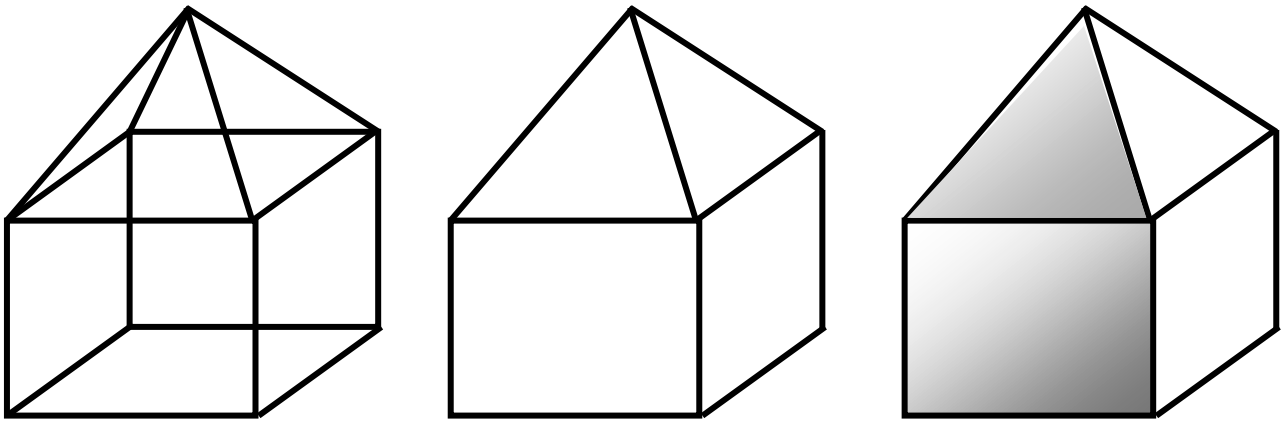
11 Визуализация объемных изображений

Любой объект, в том числе и объемный, может быть изображен различными способами. Условно разделим способы визуализации по характеру изображений и по степени сложности соответствующих алгоритмов на такие уровни:

1. Каркасная («проволочная») модель (рис.11.1, а).
2. Показ поверхностей в виде многогранников с плоскими гранями или сплайнов с удалением невидимых частей объектов (рис.11.1, б).
3. То же, что и для второго уровня, плюс сложное закрашивание объектов для имитации отражения света, затенения, прозрачности, использование текстур (рис.11.1, в).



а) б) в)

Рисунок 11.1 – Визуализация объемных изображений

11.1 Каркасная визуализация

Каркас обычно состоит из отрезков прямых линий (соответствует многограннику), хотя можно строить каркас и на основе кривых, в частности сплайновых кривых. Все ребра, показанные в окне вывода, видны – как ближние, так и дальние.

Для построения каркасного изображения надо знать координаты всех

вершин в мировой системе координат. Потом преобразовать координаты каждой вершины в экранные координаты в соответствии с выбранной проекцией. Затем выполнить цикл вывода в плоскости экрана всех ребер как отрезков прямых (или кривых), соединяющих вершины.

11.2 Показ с удалением невидимых частей объектов

Для построения более-менее реалистичного изображения трехмерных сцен необходимо уметь удалять невидимые части объектов (ребра и грани). Задача удаления невидимых линий и поверхностей является одной из наиболее сложных в машинной графике. Алгоритмы удаления невидимых линий и поверхностей служат для определения линии ребер, поверхностей или объемов, которые видимы или невидимы для наблюдателя, находящегося в заданной точке пространства.

Алгоритмы удаления невидимых линий или поверхностей можно классифицировать по ***способу выбора системы координат или пространства, в котором они работают.***

1) Алгоритмы, работающие в *объектном пространстве*, имеют дело с физической системой координат, в которой описаны эти объекты. Основная идея данных алгоритмов основана на непосредственном сравнении объектов друг с другом для выяснения того, какие части объектов являются видимыми. При этом получаются весьма точные результаты, ограниченные, вообще говоря, лишь точностью вычислений. Полученные изображения можно свободно увеличивать во много раз. Алгоритмы, работающие в объектном пространстве, особенно полезны в тех приложениях, где необходима высокая точность.
2) Алгоритмы же, работающие в *пространстве изображения*, имеют

дело с системой координат того экрана, на котором объекты визуализируются. Основная идея этих алгоритмов основана на определении для каждого пикселя экрана ближайшего к нему объекта (вдоль направления проецирования). При этом точность вычислений ограничена разрешающей способностью экрана. Результаты, полученные в пространстве изображения, а затем увеличенные во много раз, не будут соответствовать исходной сцене. Алгоритмы, формирующие список приоритетов, работают попеременно в обеих упомянутых системах координат.

