Компьютерная графика

Курс лекций

Тема №9. Формирование реалистичных изображений.

Средства повышения реалистичности изображения

- использование освещения:
  - модели освещения (диффузное, направленное);
  - отражение (диффузное, зеркальное) – свойства материала;
  - преломление;
  - прозрачность (blending, комбинация цветов в различных режимах), цвет;
  - построение теней;
- фактуры (текстуры) – нанесение некоторого изображения на поверхность или внесение возмущения в параметры поверхности;
- мультитекстурирование (multi-texturing), наложение карты окружения (environment mapping);
- движение, анимация;
- использование фракталов, шумовых функций (шум Перлина – Perlin noise);
- ...

Анимация

- анимация - изменение положения и свойств (формы, цвета, прозрачности и т.п.) объектов;
- методы анимации:
  - *анимация реального времени*: построение и отображение последовательности сцен по мере генерации в реальном времени;
  - *покадровая анимация*: сохранение последовательности сгенерированных кадров с последующим отображением;
- анимация и OpenGL: двойная буферизация.

Анимация: процедура

- *раскадровка*: определение набора базовых событий, которые должны произойти в сцене;
- *определение объектов* (через базовые формы, такие как полигональные или сплайновые поверхности) и *спецификация действия* (определение траекторий движения объектов и камеры);
- *определение ключевых кадров*: подробное изображение сцены в определенный момент анимационной последовательности, выбор которого задается:
  - крайними положениями движения;
  - исходя из необходимости обеспечить не слишком большой интервал между кадрами;
- *генерация промежуточных кадров* (последовательностей кадров между ключевыми кадрами, при этом траектория задается на основе кинематического описания или физических законов).

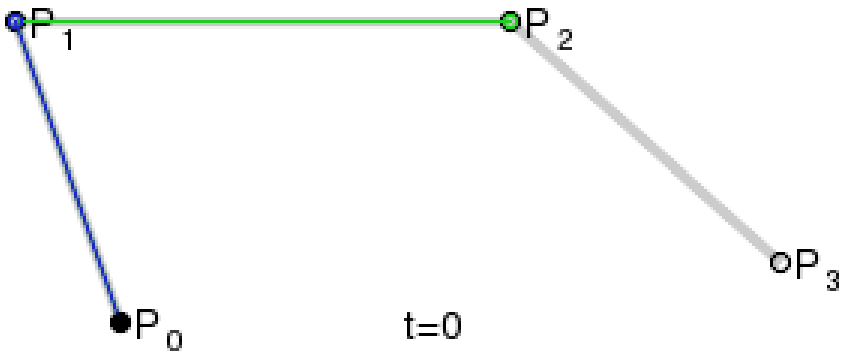
Движение

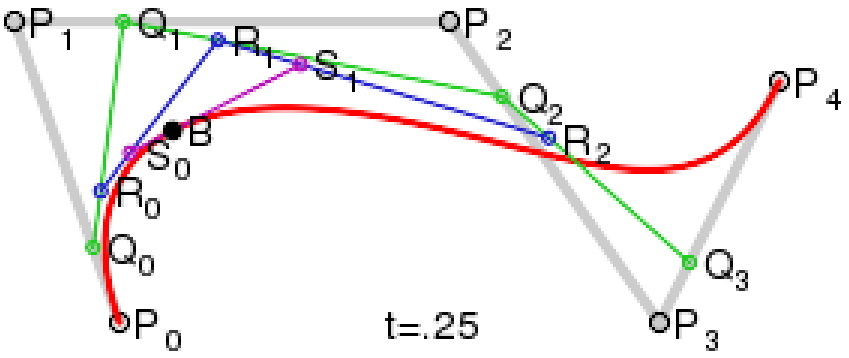
- синхронизация (timing) - связь реального времени с частотой смены кадра;
- спецификация движения:
  - прямая спецификация движения: явное задание матриц геометрических преобразований (углов движения и векторов перемещения) объектов сцены, или задание аппроксимирующих кривых;
  - целенаправленная спецификация: определение набора действий и поддействий, приводящих к некоторому результату;
  - спецификация на основе физических законов:
    - кинематическое описание (задание положения, скорости, ускорения; или задание траектории);
    - обратное кинематическое описание (определение параметров движения через заданное начальное и конечное положение объекта);
    - динамическое описание (определение сил, использование законов сохранения) – физическое моделирование;
    - обратное динамическое описание (определение действующих сил через заданное начальное и конечное положение объекта).

Твининг-анимация

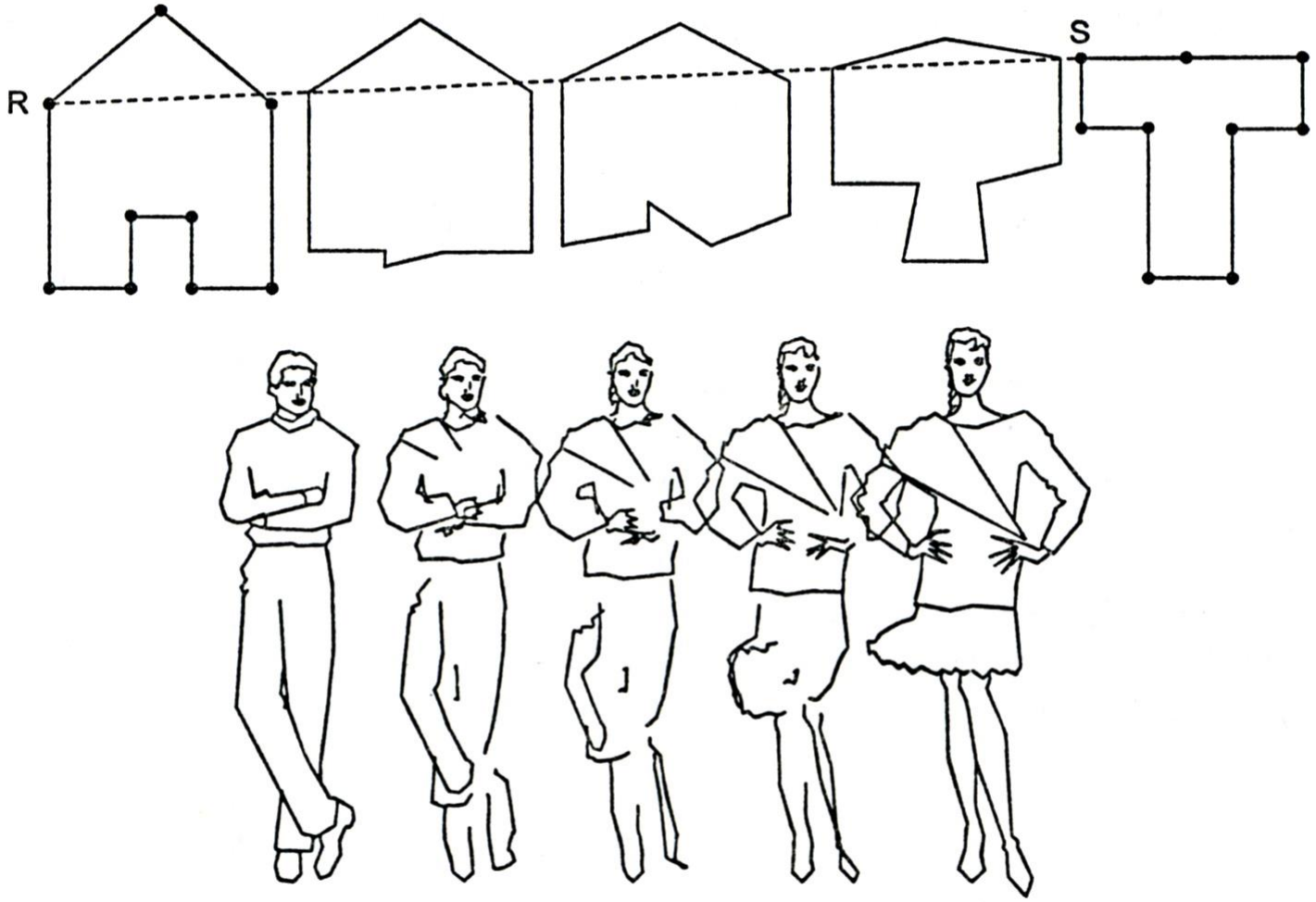
- *твининг-анимация*: преобразование формы фигуры (ломаной), при котором каждая точка (вершина) перемещается по траекториям, представляющим собой кривые Безье:
  - линейная: (1-t) + t
  - квадратичная: $((1-t) + t)^2 = (1-t)^2 + 2(1-t)t + t^2$
  - кубическая: $((1-t) + t)^3 = (1-t)^3 + 3(1-t)^2t + 3(1-t)t^2 + t^3$

$$\mathbf{B}(t) = \sum_{i=0}^{n} \mathbf{P}_i \mathbf{b}_{i,n}(t), \quad t \in [0,1]$$





Твининг-анимация: пример



$p(t) = (1 - t) \cdot A + t \cdot B$, где $t \in [0, 1]$

Учет физических законов

- время (синхронизация: связь реального времени с частотой смены кадра);
- масса (определение центра масс):

$$\boldsymbol{r_c} = \mathrm{SUM}(\boldsymbol{c_i} \cdot m_i) \,/\, \mathrm{SUM}(m_i)$$

