## СПИСОК ЛИТЕРАТУРЫ

1. **Meagner D.** Geometric modeling using oct-tree encoding // Comp. Graph. and Image Proc.— 1982.— 18, N 2.— P. 129.
2. **Sammet H., Tamminen M.** Bintrees, CSG trees and time // Comp. Graph.— 1985.— 19, N 3.— P. 121.
3. **Jackins C. L., Tanimoto S. L.** Oct-trees and their use in representing three-dimensional objects // Comp. Graph. and Image Proc.— 1980.— 14, N 3.— P. 249.
4. **Walsh T. R.** On the size of quadtrees generalized to d-dimensional binary pictures // Comp. and Maths. with Appl.— 1985.— 11, N 11.— P. 1089.


*Поступила в редакцию 17 января 1990 г.*

УДК 681.3.06

**Е. П. КУЗЬМИН**
*(Москва)*

# ВИЗУАЛИЗАЦИЯ ИЕРАРХИЧЕСКИХ СТРУКТУР

**Введение.** В отличие от трассировки ячеистых структур [1, 2], где получены эффективные целочисленные алгоритмы генерации трассы луча, существующие методы построения трассы луча для иерархических структур [3, 4] основаны на вещественной арифметике и многократно используют поиск по дереву, что ограничивает область их применения лишь наполненными деревьями низкого разрешения.

В этих алгоритмах определяется последовательность терминальных узлов дерева методом поиска узла, содержащего фиксированную точку. Данный подход приводит к высокой зависимости времени трассировки от разрешения сцены, что дает основание считать подобный метод неперспективным [1].

**Построение трассы луча.** Рассмотрим трассируемый луч в мировом пространстве с иерархией. Алгоритм построения трассы этого луча основан на процедурной генерации иерархического дерева трассы луча на основе иерархического дерева сцены. Генерация производится в направлении удаления от наблюдателя до прихода на поверхность или выхода из мирового куба.

Тогда алгоритм построения трассы выглядит следующим образом:

```
трассировка (узел, луч)
{
  для (всех узлов_сыновей в порядке удаления от наблюдателя)
   выполнить
   {
      если (узел_сын не пуст и пересекается с лучом)
      {
     если (узел_сын — терминальный)
      {
    трассировка_узла (узел_сын, луч)
      }
     иначе
      {
     трассировка (узел_сыи, луч)
      }
      }
   }
}
```

Процедура **трассировка_узла ( )** производит обычную лучевую трассировку сцены внутри узла. При этом при обнаружении пересечения с объектом осуществляется полный выход из трассировки. Особенно упрощается эта процедура в случае чистых иерархических деревьев, где она сводится лишь к проверке заполненности узла.

Упорядоченность сыновей по удалению от наблюдателя легко определить при инициализации трассировки вдоль данного направления, так как он остается неизменным для всех узлов.

Основная процедура предлагаемого метода — проверка пересечения прямой и прямоугольного параллелепипеда — оболочки узла.

**Реализация алгоритма трассировки.** При реализации теста пересечения прямой и узла наиболее эффективно использование параметризации вдоль луча.

Рассмотрим луч с координатами и грани мирового куба. Тогда в общем случае пары плоскостей, содержащие противоположные грани, определят три отрезка на луче. Если эти отрезки имеют непустое пересечение, то наш узел пересекается с лучом. В противном случае луч не задевает узел.

При переходе к сыновьям границы отрезков, соответствующие новым плоскостям деления, вычисляются как среднее арифметическое координат отрезков узла-отца. Таким образом, при трассировке луча выполняются лишь операции сложения и сдвига.

Алгоритмы поиска пересечений особенно просты для бинарных иерархических структур. В этом случае для определения инцидентности луча с обоими сыновьями требуются лишь сложение, сдвиг и два сравнения. А для восьмеричных деревьев полный анализ расположения луча относительно восьми сыновей требует максимум трех сложений и сдвигов, а также шести сравнений. Таким образом, для перехода к узлу необходимо в среднем полторы операции.

С помощью масштабирования координат вдоль луча данный процесс допускает полностью целочисленную реализацию.

При трассировке пучка параллельных лучей, соответствующего целочисленным узлам решетки экрана, границы отрезков пересечения с плоскостями мирового куба на каждом луче также могут определяться при помощи одних сложений. Действительно, при переходе к соседнему лучу координаты концов изменяются всегда на одну и ту же величину, которая подсчитывается при инициализации трассировки.

