых полупрозрачных объектов, но не подходит для рисования пересекающихся объектов и объектов сложной формы. Для получения правильного изображения необходимо использовать алгоритм, основанный на принципе порядконезависимой прозрачности (order-independent transparency, OIT).

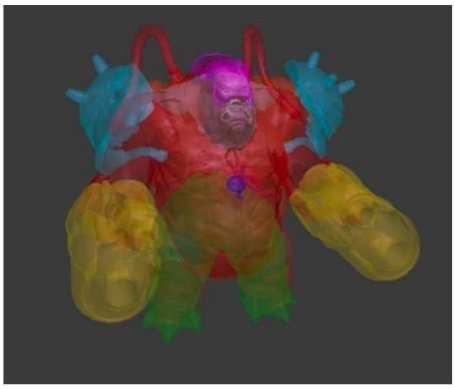

Рисунок 1 – Неправильный порядок смешивания цветов

Существует множество алгоритмов, реализующих принцип OIT. Одним из таких алгоритмов является Depth Peeling [2]. В данном алгоритме сцена рисуется по слоям. В первом слое хранятся фрагменты, расположенные ближе всего к наблюдателю, во втором слое – вторые по близости, и т.д. Затем слои смешиваются от дальних к ближним. Алгоритм прост в реализации, однако имеет ряд недостатков: многократная отрисовка сцены, необходимость хранить каждый слой отдельно.

Вторым является алгоритм, использующий трёхмерный массив для хранения всех фрагментов 3D-сцены. Для каждого пикселя изображения соответствующие фрагменты записываются в массив вдоль оси Z. Так как количество необходимых слоёв неизвестно до начала отрисовки и сложность глубины сцены обычно неравномерна, большая часть массива не используется, или возникает переполнение. В отличие от предыдущего алгоритма, данный алгоритм производит однократную отрисовку сцены для прозрачных объектов.

Для решения проблем скорости отрисовки и использования памяти предлагается следующий алгоритм:

Шаг 0. До начала отрисовки выполняется разделение треугольников на две группы: прозрачные и непрозрачные. Данный шаг выполняется один раз, и результат разделения сохраняется. При последующих отрисовках данный шаг выполнять не нужно.

Шаг 1. Выполняется заполнение буфера видимости [1] и буфера глубины для непрозрачных треугольников. Буфер видимости хранит номера видимых на итоговом изображении треугольников, а буфера глубины – глубины ближайших к наблюдателю фрагментов. На данном этапе прозрачные треугольники не рисуются.

Шаг 2. Выполняется определение сложности глубины сцены. На данном этапе происходит подсчёт количества прозрачных фрагментов для каждого пикселя изображения. В процессе выполнения данного шага используется буфер глубины, полученный на первом шаге. Если прозрачный фрагмент расположен позади непрозрачного, этот фрагмент не учитывается при подсчёте.

Шаг 3. Вычисление префиксной суммы и выделение памяти для буфера прозрачных

фрагментов. Префиксная сумма представляет собой последовательность чисел, каждый элемент которой является суммой всех предыдущих элементов входной последовательности (буфера сложности глубины сцены). Последний элемент последовательности содержит сумму всех элементов входной последовательности, что является общим количеством всех прозрачных фрагментов сцены, которые необходимо визуализировать. Далее выделяется память для буфера прозрачных фрагментов, размер которого задаётся последним элементом префиксной суммы. Буфер прозрачных фрагментов хранит номера треугольников, которым принадлежат фрагменты, а также глубины фрагментов. Таким образом, выделенная память будет использоваться максимально эффективно: устраняются проблемы неиспользуемой памяти и переполнения буфера.

Шаг 4. Выполняется заполнение буфера прозрачных фрагментов. Для этого используются два вспомогательных буфера: буфер префиксной суммы и буфер количества фрагментов. Буфер префиксной суммы, полученный на шаге 3, хранит индексы первых элементов списков фрагментов для каждого пикселя изображения в буфере фрагментов. Буфер количества фрагментов хранит текущее количество фрагментов в каждом списке фрагментов. В результате выполнения данного шага буфер количества фрагментов должен иметь такое же содержимое, что и буфер сложности глубины сцены, полученный на шаге 2.

Шаг 5. Вычисление освещения и смешивание фрагментов. Сначала выполняется сортировка фрагментов, полученных на предыдущем шаге, в порядке от ближнего к дальнему. Далее для каждого слоя производится последовательное вычисление освещения (цвета) и смешивание полученного цвета и непрозрачности фрагмента с итоговым цветом и непрозрачностью пикселя по формулам:

$$color = color + (1 - alpha) * fragmentColor \quad (1),$$
$$alpha = alpha + (1 - alpha) * fragmentAlpha \quad (2),$$

где $color$ – итоговый цвет пикселя, $alpha$ – итоговая непрозрачность пикселя, $fragmentColor$ – цвет текущего фрагмента, $fragmentAlpha$ – непрозрачность текущего фрагмента.

Стоит отметить, что формула (1) применяется для цветов с предварительно умноженной непрозрачностью. Это значит, что перед началом вычислений, каждый компонент цвета умножается на непрозрачность. Такой подход даёт правильное смешение цветов, в отличие от прямой непрозрачности, где компоненты цвета остаются неизменными.

Изначально $color$ и $alpha$ устанавливаются в ноль. $alpha$, равная нулю, означает полностью прозрачный пиксель, а равная единице – полностью непрозрачный.

После смешивания прозрачных фрагментов происходит вычисление освещения для непрозрачных фрагментов, а затем прозрачные фрагменты смешиваются с непрозрачными по формулам (1) и (2). Если в текущем пикселе отсутствует непрозрачный фрагмент, смешивание происходит с фоновым цветом или изображением.

Шаг 6. Вывод цвета на экран. Полученный на предыдущем шаге цвет выводится на экран.

На рисунке 2 представлен результат работы алгоритма. Все фрагменты смешаны в правильном порядке, получено верное изображение.

Рисунок 2 – Результат работы алгоритма

В результате выполненной работы разработано программное средство, реализующее представленный алгоритм, который позволяет визуализировать трёхмерные сцены, содержащие полупрозрачные объекты любой сложности. В ходе сравнения данного алгоритма с другими описанными алгоритмами были получены следующие результаты: расход памяти уменьшился в среднем на 80-90%, производительность увеличилась в среднем в 8-9 раз.

Список использованных источников:

1. Красковский П. Н., Серебряная Л. В. Метод отложенного затенения трехмерных сцен, использующий буфер видимости. – 2021.

2. Bavoil L., Myers K. Order independent transparency with dual depth peeling //NVIDIA OpenGL SDK. – 2008. – Т. 1. – С. 12.