11.2.1 Рассмотрим некоторые алгоритмы, работающие в объектном пространстве.

Алгоритм «Отсечение нелицевых граней», использующий нормали к плоскости

Напомним, что экран (плоскость проекции) находиться между наблюдателем и объектом (рис. 11.2). Если камера (глаз) находиться в точке E, то для каждой точки P объекта прямая PE (проецирующий луч) пересекает экран в точке P' (координаты проекции).

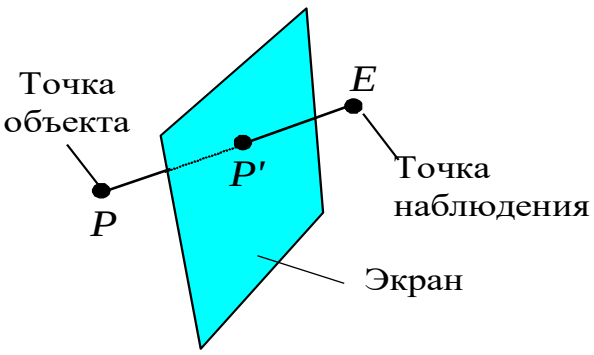


Рис. 11.2. Проекция точки объекта на экран

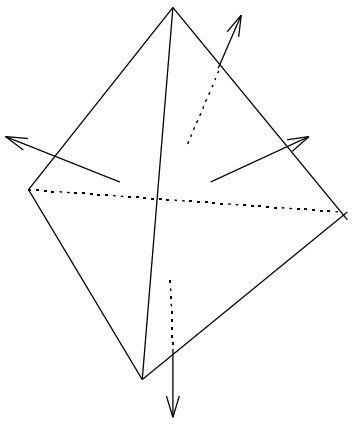


Рис. 11.3. Определение нелицевых граней

Пусть для каждой грани некоторой фигуры задан единичный вектор внешней нормали. Если вектор нормали грани составляет с направлением

проецирования (направлено на наблюдателя) тупой угол, то такая грань не может быть видна и называется нелицевой. В случае, когда данный угол является острым, грань видна и называется лицевой (см. рис. 11.3).

В случае, когда трехмерная сцена представляет собой один выпуклый многогранник, данный алгоритм полностью решает задачу удаления невидимых граней.

В общем случае описанная проверка не решает задачу полностью, но позволяет значительно сократить количество рассматриваемых граней.

Для расчета угла можно использовать формулу: $\cos(\bar{a},\bar{b}) = \frac{(\bar{a},\bar{b})}{|\bar{a}| * |\bar{b}|}$

Алгоритм Робертса

Алгоритм Робертса представляет собой первое известное решение задачи об удалении невидимых линий. Причем данный алгоритм позволяет производить удаление невидимых частей не только в одном теле, но и в целой сцене, состоящей из набора тел.

Алгоритм прежде всего удаляет из каждого тела те ребра или грани, которые экранируются самим телом. Затем каждое из видимых ребер каждого тела сравнивается с каждым из оставшихся тел для определения того, какая его часть или части, если таковые есть, экранируются этими телами. Поэтому вычислительная трудоемкость алгоритма Робертса растет теоретически как квадрат числа объектов.

В алгоритме Робертса требуется, чтобы все изображаемые тела или объекты были ***выпуклыми***. Невыпуклые тела должны быть разбиты на выпуклые части. Каждое выпуклое многогранное тело с плоскими гранями должно представляться набором пересекающихся плоскостей.

Чтобы алгоритм Робертса работал корректно, необходимо выполнение следующих предварительных условий:

1. Тело ограничено плоскостями.
2. Тело выпукло.
3. Нормали всех граней направлены внутрь тела.

Уравнение произвольной плоскости в трехмерном пространстве имеет вид

$$ax + by + cz + d = 0 \qquad (11.1)$$

В матричной форме этот результат выглядит так:

$$[x\ y\ z\ 1] \cdot \begin{bmatrix} a \\ b \\ c \\ d \end{bmatrix} = 0 \qquad (11.2)$$

или

$$[x\ y\ z\ 1] \cdot [P]^T = 0 \qquad (11.3)$$

где $[P] = [a\ b\ c\ d]$ представляет собой плоскость, $[a\ b\ c]$ – координаты вектора-нормали данной плоскости.