- скорость (привязывается к частоте смены кадров):

$$v_x = \cos(\alpha) \cdot |\boldsymbol{v}|$$
$$v_y = \sin(\alpha) \cdot |\boldsymbol{v}|$$

  - в программе:

```
p.x += v.x; p.y += v.y;
```

- ускорение:

$$\boldsymbol{a} = d/dt \cdot \boldsymbol{v} \qquad (\boldsymbol{F} = d/dt \cdot (m\boldsymbol{v}))$$
$$vt = v_0 + a \cdot t$$
$$xt = x_0 + v_0 \cdot t + a \cdot t^2 / 2$$

  - в программе:

```
p += velocity; velocity += acceleration;
```

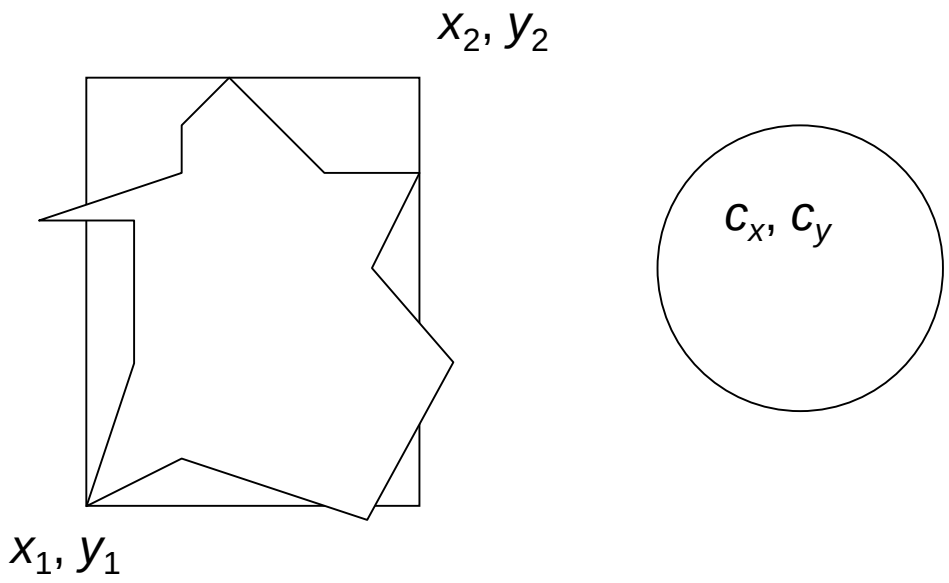
Обнаружение столкновений

- обнаружение столкновения двух окружностей:

$$(p_1.x - p_2.x)^2 + (p_1.y - p_2.y)^2 < (r_1 + r_2)^2$$

- обнаружение столкновения окружности и многоугольника (в частном случае – регулярного прямоугольника):

$$(x_1 - r < c.x < x_2 + r) \;\&\; (y_1 - r < c.y < y_2 + r)$$



Абсолютно упругие соударения

- при абсолютно упругих соударениях:
  - не происходит деформации объектов при соударении;
  - применимы законы сохранения импульса и энергии;
- центральное соударение шаров:
  - законы сохранения импульса и массы:

    $v_{1i} \cdot m_1 + v_{2i} \cdot m_2 = v_{1f} \cdot m_1 + v_{2f} \cdot m_2$

    $m_1 \cdot v_{1i}^2 + m_2 \cdot v_{2i}^2 = m_1 \cdot v_{1f}^2 + m_2 \cdot v_{2f}^2$
  - решение:

    $v_{1f} = (2 \cdot m_2 \cdot v_{2i} + (m_1 - m_2) \cdot v_{1i}) / (m_1 + m_2)$

    $v_{2f} = (2 \cdot m_2 \cdot v_{1i} + (m_2 - m_1) \cdot v_{2i}) / (m_1 + m_2)$
- соударение шара с наклонной плоскостью:

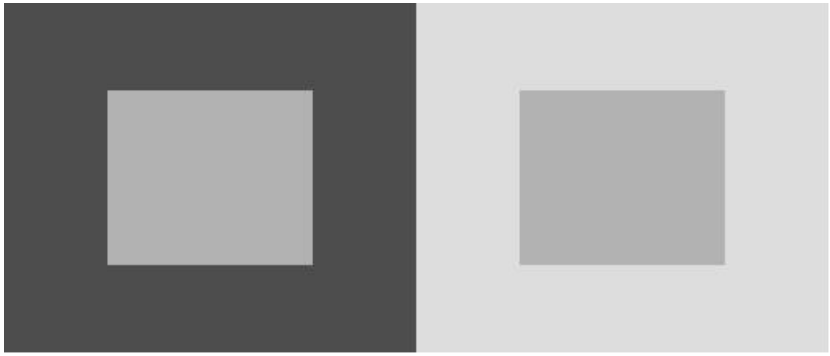
  $y + k \cdot x + b = 0$

  $(x - x_c)^2 + (y - y_c)^2 = R^2$

  + 2 параметрических уравнения перемещения центра

  + D = 0 (касание)
- соударение шара с горизонтальной / вертикальной плоскостью:
  - угол падения равен углу отражения;
  - модуль скорости после соударения не изменяется.

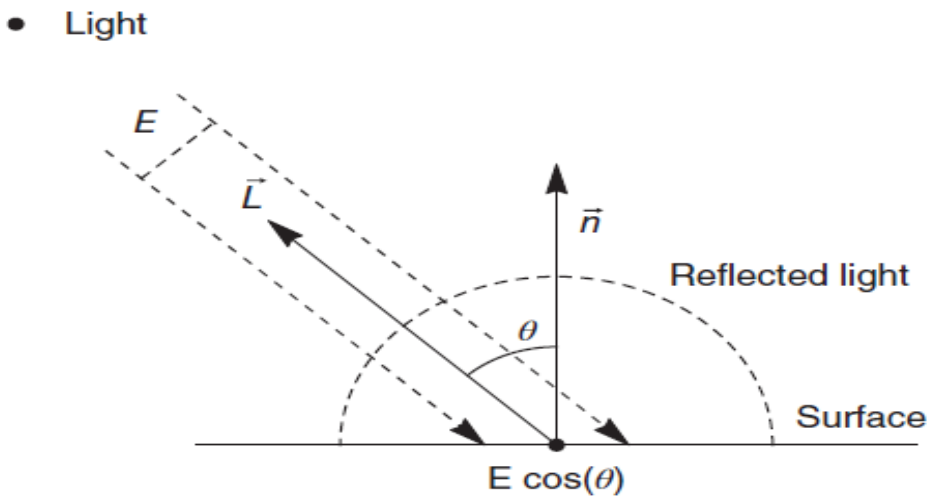
Построение реалистических изображений



- восприятие света включает как физические, так и психологические процессы, что необходимо учитывать при построении реалистических изображений;
- примеры влияния восприятия:
  - эффект полос Маха (Mach bands);
  - одновременный контраст;
- при моделировании освещения используют упрощенные эмпирические модели, отражающие компромисс между реалистичностью и объёмом необходимых вычислений:
  - прямолинейное распространение света (light rays);
  - точечные источники света;
  - простая модель освещения (модель Фонга), BRDF (bidirectional reflectance distribution function), излучательность (radiosity);
  - интерполяционное закрашивание (Гуро, Фонг).

Освещение

- световая энергия, падающая на поверхность, может быть:
  - поглощена (absorption);
  - отражена (reflection);
  - пропущена (transmission);
- количество поглощенной, отраженной или пропущенной энергии зависит от длины волны света (соответственно, изменяется распределение энергии и объект выглядит цветным, а цвет объекта определяется поглощаемыми длинами волн);
- если объект поглощает весь падающий свет, то он невидим и называется *абсолютно черным телом;*
- свойства отраженного света зависят:
  - от строения, направления и формы источника света;
  - от ориентации и свойств поверхности (диффузное и зеркальное отражение).



Диффузное отражение (diffuse scattering): закон косинусов Ламберта

- диффузное отражение света происходит, когда свет как бы проникает под поверхность объекта, поглощается, а затем вновь испускается;
- диффузно отраженный свет рассеивается равномерно по всем направлениям, следовательно, положение наблюдателя не имеет значения;
- свет точечного источника отражается от идеального рассеивателя по закону косинусов Ламберта (Lambert): интенсивность отраженного света пропорциональна косинусу угла между направлением света и нормалью к поверхности:

$$I = I_l k_d \cos\theta$$

  - где **I** — интенсивность отраженного света,
  - $\mathbf{I_l}$ — интенсивность точечного источника,
  - $\mathbf{k_d}$ — коэффициент диффузного отражения ($0 \le \mathbf{k_d} \le 1$),
  - **θ** — угол между направлением света и нормалью к поверхности

- цвет определяется свойствами материала;
- коэффициент диффузного отражения k_d зависит от материала и длины волны света, но в простых моделях освещения обычно считается постоянным.

