5.4. Алгоритм, использующий z–буфер

Алгоритм, использующий *z-буфер* это один из простейших алгоритмов удаления невидимых поверхностей. Впервые он был предложен Кэтмулом. Работает этот алгоритм в пространстве изображения. Идея *z*-буфера является простым обобщением идеи о буфере кадра. Буфер кадра используется для запоминания атрибутов (интенсивности) каждого пиксела в пространстве изображения, z-буфер - это отдельный буфер глубины, используемый для запоминания координаты *z* или глубины каждого видимого пиксела в пространстве изображения. В процессе работы глубина или значение *z* каждого нового пиксела, который нужно занести в буфер кадра, сравнивается с глубиной того пиксела, который уже занесен в *z*-буфер. Если это сравнение показывает, что новый пиксел расположен впереди пиксела, находящегося в буфере кадра, то новый пиксел заносится в этот буфер и, кроме того, производится корректировка *z*-буфера новым значением *z*. Если же сравнение дает противоположный результат, то никаких действий не производится. По сути, алгоритм является поиском по *x* и *y* наибольшего значения функции *z (x, y)*.

Главное преимущество алгоритма – его простота. Кроме того, этот алгоритм решает задачу об удалении невидимых поверхностей и делает тривиальной визуализацию пересечений сложных поверхностей. Сцены могут быть любой сложности. Поскольку габариты пространства изображения фиксированы, оценка вычислительной трудоемкости алгоритма не более чем линейна. Поскольку элементы сцены или картинки можно заносить в буфер кадра или в *z*-буфер в произвольном порядке, их не нужно предварительно сортировать по приоритету глубины. Поэтому экономится вычислительное время, затрачиваемое на сортировку по глубине.

Основной недостаток алгоритма - большой объем требуемой памяти. Если сцена подвергается видовому преобразованию и отсекается до фиксированного диапазона значений координат *z*, то можно использовать *z*-буфер с фиксированной точностью. Информацию о глубине нужно обрабатывать с большей точностью, чем координатную информацию на плоскости *(x, y)*; обычно бывает достаточно 20-ти бит. Буфер кадра размером 512×512×24 бит в комбинации с *z*-буфером размером 512×512×20 бит требует почти 1.5 мегабайт памяти. Однако снижение цен на память делает экономически оправданным создание специализированных запоминающих устройств для *z*-буфера и связанной с ним аппаратуры.

Альтернативой созданию специальной памяти для *z*-буфера является использование для этой цели оперативной памяти. Уменьшение требуемой памяти достигается разбиением пространства изображения на 4, 16 или больше квадратов или полос. В предельном варианте можно использовать *z*-буфер размером в одну строку развертки. Для последнего случая имеется интересный алгоритм построчного сканирования. Поскольку каждый элемент сцены обрабатывается много раз, то сегментирование *z*-буфера, вообще говоря, приводит к увеличению времени, необходимого для обработки сцены. Однако сортировка на плоскости, позволяющая не обрабатывать все многоугольники в каждом из квадратов или полос, может значительно сократить этот рост.

Другой недостаток алгоритма *z*-буфера состоит в трудоемкости и высокой стоимости устранения лестничного эффекта, а также реализации эффектов прозрачности и просвечивания. Поскольку алгоритм заносит пикселы в буфер кадра в

произвольном порядке, то нелегко получить информацию, необходимую для методов устранения лестничного эффекта, основывающихся на предварительной фильтрации. При реализации эффектов прозрачности и просвечивания пикселы могут заноситься в буфер кадра в некорректном порядке, что ведет к локальным ошибкам.

Формальное описание алгоритма *z*-буфера таково:

1. Заполнить буфер кадра фоновым значением интенсивности или цвета.
2. Заполнить *z*-буфер минимальным значением *z*.
3. Преобразовать каждый многоугольник в растровую форму в произвольном порядке.
4. Для каждого *Пиксел*(*x,y*) в многоугольнике вычислить его глубину *z*(*x,y*).
5. Сравнить глубину *z*(*x,y*) со значением *Zбуфер*(*x,y*), хранящимся в *z*-буфере в этой же позиции.

Если *z*(*x,y*) > *Zбуфер* (*x,y*), то записать атрибут этого многоугольника (интенсивность, цвет и т. п.) в буфер кадра и заменить *Zбуфер*(*x,y*) на *z*(*x,y*). В противном случае никаких действий не производить.

На псевдокоде алгоритм можно представить так:

```
for all objects
    for all covered pixels
        compare z
```

В качестве предварительного шага там, где это целесообразно, применяется удаление нелицевых граней.

Если известно уравнение плоскости, несущей каждый многоугольник, то вычисление глубины каждого пиксела на сканирующей строке можно проделать пошаговым способом. Грань при этом рисуется последовательно (строка за строкой). Для нахождения необходимых значений используется линейная интерполяция (рис. 5.11).

