Лекция 18

Текстуры

Нанесение текстур

Различают два вида текстур:

- Процедурные
- Проективные (наносятся на грань объекта)

Процедурные текстуры

Рассмотрим простой пример: есть домик с кирпичными стенами. Решить задачу описания грани домика достаточно сложно. Можно было бы описать стенку, но это тоже сложно, поэтому эту стенку рисуют отдельно, а потом накладывают в качестве текстуры на нужную грань.

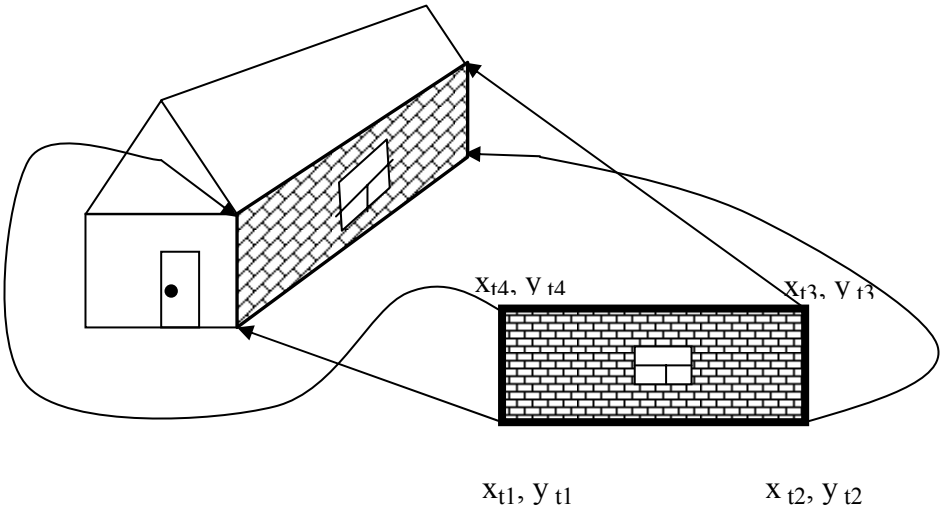
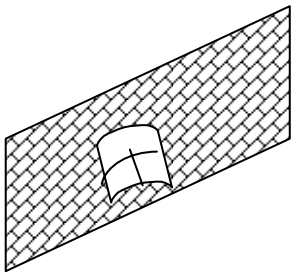


Рис. 4.1.1

индекс t означает «текстурный»



В ряде случаев могут получаться искажения

Рис. 4.1.2

Чем больше перспективное искажение, тем больше эти искажающие эффекты.

Решается следующая задача: в плоскости изображения имеется некоторый прямоугольник и bmp-картинка, которую нам надо вписать в этот прямоугольник. Эту задачу можно сформулировать иначе: имеется некоторый многоугольник и картинка, ему соответствующая. Многоугольник задан текстурными координатами, по которым из текстурного поля вырезается определённый кусок и наносится на объект.

Коррекция текстуры

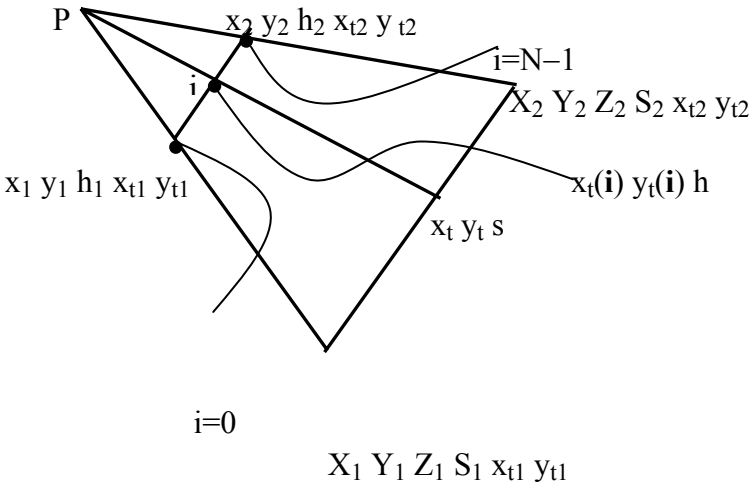


Рис. 4.2

$$\begin{cases} x_t = x_{t1} + \dfrac{s - s_1}{s_2 - s_1}\,(x_{t2} - x_{t1}) \\ y_t = y_{t1} + \dfrac{s - s_1}{s_2 - s_1}\,(y_{t2} - y_{t1}) \end{cases}$$

Линейная интерполяция:

$$h = \frac{A}{S} - B\,; \quad h_1 = \frac{A}{S_1} - B\,; \quad h_2 = \frac{A}{S_2} - B\,;$$

После подстановки всех формул получаем:

$$\begin{cases} x_t(\mathbf{i}) = x_{t1} + \dfrac{\mathbf{i}}{N\text{-}1}\,(x_{t2} - x_{t1})\left(\dfrac{h_2 + B}{h(\mathbf{i}) + B}\right) \\ y_t(\mathbf{i}) = y_{t1} + \dfrac{\mathbf{i}}{N\text{-}1}\,(y_{t2} - y_{t1})\left(\dfrac{h_2 + B}{h(\mathbf{i}) + B}\right) \\ h(\mathbf{i}) = h_1 + \dfrac{\mathbf{i}}{N-1}\,(h_2 - h_1) \end{cases}$$

коэффициент, стоящий в первых двух уравнениях системы в скобках, – это *поправочный* коэффициент, выполняющий коррекцию текстурных координат