Поэтому любое выпуклое твердое тело можно выразить матрицей тела, состоящей из коэффициентов уравнений плоскостей, т. е.

$$[V] = \begin{bmatrix} a_1 & a_2 & \dots & a_n \\ b_1 & b_2 & \dots & b_n \\ c_1 & c_2 & \dots & c_n \\ d_1 & d_2 & \dots & d_n \end{bmatrix}, \qquad (11.4)$$

где каждый столбец содержит коэффициенты одной плоскости.

Напомним, что любая точка пространства представима в однородных координатах вектором

$$[S]=[x_s \quad y_s \quad z_s \quad 1] \qquad (11.5)$$

Более того, если точка $[S]$ лежит на плоскости, *то* $ax_s + by_s + cz_s + d = 0$. Другими словами, скалярное произведение $[S]\cdot[P]^T$ равно 0 ($[S]\cdot[P]^T = 0$). Если же $[S]$ не лежит на плоскости, то знак этого скалярного произведения показывает, по какую сторону от плоскости расположена точка. ***В алгоритме Робертса предполагается, что точки, лежащие внутри тела, дают положительное скалярное произведение***.

Поэтому, важное требование к модели объекта в алгоритме Робертса заключается в достижении такой формы (знака) в функции каждой плоскости $f_n(x,y,z) = a_n x + b_n y + c_n z + d_n$, при которой справедливо $f_n(x_{вн}, y_{вн}, z_{вн}) > 0$ для любой точки ($x_{вн}$,$y_{вн}$,$z_{вн}$), заведомо лежащей внутри (принадлежащей телу) многогранника.

Достижение этого условия осуществляется путем опытной проверки знака функции относительно внутренней точки, в качестве которой может выступать точка со средним геометрическим положением относительно всех вершин многогранника.

При достижении "положительности" всех граней автоматически достигается ориентация нормали к любой из граней внутрь фигуры.

Поскольку объем вычислений в алгоритмах удаления невидимых линий или поверхностей растет с увеличением числа многоугольников, для описания поверхностей выгодно использовать многоугольники с более чем тремя сторонами. Эти многоугольники могут быть как невыпуклыми, так и неплоскими. Метод, предложенный ***Мартином Ньюэлом***, позволяет найти как точное решение для уравнений плоскостей, содержащих плоские многоугольники, так и «наилучшее» приближение для неплоских многоугольников. Этот метод эквивалентен определению нормали в каждой

вершине многоугольника посредством векторного произведения прилежащих ребер и усреднения результатов.

Пусть *a, b, c, d* - коэффициенты уравнения плоскости, вычисляемые по формулам

$$a = \sum_{i=1}^{n} (y_i - y_j)(z_i + z_j)$$
$$b = \sum_{i=1}^{n} (z_i - z_j)(x_i + x_j) \qquad (11.6)$$
$$c = \sum_{i=1}^{n} (x_i - x_j)(y_i + y_j)$$

где если $i = n$, то $j = 1$, иначе $j = i + 1$. (в данном случае имеем координаты векторов плоскости)

Величина *d* вычисляется с помощью произвольной точки на плоскости. В частности, если компоненты (координаты) этой точки на плоскости (x_t, y_t, z_t), то

$$d = -(ax_t + by_t + cz_t) \qquad (11.7)$$

Тот факт, что плоскости имеют бесконечную протяженность и что скалярное произведение точки на матрицу тела отрицательно, если точка лежит вне этого тела, позволяет предложить метод, в котором матрица тела используется для определения граней, которые экранируются самим этим телом.

Если зритель находится в бесконечности на положительной полуоси *z* и смотрит на начало координат, то его взгляд направлен в сторону отрицательной полуоси *z* (рис.11.4). В однородных координатах вектор такого направления равен:

$$[E] = [0 \quad 0 \quad -1 \quad 0] \qquad (11.8)$$

который служит, кроме того, образом точки, лежащей в бесконечности на отрицательной полуоси *z*. Фактически [*E*] представляет любую точку, лежащую на плоскости $z = -\infty$, т. е. любую точку типа $(x, y, -\infty)$.

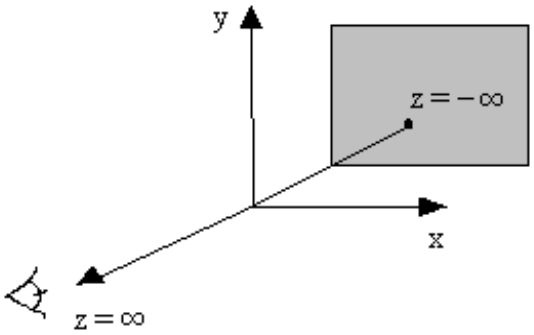


Рисунок 11.4

Поэтому, если скалярное произведение [*E*] на столбец, соответствующий какой-нибудь плоскости в матрице тела отрицательно, то [*E*] лежит по отрицательную сторону этой плоскости. Следовательно, эти плоскости невидимы из любой точки наблюдения, лежащей в плоскости $z = \infty$, а пробная точка на $z=-\infty$ экранируется самим телом. Такие плоскости называются ***нелицевыми***, а соответствующие им грани - задними.