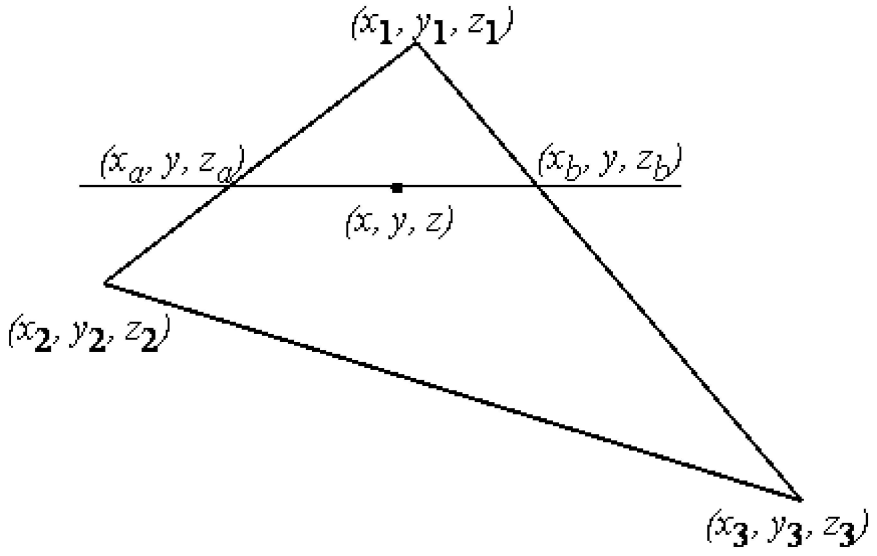

Рис. 5.11. Сканирующая строка по грани

Для рисунка y меняется от y_1 до y_2 и далее до y_3, при этом для каждой строки определяется x_a, z_a, x_b, z_b:

$$x_a = x_1 + (x_2 - x_1) \cdot \frac{y - y_1}{y_2 - y_1} ;$$

$$x_b = x_1 + (x_3 - x_1) \cdot \frac{y - y_1}{y_3 - y_1} ;$$

$$z_a = z_1 + (z_2 - z_1) \cdot \frac{y - y_1}{y_2 - y_1} ;$$

$$z_b = z_1 + (z_3 - z_1) \cdot \frac{y - y_1}{y_3 - y_1} .$$

На сканирующей строке x меняется от x_a до x_b и для каждой точки строки определяется глубина z:

$$z = z_a + (z_b - z_a) \cdot \frac{x - x_a}{x_b - x_a}$$

Реализация алгоритма вдоль сканирующей строки позволяет совместить алгоритм z-буфера с алгоритмами растровой развертки ребер и алгоритмами закраски грани.

Проиллюстрируем работу алгоритма на примере для рис. 5.12.

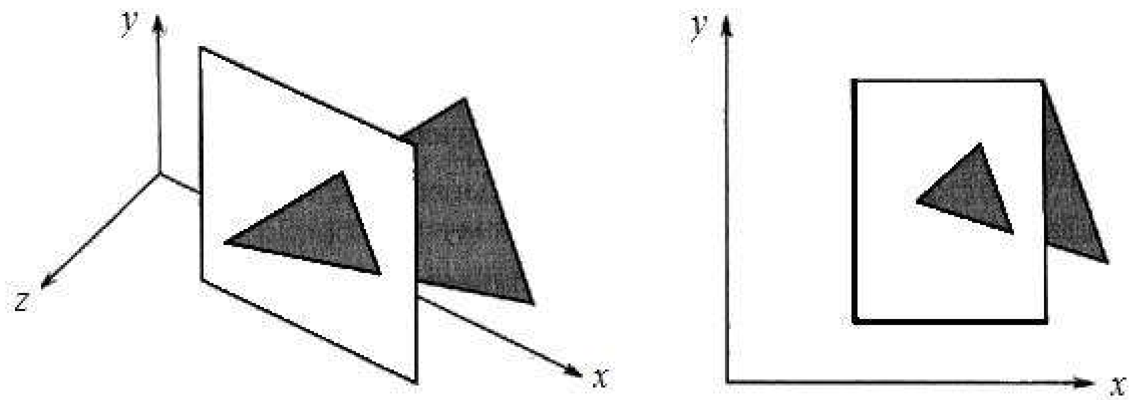

Рис. 5.12. Протыкающий треугольник

В начале в буфере кадра и в z-буфере содержатся нули. После растровой развертки прямоугольника содержимое буфера кадра будет иметь вид

0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	1	1	1	1	1	1	1	1	1	1	1	1	1	0	0	0
0	0	0	1	1	1	1	1	1	1	1	1	1	1	1	1	0	0	0
0	0	0	1	1	1	1	1	1	1	1	1	1	1	1	1	0	0	0
0	0	0	1	1	1	1	1	1	1	1	1	1	1	1	1	0	0	0