Рассеянный свет (фоновый, ambient light)

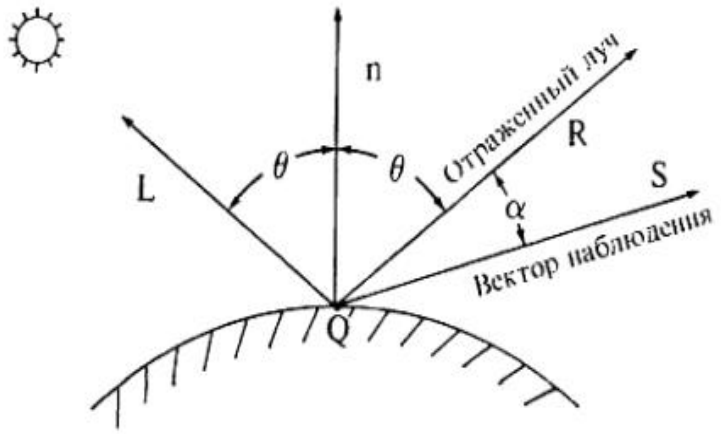
- на объекты реальных сцен падает рассеянный свет, отраженный от окружающей обстановки;
- рассеянному свету соответствует распределенный источник;
- поскольку для расчета таких источников требуются большие вычислительные затраты, в машинной графике они обычно заменяются на константный коэффициент рассеяния:

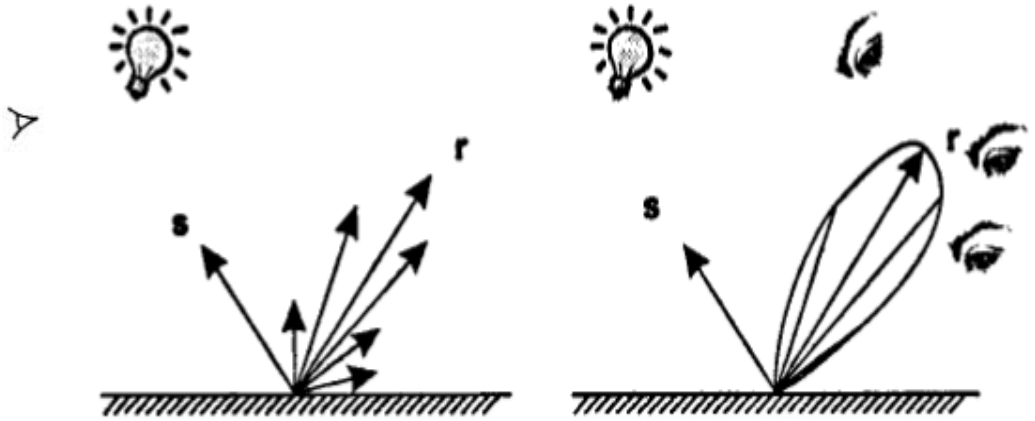
$$I = I_a k_a$$

  - где **I** — результирующая составляющая интенсивности рассеянного света,
  - $\mathbf{I_a}$ — интенсивность рассеянного света,
  - $\mathbf{k_a}$ — коэффициент диффузного отражения рассеянного света ($0 \le \mathbf{k_a} \le 1$),
- цвет определяется свойствами материала.

Зеркальное отражение (specular reflection)

- зеркальное отражение происходит от внешней поверхности объекта (за исключением металлов и некоторых твердых красителей), следовательно, отраженный луч сохраняет свойства падающего: цвет определяется свойствами источника;
- интенсивность зеркально отраженного света зависит от:
  - угла падения;
  - длины волны падающего света;
  - свойств вещества;
- зеркальное отражение света является направленным:
  - угол отражения от идеальной отражающей поверхности (зеркала) равен углу падения, в любом другом положении наблюдатель не видит зеркально отраженный свет;
  - если поверхность не идеальна, то количество света, достигающее наблюдателя, зависит от пространственного распределения зеркального отраженного света (у гладких поверхностей распределение узкое или сфокусированное, у шероховатых — более широкое).



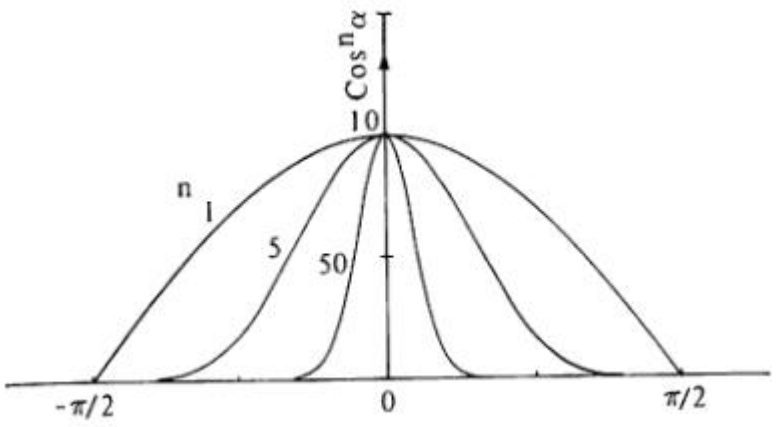


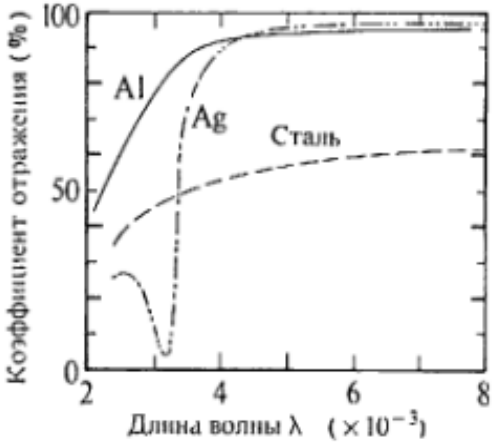
Зеркальное отражение: модель Фонга

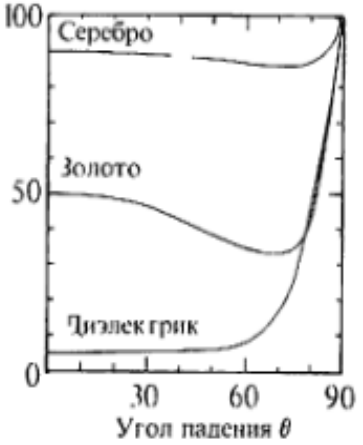
- в простых моделях освещения обычно пользуются эмпирической моделью Фонга (Phong), так как физические свойства зеркального отражения очень сложны:

$$I_s = I_l w(\theta, \lambda) \cos^n \alpha$$

  - где ***w (θ, λ)*** — кривая отражения, представляющая отношение зеркально отраженного света к падающему как функцию угла падения ***θ*** и длины волны ***λ***; (коэффициент зеркального отражения зависит от угла падения, однако даже при перпендикулярном падении зеркально отражается только часть света, а остальное либо поглощается, либо диффузно отражается)
  - ***n*** - степень, аппроксимирующая пространственное распределение зеркально отраженного света (большие значения ***n*** дают сфокусированные пространственные распределения характеристик металлов и других блестящих поверхностей, а малые — более широкие распределения для неметаллических поверхностей)
- при падении под скользящим углом (в 90°) отражается весь падающий свет (коэффициент отражения 100%);
- функция ***w (θ, λ)*** довольно сложна, поэтому ее обычно заменяют константой $\mathbf{k_s}$, которая либо выбирается из эстетических соображений, либо определяется экспериментально.





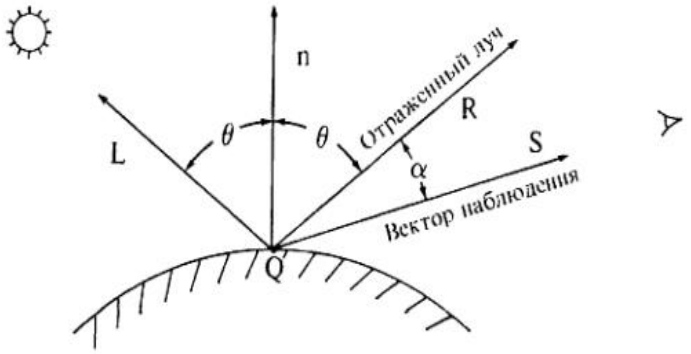


Вычисление вектора отражения и промежуточного вектора

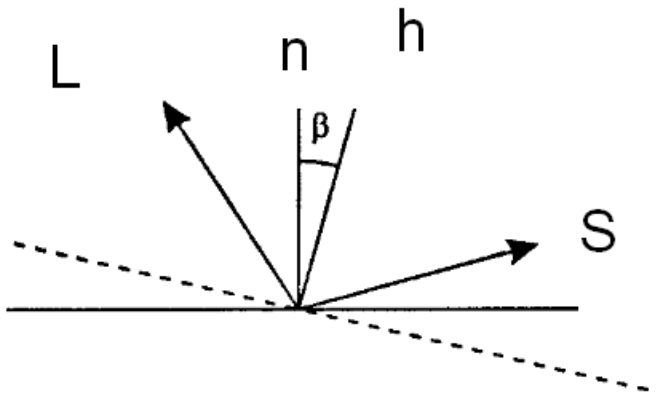
- вектор отражения вычисляется по следующей формуле:

$$\vec{R} = -\vec{L} + 2\frac{(\vec{L}\cdot\vec{n})}{\left|\vec{n}\right|^2}\vec{n}$$

  - где **L** – вектор, направленный из точки падения в направлении источника,
  - **n** – нормаль к поверхности в точке падения,
  - **R** – вектор зеркального отражения



- вычисление промежуточного вектора (Блинн):
  - *промежуточный вектор* (halfway vector) представляет собой направление воображаемой нормали, при котором вектор отражения совпал бы с вектором наблюдения: **H** = **L** + **S**
  - промежуточный вектор можно использовать вместо вектора отражения для ускорения вычислений: угол между **H** и **n** вдвое больше угла между **R** и **S**, и (**H**·**n**) можно использовать в компоненте зеркального отражения.