Следовательно, $$[E]\cdot[V]<0 \qquad (11.9)$$

является условием того, что плоскости — нелицевые, а их грани – задние.

Заметим, что для аксонометрических проекций (точка наблюдения в бесконечности) это эквивалентно поиску положительных значений в третьей строке матрицы тела.

Этот метод является простейшим алгоритмом удаления невидимых поверхностей для тел, представляющих собой одиночные выпуклые многогранники. Метод эквивалентен вычислению нормали к поверхности для

каждого отдельного многоугольника. Отрицательность нормали к поверхности показывает, что нормаль направлена в сторону от наблюдателя и, следовател

ьно, данная грань не видна.

В общем виде алгоритм Робертса (для одного выпуклого тела, наблюдатель находится в точке z=∞) можно представить так:

1) Сформировать многоугольники (грани и ребра), исходя из списка вершин тела.
2) Вычислить уравнение плоскости для каждой полигональной грани тела.
3) Проверить знак уравнения плоскости:
   a. Взять любую точку внутри тела, например, усреднив координаты его вершин.
   b. Вычислить скалярное произведение уравнения плоскости и точки внутри тела.
   c. Если это скалярное произведение < 0, то изменить знак уравнения этой плоскости (чтобы отразить правильное направление нормали).
4) Сформировать матрицу тела.
5) Умножить ее на матрицы видового преобразования и проецирования.
6) Определить нелицевые плоскости:
   a. Вычислить скалярное произведение пробной точки, лежащей в «минус» бесконечности, на преобразованную матрицу тела.
   b. Если это скалярное произведение < 0, то плоскость невидима.
   c. Удалить весь многоугольник, лежащий в этой плоскости.

Общий вид алгоритма Робертса (для одного выпуклого тела, наблюдатель находится в произвольной точке P).

Пусть F_1, F_2, ..., F_n - грани многогранника, заданного координатами своих вершин V1, V2, ..., Vm. Рассмотрим одну из граней. Обозначим вершины, инцидентные грани, через V1, V2, ..., Vk. Найдем вектор нормали к грани, вычислив векторное произведение любых двух смежных ребер этой грани $V_1V_2 = [x_1,y_1,z_1]$ и $V_2V_3 = [x_2,y_2,z_2]$:

$$n_i = [V_1V_2, \quad V_2V_3] = \begin{vmatrix} i & j & k \\ x_1 & y_1 & z_1 \\ x_2 & y_2 & z_2 \end{vmatrix} = (y_1z_2 - y_2z_1)\cdot i + (z_1x_2 - z_2x_1)\cdot j + (x_1y_2 - x_2y_1)\cdot k =$$

$$= A_i \cdot i + B_i \cdot j + C_i \cdot k$$

Тогда функция, описывающая плоскость грани, имеет вид:

$$f_i(x,y,z) = A_ix + B_iy + C_iz + D$$

Величина D вычисляется с помощью произвольной точки на плоскости. В частности, если компоненты этой точки на плоскости (x_t, y_t, z_t), то:

$$D = -(ax_t + by_t + cz_t)$$

Так как многогранник выпуклый, коэффициенты A_i, B_i, C_i легко выбрать так, чтобы $n_i(A_i, B_i, C_i)$ был вектором нормали. Для этого найдем какую-либо внутреннюю точку, например, барицентр многогранника:

$$W = (V_1 + V_2 + ... + V_m)/m$$

Если скалярное произведение уравнения плоскости и этой точки меньше 0, то необходимо поменять знак уравнения этой плоскости, чтобы отразить правильное направление внешней нормали.

Остается только вычислить скалярное произведение уравнения плоскости на точку, в которой находится наблюдатель. Если это скалярное произведение меньше 0, то плоскость невидима и необходимо удалить весь многоугольник, лежащий в этой плоскости.

Запись этого алгоритма на псеводокоде приводится ниже.