При программировании эти формулы можно упростить:

$$fx(\mathbf{i}) = \frac{\mathbf{i}}{N-1}\,(x_{t2} - x_{t1})\,(h_2 + B); \quad fx(\mathbf{i+1}) = fx(\mathbf{i}) + \Delta fx$$

$$fy(\mathbf{i}) = \frac{\mathbf{i}}{N-1}\,(y_{t2} - y_{t1})\,(h_2 + B); \quad fy(\mathbf{i+1}) = fy(\mathbf{i}) + \Delta fy$$

$$\Delta fx = \frac{(x_{t2} - x_{t1})\,(h_2 + B)}{N-1}; \quad \Delta fy = \frac{(y_{t2} - y_{t1})\,(h_2 + B)}{N-1};$$

Тогда будем иметь:

$$\begin{cases} x_t(\mathbf{i}) = x_{t1} + \dfrac{fx(\mathbf{i})}{h(\mathbf{i})+B} \\ y_t(\mathbf{i}) = y_{t1} + \dfrac{fy(\mathbf{i})}{h(\mathbf{i})+B} \end{cases}$$

Билинейная интерполяция

При обращении к полю текстуры с дробными координатами мы округляем их до целых. В результате получается несколько кривое изображение. Можно с этим бороться, используя билинейную интерполяцию. Она основана на том, что яркость в точке находится по яркости 4-х соседних точек, а в пространственных координатах это позволит избежать искривлений при проецировании текстуры. Однако в координатах изображения это будет уже не линейная интерполяция.
Задача заключается в нахождении яркости в точке V.
Сначала находится яркость в точке V5 с учетом линейной интерполяции между точками V1 и V4; затем в точке V6 с учетом интерполяции между точками V2 и V3. Затем производится интерполяция между точками V5 и V6 для нахождения яркости в точке V.

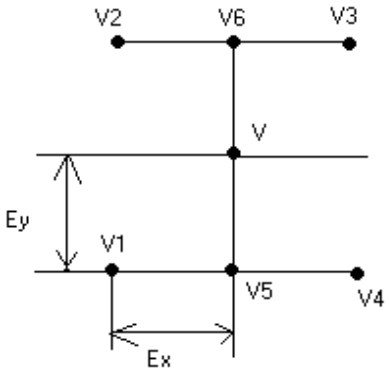
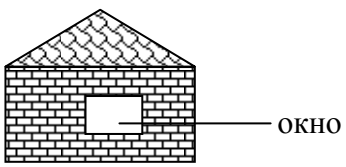


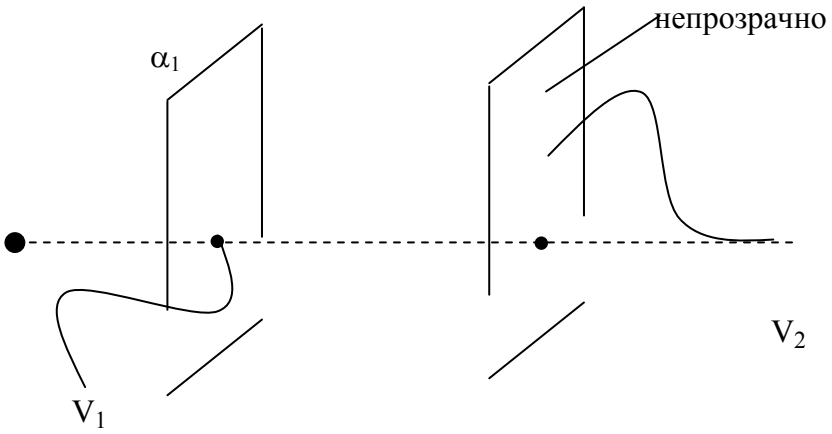
Рис. 4.3.1

V5=V1*(1-Ex) +V4*Ex
V6=V2*(1-Ex) +V3*Ex
V = V5*(1-Ey) +V6*Ey
Ex, Ey∈{0:1}

Линейная интерполяция в большинстве случаев не совсем корректна, так как можно получить эффект искажения картинки. Необходимо линейно интерполировать текстурные координаты исходя из координат изображения. Использование этого метода существенно замедляет работу алгоритма, но улучшает качество картинки.

Вопрос: как реализовать прозрачность (например, прозрачное окно в доме)?
Ответ: вводят признак прозрачности текстуры. При появлении кода прозрачности соответствующие точки игнорируются.





α_1 – коэффициент прозрачности.
$\alpha_1 \in [0, 1]$
$\alpha_1 = 0$ => прозрачно
$\alpha_1 = 1$ => непрозрачно