Учет расстояния от источника света до поверхности

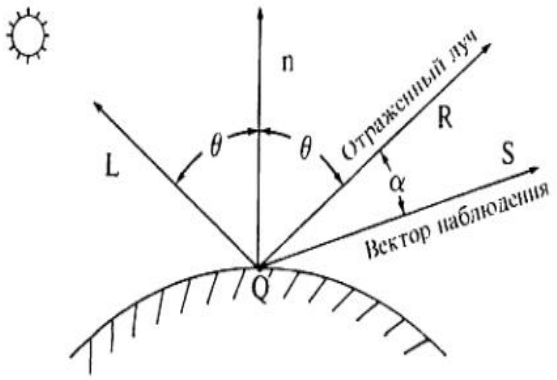
- исходя из физических законов интенсивность света обратно пропорциональна квадрату расстояния от источника до объекта, однако, если источник света находится в бесконечности, то диффузный и зеркальный члены модели освещения обратятся в нуль;
- следовательно, при реализации модели освещения применяют некоторые эмпирические коэффициенты для достижения наибольшей реалистичности построенного изображения:
  - в случае перспективного преобразования сцены в качестве коэффициента пропорциональности можно взять расстояние d от центра проекции до объекта;
  - если центр проекции лежит близко к объекту, d^2 изменяется очень быстро, т. е. у объектов, лежащих примерно на одинаковом расстоянии от источника, разница интенсивностей чрезмерно велика;
  - большей реалистичности можно добиться при линейном затухании, если предполагается, что точка наблюдения находится в бесконечности, то d определяется положением объекта, ближайшего к точке наблюдения (ближайший объект освещается с полной интенсивностью источника, а более далекие - с уменьшенной).

$$a_2 D^2 + a_1 D + a_0$$

Простая модель освещения (модель освещения Фонга)

- расчет интенсивности с учетом рассеянного, диффузного и зеркального компонентов освещения

$$I = I_a k_a + \frac{I_l}{d+K}(k_d \cos\theta + k_s \cos^n \alpha)$$



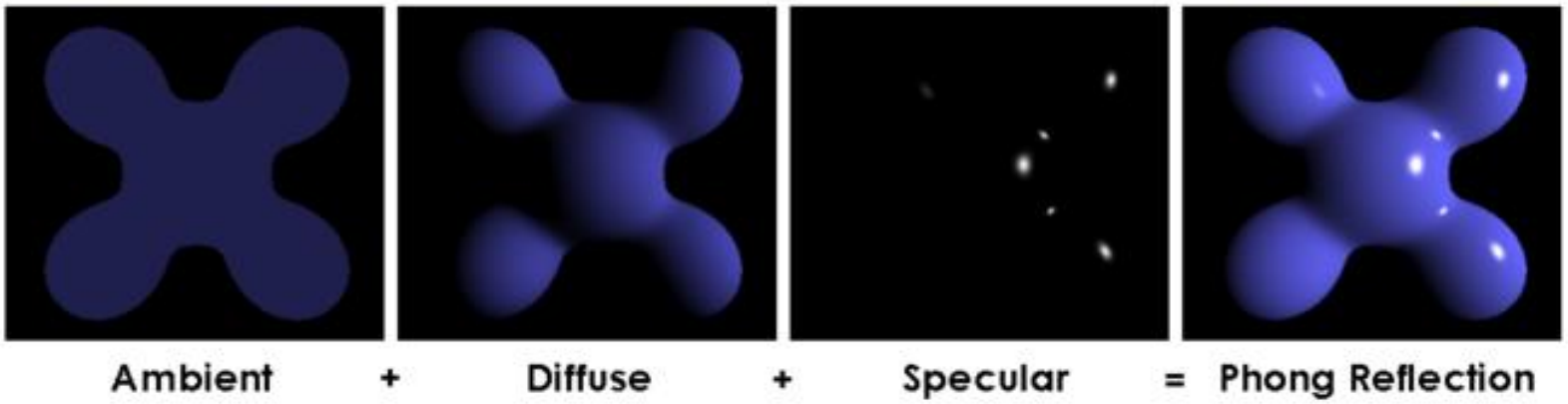
- в векторной форме

$$I = I_a k_a + \frac{I_l}{d+K}[k_d(\hat{\mathbf{n}} \cdot \hat{\mathbf{L}}) + k_s(\hat{\mathbf{R}} \cdot \hat{\mathbf{S}})^n]$$

- с учетом нескольких источников света

$$I = I_a k_a + \sum_{j=1}^{m} \frac{I_{l_j}}{d+K}(k_d \cos\theta_j + k_s \cos_j^n \alpha_j)$$

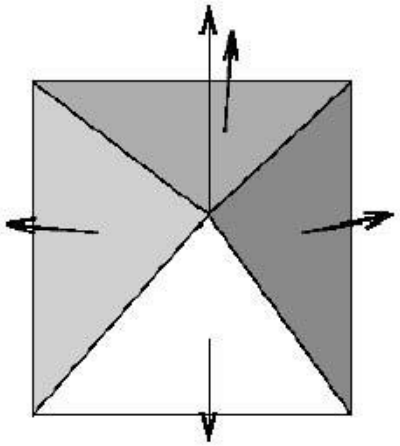
- для цветных поверхностей модель освещения применяется к каждому из трех основных цветов

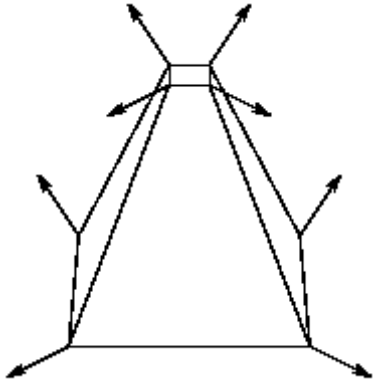


Закрашивание граней

- определение нормалей:
  - нормаль к граням (внешняя нормаль);
  - нормаль в вершине (вычисляется усреднением нормалей смежных граней);
- методы закрашивания:
  - плоское (однотонное) компоненты модели освещения вычисляются один раз для каждой грани;
  - плавное (интерполяционное):
    - метод Гуро: билинейная интерполяция интенсивности, вычисляемой в вершинах многоугольника;
    - метод Фонга: билинейная интерполяция нормалей в вершинах, на основании которых вычисляются компоненты модели освещения;
- рассматриваются выпуклые многоугольники;
- управление закрашиванием в OpenGL:

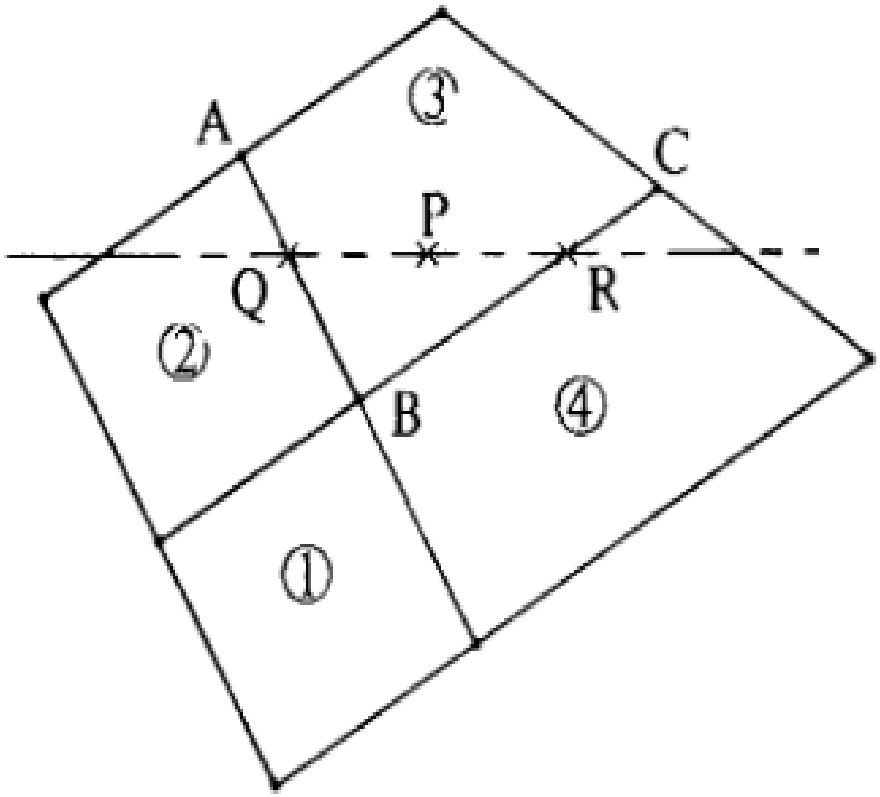
  glShadeModel(GL_FLAT | GL_SMOOTH)





Закраска Гуро (Gouraud shading)