Алгоритм Робертса
V1, V2, V3 - вершины многогранника
W - барицентр многогранника
P - точка наблюдения

```
выделим одну из граней многогранника
Vec1.X = V1.X - V2.X;          / для этой грани найдем координаты
Vec2.X = V3.X - V2.X;          двух векторов, которые лежат в
Vec1.Y = V1.Y - V2.Y;          плоскости грани
Vec2.Y = V3.Y - V2.Y;
Vec1.Z = V1.Z - V2.Z;
Vec2.Z = V3.Z - V2.Z;
A = Vec1.Y·Vec2.Z - Vec2.Y·Vec1.Z;   / вычислим коэффициенты уравнения
B = Vec1.Z·Vec2.X - Vec2.Z·Vec1.X;   плоскости
C = Vec1.X·Vec2.Y - Vec2.X·Vec1.Y;
D = -(A·V1.X + B·V1.Y + C·V1.Z);
m = -Sign(A·W.X + B·W.Y + C·W.Z + D);   / коэффициент, изменяющий знак
                                        плоскости
A = A·m;                       / корректируем направление плоскости
B = B·m;
C = C·m;
D = D·m;
if A·P.X + B·P.Y + C·P.Z + D > 0 then
грань видима; отобразить грань
else
грань невидима
```

11.2.2 Рассмотрим некоторые алгоритмы, работающие в пространстве изображения.

Метод z-буфера.

Один из самых простых алгоритмов. Этот метод используется для сложных сцен с применением ортогонального проецирования на плоскость.

Каждому пикселю экрана сопоставляется расстояние до проецируемого на него объекта (z-буфер). Для вывода на экран произвольной грани, она сначала переводиться в свое растровое представление на экране и для каждого пикселя определяется его «глубина».

Для использования метода организуются и используются две области памяти:

Z-буфер (глубины) – для хранения координаты *z* («глубины») каждого видимого на данной стадии анализа изображения пикселя картинной плоскости;

К-буфер (кадра) – для запоминания и хранения атрибутов (яркость и цвет) соответствующего пикселя.

В начальный момент в Z-буфере хранится число, обозначающее глубину фона, обычно это большое число. В К-буфере хранятся атрибуты фона.

Анализируется глубина (значение *z*) каждого нового пикселя изображения путем сравнения ее с глубиной того пикселя, который имеет ту же проекцию на картинную плоскость и уже занесен в буфер глубины.

Если *z* нового пикселя меньше *z* старого, то рассматриваемый элемент ближе к картинной плоскости. Подправляется координата *z* соответствующего элемента Z-буфера, а атрибуты нового пикселя заносятся в К-буфер.

В противном случае не делается никаких действий.

Формальное описание метода:

1. Весь буфер кадра инициализируется фоновыми значениями (интенсивности, цвета).

2. Буфер глубины инициализируется значениями глубины фона.

3. Для каждой грани сцены:

3.1. Преобразуются проекции грани в растровую форму.

3.2. Для каждого пикселя проекции вычисляется его глубина $z = z(x, y)$.

3.3. Сравнивается значение $z(x, y)$ с соответствующим значением буфера глубины $Z(x, y)$.

3.4. Если $z(x, y) < Z(x, y)$, то:

3.4.1. Записывается атрибут этого пиксела в буфер кадра.

3.4.2. Записывается z вместо Z.

3.5. Иначе никаких действий не происходит.

Алгоритм метода Z-буфера легко модифицируется для получения сечений поверхности параллельными плоскостями z_1, z_2.

Это можно получить, если в пункте 3.4 поставить условие:

если $z < Z(x, y)$ и $z_1 < z(x, y) < z_2$.

В итоге на картинной плоскости останутся только те части сцены, которые расположены между секущими плоскостями z_1, z_2.

В связи с простотой и трудоемкостью алгоритма распространены его аппаратные реализации.

Проиллюстрируем работу алгоритма на примере для рис. 11.5.

