4. Нанесение текстур

Различают два вида текстур:

- Процедурные
- Проективные (наносятся на грань объекта)

4.1 Процедурные текстуры

Рассмотрим простой пример - 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.

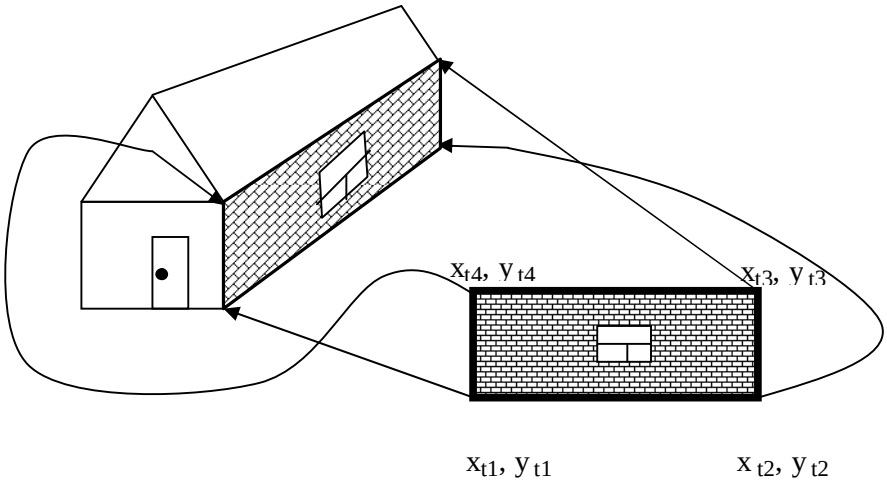


Рис. 4.1.1

индекс t означает «текстурный»

В ряде случаев могут получаться искажения

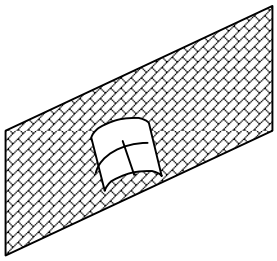


Рис. 4.1.2

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.

4.2. Коррекция текстуры

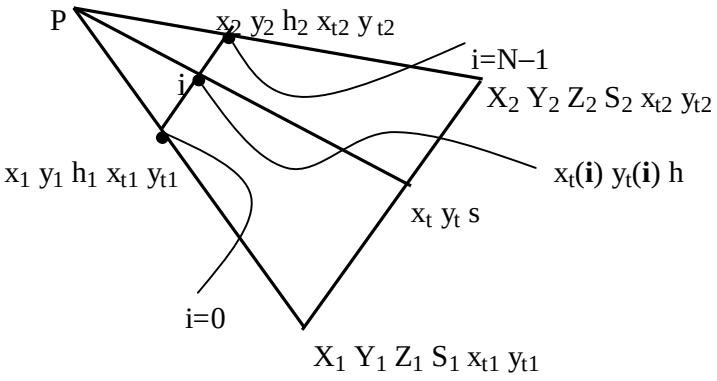


Рис. 4.2

Для i-ой точки :

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

Линейная интерполяция: $S = \dfrac{A}{h + B}$;

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

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:

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

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

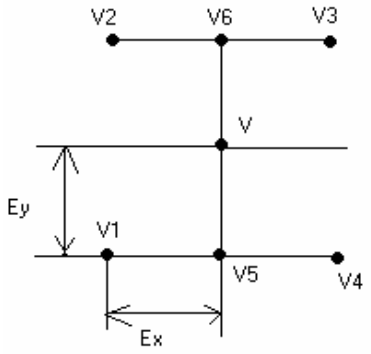
$$\Delta fx = \frac{(x_{t2} - x_{t1})(h_2 + B)}{N-1}; \quad \Delta fy = \frac{(y_{t2} - y_{t1})(h_2 + B)}{N-1};$$

Тогда будем иметь:

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

4.3 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.



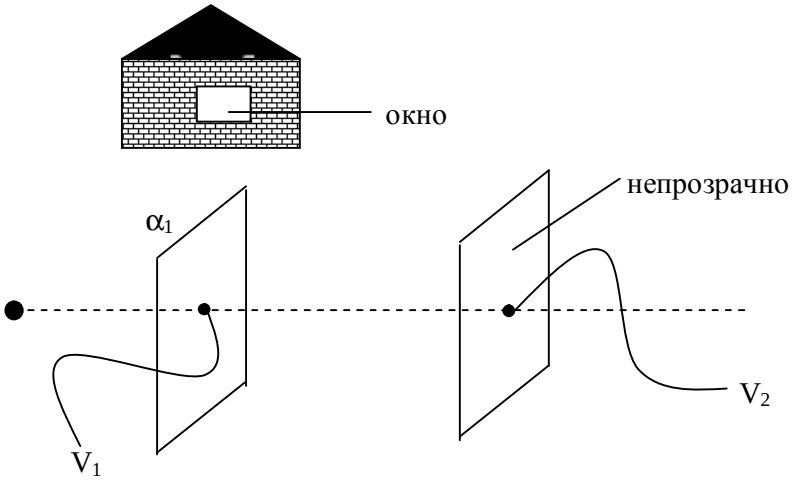
V5=V1*(1-Ex) +V4*Ex
V6=V2*(1-Ex) +V3*Ex
V = V5*(1-Ey) +V6*Ey
Ex, Ey∈ {0:1}

Рис. 4.3.1

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

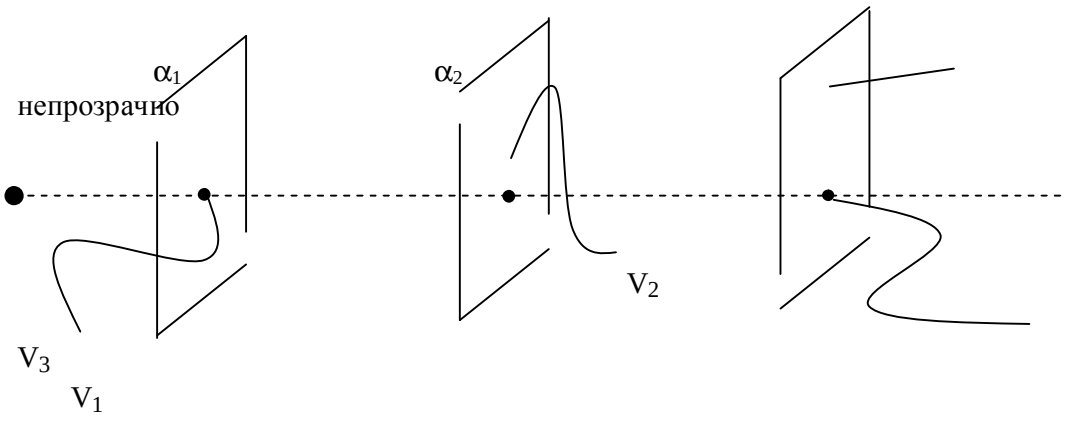


Рис. 4.3.3

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

4.4 Виды текстур

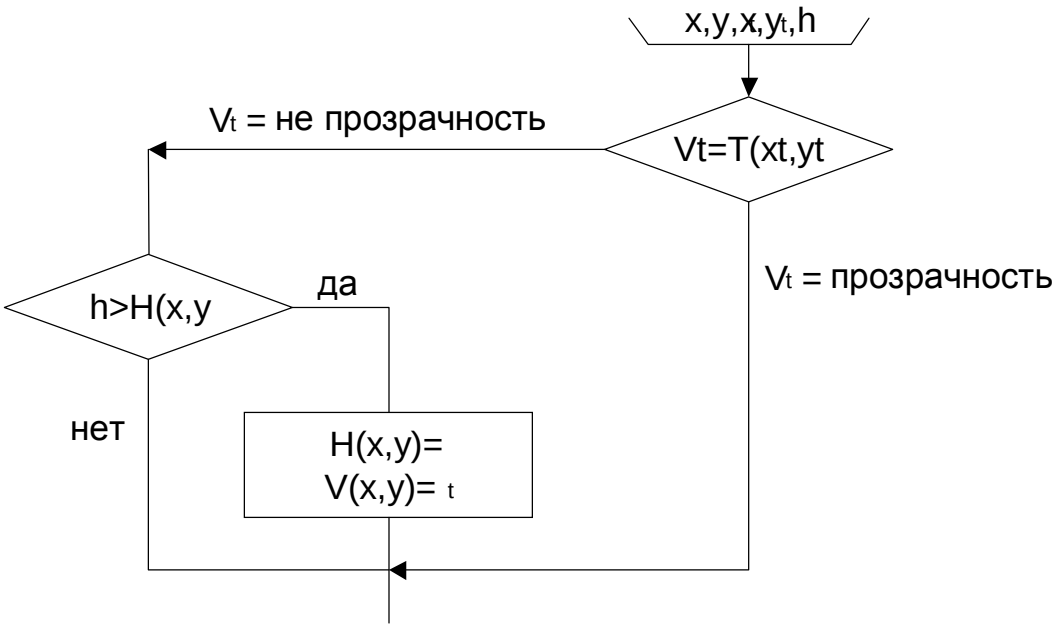
1.текстуры с мультиразрешением (мультиразрешение –представление с различной степенью детализации)

1. прозрачные текстуры
2. полупрозрачные текстуры
3. текстуры в тумане
4. циклические текстуры
5. динамические текстуры

6. рельефные текстуры

Прозрачные

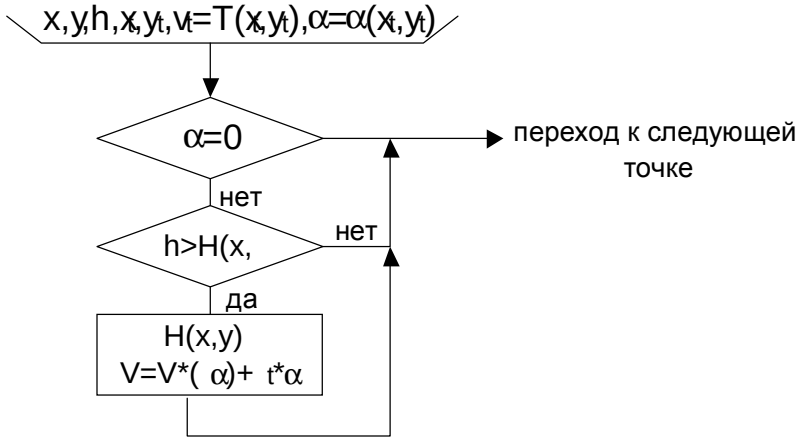
Для получения прозрачной текстуры воспользуемся следующим методом: зарезервируем один код V_t под признак прозрачности, то есть эта точка не будет заноситься в буфер изображения. Нижеприведенный алгоритм отображает обработку одной точки при использовании прозрачной текстуры:



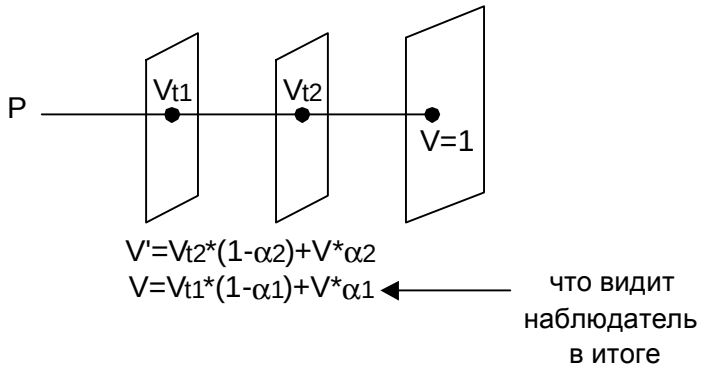
Полупрозрачные

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:



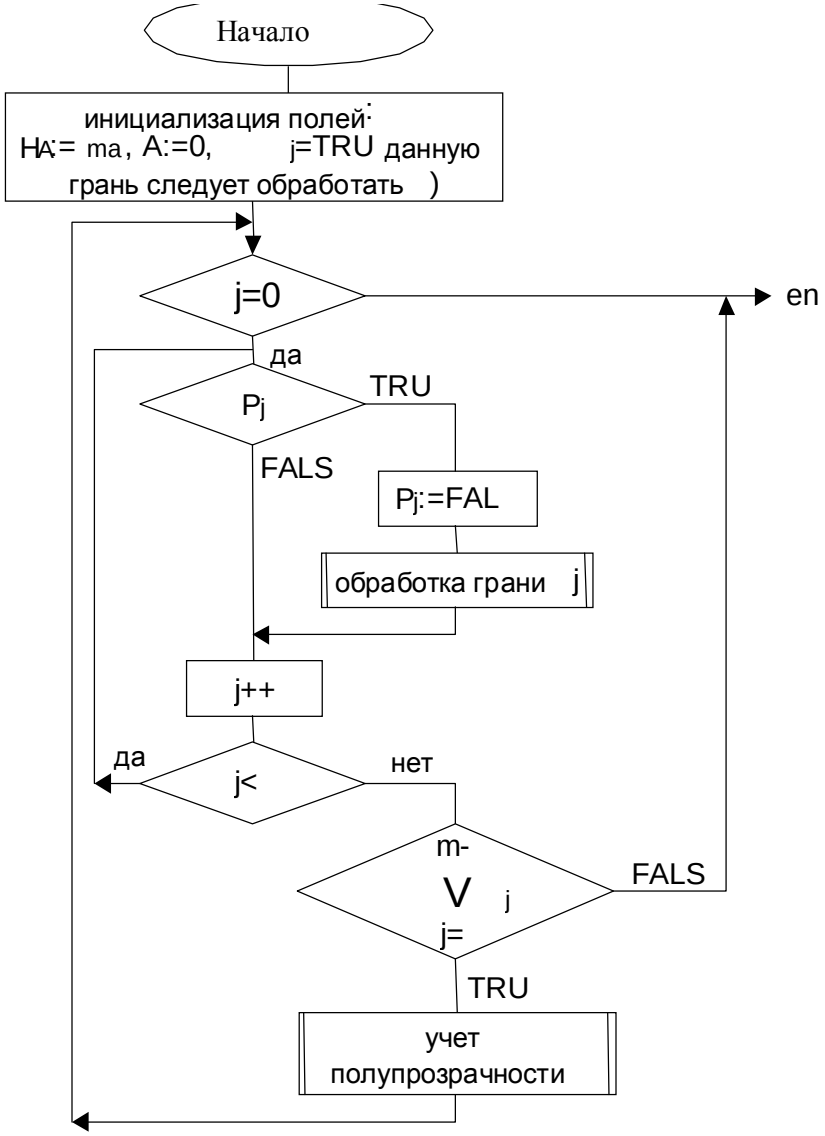
Наложение двух полупрозрачных граней:



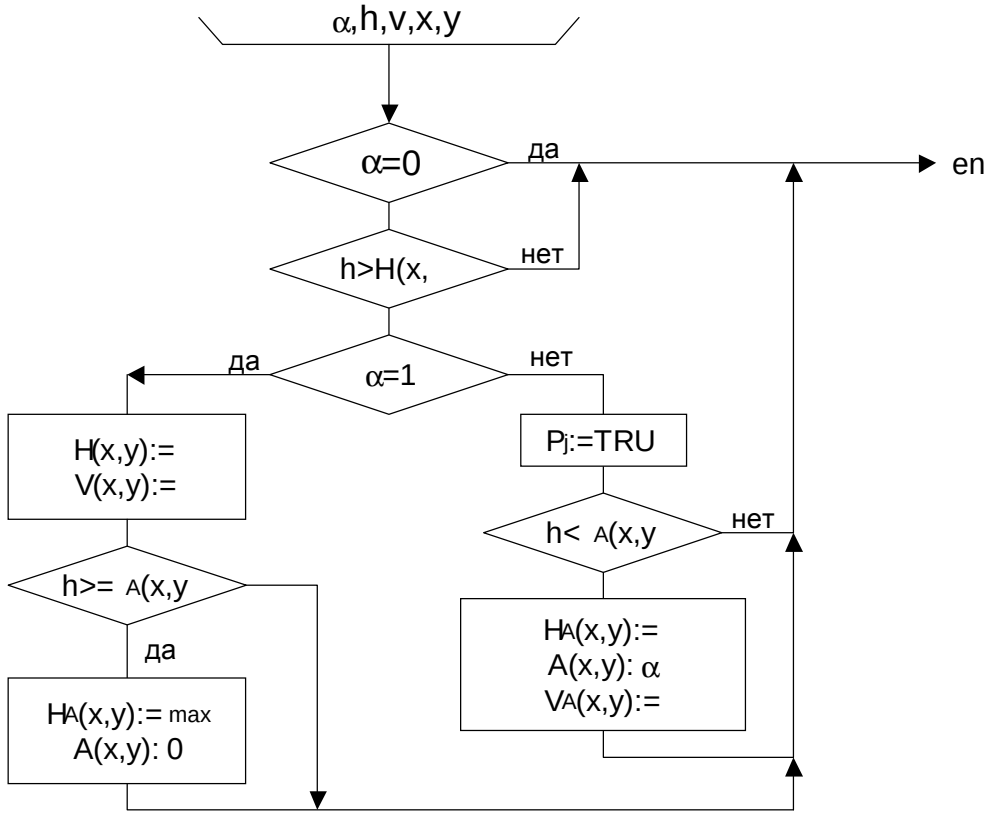
Полупрозрачнсть:

H – обычный буфер
A – α– буфер (0...1=α, 0- прозрачно, 1- не прозрачно)
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)
H:=0; V:=0 - фон

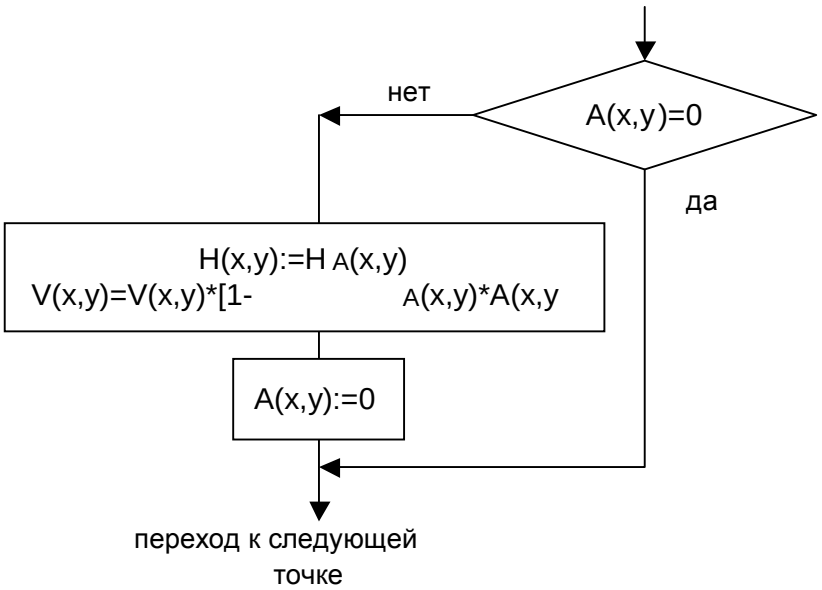
Алгоритм:



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 былиабсолютноодинаковыми.