0	0	0	1	1	1	1	1	1	1	1	1	1	1	1	1	0	0	0
0	0	0	1	1	1	1	1	1	1	1	1	1	1	1	1	0	0	0
0	0	0	1	1	1	1	1	1	1	1	1	1	1	1	1	0	0	0
0	0	0	1	1	1	1	1	1	1	1	1	1	1	1	1	0	0	0
0	0	0	1	1	1	1	1	1	1	1	1	1	1	1	1	0	0	0
0	0	0	1	1	1	1	1	1	1	1	1	1	1	1	1	0	0	0
0	0	0	1	1	1	1	1	1	1	1	1	1	1	1	1	0	0	0
0	0	0	1	1	1	1	1	1	1	1	1	1	1	1	1	0	0	0
0	0	0	1	1	1	1	1	1	1	1	1	1	1	1	1	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Содержимое *z*-буфера таково:

0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	10	10	10	10	10	10	10	10	10	10	10	10	10	0	0	0
0	0	0	10	10	10	10	10	10	10	10	10	10	10	10	10	0	0	0
0	0	0	10	10	10	10	10	10	10	10	10	10	10	10	10	0	0	0
0	0	0	10	10	10	10	10	10	10	10	10	10	10	10	10	0	0	0
0	0	0	10	10	10	10	10	10	10	10	10	10	10	10	10	0	0	0
0	0	0	10	10	10	10	10	10	10	10	10	10	10	10	10	0	0	0
0	0	0	10	10	10	10	10	10	10	10	10	10	10	10	10	0	0	0
0	0	0	10	10	10	10	10	10	10	10	10	10	10	10	10	0	0	0
0	0	0	10	10	10	10	10	10	10	10	10	10	10	10	10	0	0	0
0	0	0	10	10	10	10	10	10	10	10	10	10	10	10	10	0	0	0
0	0	0	10	10	10	10	10	10	10	10	10	10	10	10	10	0	0	0
0	0	0	10	10	10	10	10	10	10	10	10	10	10	10	10	0	0	0
0	0	0	10	10	10	10	10	10	10	10	10	10	10	10	10	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

При обработке треугольника преобразование его в растровую форму и сравнение по глубине дает новое значение буфера кадра:

0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	1	1	1	1	1	1	1	1	1	1	1	1	1	2	0	0
0	0	0	1	1	1	1	1	1	1	1	1	1	1	1	1	2	0	0
0	0	0	1	1	1	1	1	1	1	1	1	1	1	1	1	2	0	0
0	0	0	1	1	1	1	1	1	1	1	1	1	1	1	1	2	0	0
0	0	0	1	1	1	1	1	1	1	1	2	1	1	1	1	2	2	0
0	0	0	1	1	1	1	1	1	1	2	2	1	1	1	1	2	2	0
0	0	0	1	1	1	1	1	1	2	2	2	2	1	1	1	2	2	0
0	0	0	1	1	1	1	1	2	2	2	2	2	1	1	1	2	2	0
0	0	0	1	1	1	1	2	2	2	2	2	2	1	1	1	2	2	0
0	0	0	1	1	1	1	1	2	2	2	2	2	2	1	1	2	2	2
0	0	0	1	1	1	1	1	1	1	1	2	2	2	1	1	2	2	2
0	0	0	1	1	1	1	1	1	1	1	1	1	1	1	1	2	2	2

0	0	0	1	1	1	1	1	1	1	1	1	1	1	1	1	0	0	2
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Новое содержимое *z*-буфера таково:

0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	10	10	10	10	10	10	10	10	10	10	10	10	10	5	0	0
0	0	0	10	10	10	10	10	10	10	10	10	10	10	10	10	5	0	0
0	0	0	10	10	10	10	10	10	10	10	10	10	10	10	10	6	0	0
0	0	0	10	10	10	10	10	10	10	10	10	10	10	10	10	6	0	0
0	0	0	10	10	10	10	10	10	10	10	11	10	10	10	10	6	5	0
0	0	0	10	10	10	10	10	10	10	12	11	10	10	10	10	7	5	0
0	0	0	10	10	10	10	10	10	13	12	11	11	10	10	10	7	6	0
0	0	0	10	10	10	10	10	14	13	12	12	11	10	10	10	7	6	0
0	0	0	10	10	10	10	15	14	13	12	12	11	10	10	10	7	6	0
0	0	0	10	10	10	10	10	14	13	12	12	11	11	10	10	8	7	5
0	0	0	10	10	10	10	10	10	10	10	12	11	11	10	10	8	7	5
0	0	0	10	10	10	10	10	10	10	10	10	10	10	10	10	8	7	6
0	0	0	10	10	10	10	10	10	10	10	10	10	10	10	10	0	0	6
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0