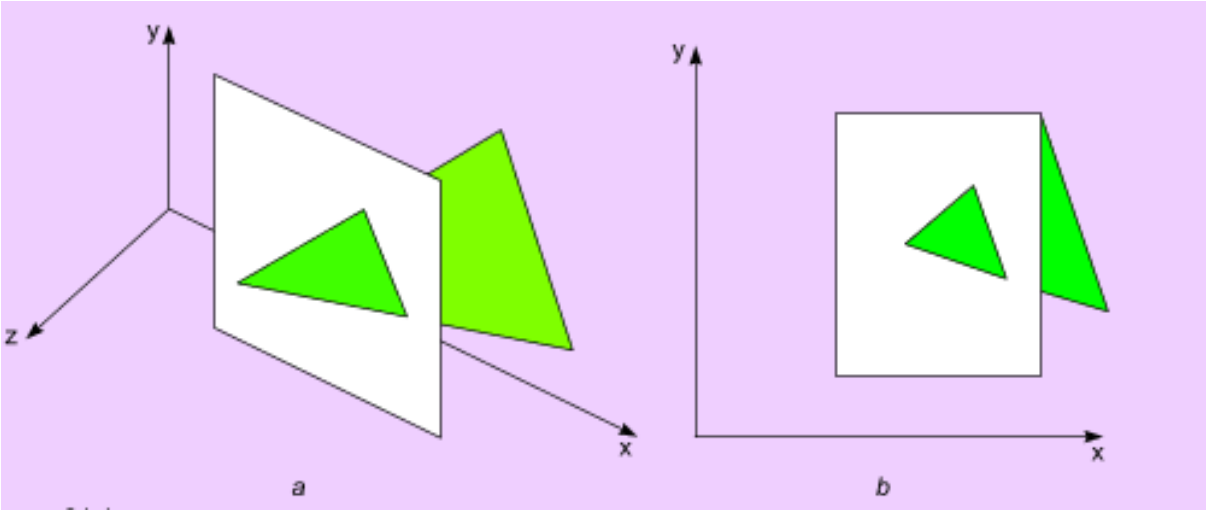


Рис. 11.5. Протыкающий треугольник

Многоугольник, координаты угловых точек которого равны $P_1(10, 5, 10)$, $P_2(10, 25, 10)$, $P_3(25, 25, 10)$, $P_4(25, 5, 10)$ и треугольник с вершинами $P_5(15, 15, 15)$, $P_6(25, 25, 5)$, $P_7(30, 10, 5)$.

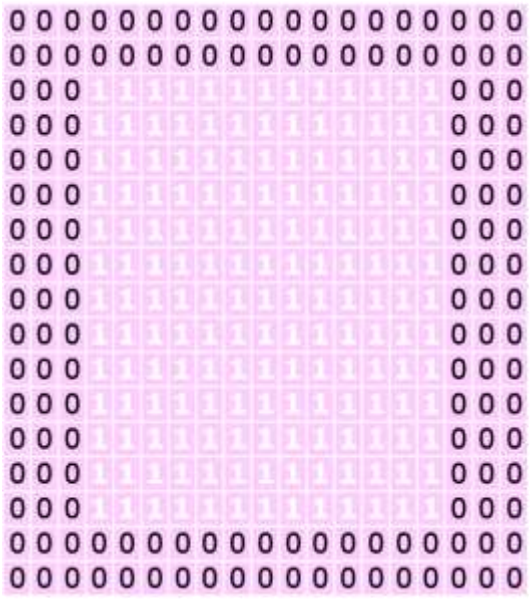
Треугольник протыкает прямоугольник

Буфер кадра:
фон обозначен через 0,
прямоугольник — через 1,
а треугольник — через 2
z-буфер: плоскости размером 32 * 32 бит.

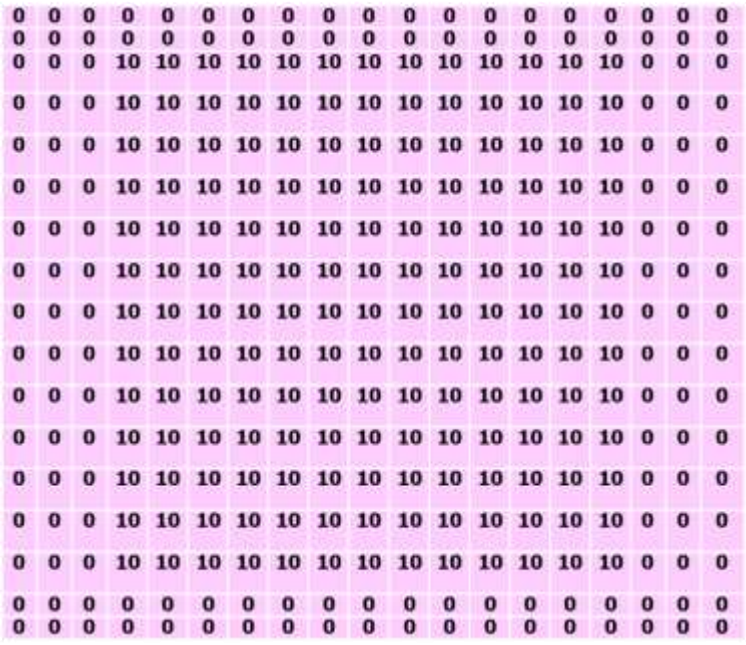
Вначале и в буфере кадра, и в z-буфере содержатся нули