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:

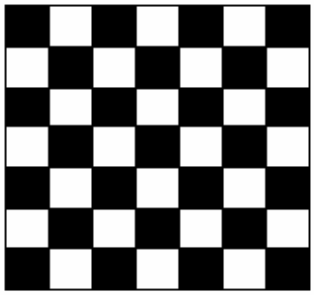


Рис 4.4.1

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):

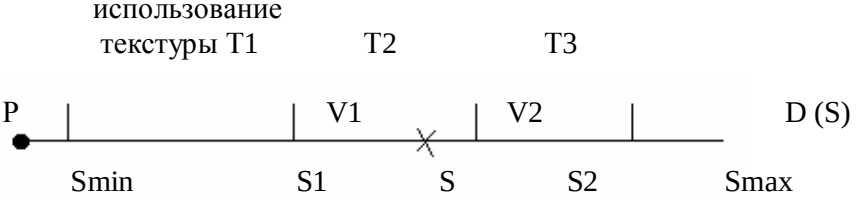


Рис 4.4.2

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

Начало координат находится в центре плоскости изображения

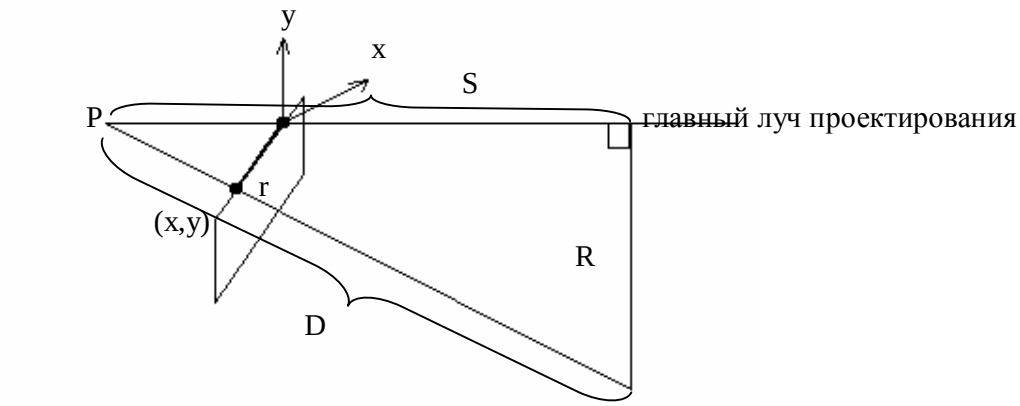


Рис 4.4.3

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

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

Отсюда:

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

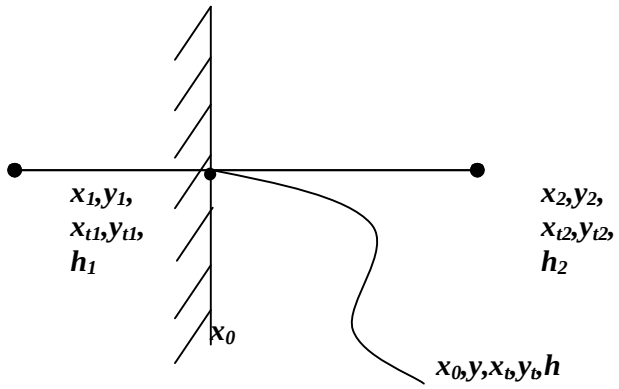
Вычисляем h = H(x,y), а затем – глубину S = A / (h + B).
Для пространственных координат получим:

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

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

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

Отсечение текстурных координат по полю вывода



$$h = h_1 + (h_2 - h_1)\frac{x_0 - x_1}{x_2 - x_1}; \qquad S = \frac{A}{h+B};$$

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

$$\begin{cases} x_t = x_{t1} + (x_{t2} - x_{t1})\dfrac{x_0 - x_1}{x_2 - x_1}\dfrac{h_2 + B}{h + B} \\ y_t = y_{t1} + (y_{t2} - y_{t1})\dfrac{x_0 - x_1}{x_2 - x_1}\dfrac{h_2 + B}{h + B} \end{cases}$$

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.

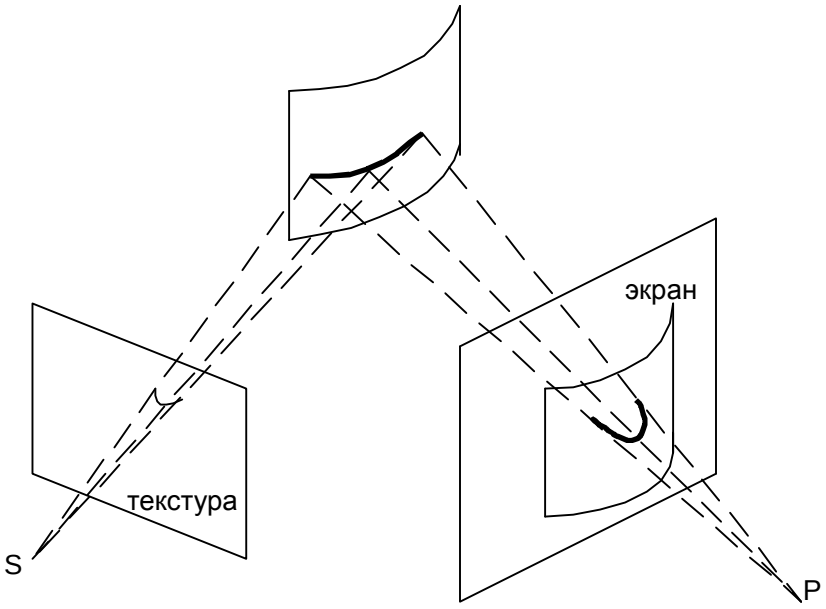


Рис 4.4.4

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.

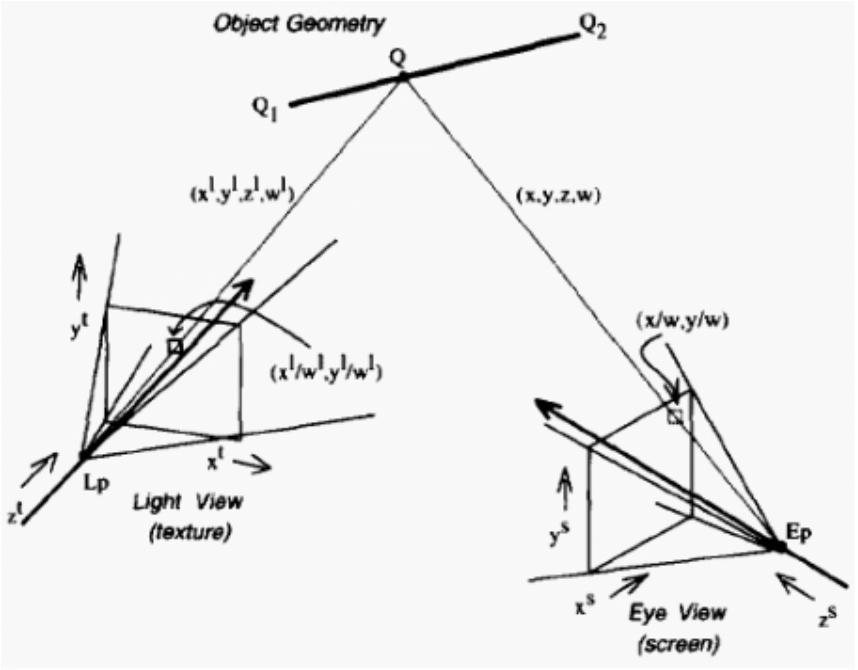


Рис 5.2

На рис. 4.4.5 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:

$Q_1 = (x_1, y_1, z_1, w_1)$ 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 $1/w$.

4.5 Туман, тени

Туман

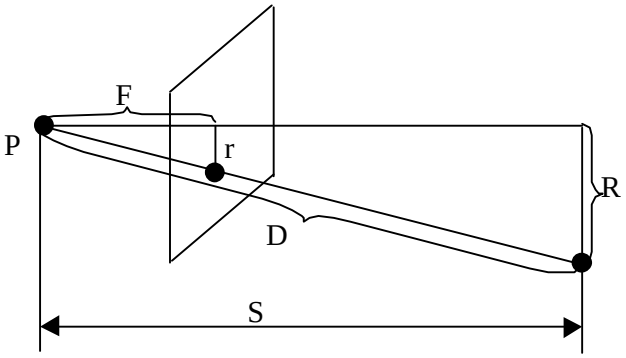
Туман используется для создания атмосферных эффектов. Туман используется для создания дымки и скрытия удаленных объектов. В первом случае повышается реалистичность сцены, во втором – понижается ее сложность. С туманом объекты могут рендериться с разной степенью детализации в зависимости от расстояния до наблюдателя. Туман работает по

принципу: чем дальше объект, тем больше туман его поглощает. Поэтому для удаленных объектов разумно использовать меньше полигонов, чем для близких.

Туман можно разделить на **полигонный** и **пиксельный**. Полигонный метод линейно интерполирует уровень тумана по значениям в вершинах для получения уровня тумана в каждой точке полигона. Этот метод хорош только для маленьких полигонов. Пиксельный метод рассчитывает уровень тумана для каждого пиксела, и для больших полигонов дает более реалистичное изображение.

Рассмотрим пиксельный туман. Туман также можно разделить и по другому признаку – на **линейный** и **экспоненциальный** (или табличный). При линейном тумане степень поглощения объекта туманом линейно зависит от расстояния до наблюдателя, а при экспоненциальном тумане – рассчитывается на основании таблицы.

Расстояние до точки.



D – расстояние от наблюдателя (P) до точки (или дистанция).

Рис 4.5.1

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

отсюда получаем

$$D=\sqrt{S^2+R^2}$$

Итак,

$$D=S\cdot\sqrt{1+\left(\frac{r}{F}\right)^2}$$

где $r=\sqrt{x^2+y^2}$

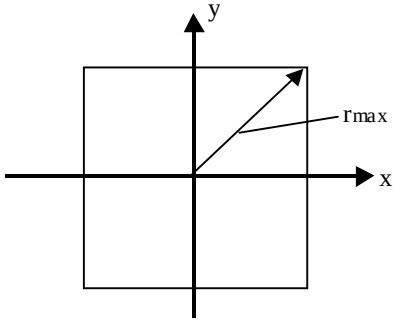
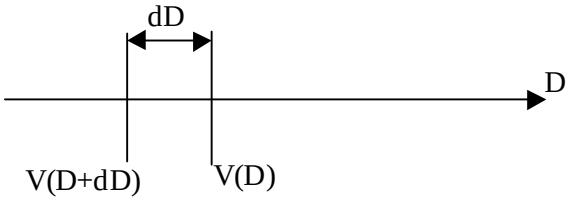


Рис 4.5.2

Подставляя в формулу получаем уравнение расстояния до точки:

$$D=S\cdot\sqrt{1+\frac{x^2+y^2}{F^2}},\qquad \text{где } S=\frac{A}{h+B}$$



V(D) – яркость в данной точке
dD – расстояние, на которое добавляем туман

V(D+dD) – яркость в точке с учетом добавленного тумана
Формула яркости в точке с учетом добавленного тумана.

$V(D+dD) = V(D) \cdot (1 - s \cdot dD) + s \cdot dD \cdot V_T$,

где Vт – яркость тумана

σ - плотность тумана

$dV = V(D+dD) - V(D) = -s \cdot VdD + s \cdot V_T dD$

решим дифференциальное уравнение

$dV = s\,VdD + s\,V_T dD$ отсюда получим

$V = V_T + C \cdot e^{-sD}$

В нулевой точке, то есть в точке, где находится наблюдатель тумана нет, поэтому получаем:

$V(0) = V_0$, тогда получим: $V = V_T + (V_0 - V_T) \cdot e^{-sD}$

$s \equiv \frac{1}{D_T}$, где Dт – дистанция тумана

Получаем уравнение вычисления тумана:

$V = V_T + (V_0 - V_T) \cdot e^{-\frac{D}{D_T}}$. На дистанции Dт туман увеличивается в e раз.

Примем соглашение, что $S\max >\approx 3 \cdot D_T$, так как отображать объекты, находящиеся на расстоянии больше, чем 3Dт, нет смысла (из-за тумана не будет видно).

Экспоненциальный туман.

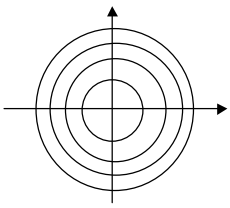
Значение функции $e^{-\frac{D}{D_T}}$ можно вычислять таблично:

$e^{-\frac{D}{D_T}} = e^{-\frac{A}{(h+B)\cdot D_T}\cdot\sqrt{1+\frac{r^2}{F^2}}} \equiv \mathrm{E}(h, r^2)$

h – по старшим разрядам, например, 6 разрядов $A_h = \frac{h}{h_{\max}} \cdot (2^6 - 1)$

r – делим на зоны, r = $r_0 \ldots r_{max}$, например, на 16 зон (4 разряда)

$A_r = \frac{r}{r_{\max}} \cdot (2^4 - 1) \quad r_{\max} = \sqrt{x_{\max}^2 + y_{\max}^2}$



Итого, например, получаем 10-ти разрядную таблицу.

Тени

В компьютерной графике реального времени часто объекты изображаются без теней, что приводит к тому, что объект как бы не закреплён в окружающей его обстановке. Тень несёт очень много информации - фактически, она представляет объект с другой точки зрения и закрепляет его в сцене.

Тени бывают двух типов:

Первый:

При построении тени учитывается знак скалярного произведения нормали к грани и направления источника света.

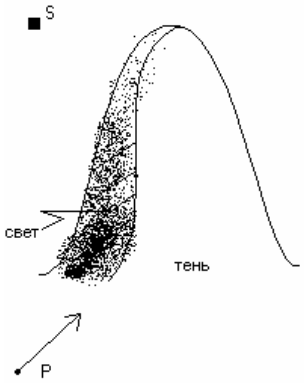


Рис 4.5.3

На рис 4.5.3 представлены следующие обозначения, которые будут использоваться в дальнейшем:

s – точечный источник света.

p – положение наблюдателя.

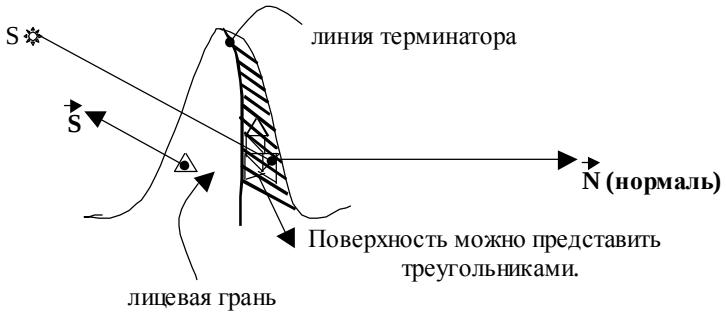


Рис 4.5.4

$\vec{S}$ – вектор, направленный на S.

Наблюдатель увидит линию терминатора – границу раздела света и тени.

Если угол между $\vec{S}$ и $\vec{N} > 90$, то источник света падает не на лицевую грань.

Второй: Для построения используются алгоритм теневого объема, алгоритм дополнительного буфера глубины.

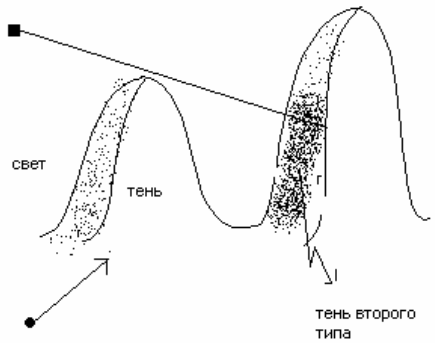


Рис 4.5.5

Тень второго типа – тень, отбрасываемая одним объектом на другой.

Суть алгоритма доп. буфера глубины:

Наблюдателя помещаем в S (некоторого вспомогательного наблюдателя). Сначала строим вспомогательный буфер глубины для вспомогательного наблюдателя. Затем производится сравнение точки с

заданными координатами (построенной в пространственных координатах) с координатами, видных наблюдателю точек.

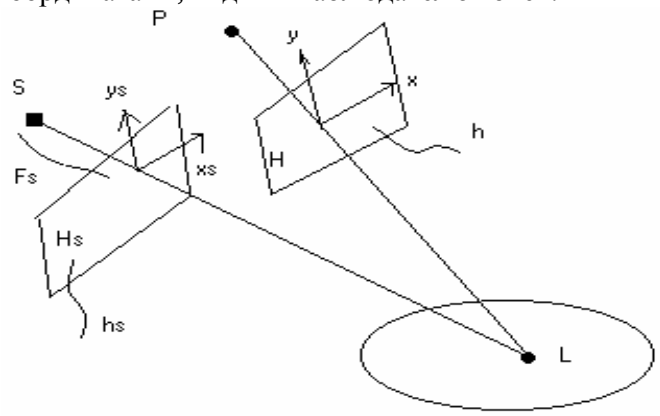


Рис 4.5.6

При построении второго буфера глубины используем ту же точку фиксации взгляда. (куда смотрим мы, туда и вспомогательный наблюдатель.)

1 Этап – построение Hs.

2 Этап – построение H,V.

Обработка каждой текущей точки будет производиться в соответствии со следующим алгоритмом. (когда мы идём по сканированной строке). Для каждой текущей точки в плоскости изображения известно: x, y – координаты в плоскости изображения, h - текущее значение параметра глубины, а так же U - яркость, U_0 -яркость для точки в тени.Заполняется буфер для наблюдателя (S). Теперь для ответа на вопрос видна ли точка с координатами x,y,h пересчитываем координаты и помещаем в H_S -> x_s, y_s, h_s. Обращаемся в H_S (x_s, y_s), где по результату сравнения (условие (**) на рис 4.5.7) проверяем - освещена точка или нет. Если условие не выполняется, то точка не освещена, и имеет место тень второго типа, иначе точка видна по отношению к наблюдателю, находящемуся в источнике света.