- метод Гуро основан на билинейной интерполяции интенсивности, вычисляемой в вершинах многоугольника:



$$I_Q = uI_A + (1-u)I_B \qquad 0 \le u \le 1$$

$$I_R = wI_B + (1-w)I_C \qquad 0 \le w \le 1$$

$$I_P = tI_Q + (1-t)I_R \qquad 0 \le t \le 1$$

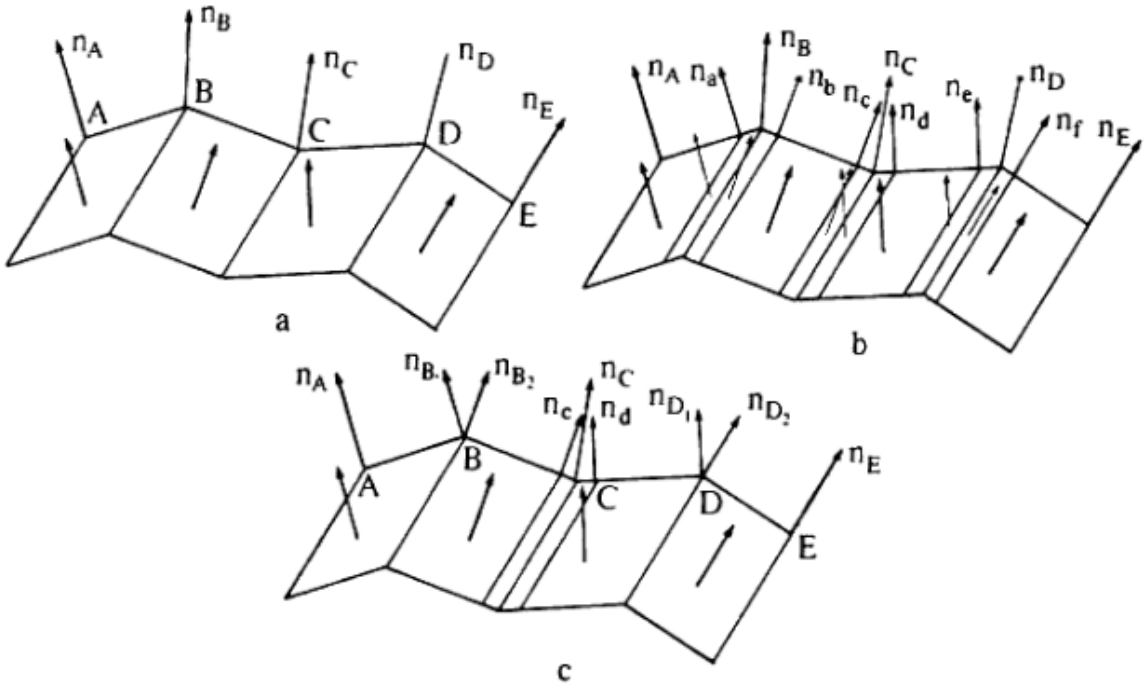
$$I_{P_2} = t_2 I_Q + (1-t_2)I_R$$

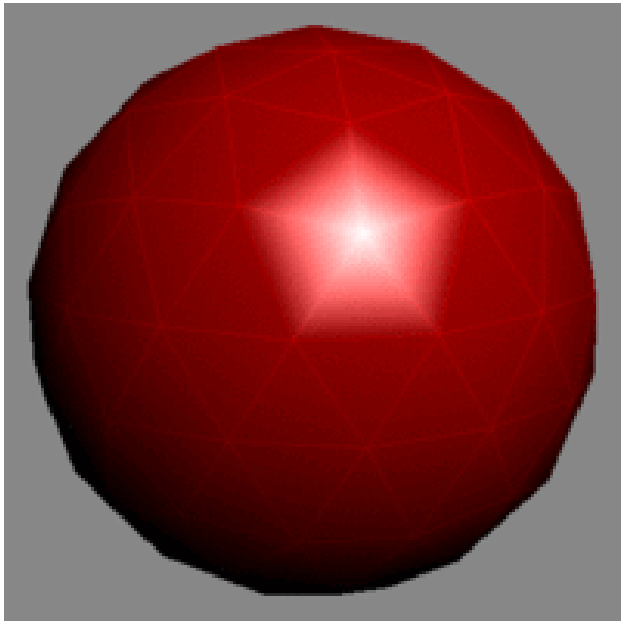
$$I_{P_1} = t_1 I_Q + (1-t_1)I_R$$

$$I_{P_2} = I_{P_1} + (I_Q - I_R)(t_2 - t_1) = I_{P_1} + \Delta I\,\Delta t$$

Закраска Гуро: анализ

- эффективность сопоставима с плоским закрашиванием, так как не требуется значительного объема вычислений;
- не является реализацией физического закона →не учитывает блики;
- результат зависит от выбора нормалей (проблемы с равномерно волнистой поверхностью);
- возможно появление эффекта полос Маха.





Закраска Фонга (Phong shading)

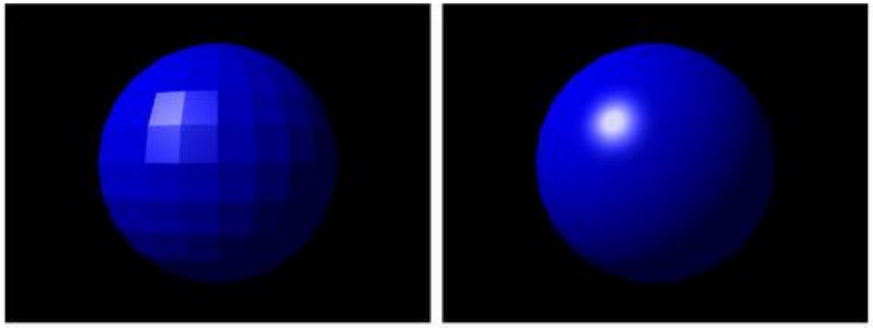
- метод Фонга основан на билинейной интерполяции нормалей в вершинах, на основании которых вычисляются компоненты модели освещения;
- свойства:
  - достигается лучшая локальная аппроксимация кривизны поверхности;
  - необходима нормализация вычисляемых векторов;
  - метод требует значительного объема вычислений;
  - возможно появление эффекта полос Маха.

$$\mathbf{n}_Q = u\mathbf{n}_A + (1-u)\mathbf{n}_B \qquad 0 \le u \le 1$$

$$\mathbf{n}_R = w\mathbf{n}_B + (1-w)\mathbf{n}_C \qquad 0 \le w \le 1$$

$$\mathbf{n}_P = t\mathbf{n}_Q + (1-t)\mathbf{n}_R \qquad 0 \le t \le 1$$

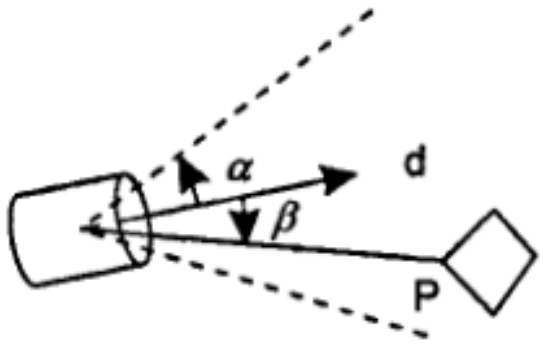
$$\mathbf{n}_{P_2} = \mathbf{n}_{P_1} + (\mathbf{n}_Q - \mathbf{n}_R)(t_2 - t_1) = \mathbf{n}_{P_1} + \Delta\mathbf{n}^*\Delta t$$



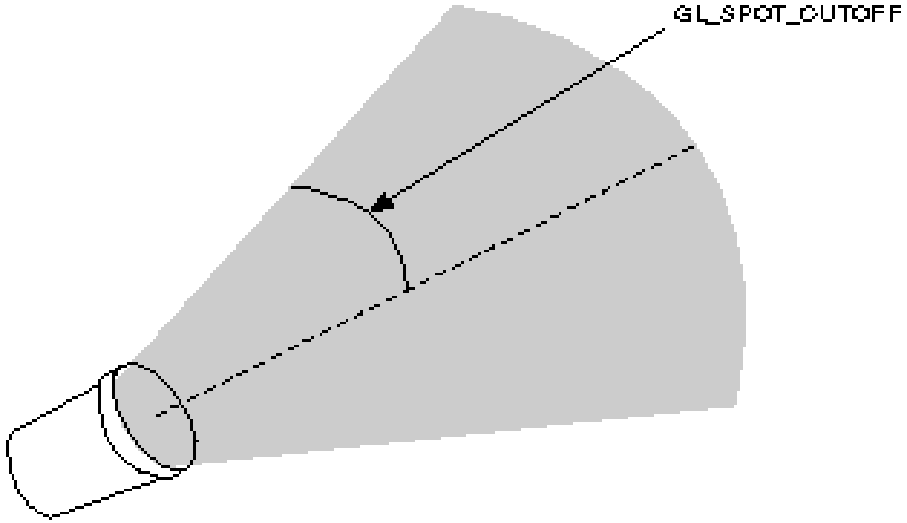
FLAT SHADING PHONG SHADING

Использование направленного сфокусированного источника света

- направленный сфокусированный источник (прожектор) представляет собой источник света, излучающий внутри некоторого конуса (с вершиной в источнике);
- для определения направленного сфокусированного источника света задаются:
  - направление (ориентация оси конуса);
  - угол при вершине конуса (или его половина) – *угол пропускания;*
  - степень сфокусированности источника (задает распределение света внутри конуса).