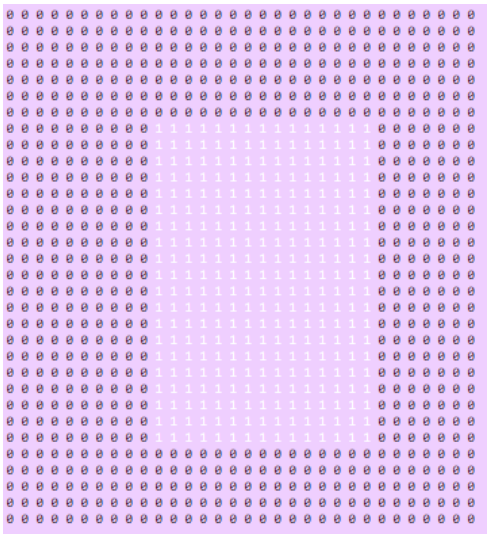
После растровой развертки прямоугольника содержимое буфера кадра будет иметь вид:

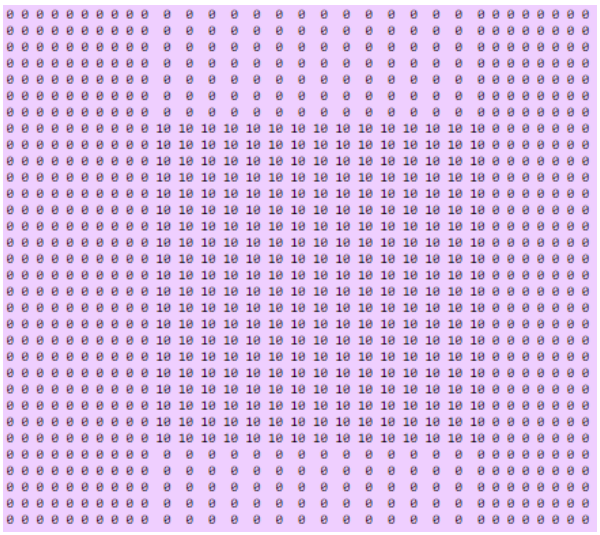
буфера кадра



Z-буфер

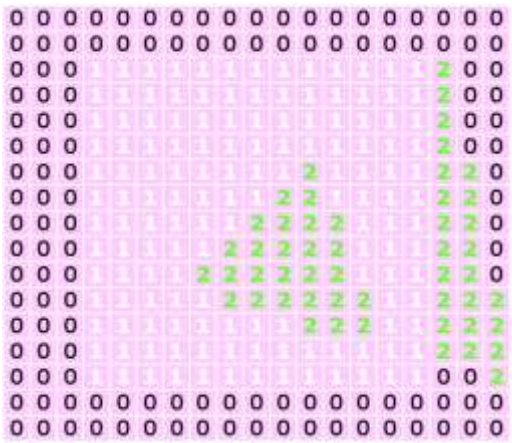




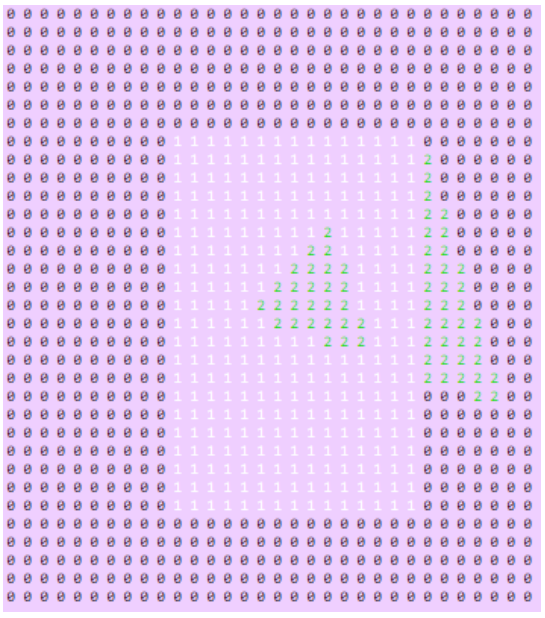


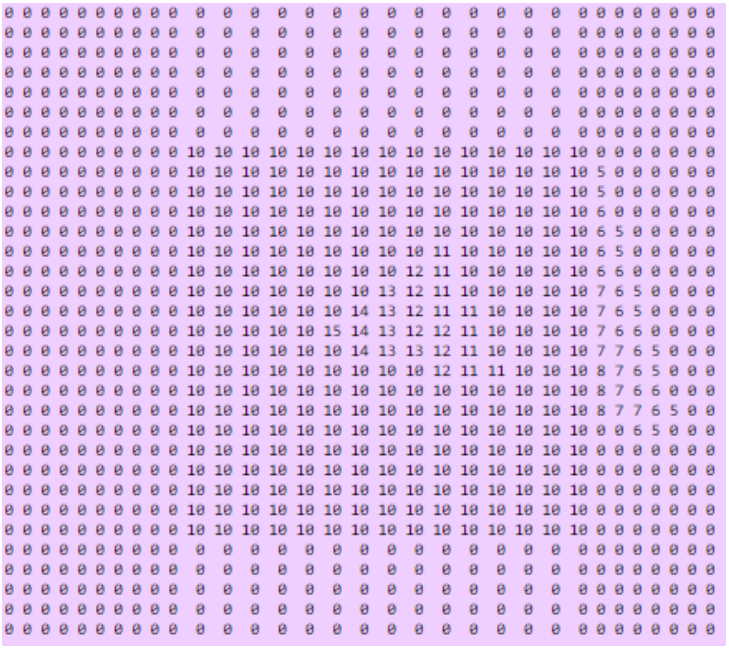
При обработке треугольника преобразование его в растровую форму и сравнение по глубине дает новое значение буфера кадра новое содержимое z-буфера:

буфера кадра



Z-буфер





Алгоритм Варнака.

Основан на «разбиении» экрана на части. Сначала разделим картинную плоскость на 4 равные части (Рис. 11.5). Возможны следующие ситуации:

1) часть экрана полностью накрывается проекцией ближайшей грани;

2) часть экрана не накрывается проекцией ни одной грани;

3) ни первое, ни второе условия не выполняются.

В первом и втором случаях часть экрана полностью закрашивается соответствующим цветом. В третьем случае данная часть разбивается еще на 4 части, для каждой из которой вновь выполняется проверка, и так далее. Разбиение можно выполнять пока размер части не будет соответствовать одному пикселю (если до этого дошло, то пиксель закрашивается цветом ближайшей к нему грани).

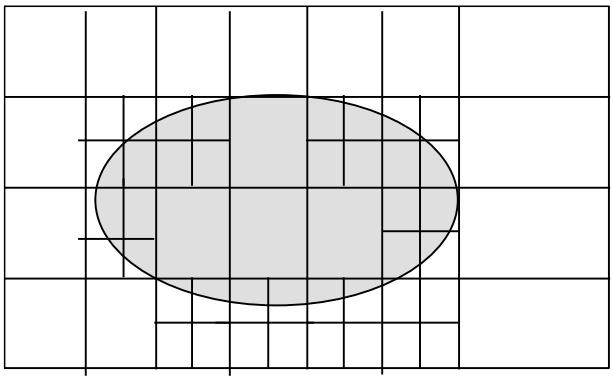


Рис. 11.5. Разбиение экрана на части