Не представляет труда также и трассировка луча из точки. Если считать, что в этой точке расположено начало координат на луче, то при трассировке необходима еще одна дополнительная проверка пересечения трех вышеуказанных отрезков и луча от точки в положительном направлении. Этот вид трассировки применяется для отслеживания вторичных лучей, а также при центральной проекции.

**Заключение.** Рассмотрен эффективный целочисленный алгоритм лучевой трассировки иерархических структур. Применение алгоритма позволяет получить изображение размером $1024 \times 1024$ пиксела чистого бинарного дерева с разрешением $1024^3$ на процессоре MC68020 в пределах 15 мин. При разрешении порядка 256 визуализация занимает около 10—20 с.

Предложенный алгоритм применим также к бинарным сетям.

Следует также отметить, что дальнейшее значительное ускорение трассировки иерархических структур происходит при использовании когерентности трассируемых лучей.

## СПИСОК ЛИТЕРАТУРЫ


1. **Fujimoto A., Tanaka T., Iwata K.** ARTS: accelerated ray-tracing system // IEEE Comp. Graph. and Appl.— 1986.— **6**, N 4.— P. 16.
2. **Snyder John M., Barr Alan H.** Ray-tracing complex models containing surface tessellations // Comp. Graph.— 1987.— **21**, N 4.— P. 119.

3. **Glassner A. S.** Space subdivision for fast ray-tracing // IEEE Comp. Graph. and Appl.— 1984.— 4, N 10.— P. 15.
4. **Kaplan M.** The uses of spatial coherence in ray-tracing // ACM SIGGRAPH'85, Course Notes.— 1985.— **11**, p. 22.

*Поступила в редакцию 17 января 1990 г.*

---

УДК 681.3.068

**Р. И. ВИЛЬДАНОВ, А. М. МАЦОКИН**
*(Новосибирск)*

## ГРАФИЧЕСКИЙ ПАКЕТ ДЛЯ ДВК-3

Использование средств машинной графики на ПЭВМ в значительной степени повышает эффективность труда исследователей. В нашей стране на отечественных мини-ЭВМ и ЕС ЭВМ широко используется графическая система СМОГ [1], предоставляющая необходимый набор средств по формированию и выводу графиков, карт изолиний, аксонометрических проекций поверхностей, векторных полей, чертежно-конструкторской документации.

В настоящей работе предлагается вариант этой системы для ДВК-3 [2]. Напомним, что в состав ДВК-3 входят 2 МГМД и видеомонитор «Электроника МС 6105», обеспечивающий вывод графической информации на экран и ее совмещение с имеющейся алфавитно-цифровой информацией, а также раздельный вывод графической и алфавитно-цифровой информации. Размер графического изображения — 286 строк по вертикали и 400 точек в каждой строке (50 байт). Графическая информация хранится в ОЗУ контроллера графического дисплея (КГД), имеющем информационную емкость 16 Кбайт. Логической единице в памяти соответствует светящаяся точка на экране монитора.

При адаптации графической системы СМОГ на ДВК-3 к операционной системе РАФОС основная задача — математическое описание логического графического экрана и обеспечение вывода вектора. Естественно, что эта задача является простой и при ее решении на графическом экране вводим декартову систему координат, начало которой соответствует левому нижнему углу экрана, размер по горизонтали и вертикали экрана — 399 и 285, координаты $(i, j)$ каждой точки экрана являются целочисленными. Поскольку ОЗУ КГД представляет собой линейный байтовый массив, точке $(i, j)$ при ограничениях $0 \leqslant i \leqslant 399$ и $0 \leqslant j \leqslant 285$ соответствует $n$-й байт $m$-го бита, где

$$n = (285 - j)\,50 + k;$$

$$m = i - 8k$$

($k$ — целая часть от $i/8$).

Так как ни в аппаратном, ни в программном обеспечении ДВК-3 генератор векторов не реализован, целесообразно использовать один из эффективных алгоритмов — алгоритм Брезенхэйма. Отметим, что алгоритм Брезенхэйма реализован таким образом, что генерация точек вектора происходит упорядоченно по горизонтали слева направо. Такая реализация позволяет с помощью повторного вывода вектора стереть его без появления лишних точек («мусора») при произвольном порядке задания концов отрезка.

Прежде чем перенести пакет СМОГ на ДВК-3, было разработано более простое, но, с нашей точки зрения, минимально необходимое графическое ядро (ГЯ), на основе которого можно строить то или иное программное обеспечение по решению графических задач, в том числе и