$$I = I_l \cos^{\varepsilon} \beta$$



Модель освещения OpenGL: источники света

- задание цвета и нормали в вершине:
  glColor*()
  glNormal*()
  glEnable(GL_NORMALIZE) (нормализация нормалей после применения преобразований)
- активизация источников света (до 8):
  glEnable(GL_LIGHTING)
  glEnable(GL_LIGHT0)
- определение позиции источника:
  – если (x,y,z,1) – расположен в заданной точке;
  – если (x,y,z,0) – бесконечно удален в заданном направлении (выигрыш в вычислениях), ослабление света не происходит;
  glLightfv(GL_LIGHT0, GL_POSITION, v[ ])
- определение параметров света, испускаемого источником:
  glLightfv(GL_LIGHT0, GL_AMBIENT(0,0,0,1 – для всех источников) **|**
  GL_DIFFUSE (1,1,1,1 – для GL_LIGHT0; 0,0,0,1 - иначе) **|**
  GL_SPECULAR (1,1,1,1 – для GL_LIGHT0; 0,0,0,1 - иначе)**,**
  c[ ])

Модель освещения OpenGL: источники света (продолжение)

- моделирование сфокусированного источника света
  - определения угла пропускания [по умолчанию – Pi]:
    glLightf(GL_LIGHT0, GL_SPOT_CUTOFF)
  - параметр сфокусированности: показатель степени $I = I_l * \cos^{exp}(\boldsymbol{\theta})$ [по умолчанию, 0]:
    glLightf(GL_LIGHT0, GL_SPOT_EXPONENT)
  - направление излучения источника света [по умолчанию, (0,0,-1)]:
    glLightf(GL_LIGHT0, GL_SPOT_DIRECTION)
- учет ослабления света с расстоянием: коэффициент ослабления (attenuation factor):

$$attenuation = \frac{1}{k_c + k_l \cdot D + k_q \cdot D^2}$$

glLightf(GL_LIGHT0, GL_CONSTANT_ATTENUATION, K_c [1])
glLightf(GL_LIGHT0, GL_LINEAR_ATTENUATION, K_l [0])
glLightf(GL_LIGHT0, GL_QUADRIC_ATTENUATION, K_q [0])

Модель освещения OpenGL: параметры глобальной модели освещения и свойства материалов

- параметры модели освещения:
  - цвет глобального фонового источника (по умолчанию, (0,2; 0,2; 0,2; 1)):
    glLightModelfv(GL_LIGHT_MODEL_AMBIENT, c[ ])
  - локальная или удаленная точка наблюдения: по умолчанию, v = (0,0,1), и вычисляется промежуточный вектор:
    glLightModel(GL_LIGHT_MODEL_LOCAL_VIEWER, GL_FALSE)
  - обработка граней (двусторонняя):
    glLightModel(GL_LIGHT_MODEL_TWO_SIDE, GL_TRUE)
- определение свойств материалов (эмиссионный свет: собственное свечение граней):
  glMaterialfv(GL_FRONT | GL_BACK | GL_FRONT_AND_BACK, GL_AMBIENT | GL_DIFFUSE | GL_AMBIENT_AND_DIFFUSE | GL_SPECULAR | GL_EMISSION | GL_SHININESS , c[ ])
- общий вид модели освещения с учетом всех параметров:

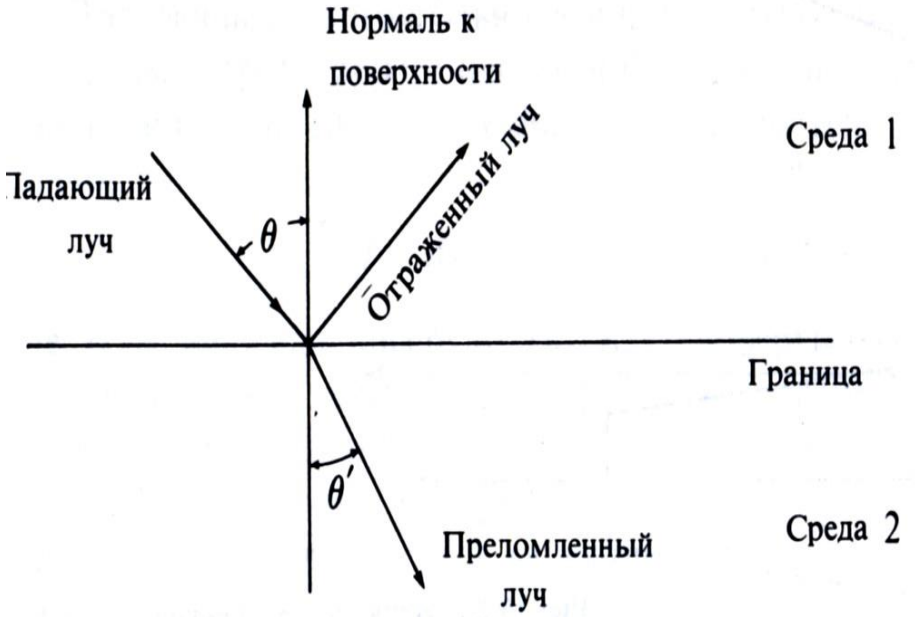
$$I_r = e_r + I_{\mathrm{mr}}\rho_{\mathrm{ar}} + \sum_i \mathrm{atten}_i \times \mathrm{spot}_i \times \left(I^i_{\mathrm{ar}}\rho_{\mathrm{ar}} + I^i_{\mathrm{dr}}\rho_{\mathrm{dr}} \times \mathrm{lambert}_i + I^i_{\mathrm{spr}}\rho_{\mathrm{sr}} \times \mathrm{phong}_i^f\right).$$

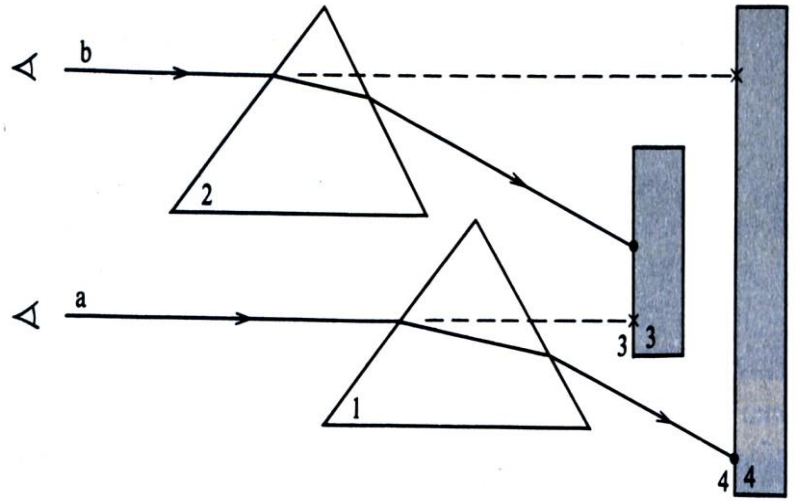
Прозрачность и преломление

- преломление рассчитывается по закону Снеллиуса:
  - падающий, преломленный и отраженный лучи лежат в одной плоскости, а углы падения и преломления связаны формулой (η_1, η_2 - показатели преломления двух сред, θ - угол падения, θ' - угол преломления):

$$\eta_1 \sin\theta = \eta_2 \sin\theta'$$

  - пропускание (как и отражение) может быть:
    - зеркальным (направленным);
    - диффузным.





Вычисление преломлённого луча

$$\vec{v}_i = -(\sin\theta_i)\vec{s} - (\cos\theta_i)\vec{n}$$

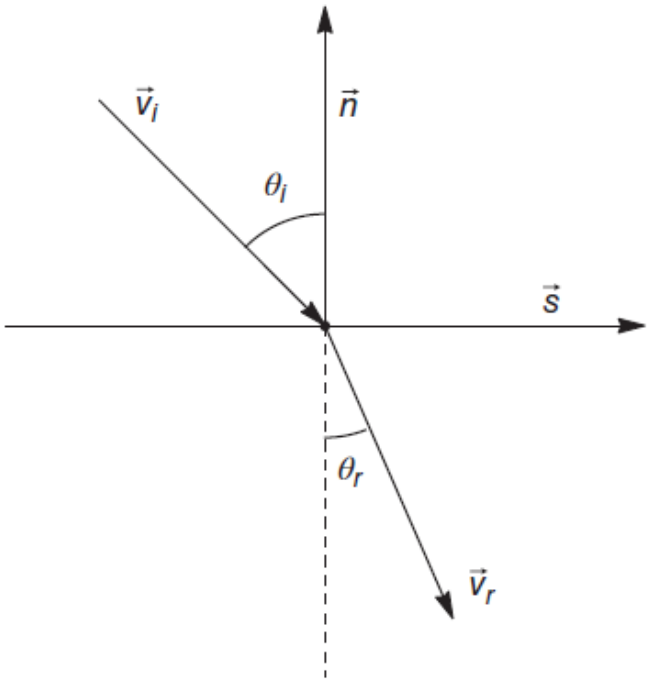
$$\vec{v}_r = -(\sin\theta_r)\vec{s} - (\cos\theta_r)\vec{n}$$