Рис. 4.3.2

В этом случае $V_\Sigma = \alpha_1 V_1 + V_2 (1-\alpha_1)$

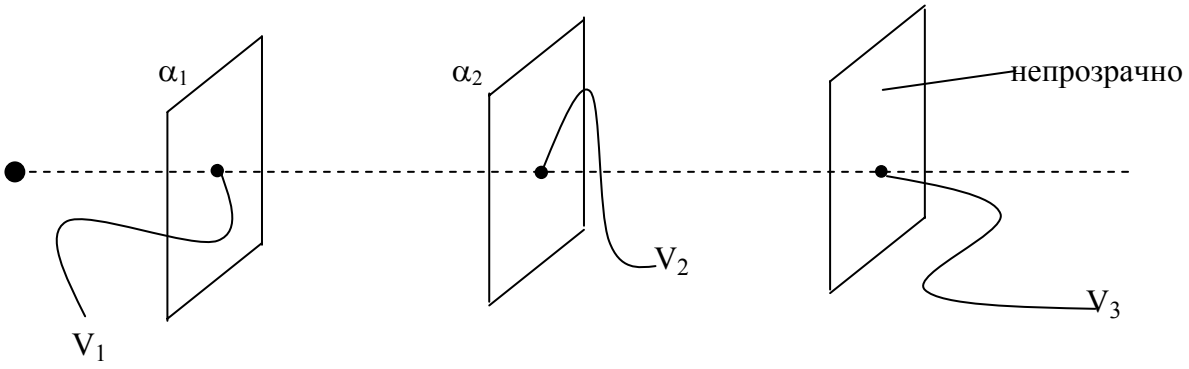


Рис. 4.3.3

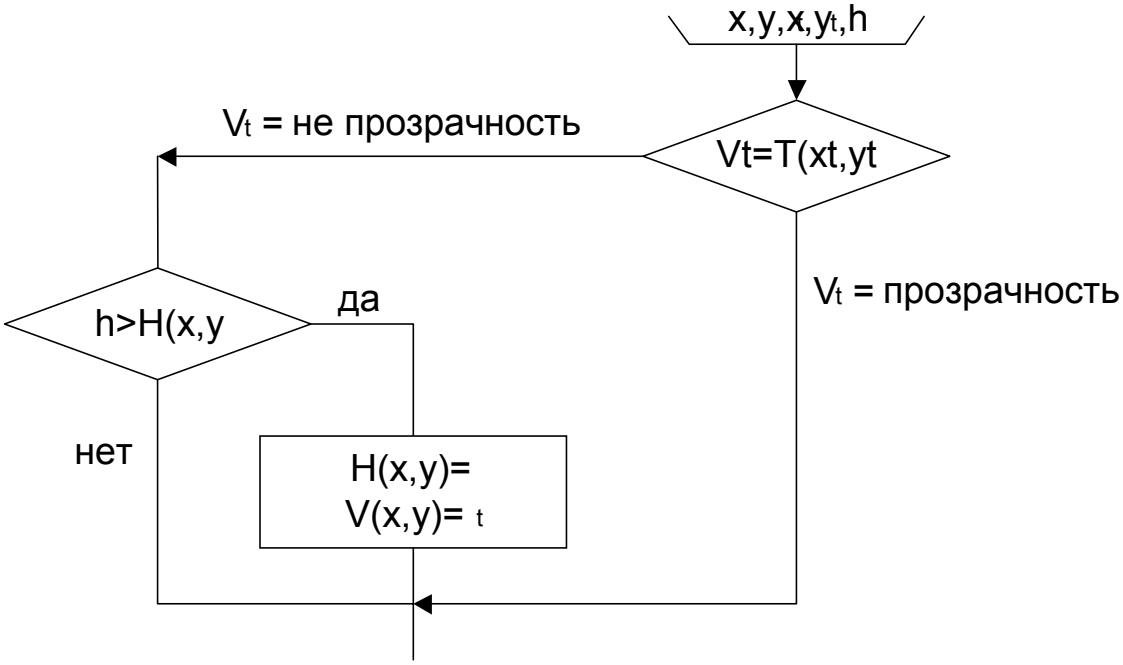
$V_\Sigma = \alpha_1 V_1 + (1-\alpha_1)(\alpha_2 V_2 + V_3 (1-\alpha_2))$

Виды текстур

1. 1. Прозрачные текстуры
2. 2. Полупрозрачные текстуры
3. 3. Циклические текстуры
4. 4. Динамические текстуры
5. 5. текстуры с мультиразрешением

Прозрачные

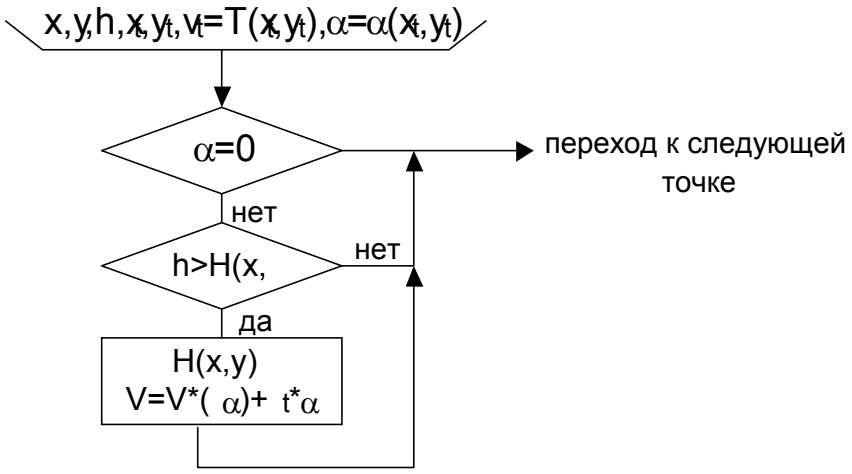
Если необходима прозрачная текстура, то можно воспользоваться следующим методом:



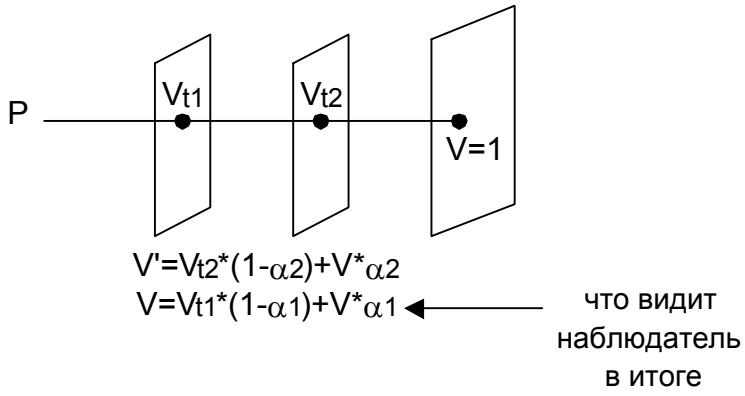
Зарезервируем один код V_t под признак прозрачности, то есть эта точка не будет заноситься в буфер изображения. Нижеприведенный алгоритм отображает обработку одной точки при использовании прозрачной текстуры:

Полупрозрачные

При применении полупрозрачной текстуры используется L-буфер, хранящий коэффициенты прозрачности всех точек текстуры, то есть размет буфера равен размеру текстуры.
Коэффициент прозрачности L = 0…1 , 0 – чисто прозрачная точка, 1 – непрозрачная точка.
Алгоритм обработки точки полупрозрачной текстуры:



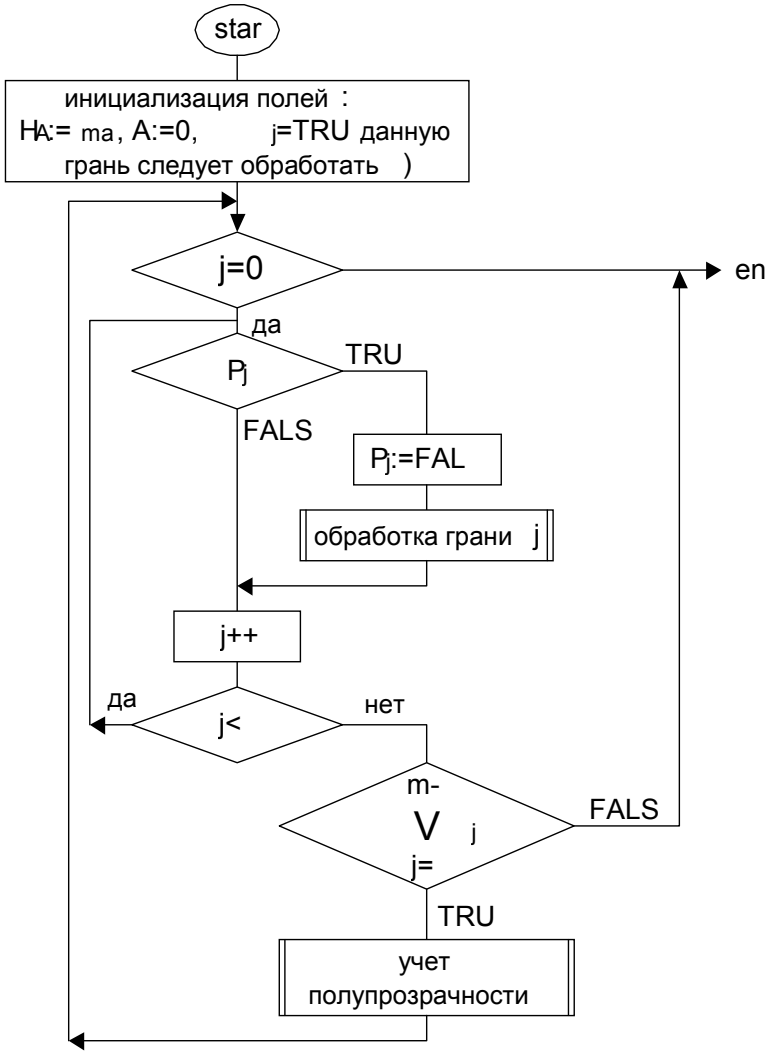
Наложение двух полупрозрачных граней:



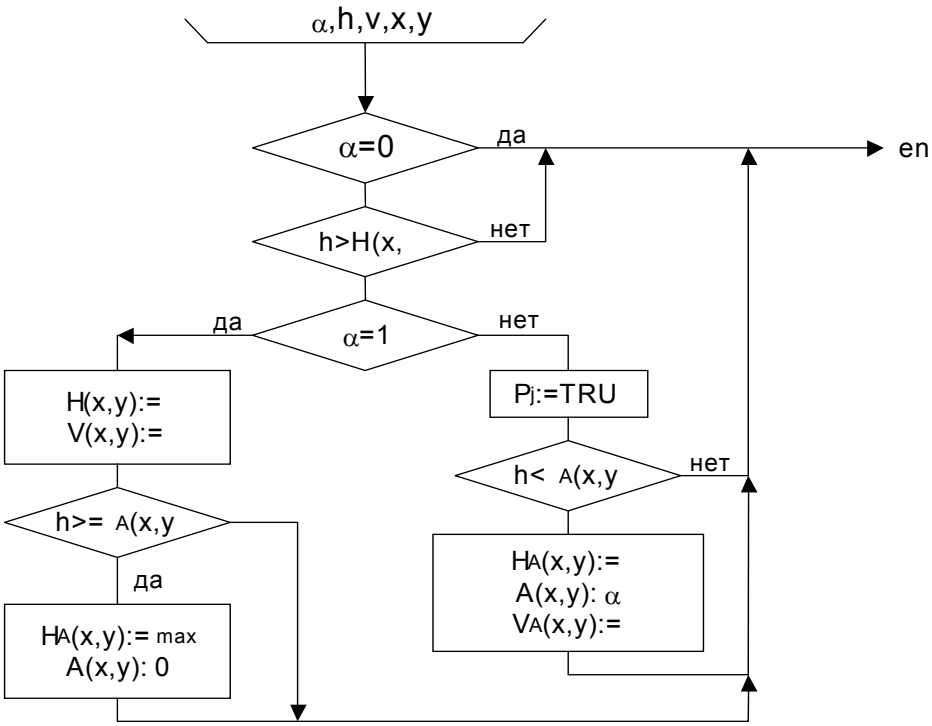
Полупрозрачнсть:

H – обычный буфер
A – α– буфер
V – поле изображения
H_A – вспомогательный буфер для полупрозрачной грани
V_A – яркость полупрозрачной грани
P_j – признак обработки j- ой грани (j = [0;m-1], если m граней)

Алгоритм:



Алгоритм обработки грани – сводится к обработке текущей точки:



Алгоритм учета полупрозрачности - перебор всех точек изображения - для каждой точки с координатами x,y:

Циклические

Допустим, что нам необходимо изобразить поверхность моря. Можно взять большую структуру на всю поверхность моря, но это очень громоздко и сложно. Вместо этого можно взять небольшой фрагмент и составить поверхность из нескольких таких фрагментов (т.е. размножить исходный текстурный фрагмент). При этом необходимо, чтобы вертикальные стороны были абсолютно одинаковыми.

Динамические

Как можно показать, что море волнуется? У нас есть несколько текстур моря, и мы генерируем изображение с учетом изменения текстур. Т.е. в первом кадре накладывают первую текстуру, в следующем вторую и т.д. (каждый кадр берёт текстуру из своего файла).

Текстуры с мультиразрешением

Рассмотрим текстуру шахматного поля:

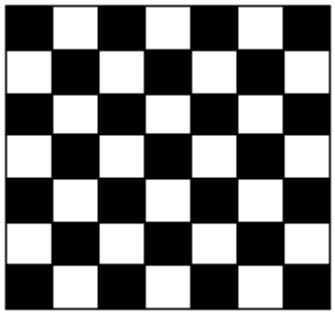


Рис 4.4.5.1

При большом удалении мы будем иметь чередование черных и белых точек, и скорее всего нарушится порядок клеток шахматного поля. На самом деле при большом удалении мы должны наблюдать серый фон, следовательно, надо иметь несколько текстур с разным расширением. Все поле видимости делится на несколько областей, которые нумеруются .Для каждой области удаления используется своя текстура – чем ближе она расположена к наблюдателю, тем большее разрешение имеет. Оценив расстояние до объекта, надо использовать либо текстуру для малых удалений, либо – для больших удалений.

Трилинейная интерполяция:

Суть: использование текстур в зависимости удаления от наблюдателя P (то есть дистанция D):

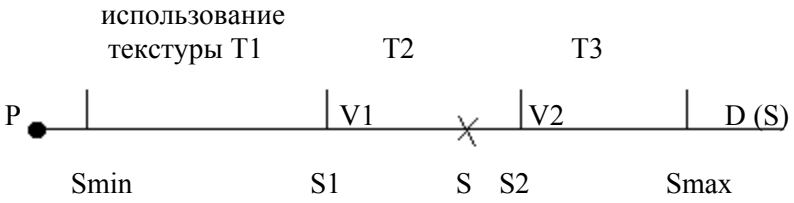
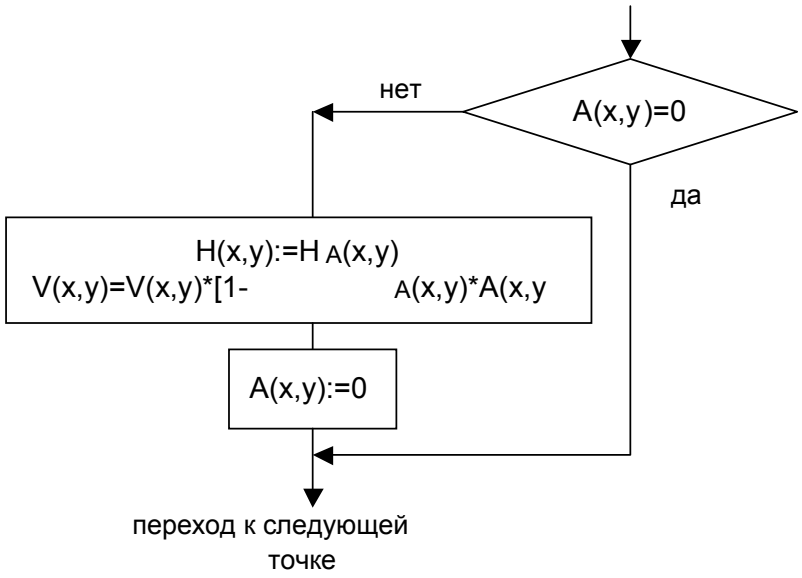


Рис 4.4.5.2

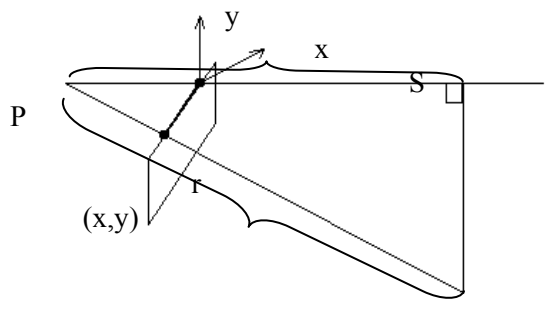
Проще работать с глубиной отрезка S, но корректнее с D.