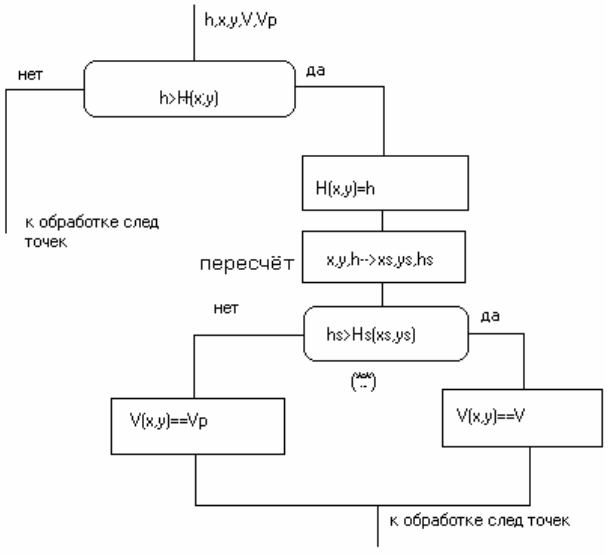


Рис 4.5.7

Пересчет точек из одной системы координат в другую осуществляется следующим образом.

$$\left(\frac{-X}{F}\ \frac{-Y}{F}\ -S^{-1}\ 1\right) = (x - x_p\quad y - y_p\quad z - z_p\quad 1) * Q_p$$

$$\left(-\frac{X_s}{F_s}\ \frac{-Y_s}{F_s}\ -S_s^{-1}\ 1\right) = (x - x_s\quad y - y_s\quad z - z_s\quad 1) * Q_s$$

где:

$$Qp = \begin{vmatrix} a1 & a2 & 0 & a3 \\ b1 & b2 & 0 & b3 \\ c1 & c2 & 0 & c3 \\ 0 & 0 & 0 & 1 \end{vmatrix} \quad Qs = \begin{vmatrix} a1s & a2s & 0 & a3s \\ b1s & b2s & 0 & b3s \\ c1s & c2s & 0 & c3s \\ 0 & 0 & 0 & 1 \end{vmatrix}$$

Из первого уравнения найдем пространственные координаты.

$$(x - x_p\quad y - y_p\quad z - z_p\quad 1) = \left(\frac{-X}{F}\ \frac{-Y}{F}\ -S^{-1}\ 1\right) * Q^T$$

Чтобы перейти в другую систему координат необходимо осуществить параллельный перенос

$$(x - x_s\quad y - y_s\quad z - z_s\quad 1) = (x - x_p\quad y - y_p\quad z - z_p\quad 1) * T_{sp}$$

$$\text{где } T_{sp} = \begin{vmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ x_p - x_s & y_p - y_s & z_p - z_s & 1 \end{vmatrix}$$

Тогда получим в матричном виде:

$$\left(-\frac{X_s}{F}\ \frac{-Y_s}{F}\ -S_s^{-1}\ 1\right) = \left(\frac{-X}{F}\ \frac{-Y}{F}\ -S^{-1}\ 1\right) * Q^T * T_{sp} * Q_s \quad (*)$$

$$M = T_{sp} * Q^T * Q_s = \begin{vmatrix} d1 & d2 & 0 & d3 \\ l1 & l2 & 0 & l3 \\ g1 & g2 & 0 & g3 \\ f1 & f2 & 0 & f3 \end{vmatrix}$$

Мы выполняем пересчет $h \longrightarrow S$ из формулы $\boxed{S = \frac{A}{h + B}}$ и далее получаем

$$\boxed{h_s = \frac{A}{S_s} - B}$$

Перед этим из выражения (*) мы получаем S_S через S
Для расчета коэффициентов применим следующие формулы:

$$\begin{bmatrix} d1 & d2 & d3 \\ l1 & l2 & l3 \\ f1 & f2 & f3 \end{bmatrix} = \begin{bmatrix} a1 & b1 & c1 \\ a2 & b2 & c2 \\ a3 & b3 & c3 \end{bmatrix} \begin{bmatrix} a1s & b1s & c1s \\ a2s & b2s & c2s \\ a3s & b3s & c3s \end{bmatrix}$$

$$(g1 \quad g2 \quad g3) = (x_p - x_s \quad y_p - y_s \quad z_p - z_s \quad 1) \begin{bmatrix} a1s & a2s & a3s \\ b1s & b2s & b3s \\ c1s & c2s & c3s \end{bmatrix}$$

$$\begin{cases} x_s = -\frac{F_s}{S_s}\left(\frac{S}{F}(x * d1 + y * l1) + g1 - Sf1\right) \\ y_s = -\frac{F_s}{S_s}\left(\frac{S}{F}(x * d2 + y * l2) + g2 - Sf2\right) \\ S = \left(\frac{S}{F}(x * d3 + y * l3) + g3 - Sf3\right) \end{cases}$$

Однако, процесс решения задачи «в лоб» достаточно сложен. Можно использовать другой путь, а именно: свести эту задачу к задаче проективной структуры и воспользоваться формулами для проективных структур.

С помощью текстурирования решается задача закраски объектов, состоящих из отдельных граней, что будет рассмотрено ниже в пункте 5.2.

. В отличие от локального освещения, тени являются одним из эффектов глобального освещения. Т.к. основным местом использования алгоритмов теней в реальном времени являются компьютерные игры, то в дальнейшем, для удобства, будут приводиться примеры именно на них. Существует несколько основных подходов к построению теней, рассмотрим алгоритм «***Преобразование модели "на землю" и отрисовка её как тени***». Фактически, это первый алгоритм построения тени, который был применён в играх. (Turok II, Shogo, etc.). Он отличается простотой реализации и хорошим качеством получаемой тени. Этот алгоритм был впервые описан Джимом Блинном [BLIN88]. В своей статье он описал уравнения для проектирования полигона "на землю", т.е. на плоскость z=0, в направлении от источника света. Он рассмотрел два случая:

1. Источник на бесконечности (параллельный направленный свет)
2. Локальный источник (точечный источник недалеко от объекта)

Этот метод использует геометрическое взаимоотношение источника света и полигона, т.е. подобные треугольники, для вычисления проекции каждого полигона модели "на землю". "Теневые полигоны" должны быть рассчитаны для каждого источника света, т.е. если объект освещается N источниками света, то необходимо рассчитать N его "теневых проекций".

Источник на бесконечности

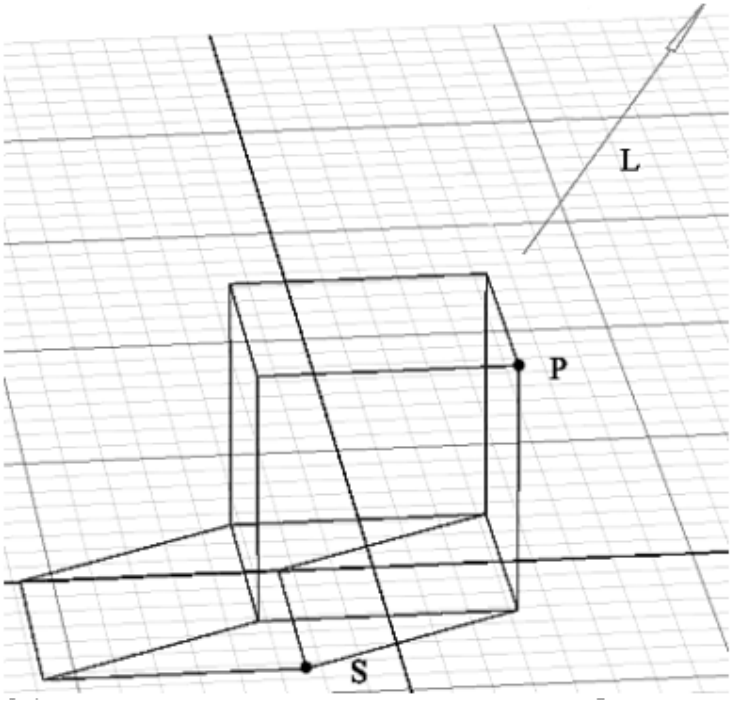


Рис 4.5.7 Параллельная тень от источника света на бесконечности

В случае бесконечно удалённого источника света мы предполагаем, что лучи света, приходящие к объекту, полностью параллельны. Это позволит нам решить уравнение проекции только раз и применять полученное решение ко всем вершинам объекта.

Общая постановка задачи:

Имея точку источника света (x_S, y_S, z_S) и вершину объекта (x_P, y_P, z_P) , мы хотим получить проекцию вершины объекта на плоскость z=0, т.е. точку тени (x_5, y_5, z_5).

Из подобных треугольников получаем:

$$\frac{x_p - x_5}{z_p - z_5} = \frac{x_s - x_p}{z_s - z_p} \qquad (1)$$

решая это уравнение относительно x_5, получаем:

$$x_5 = x_p - (z_p - z_5)\left(\frac{x_s - x_p}{z_s - z_p}\right) \qquad (2)$$

если принять, что L это вектор из точки P к источнику света, то точку S можно выразить как

$$S = P - \alpha L \qquad (3)$$

т.к. мы производим проекцию на плоскость z=0, то уравнение (3) можно переписать в следующем виде: $0 = z_p - \alpha z_s$ (4) или $\alpha = z_p / z_s$ (5)

решая (3) относительно x_5 и y_5, получаем:

$$x_5 = x_p - \frac{z_p}{z_s} * x_s$$
$$y_5 = y_p - \frac{z_p}{z_s} * y_s \qquad (6)$$

или в матричной форме

$$M_5 = \begin{vmatrix} 1 & 0 & -x_s / z_s & 0 \\ 0 & 1 & -y_s / z_s & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{vmatrix} \qquad (7)$$

Теперь имея координаты точки P в мировом координатном пространстве, можно получить её проекцию на плоскость z=0 просто путём умножения на матрицу M_5: $S = M_5 * P$ (8)

Локальный источник

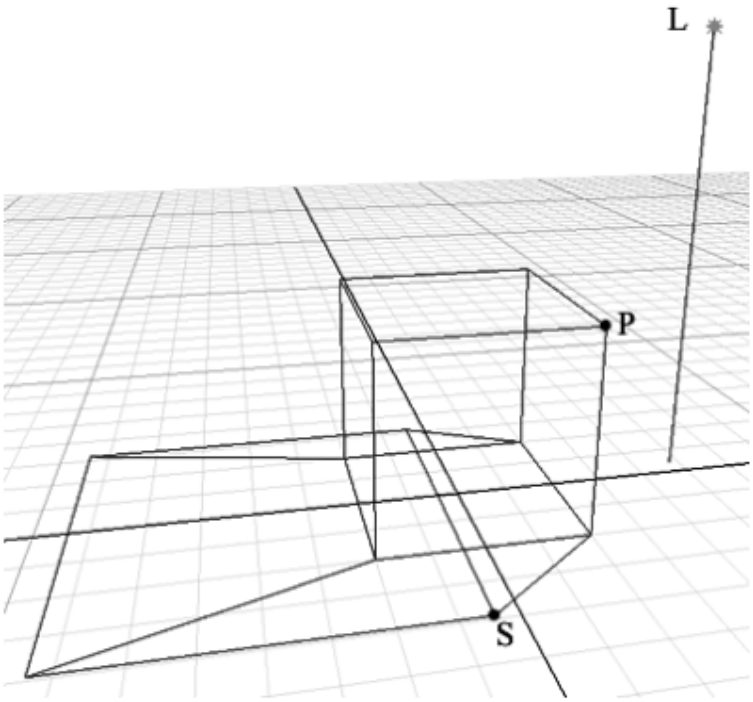


Рис 4.5.8 Перспективная тень от локального источника

Уравнение (6) для бесконечно удалённого источника света может быть обобщено для случая, когда источник света находится на конечном расстоянии от объекта. В этом случае нам понадобятся дополнительные вычисления на каждую вершину, т.к. каждая вершина имеет, в общем случае, своё собственное направление на источник света. Тем не менее, в этом случае мы тоже можем перенести большую часть вычислений в матрицу M_5.

Если L это точка расположения источника света, то (3) принимает вид: $S = P + \alpha(P - L)$ (9)

и снова нам необходимо произвести проекцию на плоскость z=0, т.ч.

$$\alpha = \frac{-z_p}{z_p - z_s} \qquad (10)$$

и

$$x_5 = \frac{x_s z_p - x_p z_s}{z_p - z_s}, \quad y_5 = \frac{y_s z_p - y_p z_s}{z_p - z_s} \qquad (11)$$

Если использовать гомогенизацию после преобразования, то (11) можно записать в виде матрицы

$$M_{5k} = \begin{pmatrix} -z_s & 0 & x_s & 0 \\ 0 & -z_s & 0 & 0 \\ 0 & 0 & -y_s & 0 \\ 0 & 0 & 1 & -z_s \end{pmatrix} \qquad (12)$$

Опять, имея координаты точки P в мировом координатном пространстве, можно записать:

$$S_k = M_{5k} * P \qquad (13)$$

после чего провести гомогенизацию точки S_k для получения проекции точки P на плоскость z=0.

Существует несколько ситуаций, когда ***тени не нужны***:

1. Когда нет источника света.
2. Когда совмещены наблюдатель и источник света.
3. Когда солнце в Зените.

5. Моделирование освещения.

5.1 Основные законы освещения

Закон Ламберта (диффузного отражения)

Если есть некоторая поверхность и в некоторую точку этой поверхности, у которой есть нормаль $\vec{N}$, направлен луч от источника света. Для наблюдателя, находящегося в любой точке, яркость точки, которую он видит, будет выражаться следующим образом. $V = E \cdot I \cdot \cos q + E \cdot I_0$, где **V** - яркость (для ч/б); **E** – коэффициент диффузного отражения поверхности (Альберта). $E \in [0;1]$, **I** – освещённость точки, $\mathbf{I_0}$ – фоновая освещенность (рассеянный свет), q - угол между нормалью ($\vec{N}$) и вектором, направленным на источник света ($\vec{S}$), $\cos q = \frac{\vec{S} \cdot \vec{N}}{|\vec{S}| \cdot |\vec{N}|}$.

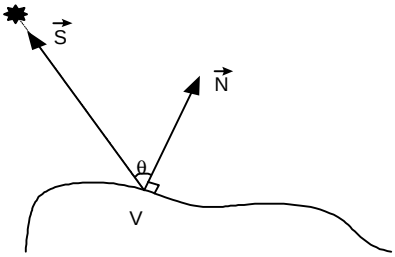


Рис. 5.1.1

Данный метод не учитывает отражения света, поэтому место положения наблюдателя не играет роли. При помощи этого метода лучше всего моделируется матовые поверхности.

Рассмотренный ранее закон Ламберта можно записать в удобной форме.

$$V = V_{MAX} \cdot E \cdot \left((1-e) \cdot \cos q + e \right),$$

где ε – доля рассеянного света (рекомендуется $e \in [0.1;0.2]$).

Рассматриваются два вида источников света:

а) точечный источник света:

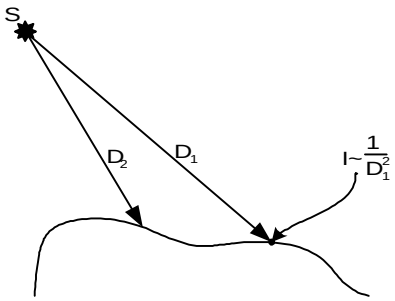


Рис. 5.1.2

б) параллельный пучок света: от удаленного источника

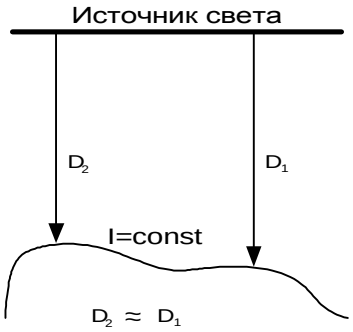


Рис.5.1.3

Закон Фонга (закон зеркального отражения)

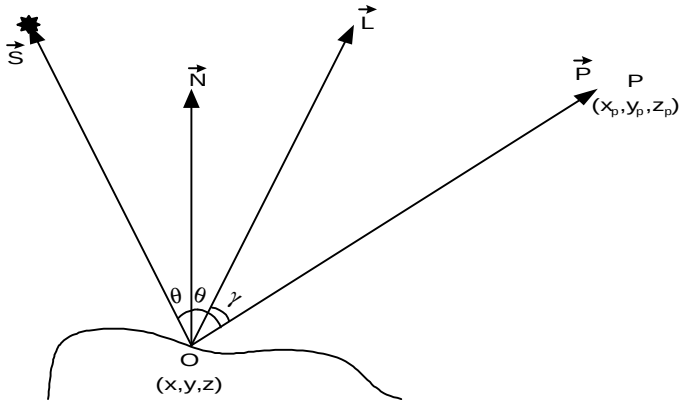


Рис.5.1.4

$\vec{N}$ – нормаль к поверхности в точке (x,y,z);

$\vec{S}$ – падающий луч от источника S;

$\vec{L}$ – отраженный луч света;

$\vec{P}$ – направление на наблюдателя P (x_P,y_P,z_P);

θ – угол падения и отражения;

γ – угол между отраженным лучом и направлением на наблюдателя.

$\boxed{V_\Phi = E \cdot I \cdot \left|\cos^n g\right|;}$ формула для определения зеркальной составляющей V, где n – степень зеркальности поверхности, $n \in [1;200]$.

Чем больше n тем больше зеркальные свойства поверхности.

$\cos q = \frac{\vec{L} \cdot \vec{P}}{\left|\vec{L}\right| \cdot \left|\vec{P}\right|}; \vec{P} = [x_p - x_0, y_p - y_0, z_p - z_0]; \vec{S} = [x_S - x_0, y_S - y_0, z_S - z_0]; \vec{L} = 2\left(\vec{N} \cdot \vec{S}\right) \cdot \vec{N} - \vec{S};$

$\left|\vec{L}\right| = 1; \left|\vec{N}\right| = 1; \left|\vec{S}\right| = 1;$

Вектора $\vec{L}, \vec{S}, \vec{N}$ нормированные и лежат в одной плоскости (см. закон отражения света)

Пусть I = const, тогда $\boxed{V = V_{MAX} \cdot E\left((1 - e - e_\Phi) \cdot \left|\cos q\right| + e_\Phi \cdot \left|\cos^n g\right| + e\right)}$

ε – доля рассеянного света,

ε_Φ – доля отраженного света, $e_\Phi \in [0;1]$; $0 \le e + e_\Phi \le 1$

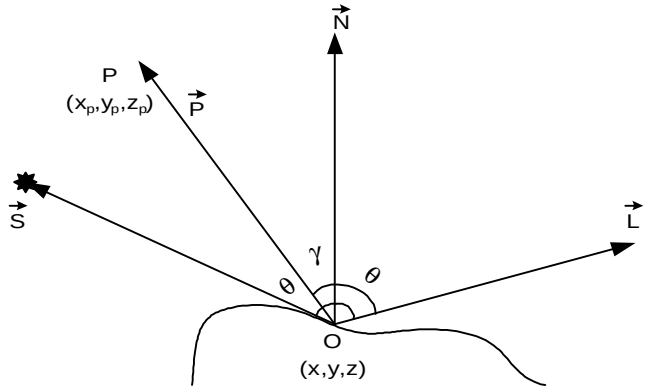


Рис. 5.1.5

Если угол $\gamma > 90°$, то не надо учитывать зеркальную составляющую.

Лунная модель

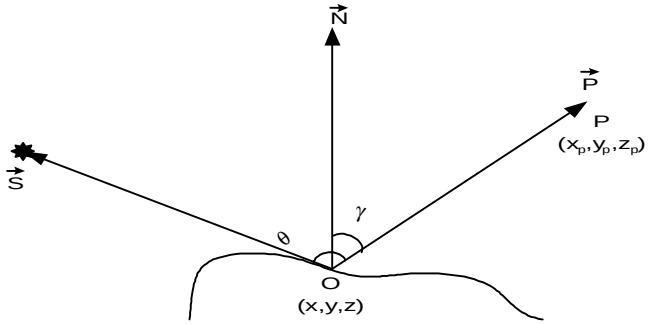


Рис. 5.1.6

$\boxed{V_Д = E \cdot I \cdot \left[\frac{\left|\cos q\right|}{1 + L \cdot \left|\cos g\right|}\right]}$ – формула Гуро

$V_Д$ – диффузная составляющая света. Если L = 0 , получается закон Ламберта. L >0.

По сравнению с методом Ламберта эта модель уменьшает яркость точек, на которые мы смотрим под углом 90°, и увеличивает яркость тех точек, на которые мы смотрим вскользь.