$$\sin\theta_r = \frac{\eta_i}{\eta_r}\sin\theta_i$$

$$\cos^2\theta_r = 1 - \frac{\eta_i^2}{\eta_r^2}(1 - \cos^2\theta_i)$$

$$-\cos\theta_i = \vec{v}_i \cdot \vec{n}$$

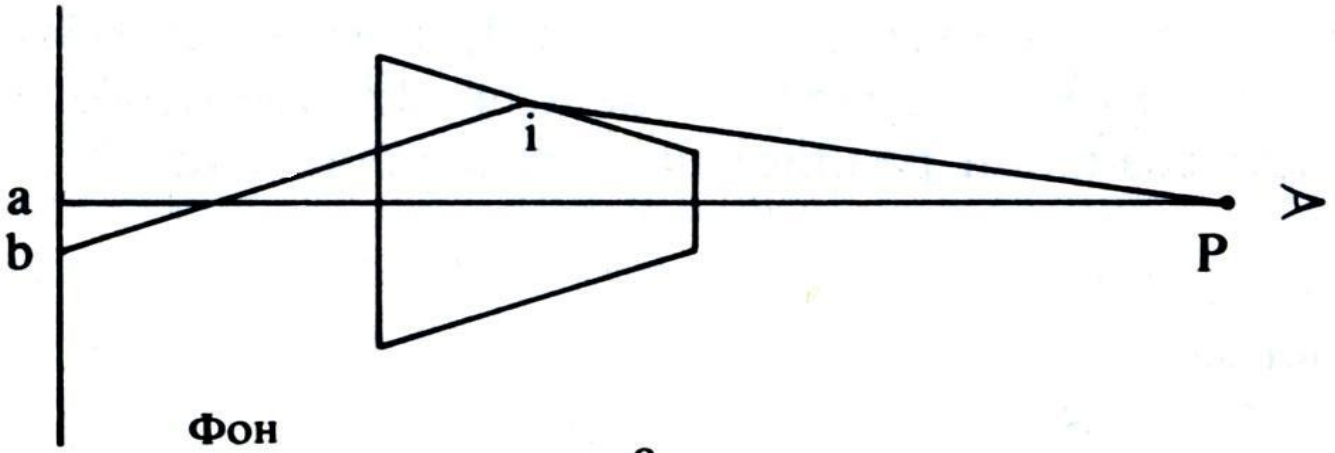
$$-(\sin\theta_i)\vec{s} = \vec{v}_i - (\vec{v}_i \cdot \vec{n})\vec{n}$$



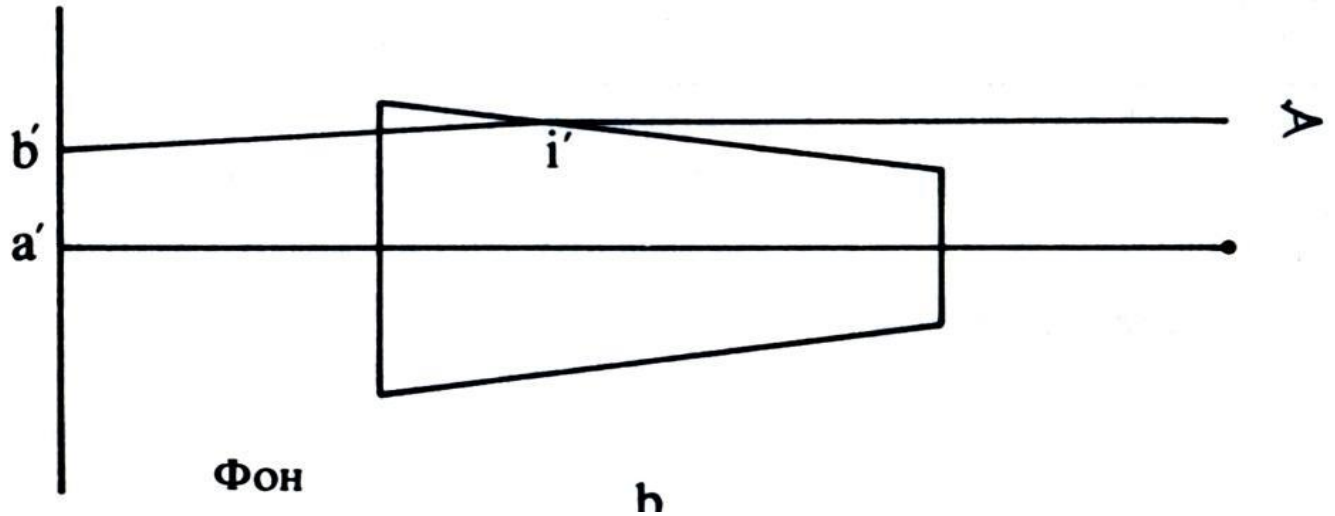
$$\vec{v}_r = \frac{\eta_i}{\eta_r}(\vec{v}_i - (\vec{v}_i \cdot \vec{n})\,\vec{n}) - \left(1 - \frac{\eta_i^2}{\eta_r^2}(1 - (\vec{v}_i \cdot \vec{n})^2)\right)^{\frac{1}{2}} \vec{n}$$

* предполагается, что все векторы – единичной длины

Эффекты преломления и влияние перспективы



a



b

Альфа-канал

- альфа-канал (alpha channel, Alvy Ray Smith) определяет прозрачность пиксела;
- простое пропускание можно встроить в любой алгоритм удаления невидимых поверхностей (кроме алгоритма Z-буфера);
- смешивание интенсивностей для двух ближайших граней производится по следующей формуле (если рассматриваются более двух граней, формула применяется рекуррентно):

$$\mathbf{I = I_1\alpha + I_2(1-\alpha),\ \alpha \in [0;1]}\ ,\ \text{где}$$

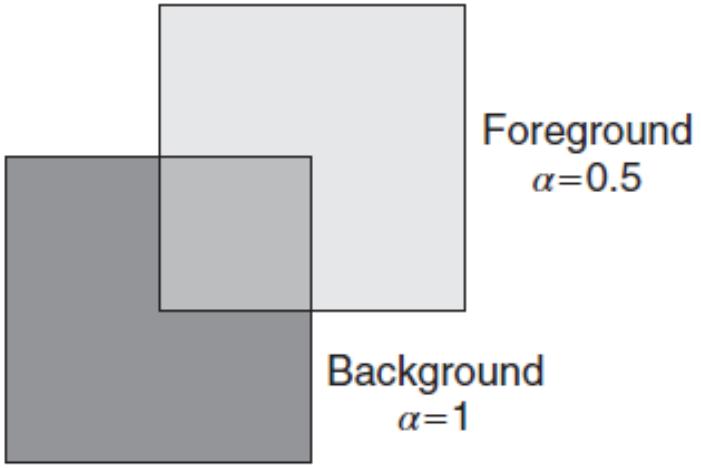
I_1 – интенсивность ближней грани,

I_2 – интенсивность граней, расположенных за ближней гранью,

α – коэффициент прозрачности ближней грани;

- ограничения:
  - расчет производится до перспективного преобразования;
  - корректность зависит от порядка занесения граней в буфер.

Комбинирование цветов с учётом прозрачности



$$B = (r_B, g_B, b_B)$$

$$F = (r_f, g_f, b_f)$$

$$P = \alpha F + (1 - \alpha)B$$

$$P = F \triangleright B$$

- over operator

$$P = (F_2 \triangleright F_1) \triangleright B$$

$$P = (F_2 \triangleright F_1) \triangleright B = F_2 \triangleright (F_1 \triangleright B)$$

$$(1 - \alpha^*) = 1 - \alpha_2 - \alpha_1 + \alpha_2\alpha_1$$

$$\alpha^*(F_2 \triangleright F_1) = \alpha_2 F_2 + (1 - \alpha_2)\alpha_1 F_1$$

$$\alpha^* = \alpha_2 + (1 - \alpha_2)\alpha_1$$

$$F_2 \triangleright F_1 = \frac{\alpha_2}{\alpha^*} F_2 + \left(1 - \frac{\alpha_2}{\alpha^*}\right) F_1$$

premultiplied alpha form $(\alpha r, \alpha g, \alpha b, \alpha)$

$$\overline{F}_2 \triangleright \overline{F}_1 = \overline{F}_2 + (1 - \alpha_2)\overline{F}_1$$

Прозрачность и OpenGL

- во время цветового наложения цветовые величины входящего фрагмента (*источника, source*) комбинируются с цветовыми величинами соответствующего, сохраненного к текущему моменту, пикселя (*приемника, destination*) в два этапа:
  - задание способа вычисления факторов источника и приемника (факторы представляют собой четверки RGBA, которые умножаются на R, G, B и A величины источника и приемника соответственно): (*Sr, Sg, Sb, Sa*) и (*Dr, Dg, Db, Da*);
  - задание способа комбинирования компонентов двух наборов RGBA;
- результирующие цветовые величины после цветового наложения могут быть получены из следующего уравнения:

$$(R_s S_r + R_d D_r, G_s S_g + G_d D_g, B_s S_b + B_d D_b, A_s S_a + A_d D_a)$$

- каждый компонент этой четверки приводится к диапазону [0,1].