В точках S1 и S2 при нетрилинейной интерполяции будет происходить резкий скачок из T2 в T3. Если мы используем трилинейную интерполяцию, то эти переходы будут плавными, и в точке S будет среднее между T2 и T3, причем в большей степени будет содержание T3:

$$V = V1 + \frac{S-S1}{S2-S1} * (V2 - V1)$$

, при расчете V1 и V2 используется билинейная интерполяция



главный луч проектирования

R Начало координат находится в центре плоскости изображения

D

Рис 4.4.5.3

$$\frac{F}{r} = \frac{S}{R}$$

$$D = \sqrt{S^2 + R^2} \qquad r = \sqrt{x^2 + y^2}$$

Отсюда:

$$D = S * \sqrt{1 + \frac{x^2 + y^2}{F^2}}$$

Вычисляем h = H(x,y), а затем – глубину S = A / (h + B).
Для пространственных координат получим:

$$\left(\frac{-x}{F} \quad \frac{-y}{F} \quad -S^{-1} \quad 1 \right) = \left(X-X_p \quad Y-Y_p \quad Z-Z_p \quad 1 \right) * Q$$

$$\Rightarrow \left(X-X_p \quad Y-Y_p \quad Z-Z_p \quad 1 \right) = \left(\frac{-x}{F} \quad \frac{-y}{F} \quad -S^{-1} \quad 1 \right) * Q^T$$

Величина S вычисляется через буфер глубины: S = A / (H(x,y) + B).

Проективные текстуры

Рассмотрим общий случай, когда текстура проецируется на поверхность, которая затем проецируется на 2-х мерный экран. Мы проецируем проектором некое изображение на поверхность, а затем смотрим на нее из произвольной точки (см. рис.1). Т.е. снова проецируем изображение, на этот раз уже с поверхности на наблюдателя. При построении изображения эта ситуация моделируется крайне просто - проекция примитивов поверхности на экран дело обычное, а роль второй проекции (проецирование изображения на поверхность) играет привязка соответствующего места текстуры с изображением на примитивы.

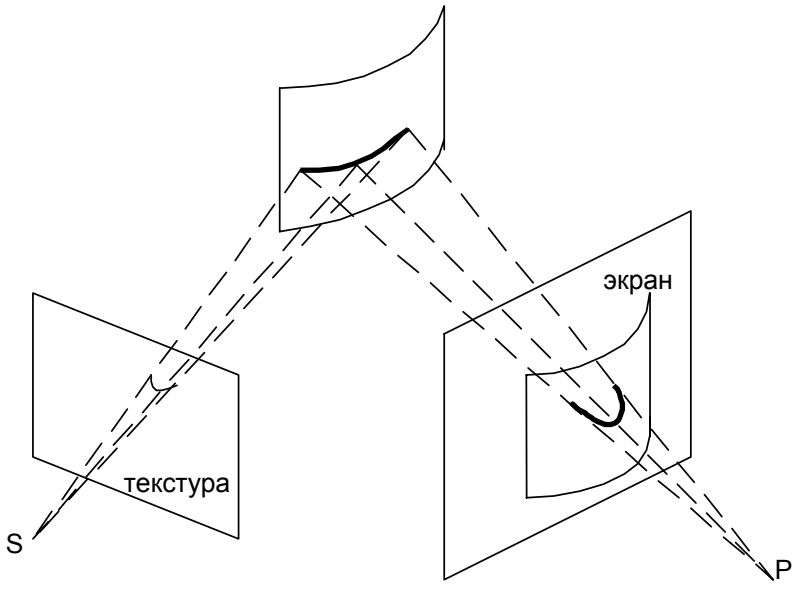


Рис5.1

Нам осталось лишь научиться правильно привязывать текстуру с исходным изображением к нашей поверхности. Всего мы имеем дело с четырьмя координатными системами.

1. Наблюдательская система ("clip" или "projection") - является обычным для графики 4-х координатным представлением 3-х мерного (объемного) пространства. Координаты зовутся x, y, z, w. Начало этой координатной системы лежит в точке наблюдения.
2. Экранная система ("screen") - 2-х мерный экран, который и видит наблюдатель. Эти координаты получаются из наблюдательской системы путём деления x и y на w – $x^5 = x / w$, $y^5 = y / w$, (индекс "s" у получающихся координат обозначает экранную систему).
3. Система источника света ("light") - это вторая объемная система координат (x^t, y^t, z^t и w^t,). В начале этой системы координат находится источник света.
4. Текстурная система (texture) - координаты на плоскости проецируемой текстуры (тот слайд, сквозь который светит воображаемый источник света). Текстурные координаты получаются как $x^t = x^t / w^t$, $y^t = y^t / w^t$ (также можно вычислить $z^t = z^t / w^t$, если мы решили не ограничиваться плоской текстурой).

Наша задача: имея точку (x^5, y^5) на экране, нам необходимо найти соответствующую ей точку (x^t, y^t) на текстуре.