5.2. Применение законов освещения при синтезе объекта изображения.

В большинстве случаев объекты задаются набором плоских выпуклых граней. Поэтому при построении изображения естественно воспользоваться этой простотой. Существуют три простейших метода закраски, дающих достаточно приемлемые результаты - метод постоянного закрашивания, метод Гуро, метод Фонга.

Flat- метод постоянного закрашивания

самый простой из всех методов.

Основная идея: каждая грань закрашивается одним цветом.

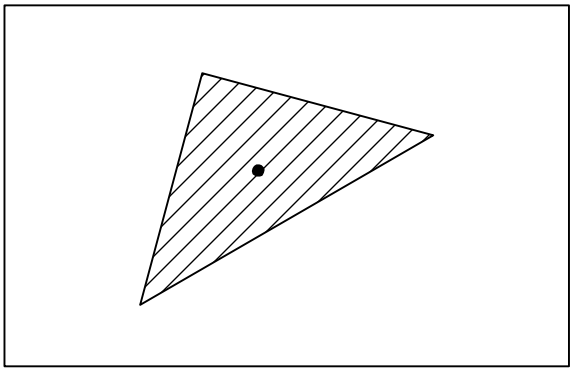


Рис. 5.2.1

Рассчитывается яркость в одной точке (например, в центре тяжести для выпуклых многоугольников) грани (по Ламберту) и производится заливка грани полученным цветом.

Метод закраски Гуро

Обеспечивает непрерывность освещенности. Основная идея: заливка осуществляется с учетом линейной интерполяции яркости, вычисляется яркость только для вершин многоугольника.

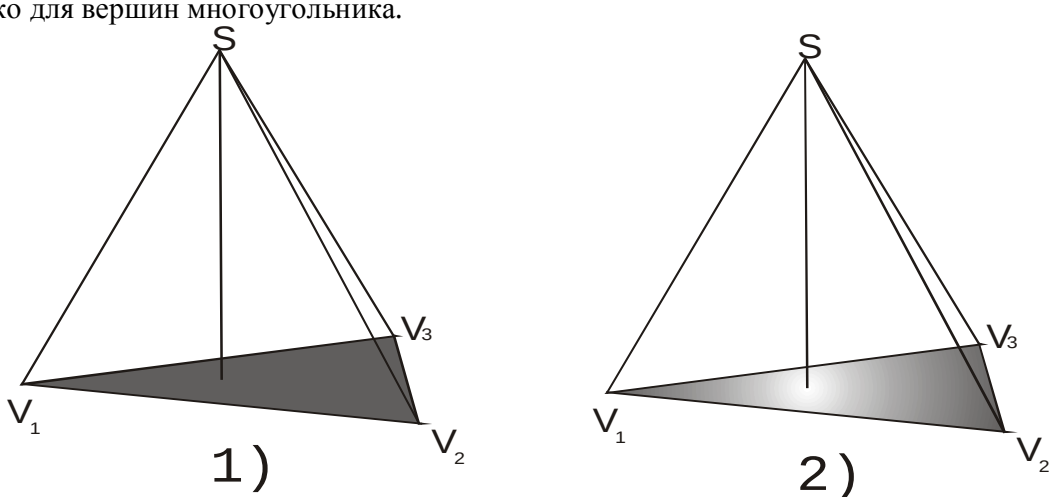


Рис. 5.2.2

Пусть задана плоская грань V_1 V_2 V_3(рис 5.2.2). Найдем значение освещенности в каждой ее вершине. Обозначим получившиеся значения через I_1, I_2, I_3. Рисуя грань V_1 V_2 V_3 построчно (рис 5.2.3), находим значения освещенности в конце каждого горизонтального отрезка путем линейной интерполяции значений вдоль ребер. При рисовании очередного отрезка AB будем считать, интенсивность изменяется от I(A) до I(B) линейно.

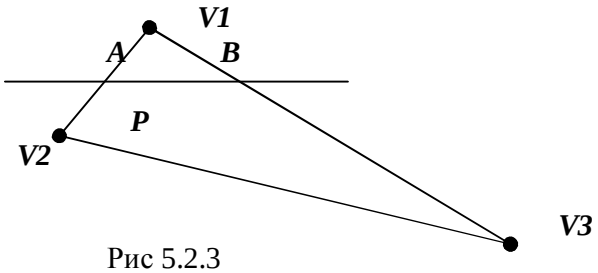


Рис 5.2.3

Недостаток метода в том, что если источник света проецирутся в плоскость многоугольника, то, после использования этого метода заливки, будет получен результат рис. 5.2.2 (а), хотя должно быть рис.5.2.2 (б).

Закраска по Фонгу

Основная идея: для каждой точки изображения устанавливаются пространственные координаты, исходя из которых, считаем γ и получаем яркость для точки.

Недостаток метода – большая сложность вычислений.

Моделирование освещения методом наложения текстуры

Задачу закраски объектов, состоящих из отдельных граней можно решить с помощью текстурирования. Можно упростить вычисления, сведя метод Фонга к процедуре нанесения текстуры.

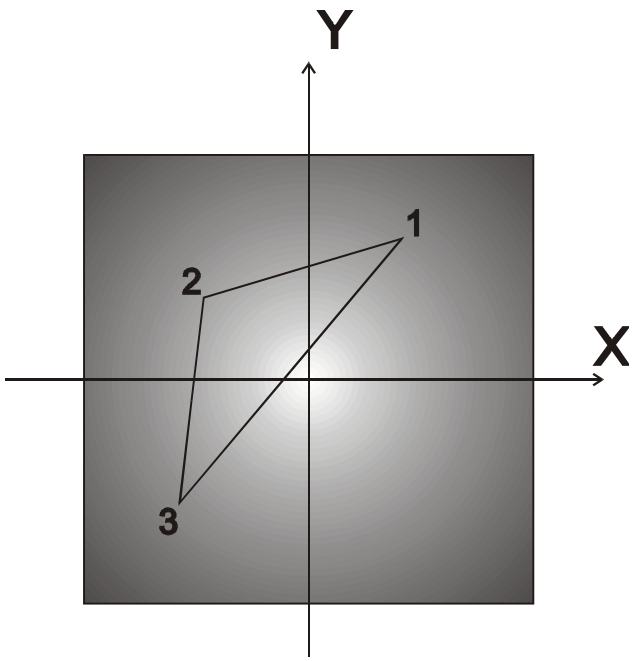


Рис. 5.2.4

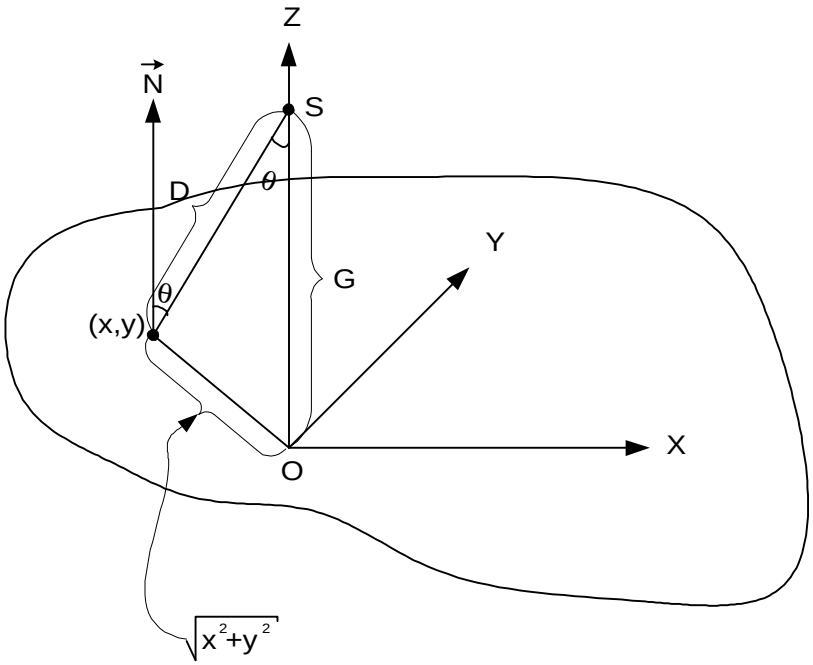


Рис. 5.2.5

Основная идея: в памяти рассчитывается текстура рис. 5.2.4, затем заливка объектов осуществляется с использованием полученной текстуры.

- Рассчет вспомогательной текстуры.

Для каждой точки текстуры рассчитывается яркость по формуле

$$V = I \cdot E \cdot \cos q, E \in [0;1]; I = \frac{I_{MAX}}{D^2}$$

$$V = E \cdot I_{MAX} \cdot \frac{\cos q}{D^2}$$

Пусть под яркость отведен 1 байт, т.е. $V \in \{0,1,...,255\}$ – $V_{MAX} = 255$.

Максимальная яркость будет в точке максимально приближенной к источнику света, т.е. $V_{MAX} = \frac{I_{MAX}}{D^2_{MIN}}$. $D^2_{MIN} = G^2; I_{MAX} = V_{MAX} \cdot G^2$

$$V_{(x,y)} = V_{MAX} \cdot E \cdot \frac{G^2}{D^2} \cdot \cos q; h = \frac{G^2}{D^2} \cdot \cos q; \cos q = \frac{G}{D}; D = \sqrt{x^2 + y^2 + G^2}$$

$$h_0(x,y) = \frac{G^2}{x^2 + y^2 + G^2} \cdot \frac{G}{\sqrt{x^2 + y^2 + G^2}} = \left(\frac{G}{\sqrt{x^2 + y^2 + G^2}} \right)^3$$

Желательно иметь η в пределах [0,1]. Для этого домножим выражение для η_0 домножим на константу G^2. $h_0(x,y) = \frac{1}{\left[\left(\frac{x}{G} \right)^2 + \left(\frac{y}{G} \right)^2 + 1 \right]^{3/2}}$

Если соответсявующим образом просматривать η, то получится яркость соответствующей точки в вспомогательной текстуре.

Рассчет координатных точек для произвольного треугольника.

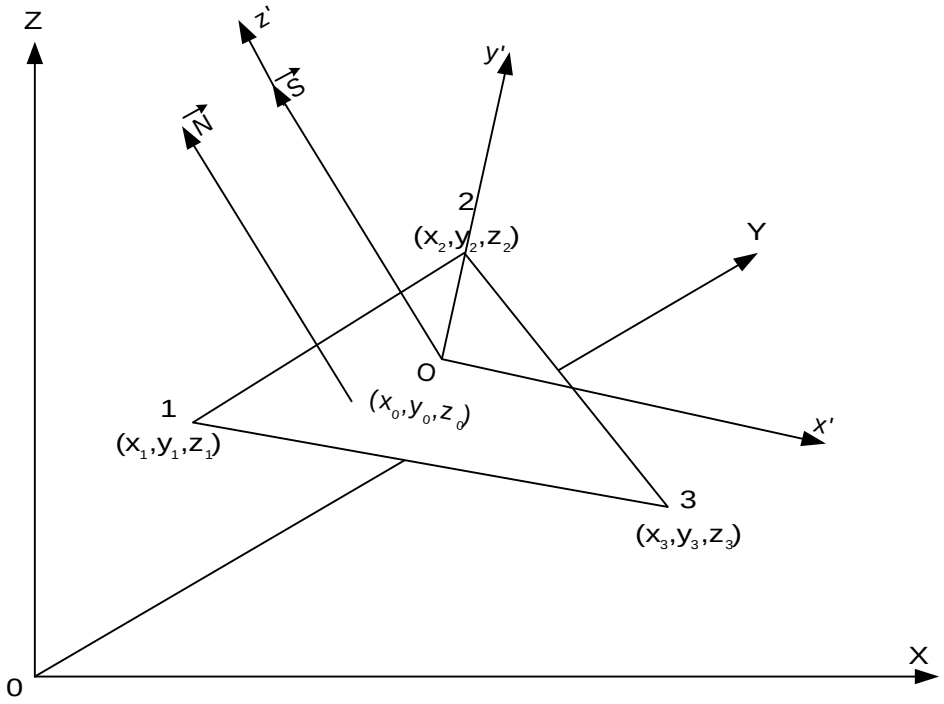


Рис. 5.2.5

В мировой системе координат задан произвольный треугольник рис. 5.2.6 необходимо провести его заливку с учетом освещенности.

Для этого строится система координат (x',y'z') с началом в точке O(x_0,y_0,z_0), таким образом, что ось OZ проходит через источник света S и параллельна нормали $\vec{N}$, а OX и OY лежат в плоскости треугольника.

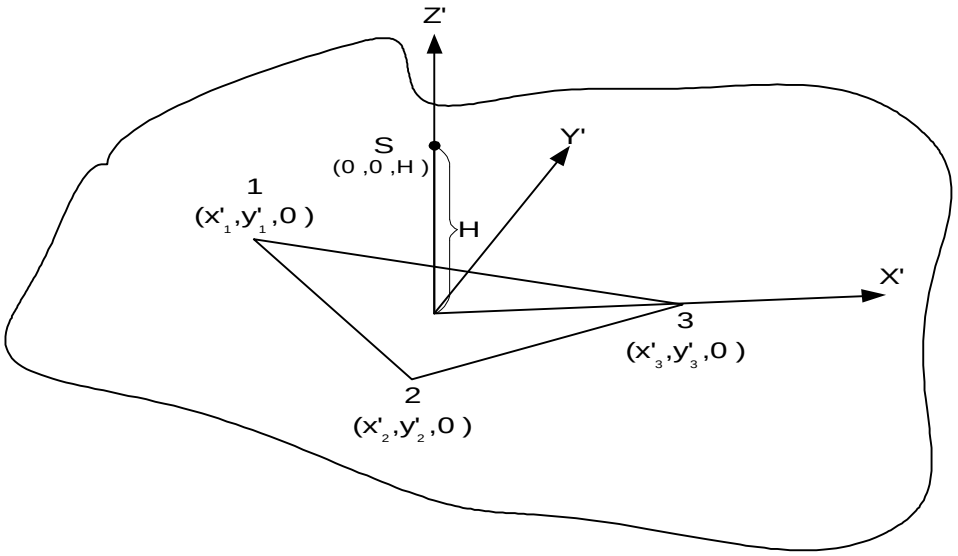


Рис. 5.2.6

$[x';y';z'] = [x - x_0; y - y_0; z - z_0] \cdot M$, где M – матрица преобразования.

Найдем такую матрицу M, чтобы точки 1,2,3,S проецировались в точки с координатами которые озображены на рис. 5.2.6.

$N_V = (V_2 - V_1) \times (V_3 - V_1)$ – ненормированный вектор нормали

Пронормируем этот вектор: $N = \frac{N_V}{|N_V|}$; N(N_X,N_Y,N_Z)

Опираясь на это выражение, вычисляем матрицу M:

1)

$$M_1 = \begin{bmatrix} \sqrt{N_Z^2 + N_Y^2} & \frac{-N_Y \cdot N_X}{\sqrt{N_Z^2 + N_Y^2}} & \frac{-N_Z \cdot N_X}{\sqrt{N_Z^2 + N_Y^2}} \\ 0 & \frac{N_Z}{\sqrt{N_Z^2 + N_Y^2}} & \frac{-N_Y}{\sqrt{N_Z^2 + N_Y^2}} \\ N_X & N_Y & N_Z \end{bmatrix}$$

данная формула используется когда составляющая нормали N_X = min;

2)

$$M_2 = \begin{bmatrix} \frac{-N_Y \cdot N_X}{\sqrt{N_Z^2 + N_X^2}} & \sqrt{N_Z^2 + N_X^2} & \frac{-N_Y \cdot N_Z}{\sqrt{N_Z^2 + N_X^2}} \\ \frac{N_Z}{\sqrt{N_Z^2 + N_X^2}} & 0 & \frac{-N_X}{\sqrt{N_Z^2 + N_X^2}} \\ N_X & N_Y & N_Z \end{bmatrix}$$

данная формула используется когда составляющая нормали N_Y = min;

3)

$$M_3 = \begin{bmatrix} \frac{N_Z \cdot N_X}{\sqrt{N_Y^2 + N_X^2}} & \frac{N_Z}{\sqrt{N_Y^2 + N_X^2}} & -\sqrt{N_Y^2 + N_X^2} \\ \frac{-N_Y}{\sqrt{N_Y^2 + N_X^2}} & \frac{N_X}{\sqrt{N_Y^2 + N_X^2}} & 0 \\ N_X & N_Y & N_Z \end{bmatrix}$$

данная формула используется когда составляющая нормали N_Z = min;

Для окончательного пересчета координат вершин треугольника будем пользоваться M умноженной на $\frac{G}{H}$; $M_f = \frac{G}{H} \cdot M$; где M=$M_1$, M_2, M_3.

M_f – матрица Фонга. Таким образом координаты в текстурном поле :

$$[x_{1f}; y_{1f}; 0] = [x_1 - x_0; y_1 - y_0; z_1 - z_0] \cdot M_f;$$
$$[x_{2f}; y_{2f}; 0] = [x_2 - x_0; y_2 - y_0; z_2 - z_0] \cdot M_f;$$
$$[x_{3f}; y_{3f}; 0] = [x_3 - x_0; y_3 - y_0; z_3 - z_0] \cdot M_f;$$
$$[0; 0; G] = [x_S - x_0; y_S - y_0; z_S - z_0] \cdot M_f;$$

Последняя строка используется для контроля вычислений.

Схема закраски фигуры с учетом освещенности с использованием нанесения текстур.

1) Вычисляем η для каждой точки текстуры и записываем полученные результаты в таблицу, которую храним как текстурное поле.

$$h_0(x,y)=\frac{1}{\left[\left(\frac{x}{G}\right)^2+\left(\frac{y}{G}\right)^2+1\right]^{3/2}}$$

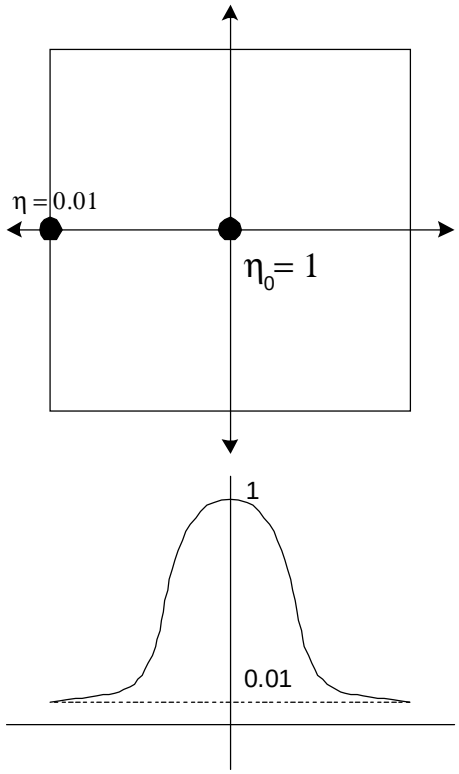


Рис 5.2.7

На рис. 5.2.7 показан примерный диапазон и характер изменения η.