Метод построчного сканирования.

Все изображение на экране представляет собой набор вертикальных линий (полосу пикселов). Рассмотрим сечение трехмерной сцены плоскостью, проходящей через такую линию и центр проекции (точку наблюдения). Результатом такого сечения будет набор отрезков, которые необходимо спроецировать на экран. Исходная задача свелась к удалению невидимых отрезков на каждой линии.

Данный алгоритм может быть использован при реализации перемещения по прямоугольному лабиринту. В случае если высота пола и потолка одинакова, а стены вертикальны, задача вообще сводиться к двумерной (см. рис. 11.6 вид сверху). Разложим изображение на экране в ряд вертикальных линий. Каждая такая линия определяется пересечением вертикальной полуплоскости, проходящей через центр проекции (камеру) и заданную вертикальную линию. Видимым будет ближайшее пересечение со стенами лабиринта.

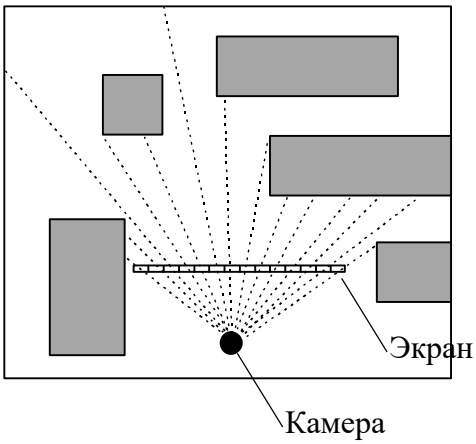


Рис. 11.6. Вид сверху на лабиринт

Каждая вертикальная линия может состоять из трех частей – пола, стены и потолка. Решая задачу в двумерном пространстве, определяем расстояние до ближайшей стены. Часть линии закрашиваем цветом пола, часть цветом стены (в зависимости от расстояния можно менять интенсивность цвета), часть цветом потолка.