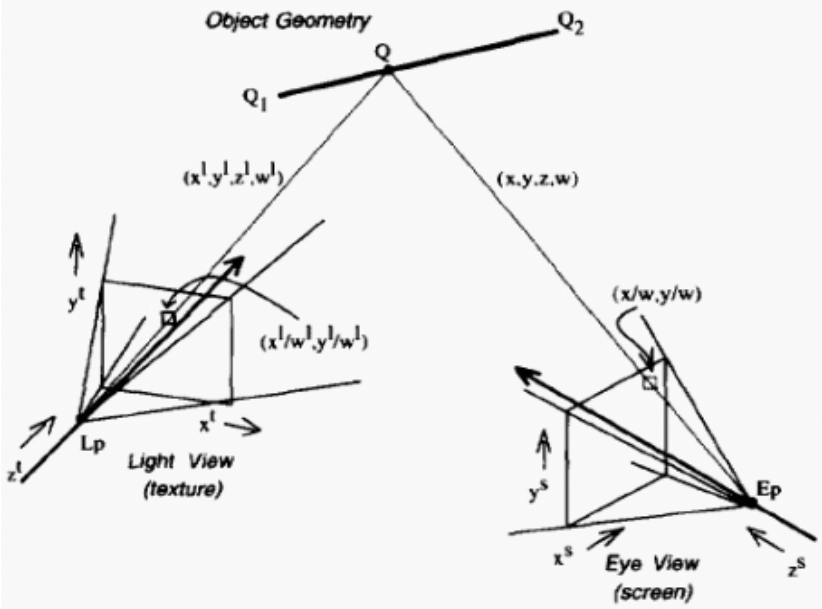


Рис 5.2

На рис. 5.2 показан сегмент линии в нашем трехмерном пространстве и его проекция на 2-х мерный экран. Этот сегмент - горизонтальная полоса сканирования на экране, расположенная между двумя рёбрами полигона. Координаты его концов в наблюдательской системе:

$$Q_1 = (x_1, y_1, z_1, w_1) \text{ и } Q_2 = (x_2, y_2, z_2, w_2)$$

Произвольная точка Q на этом отрезке может быть задана параметрическим образом:

$$Q = (1-t)Q_1 + tQ_2 \qquad (1)$$

для $t \in [0,1]$. В экранной системе координат (*screen)* эта точка задется следующим образом:

$$Q^s = (1-t^s)Q_1^s + t^s Q_2^s \qquad (2)$$

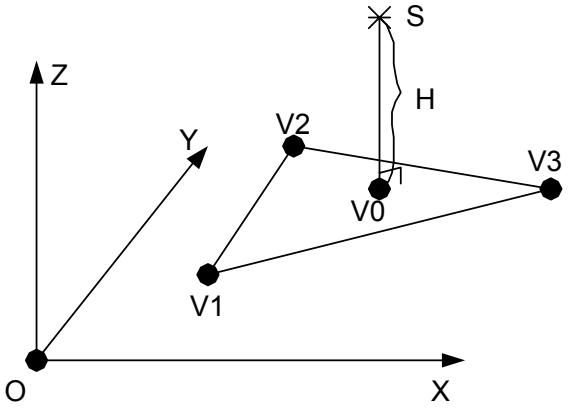
где

$Q_1^s = Q_1/w_1$ и $Q_2^s = Q_2/w_2$ (общее правило преобразования из наблюдательской системы в экранную)

Нам необходимо найти координаты нашей произвольной точки отрезка в координатной системе источника света. Будем считать, что, так или иначе, мы уже определили координаты концов отрезка в системе источника света. Для начала нам необходимо найти параметр t , соответствующий t^5 (в общем случае экранное t не равно t^5 наблюдательскому). Для этого запишем

$$Q^s = (1-t^s)Q_1/w_1 + t^s Q_2/w_2 = \frac{(1-t)Q_1 + tQ_2}{(1-t)w_1 + tw_2} \qquad (3)$$

и решим относительно t . Для тех, кому интересно, приведем все рассуждения:



Вычисление t:
Зададим a и b, таким образом, что $1 - t^5 = a / (a + b)$ и $t^5 = b / (a + b)$.
Зададим A и B так, что $t = A / (A + B)$ и $t = B / (A + B)$.
Тогда:

$$Q^s = \frac{aQ_1/w_1 + bQ_2/w_2}{(a+b)} = \frac{AQ_1 + BQ_2}{Aw_1 + Bw_2} \qquad (4)$$

Легко проверить, что $A = aw_1$ и $B = bw_2$ удовлетворяют этому уравнению, позволяя нам получить искомый параметр t , и, далее, координаты Q.
Продолжим. У нас есть матрица M, переводящая координаты из системы источника в наблюдательскую:

$$Q^l = MQ = \frac{A}{A+B}Q_1^l + \frac{B}{A+B}Q_2^l \qquad (5)$$

где $Q_1^l = \left(x_1^l, y_1^l, z_1^l, w_1^l\right)$ и $Q_2^l = \left(x_2^l, y_2^l, z_2^l, w_2^l\right)$ координаты в системе источника света точек Q_1 и Q_2 (концы нашего отрезка) из наблюдательской системы. В итоге мы получаем:

$$Q^t = Q^l/w^l = \frac{AQ_1^l + BQ_2^l}{Aw_1^l + Bw_2^l} = \frac{aQ_1^l/w_1 + bQ_2^l/w_2}{a\left(w_1^l/w_1\right) + b\left(w_2^l/w_2\right)} \qquad (6)$$

Уравнение (6) выражает координаты на поверхности текстуры, соответствующие любой точке сегмента выбираемой (линейно интерполируемой) параметром t в экранных координатах.
Для того, чтобы получить координаты, мы должны линейно интерполировать x^t/w, y^t/w, w^t/w.
Для каждого пикселя:

$$x^t = x^l/w^l, y^t = y^l/w^l, \text{ при этом } x^l/w^l = \frac{x^l/w}{w^l/w} \text{ и } y^l/w^l = \frac{y^l/w}{w^l/w} \qquad (7)$$

Если w^t постоянна на всём полигоне, то уравнение (7) приобретает вид

$$s = \frac{s/w}{1/w} \text{ и } t = \frac{t/w}{1/w} \qquad (8)$$

откуда мы имеем $s = x^t/w^t$, $t = y^t/w^t$. Здесь (s,t) - текстурные координаты, синонимы (x^t, y^t).

Уравнение (8) и определяет текстурные координаты, которые можно привязать к вершинам передаваемого на ускоритель полигона. В более общем сложном случае проективной текстуры, выражаемом уравнением (7), требуется деление на w^t/w, а не на $1/w$.