2) Высчитываем нормаль к поверхности треугольника

$$N_V=(V_2-V_1)\times(V_3-V_1);\ N=\frac{N_V}{|N_V|}$$

3) Пересчитываем координаты в текстурные, используя M_f

$$M_f=\frac{G}{H}\cdot M\ ;$$

$$[x_{1f}; y_{1f}; 0] = [x_1 - x_0; y_1 - y_0; z_1 - z_0] \cdot M_f;$$
$$[x_{2f}; y_{2f}; 0] = [x_2 - x_0; y_2 - y_0; z_2 - z_0] \cdot M_f;$$
$$[x_{3f}; y_{3f}; 0] = [x_3 - x_0; y_3 - y_0; z_3 - z_0] \cdot M_f;$$
$$[0; 0; G] = [x_S - x_0; y_S - y_0; z_S - z_0] \cdot M_f;$$

4) Рассчитываем яркость каждой точки.

$$V_{(x,y)} = V_{MAX} \cdot E \cdot h_0(x_f, y_f) \cdot \frac{G^2}{H^2};$$

Если учитывать рассеянный свет, то $\boxed{V_{(x,y)} = V_{MAX} \cdot E \cdot h_0(x_f, y_f) \cdot \frac{G^2}{H^2} \cdot (1-e) + e;}$, где ε - доля рассеянного света.

Рассмотрим случай, когда объект имеет гладкую форму.

Аналог алгоритма Гуро

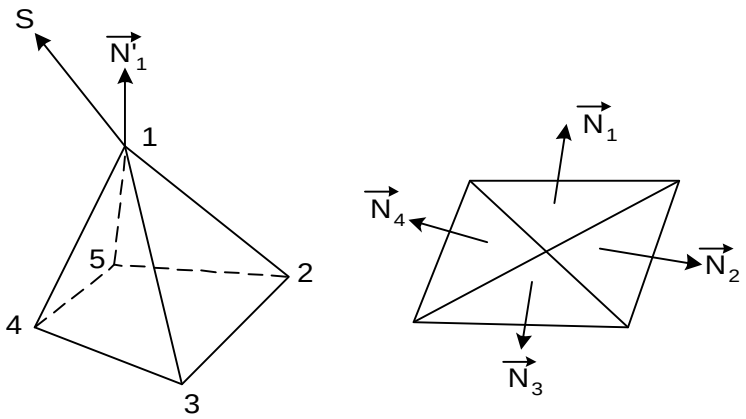


Рис. 5.2.8

Яркость рассчитывается в каждой вершине, а яркость на гранях и ребрах получается линейной интерполяцией. В качестве нормали при расчетах яркости в одной вершине используется средняя нормаль $\vec{N'}_1 = \frac{1}{n} \cdot \sum_{i=1}^{n} \vec{N}_i$, где n – число прилегающих к этой вершине граней. $\vec{N}_i$ – вектора площади, перпендикулярные соответсвующей грани и равные ее площади, таким образом учитывается то, что грани могут быть разного размера, а следовательно по-разному влиять на среднее значение нормали.

Рассчитанная таким методом яркость вершины используется для всех прилегающих к ней ребер, а следовательно со всех сторон вершины яркость одинакова и перепада яркости на ребрах не будет.

Аналог алгоритма Фонга

Гладкий объект отличается от негладкого тем, что на его поверхности задано прерывное поле единичных векторов нормали. Попытаемся построить такое поле искусственно. Для этого применим ранее описанную процедуру билинейной интерполяции(рассмотренный в разделе «Метод закраски Гуро»), но только не к значениям освещенности, а к значениям векторов нормали. В результате мы получим непрерывное поле векторов нормали, но так как эти векторы не всегда оказываются единичными, нужна нормировка.

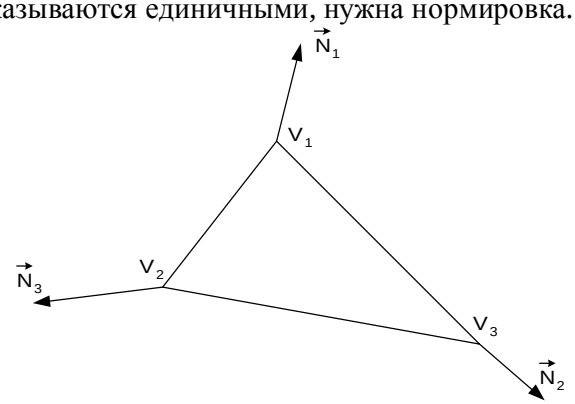


Рис. 5.2.9

Недостаток метода – если поверхность неровная, то возможны неточности рис.5.2.10

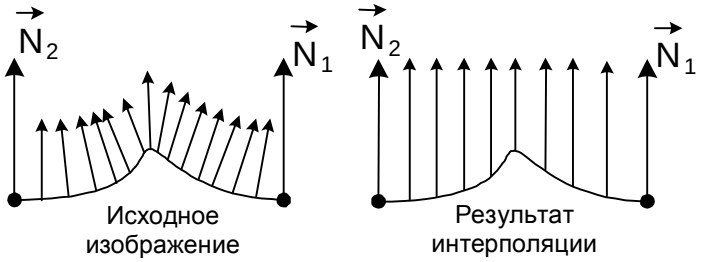


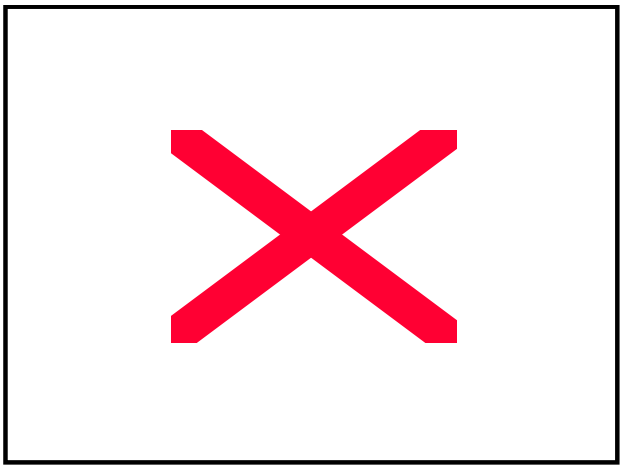
Рис. 5.2.10

Метод Фонга требует большое число вычислений, так как вычисление вектора нормали и освещенности производится лтдельно в каждой точке.

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.

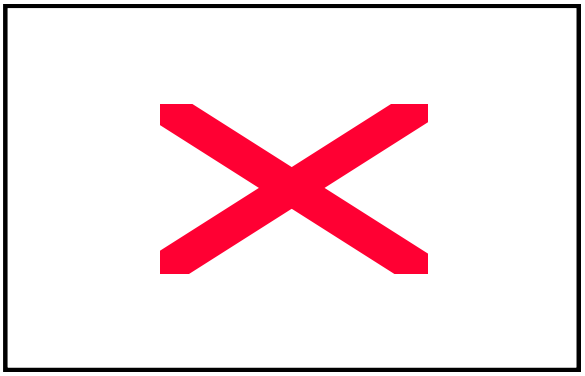
Теперь рассмотрим мировую систему координат, в которой мы имеем следующий треугольник (имеет рельефную текстуру):
S – источник света;



$$V_1(X_1, Y_1, Z_1, x_{r1}, y_{r1})$$
$$V_2(X_2, Y_2, Z_2, x_{r2}, y_{r2}),$$
$$V_3(X_3, Y_3, Z_3, x_{r3}, y_{r3})$$

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

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



Где:

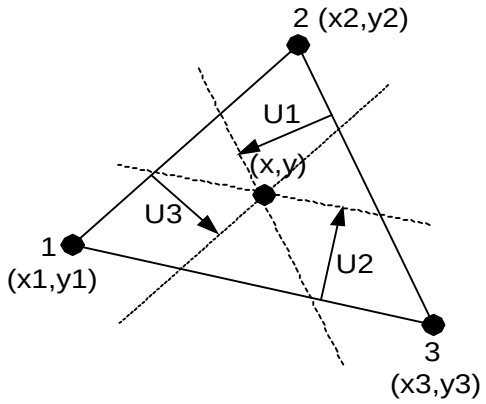
$$S(x_{r0}, y_{r0}, H); V_1(x_{r1}, y_{r1}, 0); V_2(x_{r2}, y_{r2}, 0); V_3(x_{r3}, y_{r3}, 0)$$

Воспользуемся следующими формулами:

$$N_V = (V_3 - V_1) \cdot (V_2 - V_1)$$
$$N = \frac{N_V}{|N_V|}$$
$$H = N \cdot (S - V_1)$$
$$V_0 = S - N \cdot H$$

Для нахождения координат введем локальную систему координат $U_1 U_2 U_3$ (рис).

Относительные координаты:



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

Для любой точки принадлежащей этому треугольнику:

$U_1 + U_2 + U_3 = 1$ а величина $0 < U_2 < 1$, при этом U_2 характеризует удаленность от (.) центра.

$$\begin{cases} X_0 = X_1 U_1 + X_2 U_2 + X_3 U_3 \\ Y_0 = Y_1 U_1 + Y_2 U_2 + Y_3 U_3 \quad (*) \\ Z_0 = Z_1 U_1 + Z_2 U_2 + Z_3 U_3 \end{cases}$$

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

Определим для точки V_0 относительные координаты через её пространственные координаты:

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

Эти формулы являются по сути решением системы (*).

Координаты точки в рельефной системе:

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

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

Чтобы избежать появления знаменателя равного нулю, воспользуемся альтернативными формулами, где X заменены на Z для системы (**)

Алгоритм прорисовки:

1) пересчёт координаты в рельефном поле;
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:

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

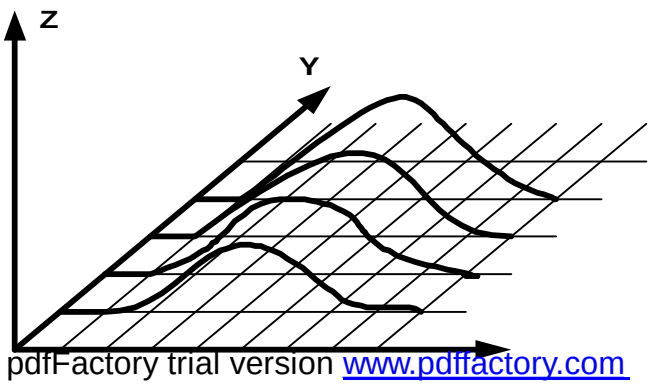
6. Удаление невидимых элементов

6.1 Синтез изображения с помощью Y-буфера.

Y –буфер – это не универсальный алгоритм для удаления невидимых элементов в проволочном каркасе. Y –буфер имеет размерность равную длине строки изображения. Туда записываются максимальные y-координаты, но уже в плоскости изображения.

Рассмотрим частный случай:

Синтез каркасных изображений с удалением невидимых элементов этого изображения.



Синтезируют изображение от ближнего плана к дальнему. Проецируют сначала отрезки, соответствующие сечению 1 рельефа. В Y-буфере будет отслеживаться у максимальное. При рисовании второго сечения проверяются текущие координаты, сравниваются с теми, что уже записаны. Это делается для того, чтобы часть изображения по сечению 2 не была ниже изображения относительно сечения 1 (рис 6.1). При записи второго сечения Y-буфер обнуляется и информация в нём обновляется.

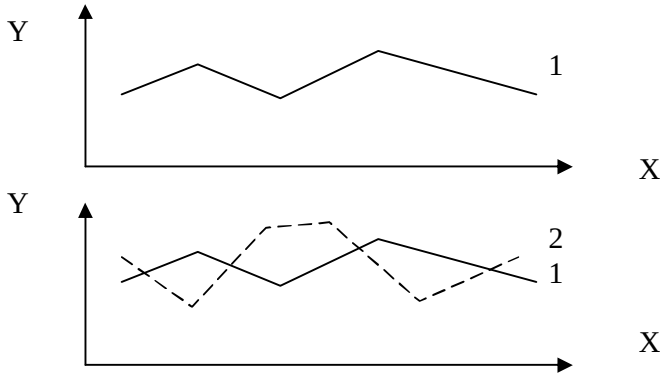
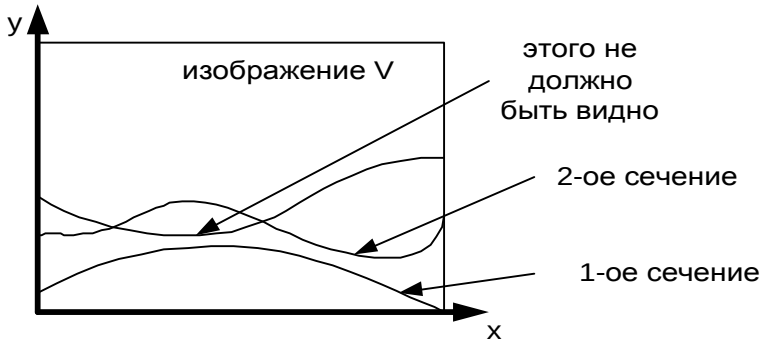
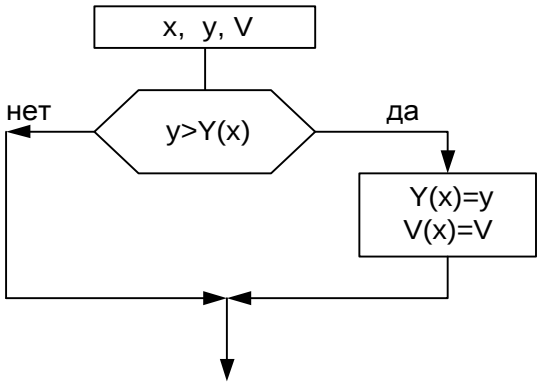


Рис 6.1.1



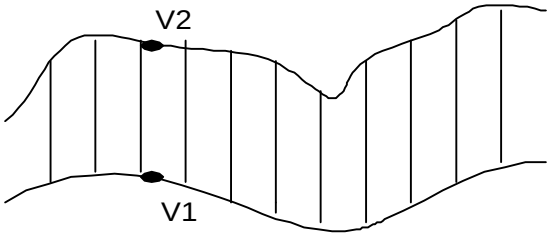
Для решения проблемы видимости ненужных частей рельефа используется



анализ текущей точки:

Примечание:

Для полутоновых изображений так же можно использовать этот алгоритм, но с закраской вертикальными линиями с учётом линейной интерполяции области между двумя соседними сечениями.



6.2 Трассировка лучей.

Всё что было рассмотрено ранее относится к проективным методам синтеза.
Альтернативой им является **трассировка лучей**, так же как и проективные методы поддерживается проективно.
Суть: отслеживание луча.
Существуют:

1) ***Обратная трассировка*** луча (отслеживание луча в направлении из глаза к источнику света) или просто ***трассировка***;
2) ***Прямая трассировка*** луча (движение вдоль луча, выходящего из источника света (рис 6.2.1)).

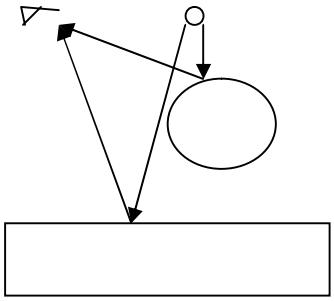


Рис 6.2.1

Это метод, при котором для каждого пиксела картинной плоскости определяется ближайшая к нему грань, для чего через этот пиксел выпускается луч, находятся все его пересечения с гранями и среди них выбирается ближайшая.Надо найти точку пересечения луча с ближайшим многоугольником (с объектом трёхмерного изображения). Эта задача решается с помощью сортировки элементов пространственной сцены.

Недостатки:

1) Для сложных сцен трассировка лучей не эффективена;
3) Струсктура вычислений сложнее, чем в проективных медодах.

Но метод трассировки лучей более эффективен, чем проективный метод для простой (гладкой) сцены, которая может быть описана математически, т. е.:

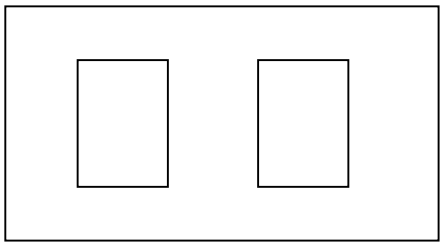
1) Решается система уравнений;
2) Находится точка пересечения;
3) Находится яркость.

Так же трассировка лучей эффективна для обработки диффузных поверхностей. Так как в этом существует диффузное отражение, которое может быть получено при использовании прямой и обратной трассировке лучей.

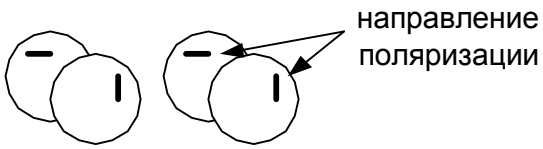
7. Синтез стереоизображений.

7.1 Методы наблюдения:

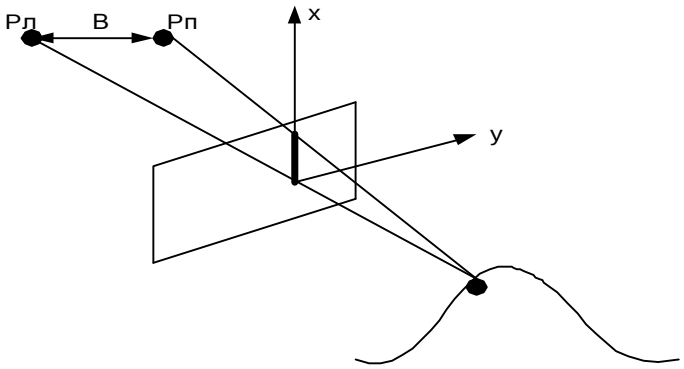
1) делим изображение на 2, одно для левого глаза другое для правого. Затем на экране синтезируются эти 2 изображения, в результате чего мы видим стерео изображение.



2) с помощью анаглифов:
   а) цветовой (Одеваем очки со светофильтрами, допустим, синий и красный. Наложение синий палитры на красную даст нам сиреневое изображение);
   б) поочерёдное представление 2-х изображений (применяются очки в виде оптических затворов);
   в) применение очков на основе жидких кристаллов (два светофильтра, с помощью изменения направления поляризации закрывается одно изображение).



Синтез:
2 центра проекции параллельно разносятся: Р-левое и Р-правое

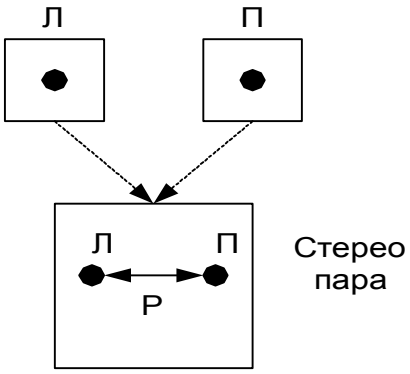


В – база

База должна лежать в одной плоскости с осью X.

Рис 7.1.1.

Это изображение для смещения только относительно оси X.



P – продольный паралакс

Рис 7.1.2

Рассмотрим пример:

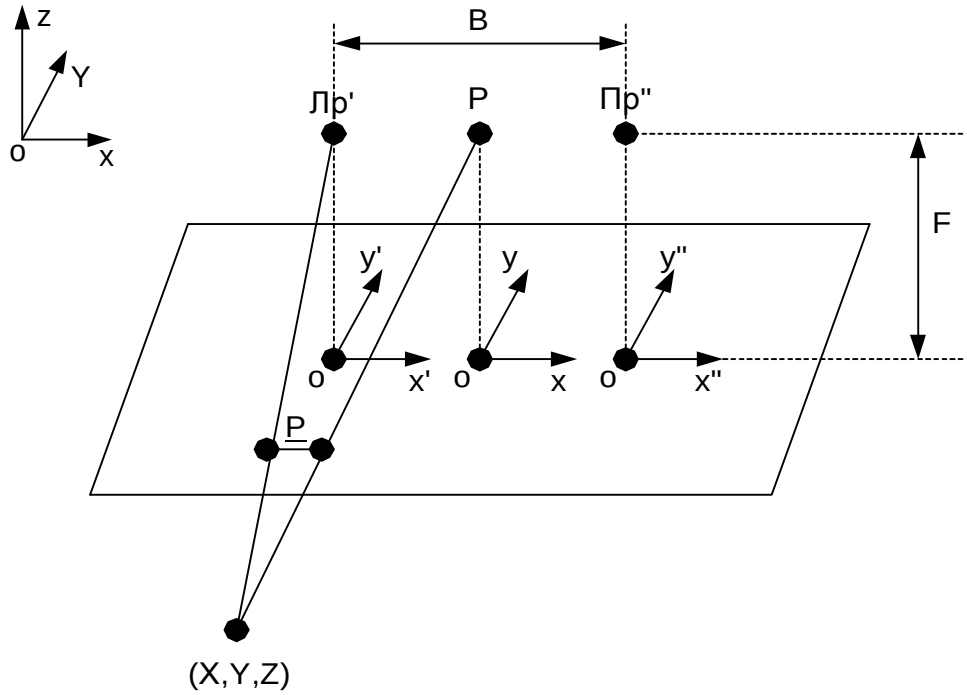


Рис 7.1.3

Координаты точек: $Л_{p'}(X_p^{'}, Y_p, Z_p); P(X_p, Y_p, Z_p); П_{p''}(X_p^{''}, Y_p, Z_p)$

Наблюдатель смотрит вертикально вниз.

Для центра проекции левого глаза и центра проекции правого глаза мы имеем следующие координаты:

$$x'' = -F\frac{X''_p - X}{Z_p - Z}; x' = -F\frac{X'_p - X}{Z_p - Z}$$

$$y' = y'' = -F\frac{Y_p - Y}{Z_p - Z}$$

$$X''_p = X_p + \frac{B}{2}; X'_p = X_p - \frac{B}{2}$$

В системе координат OXY:

$$x_{Левое} = x' - \frac{B}{2}; x_{Правое} = x'' + \frac{B}{2}$$

Используя эти формулы получаем:

$x_Л = \bar{x} + p$; $x_П = \bar{x} - p$, где

p – полупаралакс:

$p = \frac{B}{2}\left(1 - \frac{F}{Z_p - Z}\right)$, где F – фокусное расстояние

Найдём значение $\bar{x}$:

$$\bar{x} = \frac{x_Л + x_П}{2} = -F\frac{X_p - X}{Z_p - Z}$$

$p = x_П - \bar{x} = \bar{x} - x_Л$; $P = 2p$

Для каждой точки можно наити изображение для левого и правого снимка.
Пусть у нас есть:

Поле высот Z(X,Y) Фотография V(X,Y)

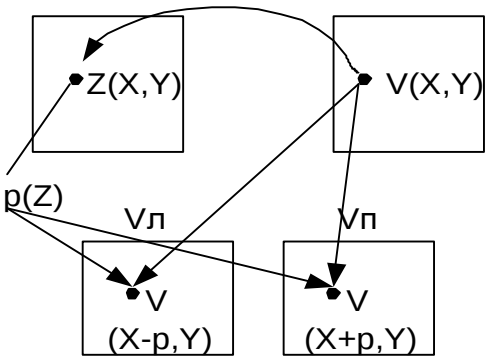


Рис 7.1.4

Для того чтобы получить желаемое необходимо:
Обоити все точки поля V и для каждой из них, обратившись в поле Z, получим высоту Z(X,Y), уже имея яркость V(X,Y). Зная высоту, подставим её в формулу полупаралакса. Затем в левое поле записываем яркость V в точку (X-p,Y), а в правое поле записываем яркость V в точку (X-p,Y).

Недостаток этого механизма:
При заполнении Vл и Vп некоторые точки могут быть не заполнены.

В качестве исходных данных мы задаём минимальный и максимальный полупаралакс:

$$p_{\min} = Z_{\max};\ p_{\max} = Z_{\min}$$

Для рассматриваемой задачи:

$p_{\min} = 0$

$p_{\max} = 10 \div 20\%$ от размера изображения по горизонтали (т.е. от $x_{\max}$)

Решив следующую систему, можно найти B и F:

$$p_{\min} = \frac{B}{2}\left(1 - \frac{F}{Z_p - Z}\right)$$

$$p_{\max} = \frac{B}{2}\left(1 - \frac{F}{Z_p}\right)$$

И далее подставить их в формулу полупаралакса:

$$p = \frac{Z_p p_{\max} - \frac{Z}{Z_{\max}}\left(p_{\max}\frac{Z_p}{Z_{\max}} - p_{\min}\left(\frac{Z_p}{Z_{\max}} - 1\right)\right)}{\frac{Z_p}{Z_{\max}} - \frac{Z}{Z_{\max}}}$$

, где

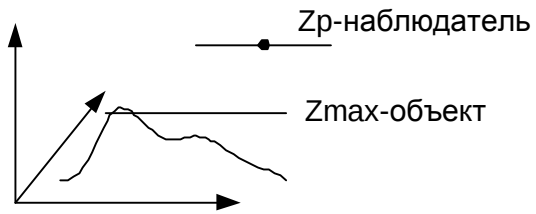


Рис 7.1.4

$\frac{Z_p}{Z_{\max}} = a > 1$, где Zp – высота наблюдателя

$$Z_p > Z_{\max} \geq Z \geq 0$$

Отсюда следует следующее:

$$p = \frac{ap_{\max} - \frac{Z}{Z_{\max}}\left(p_{\max}a - p_{\min}(a-1)\right)}{a - \frac{Z}{Z_{\max}}}$$

А при $p_{\min} = 0$:

$$p = \frac{a\, p_{\max} \left(1 - \frac{Z}{Z_{\max}} \right)}{a - \frac{Z}{Z_{\max}}}$$

Далее вместо поля вывода будем использовать буфер глубины.
В буфере глубины содержится информация об удалённости точки от наблюдателя, а в поле вывода тоже фактически содержится информация об отдалённости точки (о выс оте).

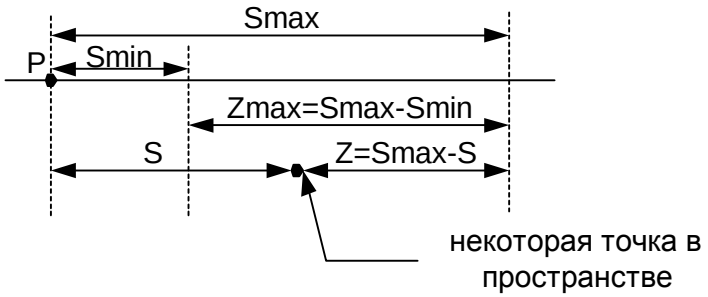


Рис 7.1.5

$$Z = S_{\max} - S$$

$$Z_p = S_{\max}$$

$$Z_{\max} = S_{\max} - S_{\min}$$

$$a = \frac{Z_p}{Z_{\max}} = \frac{S_{\max}}{S_{\max} - S_{\min}}$$

Используя формулу :

$$S = \frac{A}{h + B}$$, которая была получена ранее получаем, что:

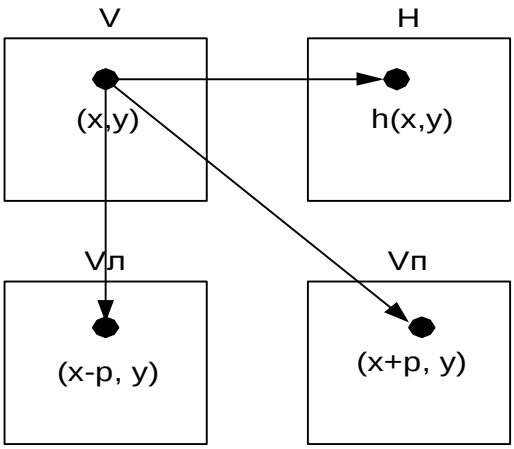
$S_{\max} = \frac{A}{h_{\min} + B}$, а так как $h_{\min} = 0$, то $S_{\min} = \frac{A}{0 + B} = \frac{A}{B}$;

$$S_{\min} = \frac{A}{h_{\max} + B}$$

Следовательно формула полупаралакса будет выглядеть следующим образом:

$$p = p_{max}\left(1 - \frac{h}{h_{max}}\right)$$

С помощью буфера глубины можно синтезировать изображение:



8.Представление пространственных форм.

Пусть надо изобразить пространственную кривую:

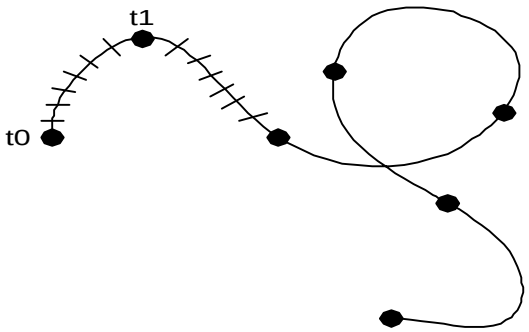


Рис 8.1.

Всю кривую разобьём на криволинейные отрезки. В пределах кусочка {t0-t1} зададим некоторый параметр t. t изменяется в диапазоне от 0 до 1.
Пространственные координаты этих кусочков:

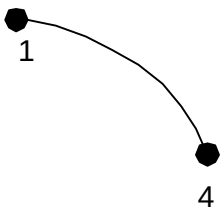
$$x(t) = a_x t^3 + b_x t^2 + c_x t + d$$
$$y(t) = a_y t^3 + b_y t^2 + c_y t + d \quad (*)$$
$$z(t) = a_z t^3 + b_z t^2 + c_z t + d$$

Но чтобы обеспечить хорошую стыковку этих кусочков нужно соблюдать следующие условия:

1) При стыковке по уровню $C^{(0)}$ - непрерывность по координатам;
2) При стыковке по уровню $C^{(1)}$ - непрерывность по первой производной;
3) При стыковке по уровню $C^{(2)}$ - непрерывность на уровне второй производной.

Математическое вычисление коэффициентов полиномов (*):

1) В форме Эрмитта:
   Для куска кривой должны быть известны:
   а) координаты начальной и конечной точек:



Координаты:
$P_1(P_{1x}, P_{1y}, P_{1z})$ и $P_4(P_{4x}, P_{4y}, P_{4z})$

Рассмотрим вычисления только относительно X:
$x(0) = P_{1x}$; $x(1) = P_{4x}$

б) производные (по каждой из координат) в начальной и конечной точках:

$$x'(0) = R_{1x} \;\; ; \;\; x'(1) = R_{4x}$$

Для нахождения коэффициентов полиномов (*) обозначим:

$$x(t) = \begin{bmatrix} t^3 & t^2 & t & 1 \end{bmatrix} * \begin{bmatrix} a_x \\ b_x \\ c_x \\ d_x \end{bmatrix}$$

$x(t) = T * Q_x$, где Q_x - матрица коэффициентов

$x(0) = P_{1x} = \begin{bmatrix} 0 & 0 & 0 & 1 \end{bmatrix} * Q_x$

$x(1) = P_{4x} = \begin{bmatrix} 1 & 1 & 1 & 1 \end{bmatrix} * Q_x$

$R_{1x} = \begin{bmatrix} 0 & 0 & 1 & 0 \end{bmatrix} * Q_x$

$R_{4x} = \begin{bmatrix} 3 & 2 & 1 & 0 \end{bmatrix} * Q_x$

Пусть :

$$G_{HX} = \begin{bmatrix} P_{1x} \\ P_{4x} \\ R_{1x} \\ R_{4x} \end{bmatrix} = \begin{bmatrix} 0 & 0 & 0 & 1 \\ 1 & 1 & 1 & 1 \\ 0 & 0 & 1 & 0 \\ 3 & 2 & 1 & 0 \end{bmatrix} * Q_x$$

G_{HX} - геометрический вектор Эрмитта (т.е. наши начальные донные).

$$Q_x = \begin{bmatrix} 2 & -2 & 1 & 1 \\ -3 & 3 & -2 & -1 \\ 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 \end{bmatrix} * G_{HX}$$

$Q_x = M_H * G_{HX}$, где M_H - матрица Эрмитта.

$x(t) = T * M_H * G_{HX}$

Анологичные вычисления производятся для Y и Z.

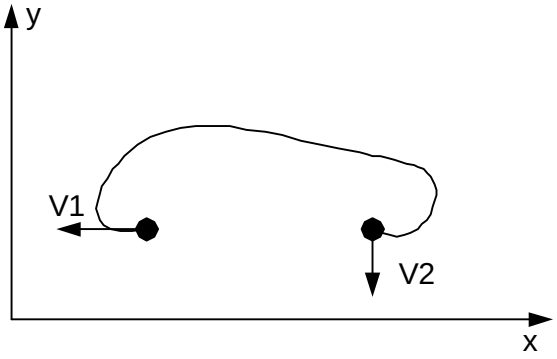
Таким образом мы получаем следующие формулы:

$P(t) = T * M_H * G_H$ и $Q = M_H * G_H$

$P(t) : x(t), y(t), z(t)$

$G_H : G_{HX,} G_{HY}, G_{HZ}$

Кривая построенная по этим данным:

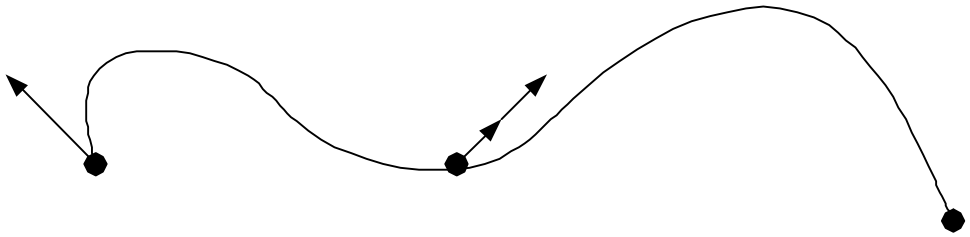


V1 и V2 – вектора скорости

Касательная к кривой задаётся отношением :

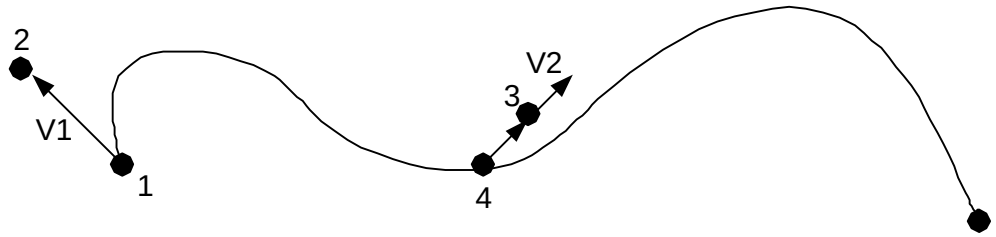
$$\frac{dy}{dx} = \frac{dy/dt}{dx/dt}$$

Если мы хотим соединить несколько кусочков, то в месте стыковки направление касательных для конца 1-го и начала 2-го отрезков должно совпадать.



Скорости могут отличаться по длине, но они должны лежать на одной касательной.

2) Задание коэффициентов в форме Безье:



В этом случае точки 2 и 3 являются управляющими(управляют формой кривой), а точки 1 и 4 являются опорными точками (кривая проходит через них).
Представление по Эрмиту:

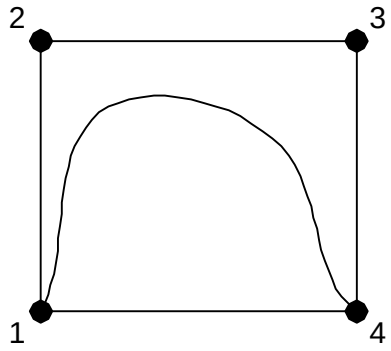
$$R_1 = p_2 - p_1 \;;\; R_4 = p_3 - p_4$$

А Безье предложил следующее:

$$R_1 = 3(p_2 - p_1)$$

$$R_4 = 3(p_4 - p_3)$$

Т.е. кривая должна выити из точки 1 и прийти в точку 4, а лежать она будет внутри четырёхугольника, образованного точками 1, 2, 3, 4.



По Эрмиту:

$p(t) = T * M_H * G_H$, где под p может подразумеваться либо x, либо y, либо z

Запишем эту формулу для Безье:

$$G_{HB} = \begin{bmatrix} p_1 \\ p_2 \\ p_3 \\ p_4 \end{bmatrix} ; \quad G_H = \begin{bmatrix} p_1 \\ p_2 \\ R_3 \\ R_4 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 3 & 3 & 0 & 0 \\ 0 & 0 & -3 & 3 \end{bmatrix} * G_{HB} = M_{HB} * G_{HB}$$

Откуда следует, что:

$$p(t) = T * M_H * G_H = M_H * M_{HB} * G_{HB},$$

где $M_H * M_{HB} = M_B$ - матрица Безье.

$$M_B = \begin{bmatrix} -1 & 3 & -3 & 1 \\ 3 & -6 & 3 & 0 \\ 3 & 3 & 0 & 0 \\ 1 & 0 & 0 & 0 \end{bmatrix}$$

$p(t) = T * M_B * G_{HB}$, где $M_B * G_{HB} = Q_p$ - матрица коэффициентов.

Когда мы имеем несколько кусочков кривой описанных Безье, то для стыковки надо соблюсти следующее условие:

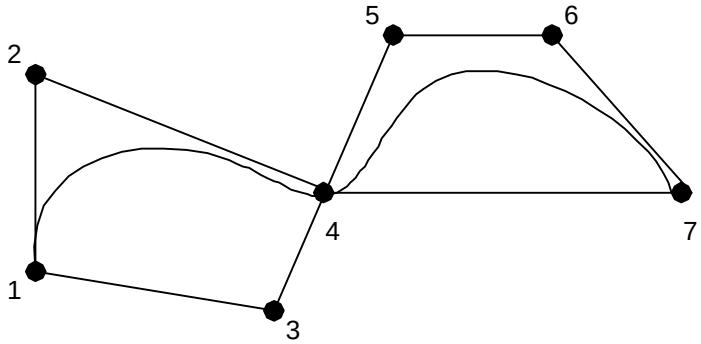


Рис 8.2

т.е точки 3 и 5 должны лежать на одной прямой (для данного случая). Это обеспечивает одинаковую касательную в точке стыковки.

3) Форма сплайна:

Идея: хотим провести гладкую прямую через набор точек.