Рельефные текстуры.

Рельефное текстурирование очень напоминает обычный процесс наложения текстуры на полигон. Только при обычном наложении текстуры мы работаем со цветом и изменяем его цветовое восприятие, а вот при рельефном текстурировании мы добавляем ощущение рельефа, объёмности плоскому полигону. Рельефное текстурирование отражает реальное положение источника света в сцене и даже изменение его местоположения.

Теперь рассмотрим мировую систему координат, в которой мы имеем следующий треугольник (имеет рельефную текстуру):

S – источник света;

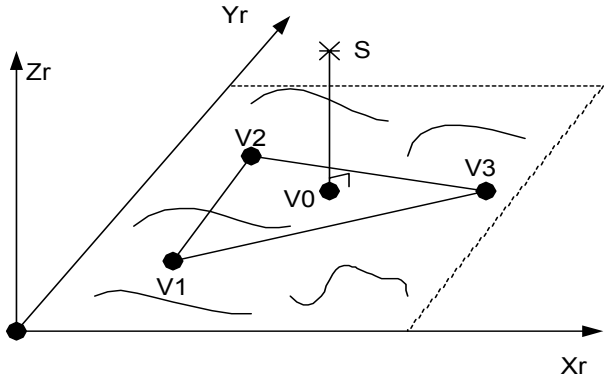
$V_1(X_1, Y_1, Z_1, x_{r1}, y_{r1})$

$V_2(X_2, Y_2, Z_2, x_{r2}, y_{r2})$

$V_3(X_3, Y_3, Z_3, x_{r3}, y_{r3})$,

где x_r, y_r - координаты связанные с рельефным полем (поле нормалей).

Наша главная задача состоит в том чтобы найти координаты точки S,а так же найти яркость для каждой точки треугольника. Для этого мы переходим в следующую систему координат (т.е. в рельефное поле).



Где:

$S(x_{r0}, y_{r0}, H); V_1(x_{r1}, y_{r1}, 0); V_2(x_{r2}, y_{r2}, 0); V_3(x_{r3}, y_{r3}, 0)$

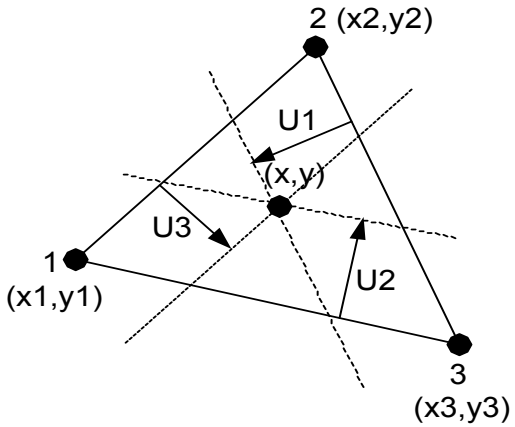
Воспользуемся следующими формулами:

$N_V = (V_3 - V_1) \cdot (V_2 - V_1)$

$N = \frac{N_V}{|N_V|}$

$H = N \cdot (S - V_1)$

$V_0 = S - N \cdot H$



Используем относительные координаты точки в пределах треугольника:

Относительные координаты:

точка (x,y) будет характеризоваться: $U_1 U_2 U_3$

Для любой точки принадлежащей этому треугольнику:

$U_1 + U_2 + U_3 = 1$

$$\begin{cases} U_1 = 1 - U_2 - U_3 \\ U_2 = \dfrac{(x_3 - x_1)(y - y_1) - (x - x_1)(y_3 - y_1)}{(x_3 - x_1)(y_2 - y_1) - (x_3 - x_1)(y_3 - y_1)} \\ U_3 = \dfrac{(x - x1)(y_2 - y_1) - (x_2 - x_1)(y - y_1)}{(x_3 - x_1)(y_2 - y_1) - (x_3 - x_1)(y_3 - y_1)} \end{cases}$$

При обратном пересчёте:

$$x = x_1 \cdot U_1 + x_2 \cdot U_2 + x_3 \cdot U_3$$
$$y = y_1 \cdot U_1 + y_2 \cdot U_2 + y_3 \cdot U_3$$

Определим для точки V_0 относительные координаты через её пространственные координаты:

$$\begin{cases} U_2 = \dfrac{(X_3 - X_1)(Y_0 - Y_1) - (X_0 - X_1)(Y_3 - Y_1)}{(X_3 - X_1)(Y_2 - Y_1) - (X_2 - X_1)(Y_3 - Y_1)} \\ U_3 = \dfrac{(X_0 - X_1)(Y_2 - Y_1) - (X_2 - X_1)(Y_0 - Y_1)}{(X_3 - X_1)(Y_3 - Y_1) - (X_2 - X_1)(Y_3 - Y_1)} \\ U_1 = 1 - U_2 - U_3 \end{cases}$$

Координаты точки в рельефной системе:

$$x_{r0} = x_{r1} \cdot U_1 + x_{r2} \cdot U_2 + x_{r3} \cdot U_3$$
$$y_{r0} = y_{r1} \cdot U_1 + y_{r2} \cdot U_2 + y_{r3} \cdot U_3$$

Алгоритм прорисовки:

1) 1) пересчёт координаты в рельефном поле;
2) 2) при закраске интерполяция (нелинейная) рельефных координат.

По рельефным координатам просчитываем нормаль, плюс имея расстояние до S высчитываем угол между векторами $\vec{S}$ и $\vec{N}$, следовательно имеем яркость точки.

Учёт освещения:

$$V = V_t \cdot \frac{V_{Ламберга}}{V_{\max}}$$