Пример 1:

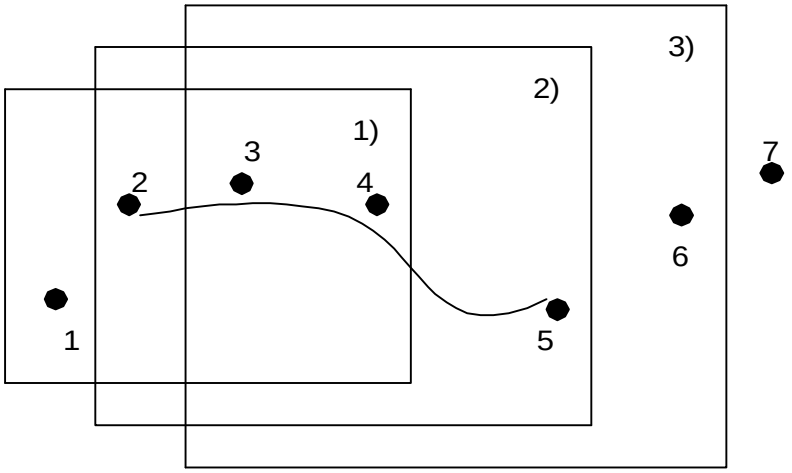


Рис 8.3

1) Берём первые 4-ре точки и по ним считаем уравнение кусочка кривой, но это уравнение опишет нам кусочек кривой между точками 2 и 3.
2) Берём следующие 4-ре точки и получаем кусочек кривой между точками 3 и 4, причём можно подобрать такие коэффициенты , что в точке 3 стыковка будет гладкостью $C^{(2)}$.

Тогда:

$$p(t) = T * M_S * G_{HS}$$

$$G_{HS} = \begin{bmatrix} p_1 \\ p_2 \\ p_3 \\ p_4 \end{bmatrix} ; \; M_{S1} = \frac{1}{6}\begin{bmatrix} -1 & 3 & -3 & 1 \\ 3 & -6 & 3 & 0 \\ -3 & -6 & 3 & 0 \\ 1 & 4 & 1 & 0 \end{bmatrix}$$

Примечание:

В рассмотренном выше примере все точки являются управляющими, т.е. кривая проходит вблизи точек, а не по ним.

Обеспечивается гладкость $C^{(2)}$.

Относительно решения проблемы кусочка кривой между точками 1 и 2 можно предложить следующие варианты:

1) можно добавить фиктивные точки;
2) сделать замкнутую кривую;
3) “слить” две точки в одну.

Пример 2:
Нам нужно чтобы кривая прошла через какие-то фиксированные точки:

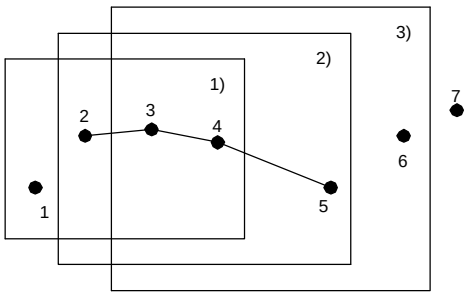


Рис 8.4

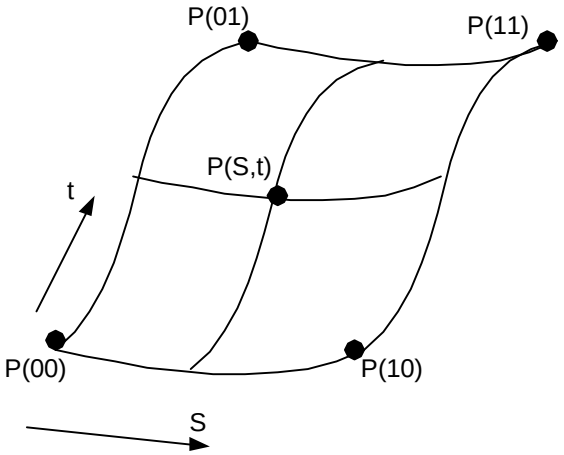
Процесс обработки тот же, что и в примере 1.
Отличие в том, что в данном случае все точки опорные и кривая проходит не вблизи , а точно по точкам. Следовательно мы проигрываем в гладкости, обеспечивая только уровень $C^{(1)}$.

Для данного случая:

$$M_{S2} = \begin{bmatrix} -\frac{1}{2} & \frac{3}{2} & -\frac{3}{2} & \frac{1}{2} \\ 1 & -\frac{5}{2} & 2 & -\frac{1}{2} \\ -\frac{1}{2} & 0 & \frac{1}{2} & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix} \quad ; \quad p(t) = T * M_S * G_{HS}$$

9. Параметрические би-кубические поверхности.

Поверхность может быть разбита на куски, каждый из которых будет описан
параметрическим би - кубическим уравнением.
Отдельно идёт работа по X, по Y, по Z для представления поверхности.



Рассмотрим параметр X:
Би-кубическое уравнение для описания X:

$$X(S,t) = a_{11}\cdot S^3\cdot t^3 + a_{12}\cdot S^3\cdot t^2 + a_{13}\cdot S^3\cdot t + a_{14}\cdot S^3 +$$
$$+ a_{21}\cdot S^2\cdot t^3 + a_{22}\cdot S^2\cdot t^2 + a_{23}\cdot S^2\cdot t + a_{24}\cdot S^2 +$$
$$+ a_{31}\cdot S\cdot t^3 + a_{32}\cdot S\cdot t^2 + a_{33}\cdot S\cdot t + a_{34}\cdot S +$$
$$+ a_{41}\cdot t^3 + a_{42}\cdot t^2 + a_{43}\cdot t + a_{44}$$

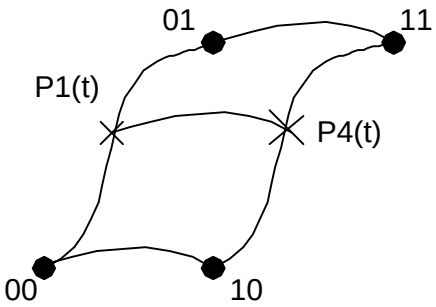
$X(S,t) = S\cdot C_X\cdot T^T$, где

$S = \begin{bmatrix} S^3 & S^2 & S & 1 \end{bmatrix}$, $T = \begin{bmatrix} t^3 & t^2 & t & 1 \end{bmatrix}$

$$C_X = \begin{bmatrix} a_{11} & a_{12} & a_{13} & a_{14} \\ a_{21} & a_{22} & a_{23} & a_{24} \\ a_{31} & a_{32} & a_{33} & a_{34} \\ a_{41} & a_{42} & a_{43} & a_{44} \end{bmatrix}$$ - матрица коэффициентов

Рассмотрим 3 способа расчёта коэффициентов:

1) В форме Эрмитта:



$$p(S,t) = S \cdot M_H \cdot G_{HX}(t)$$

Наши начальные данные:

$$G_{HX} = \begin{bmatrix} P_1(t) \\ P_4(t) \\ R_1(t) \\ R_4(t) \end{bmatrix}$$

В свою очередь каждая из этих функций может быть описана:

$$P_1(t) = T \cdot M_H \cdot \begin{bmatrix} P_{00} \\ P_{01} \\ R^t_{00} \\ R^t_{01} \end{bmatrix}$$, R – по параметру t.

$$P_{00} = \{x_{00}, y_{00}, z_{00}\}$$

$$R_{00} = \{(x'_t)_{00}, (y'_t)_{00}, (z'_t)_{00}\}$$

Для P_{01} - аналогично.

$$P_4(t) = T \cdot M_H \cdot \begin{bmatrix} P_{10} \\ P_{11} \\ R^t_{10} \\ R^t_{11} \end{bmatrix}$$

$$R_1(t) = T \cdot M_H \cdot \begin{bmatrix} R^S_{00} \\ R^S_{01} \\ R^{St}_{00} \\ R^{St}_{01} \end{bmatrix}; \quad R_4(t) = T \cdot M_H \cdot \begin{bmatrix} R^S_{10} \\ R^S_{11} \\ R^{St}_{10} \\ R^{St}_{11} \end{bmatrix}$$, где

R^S - производная по S , а R^{St} - вторая производная по t.

Матрица исходных данных выглядит следующим образом:

$$G_H = \begin{bmatrix} P_{00} & P_{10} & R_{00}^S & R_{10}^S \\ P_{01} & P_{11} & R_{01}^S & R_{11}^S \\ R_{00}^t & R_{10}^t & R_{00}^{St} & R_{10}^{St} \\ R_{01}^t & R_{11}^t & R_{01}^{St} & R_{11}^{St} \end{bmatrix}$$

$$G_H(t) = \begin{bmatrix} P_1(t) \\ P_4(t) \\ R_1(t) \\ R_4(t) \end{bmatrix} = G_H \cdot M_H^T \cdot T^T \; ; \quad p(S,t) = S \cdot M_H \cdot G_H(t)$$, следовательно :

$p(S,t) = S \cdot M_H \cdot G_H \cdot M_H^T \cdot T^T$, где $M_H \cdot G_H \cdot M_H^t = C_P$ - матрица коэффициетов полинома

Например конкретно для компоненты X:

$X(S,t) = S \cdot M_H \cdot G_{HX} \cdot M_H^T \cdot T^T$, где $M_H \cdot G_{HX} \cdot M_H^T = C_X$ - матрица коэффициентов (была приведена выше).

$$G_{HX} = \begin{bmatrix} x_{00} & x_{10} & (x'_S)_{00} & (x'_S)_{10} \\ x_{01} & x_{11} & (x'_S)_{01} & (x'_S)_{11} \\ (x'_t)_{00} & (x'_t)_{10} & (x''_{St})_{00} & (x''_{St})_{10} \\ (x'_t)_{01} & (x'_t)_{11} & (x''_{St})_{01} & (x''_{St})_{11} \end{bmatrix}$$

2) В форме Безье:

$p(S,t) = S \cdot M_B \cdot G_{BP} \cdot M^T \cdot T^T$, где геометрический вектор Безье равен:

$$G_{BP} = \begin{bmatrix} P_{11} & P_{12} & P_{13} & P_{!4} \\ P_{21} & P_{22} & P_{23} & P_{24} \\ P_{31} & P_{32} & P_{33} & P_{34} \\ P_{41} & P_{42} & P_{43} & P_{44} \end{bmatrix}$$

Из этих точек 4 опорных (т.е. через них проходит плоскость):

$$P_{11}, P_{14}, P_{41}, P_{44}$$

Все остальные точки являются управляющими.

Рассмотрим два сопряжённых куска поверхности:

Для обеспечения сшивки поверхности на уровне $C^{(1)}$ надо, чтобы следующие точки лежали на одной прямой:

13, 14, 15;
23, 24, 25;
33, 34, 35;
43, 44, 45.

Соблюдение этого условее видно на следующем рисунке:

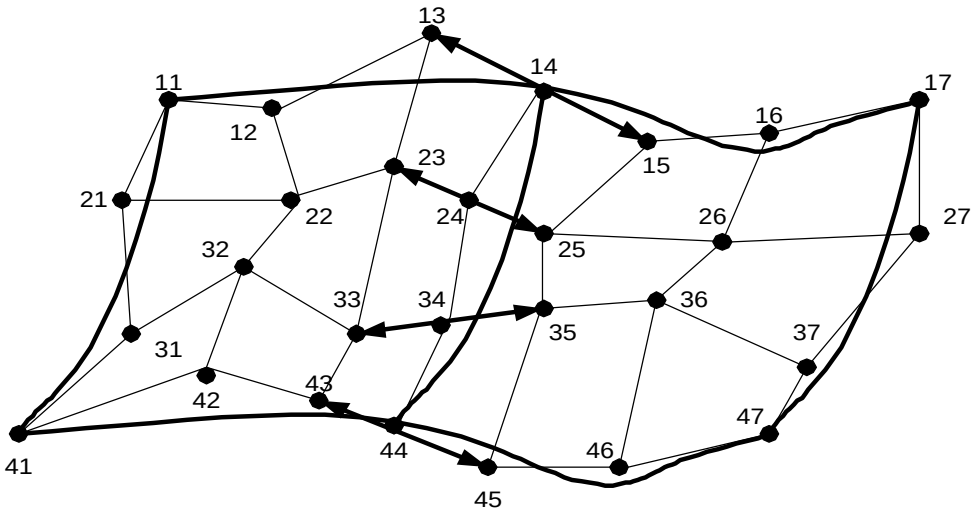


Рис 9.1

3) В форме сплайна:

$$p(S,t) = S \cdot M_S \cdot G_{SP} \cdot M_S^T \cdot T^T$$

$$G_{SP} = \begin{bmatrix} P_{11} & P_{12} & P_{13} & P_{14} \\ P_{21} & P_{22} & P_{23} & P_{24} \\ P_{31} & P_{32} & P_{33} & P_{34} \\ P_{41} & P_{42} & P_{43} & P_{44} \end{bmatrix}$$

Здесь можно обеспечить “сшивку”:

а) по уровню $C^{(2)}$, если применить: $M_S = M_{S1}$, т.е. все точки будут управляющие:

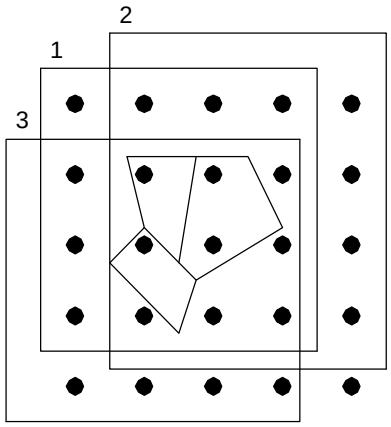
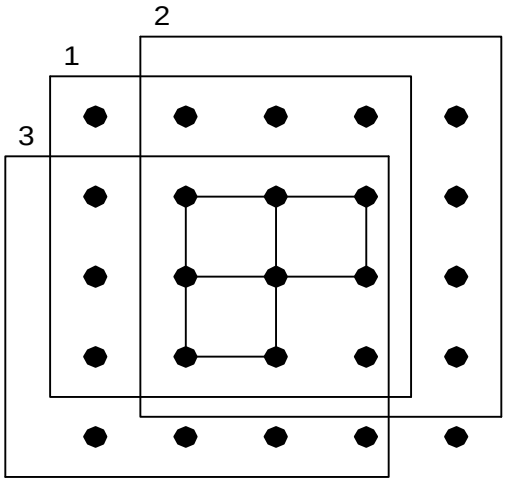


Рис 9.2

б) по уровню $C^{(1)}$, если применить : $M_S = M_{S2}$, т. е. все точки являются опорными:



Итерационные способы вычисления полиномов.

Вычисление кубического уравнения для прямой:

$$f(t) = at^3 + bt^2 + ct + d = d + t(c + t(b + ta))$$

$$d = \Delta t \; ; \; t = 0...1$$

$$\Delta f(t) \equiv f(t+d) - f(t) = 3at^2 d + t(3ad^2 + 2bd) + ad^3 + bd^2 + cd$$

$$\Delta^2 f(t) \equiv \Delta(\Delta f(t)) = \Delta f(t+d) - \Delta f(t) = 6ad^2 t + 6ad^3 + 2bd^2$$

$$\Delta^3 f(t) \equiv \Delta(\Delta^2 f(t)) = \Delta^2 f(t+d) - \Delta^2 f(t) = 6ad^3$$

$$t = rd \; ; r = 0 \ldots \frac{1}{d}$$

$$f_n = f(nd)$$

$$\Delta f_n = \Delta f(nd)$$

$$\Delta^2 f_n = \Delta^2 f(nd)$$

$$\Delta^3 f_n = 6ad^3 = const$$

$$f_{n+1} = f_n + \Delta f_n$$

$$\Delta f_{n+1} = \Delta f_n + \Delta^2 f_n$$

$$\Delta^2 f_{n+1} = \Delta^2 f_n + \Delta^3 f_n$$

Для n=0; t=0:

$$f_0 = d$$

$$\Delta f_0 = ad^3 + 6d^2 + cd$$

$$\Delta^2 f_0 = 6ad^3 + 2bd^2$$

$$\Delta^3 f_0 = 6ad^3$$

Для n=1; t=δ :

$$f_1 = f_0 + \Delta f_0$$

$$\Delta f_1 = \Delta f_0 + \Delta^2 f_0$$

$$\Delta^2 f_1 = \Delta^2 f_0 + \Delta^3 f_0$$

Для n=2 ; t=2δ :

$$f_2 = f_1 + \Delta f_1$$
$$\Delta f_2 = \Delta f_1 + \Delta^2 f_1$$
$$\Delta^2 f_2 = \Delta^2 f_1 + \Delta^3 f_0$$

Запишем эту схему вычислений в матричном виде:

$$\begin{bmatrix} f_0 \\ \Delta f_0 \\ \Delta^2 f_0 \\ \Delta^3 f_0 \end{bmatrix} = \begin{bmatrix} 0 & 0 & 0 & 1 \\ d^3 & d^2 & d & 0 \\ 6d^3 & 2d^2 & 0 & 0 \\ 6d^3 & 0 & 0 & 0 \end{bmatrix} \cdot \begin{bmatrix} a \\ b \\ c \\ d \end{bmatrix} \text{ или } D_0 = E(d) \cdot Q$$

При n=1 схема преобразования D_0 в D_1 выглядит следующим образом:

Стр.1 = Стр.1 + Стр.2
Стр.2 = Стр.2 + Стр.3
Стр.3 = Стр.3 + Стр.4

Следующая операция при n=2 :

Преобразование D_1 в D_2 происходит аналогично тому, что выше.

Для бикубических полиномов:

$$D_{00} = \begin{bmatrix} f_0 & \Delta_t f_0 & \Delta_t^2 f_0 & \Delta_t^3 f_0 \\ \Delta_S f_0 & \Delta_S \Delta_t f_0 & \Delta_S \Delta_t^2 f_0 & \Delta_S \Delta_t^3 f_0 \\ \Delta_S^2 f_0 & \Delta_S^2 \Delta_t f_0 & \Delta_S^2 \Delta_t^2 f_0 & \Delta_S^2 \Delta_t^3 f_0 \\ \Delta_S^3 f_0 & \Delta_S^3 \Delta_t f_0 & \Delta_S^3 \Delta_t^2 f_0 & \Delta_S^3 \Delta_t^3 f_0 \end{bmatrix}$$

При этом:

$$\Delta_S^3 \Delta_t^3 f_0 = const$$
$$D_{00} = E(e) \cdot C_P \cdot E^T(d)$$
$$t = d \cdot n$$
$$S = e \cdot m$$

10.**Полигональное задание пространственных форм.**

10.1 Рельефы:

- регулярные;
- нерегулярные.

1) Представление на регулярных сетках:

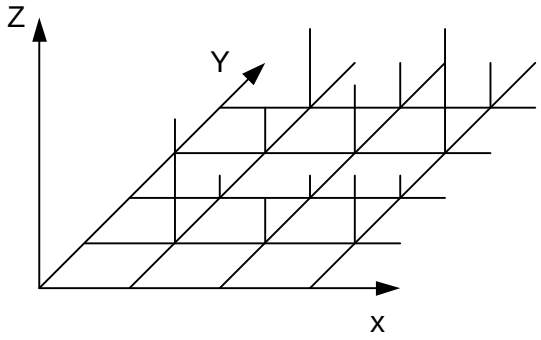


Рис 10.1.1

Вид сверху:

ТСР-триангулярная регулярная сетка:

а)

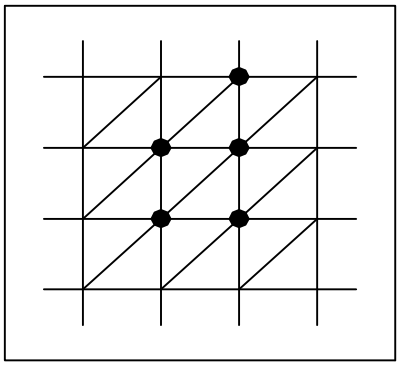


Рис 10.1.2

б) но предпочтительнее правильные треугольники:

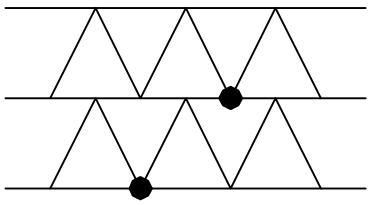


Рис 10.1.3

Плюс ТСР:

Простота паостроения.

Недостаток:

Большой обьём данных.

2) THC – триангулированная нерегулярная сетка:

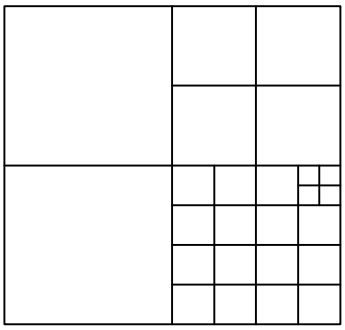


Рис 10.1.4

Здесь все точки могут быть не привязаныт ни к каким дескрипторам.

Пример:

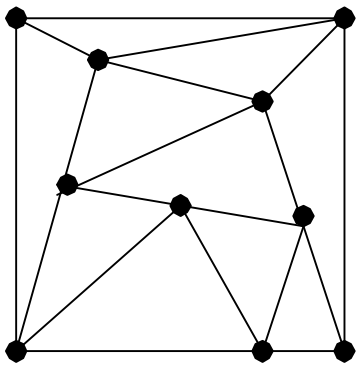


Рис 10.1.5

Метод триангуляции Делоне.

Суть :

Позволяет получать триангуляцию, все треугольники стремятся к правильной форме.

В основе метода лежит круговой критерий:

Если провести окружность вокруг 3-ч точек, то другие точки не должны попадать в него.

Алгоритм:

1) Все точки, которые надо стриангулировать, лежат внутри прямоугольника:

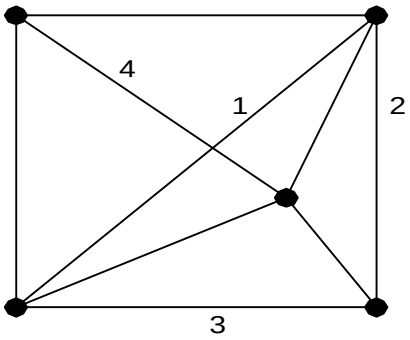


Рис 10.1.6

2) После этого проводят начальную триангуляцию: делим прямоугольник пополам;
3) Берётся точка (например A) и проводится триангуляция:
   а) Определяем , в какой треугольник попала эта точка;
   б) Делим этот треугольник на 3 треугольника;
   в) Помещаем в стек флипов 3 ребра (на рис. 1, 2, 3);
   г) Просматриваем стек флипов и , используя круговой критерий, решаем надо флиповать ребро или нет (если точка не попала в окружность, то флип не нужен, а если попала – нужен)
   Флип – переброска ребра.
   На этом основании в рассматриваемом примере произошла замена ребра 1 на ребро 4.

Примечание:
Если у нас есть N вершин, то сколько у нас будет рёбер (R) и сколько треугольников (T), а так же найдём количество соседних треугольников (K), приходящихся на одну вершину.
Пусть у нас есть треугольник. На каждую точку, взятую в этом треугольнике добавляется 2 треугольника. Следовательно, T=2N.
Число рёбер: R=3N, так как при добавлении каждой точки добавляется ещё и 3 новых ребра.
У каждой точки есть определённое количество соседних вершин, а общее количество прикркплений рёбер к треугольникам будет равно: 2R.
Следовательно: $K = \frac{2R}{N} = \frac{6N}{N} = 6$ (т.е. у каждой точки есть по “шесть соседий”)

Проверка кругового критерия может быть заменена на проверку расстояний (длины диагоналей сравниваются и выбираются более короткие),но это уже не триангуляция Делоне.

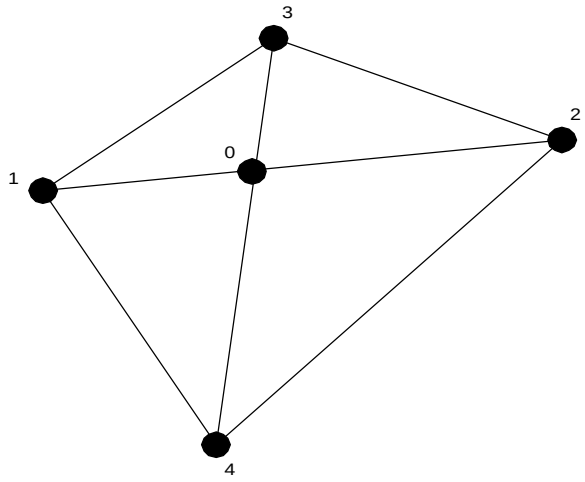


Рис 10.1.7

Флипп производится с учётом критерия фллиппа (связан с окружностью)
Решение кругового критерия можно свести к следующему решению:

Введём следующие обозначения :

d_{01} = отрезок 01; d_{02} = отрезок 02 и т. д.

Необходимо решить вопрос о том : Какое ребро выбрать? (12 или 34)
В этом нам поможет следующее выражение:

$$d_{01} \cdot d_{02} - d_{03} \cdot d_{04} = \begin{cases} < 0, \textit{оставляем } 12 \\ > 0, \textit{оставляем } 34 \\ = 0, \textit{любое} \end{cases}$$

10.2 Области Вороного.

Область Вороного строится соединением серединных перпендикуляров в исход-
ных треугольниках.

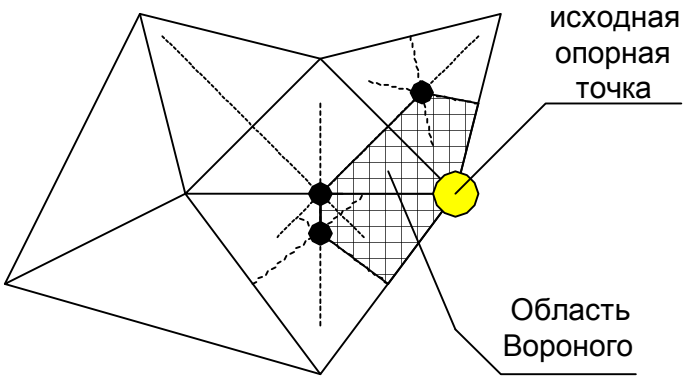


Рис 10.2.1

Свойство области Вороного:
Область Вороного гарантирует, что любая точка этой области лежит ближе к то**чке, вокруг которой строилась эта область, чем любая точка, не лежащая в этой**
области.
Дуализм проявляется в том , что:
По триангуляции Делоне можно построить области Вороного и наоборот.

10.3 Представление рельефа с мультиразрешением.

Мультиразрешение – представление с различной степенью детализации.

Основная задача:
Сортировка тачек по степени важности.

Данное представление можно использовать при изображении рельефа:
Разбиваем область на несколько зон. В ближних зонах используются триангуляция всех точек нулевого ранга(т.е. полное разрешение), далее используем триангуляцию точек за вычетом нулевого ранга, а на самом дольнем плане используем только самые важные точки.
Причём, при приближении объекта точки добавляются, а при удолении объекта удаляется так же и часть точек(ненужных).

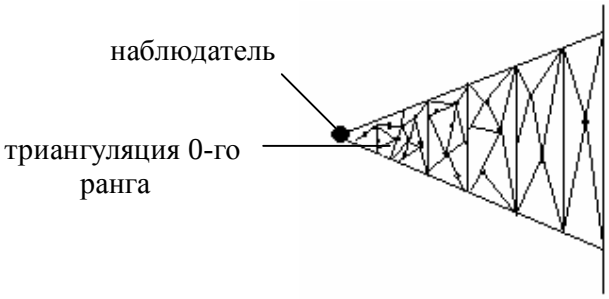


Рис 10.3.1

Существует и такое понятие, как **переменное мультиразрешение** – один объект(рельеф), но представляется с различным разрешением в разных частях этого объекта.

Теперь рассмотрим такую задачу, как удаление точки из триангуляции. Это сделать достаточно сложно,но можно.

Допустим, хотим удалить следующую точку(на рис. помечена красным):

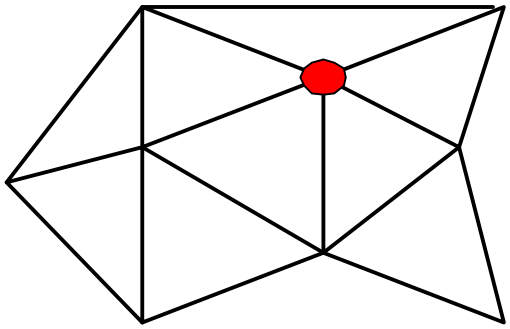


Рис 10.3.2

Для этого нужно:
“вырезать” эту точку,а образовавшуюся пустоту стриангулировать произвольным образом. Оставшиеся после удаления точки рёбра поместить в стек флипов,а затем профлиповать после чего пустота затянется.

Безусловно, сеществует несколько способов удаления точек, но самый эффективный это **Ранжирование точек.**

Принцип по которому строится этот способ состоит в следующем:
В первую очередь удаляются точки, которые лежат в одной плоскости.

Наша задача – оценить важность точки, а значит наити ценность для каждой точки.

$$S_j = \sum_{i=0}^{m-1} \left[N_{nj} (V_i - V_j) \right]^2$$

N_{nj} -нормированная нормаль для j-ой вершины;

V_i - соседние вершины (число соседних вершин m);

V_j - сама точка .

$$N_{nj} = Norm\left[\sum_{i=0}^{m-1}\left(V_i - V_j\right)*\left(V_{i-1} - V_j\right)\right]$$

Функция нормировки:

$$Norm(Vektor) = \frac{Vektor}{|Vektor|}$$

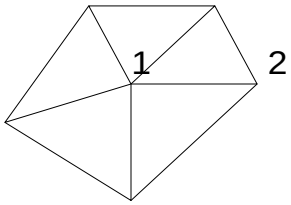
Алгоритм ранжирования точек:

1) Построить триангуляцию полного разрешения, т.е. всех вершин. Присвоить всем вершинам максимальный ранг;
2) Для каждой вершины посчитать среднюю нормаль(она считается один раз и является постоянной характеристикой вершины);
3) Для всех вершин рассчитать стоимость;
4) Установить текущий ранг = 0;
5) Удалить из триангуляции часть вершин, имеющих наименьшую стоимость. Пересчитать стоимость соседних вершин по отношению к удалённой. Присваиваем им текущий ранг;
6) Посчитать ошибку с данным рангом триангуляции, т.е. для каждой вершины (удалённой) посчитать расстояние по её триангуляции;
7) Увеличиваем текущий ранг на единицу;
8) Проверка текущего ранга(если текущий ранг не равен максимальному рангу, то возвращаемся к 5-му пункту.

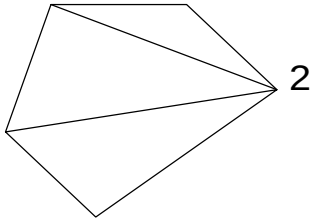
Этот алгоритм можно усложнить. Например:
Оценивать ошибки удалённых вершин,а если, вершина была удалена неудачно, то её возвращаем обратно.

10.4 Объекты.

В отличие от рельефа объект изображается с использованием одного разрешения. Как правило создаётся много моделей одного объекта.
В этом случае точка удаляется следующим образом:



Допустим, удаляем ребро {1 2}. В этом случае всё что имело связь с вершиной 1 перейдёт в вершину 2.



Т.е. происходит процесс калабса ребра: Нужно выбрать ребро, которое подвергнем калабсу. Для этого определяем стоимость рёбер. Оценку ребра можно сделать, используя среднюю нормаль, но мы рассмотрим математическую основу (т.е. основанную на квадриках).

Пусть плоскость задана нормалью $\vec{N}$ и числом D.
Тогда уравнение плоскости будет выглядеть следующим образом:

$$N_x * x + N_y * y + N_z * z + D = 0$$

D – свободный член в уравнении плоскости.
Рассмотрим расстояние от точки V до плоскости.
Пусть точка V имеет следующие координаты V(x, y, z). Тогда расстояние будет определяться следующим образом:

$$\vec{N} * \vec{V} + D$$

Но нас больше интересует квадрат этого расстояния:

$$S = \left(\vec{N} * \vec{V} + D\right)^2 = V^T \left(N * N^T\right) * V + 2 * \left(D * N^T\right) * V + D^2$$

$$V = \begin{bmatrix} x \\ y \\ z \end{bmatrix} \qquad N = \begin{bmatrix} N_x \\ N_y \\ N_z \end{bmatrix}$$

Обозначим:

$$N * N^T = A$$

$$B \equiv D * N^T$$

$$C \equiv D^2$$

Тогда выражение для квадрата расстояния будет выглядеть следующим образом:

$$S = V^T * A * V + 2 * B * V + C = S(QV)$$

Это формула вычисления квадрата расстояния с помощью квадрика плоскости и точки плоскости.

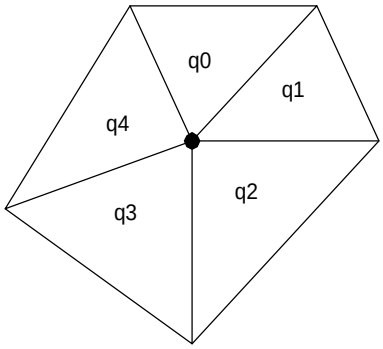
Квадрик – совокупность величин A, B, C:

$$Q = \{A, B, C\}$$

Таким образом квадрик определяется только параметрами плоскости.
При работе с объектами вводится следующее понятие:
Если к одной вершине присоединены несколько треугольников, то квадрик этой
вершины будет равен сумме всех квадриков прилегающих к этой вершине треугольников:

$$Q_V = \sum_{i=0}^{m-1} q_i$$

Например:



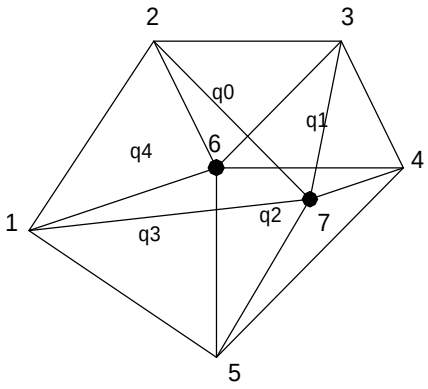
В данном случае:

$$Q_V = q0 + q1 + q2 + q3 + q4$$

А так же мажно записать следующее выражение:

$$S(q1,V) + S(q2,V) = S(q1 + q2,V)$$

Рассмотрим ещё один пример:



Допустим, мы хотим переместить точку 6 в точку 7.
Тогда:

$$Q_7 = q0 + q1 + q2 + q3 + q4$$

А для точки 7 мы можем посчитать следующее:

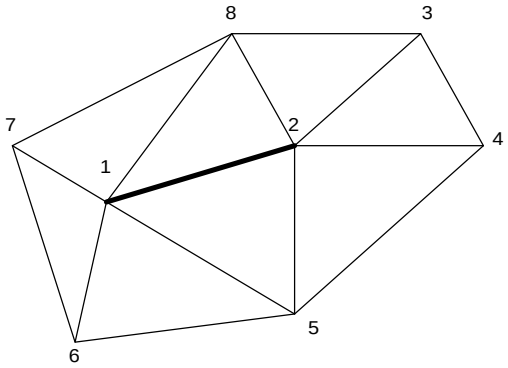
$$S(q0, V_7) + S(q1, V_7) + ... + S(q5, V_7) =$$
$$= S(Q_6, V_7) = S(Q_7, V_6)$$

Эта величина показывает на сколько вершина 7 отклоняется от окружающих плоскостей вершины 6.

Следовательно, квадрики позволяют оценить ошибки перемещения вершин, которые находятся в тренгуляции.

Механизм колабса ребра.

Цель: выбор ребра, от которого можно избавмться, но это избавление должно принести наименьшую ошибку. Рассмотрим пример:



Во-первых, нам нужно найти квадрики всех вершин, а затем и рёбер.
Для примера возьмём ребро {1-2}:

Квадрик этого ребра:

$$Q_{1-2} = Q_1 + Q_2$$

Во-вторых, для каждого ребра считаем критерий, выбирая лучший переброс:

Критерий для ребра {1-2}:

$$S_{1-2} = \min\{S(Q_{1-2}, V_1), S(Q_{1-2}, V_2)\} = \min\{S(Q_2, V_1), S(Q_1, V_2)\},$$

так как

$$S(Q_1, V_1) = 0$$
$$S(Q_2, V_2) = 0$$

Затем выбираем ребро с наименьшим значением S_{1-2}, которое и будет удалено.
Замыкание будет зависеть от следующего критерия:

$S(Q_2, V_1) < S(Q_1, V_2)$ - удаляется V_1

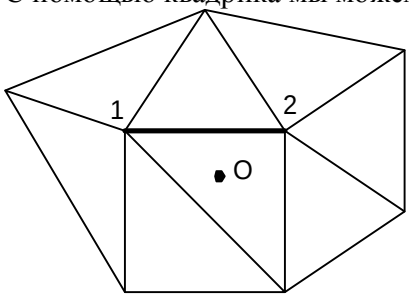
$S(Q_1, V_2) < S(Q_2, V_1)$ - удаляется V_2

Результат:

Мы выбрали такое перемещение, которое приносит наименьшую ошибку.

Примечание:
При переброске квадрики вершин изменяются, следовательно, их нужно пересчитать, а значит стоимость рёбер так же поменяется.

С помощью квадрика мы можем порождать новую вершину:



Допустим, мы хотим перекинуть все связи в точку О. Координаты этой точки нам не известны, но мы можем их найти следующим образом:

$$V_0 = \arg\min S(Q_{1-2}, V_0)$$

Получается, что $V_0^T = -B * A^{-1}$

Минимальный критерий для точки V_0:

$$S(Q, V_0) = C - B * A^{-1} * B^T = C + V_0^T * B$$

Т.е. квадрик несёт в себе информацию об оптимальной точке, в которую можно свести все связи.

Существует механизм работающий на квадриках и записи информации предыдущего шага. Т.е., допустим, мы стягиваем все связи, принадлежащие точкам 1 и 2, в точку О. При этом мы можем запомнить эту информацию и при обратном щаге уже будем знать как разлажить точку О(т.е. на точку 1 и точку 2).

10.5 Гипертриангуляция

Заключает в себе триангуляции всех уровней разрешения. Но выигрыш в её применении, по сравнению с динамической триангуляцией, невелик.

Резюме:

1) Существуют:
   а) Рельеф:
      представляется с разным уровнем разрешения (мультиразрешение) и наиболее удачна для этого триангуляция Делоне.
   б) Объект:
      представляется с постоянным уровнем разрешения, триангуляция Делоне не применяется, но используется математическое упрощение с помощью квадриков.
2) Механизм квадриков можно перекинуть на работу с рельефом, а работу с нормалью перекинуть на объекты.