Прозрачность и OpenGL: функции

- активизация режима смешивания цветов

  glEnable(GL_BLEND);

- определение факторов источника и приемника источника:

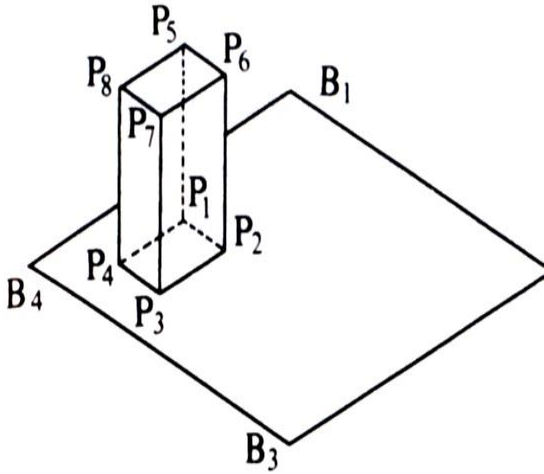
  glBlendFunc (GLenum sfactor = GL_ZERO | **GL_ONE** | GL_ONE_MINUS_DST_COLOR | GL_DST_COLOR | GL_SRC_ALPHA | GL_ONE_MINUS_SRC_ALPHA | GL_DST_ALPHA | GL_ONE_MINUS_DST_ALPHA | GL_SRC_ALPHA_SATURATE,
  GLenum dfactor = GL_ZERO | GL_ONE | GL_SRC_COLOR | GL_ONE_MINUS_SRC_COLOR | GL_SRC_ALPHA | GL_ONE_MINUS_SRC_ALPHA | GL_DST_ALPHA | GL_ONE_MINUS_DST_ALPHA**);**

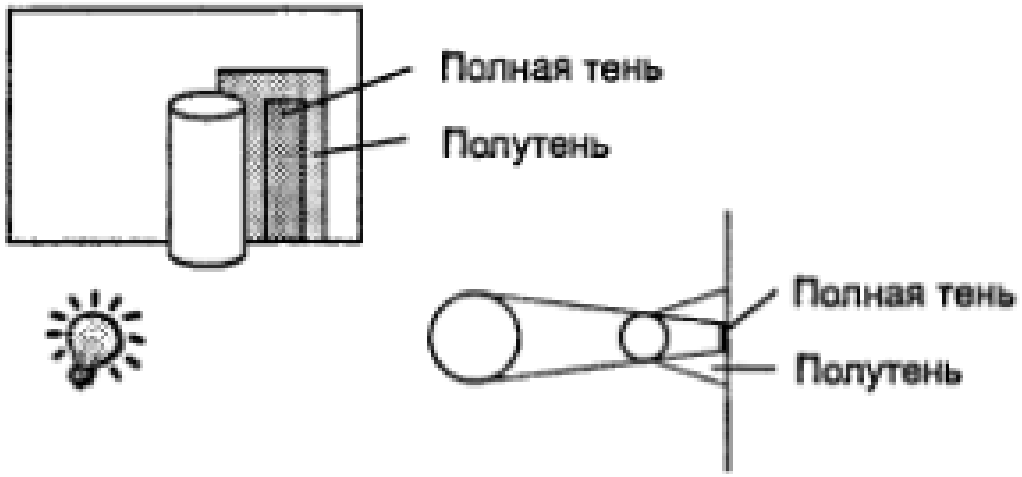
- стандартный альфа-канал:

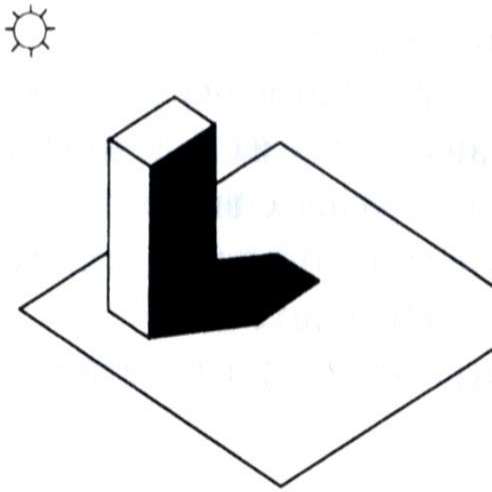
  glBlendFunc(GL_SRC_ALPHA, GL_ONE_MINUS_SRC_ALPHA)

Построение теней

- тени могут возникнуть, когда положение наблюдателя не совпадает с положением источника света;
- тень состоит из двух частей (зависит от расстояния между затененной и отбрасывающей тень гранями):
  - полной тени (центральной, темной, резко очерченной части - umbra);
  - полутени (границы между тенью и освещенной области – penumbra) – размытие границ (soft shadow);
- обычно рассматриваются точечные источники, создающие полную тень.

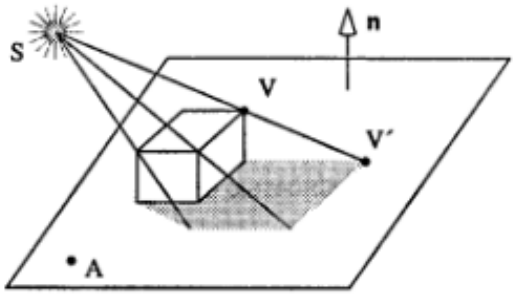






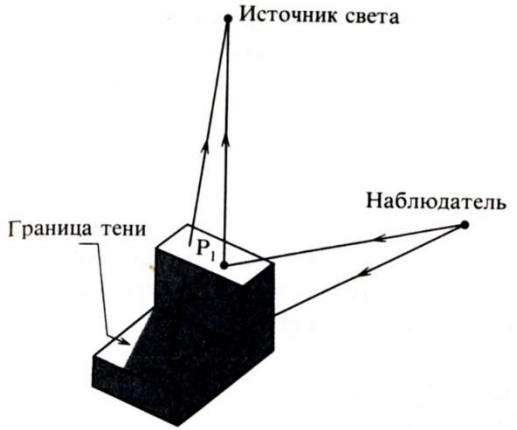
Построение теней: процедура

- могут рассматриваться два типа теней:
  - *собственные* (случай затенения гранями объекта самих себя);
  - *проекционные* (случай препятствования падения света на другие грани) – необходимо построение проекций всех нелицевых граней на сцену;
- вычислительная сложность зависит, в том числе, от положения источника света:
  - для источника в бесконечности необходимо построение ортогональных проекций;
  - для источника, расположенного на конечном расстоянии от сцены вне поля зрения – перспективное проецирование;
  - для источника, расположенного на конечном расстоянии от сцены в поле зрения – перспективное проецирование и разбиение пространства на секторы;
- двухэтапный процесс:
  - удаление невидимых поверхностей для положения каждого источника (результат не зависит от положения наблюдателя);
  - удаление невидимых поверхностей для положения наблюдателя.



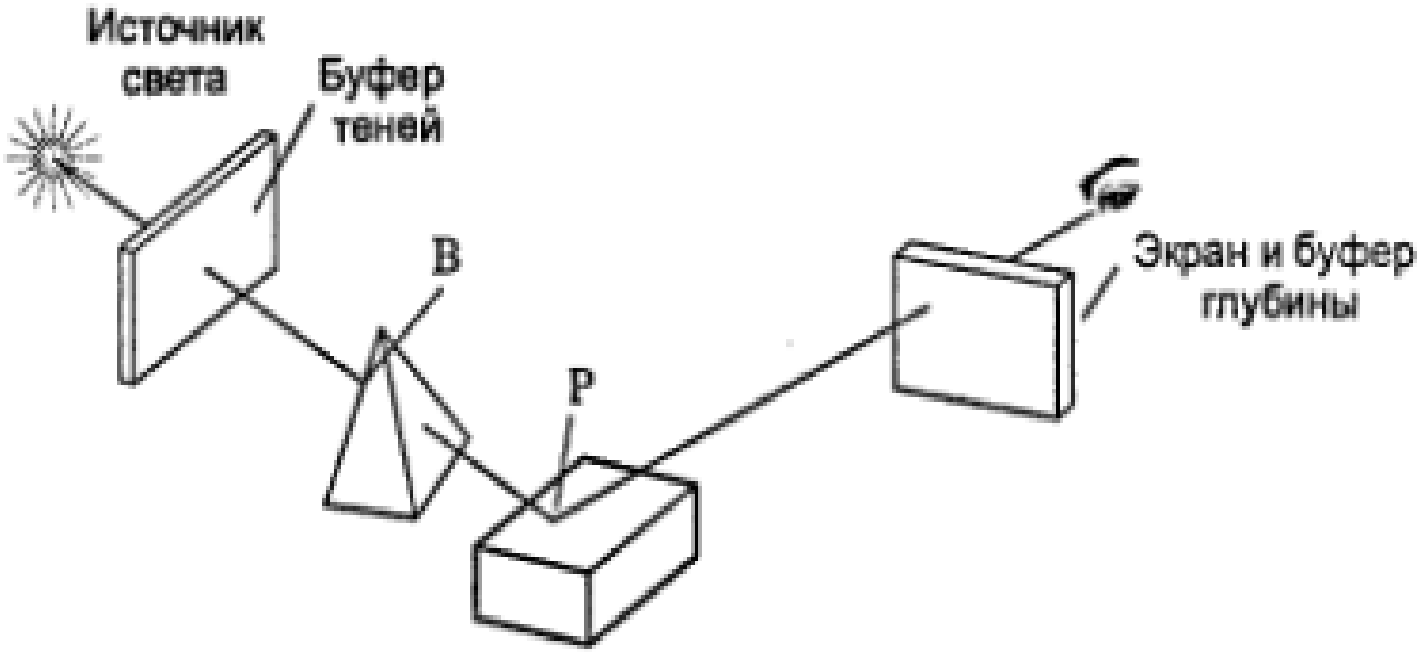
а

б

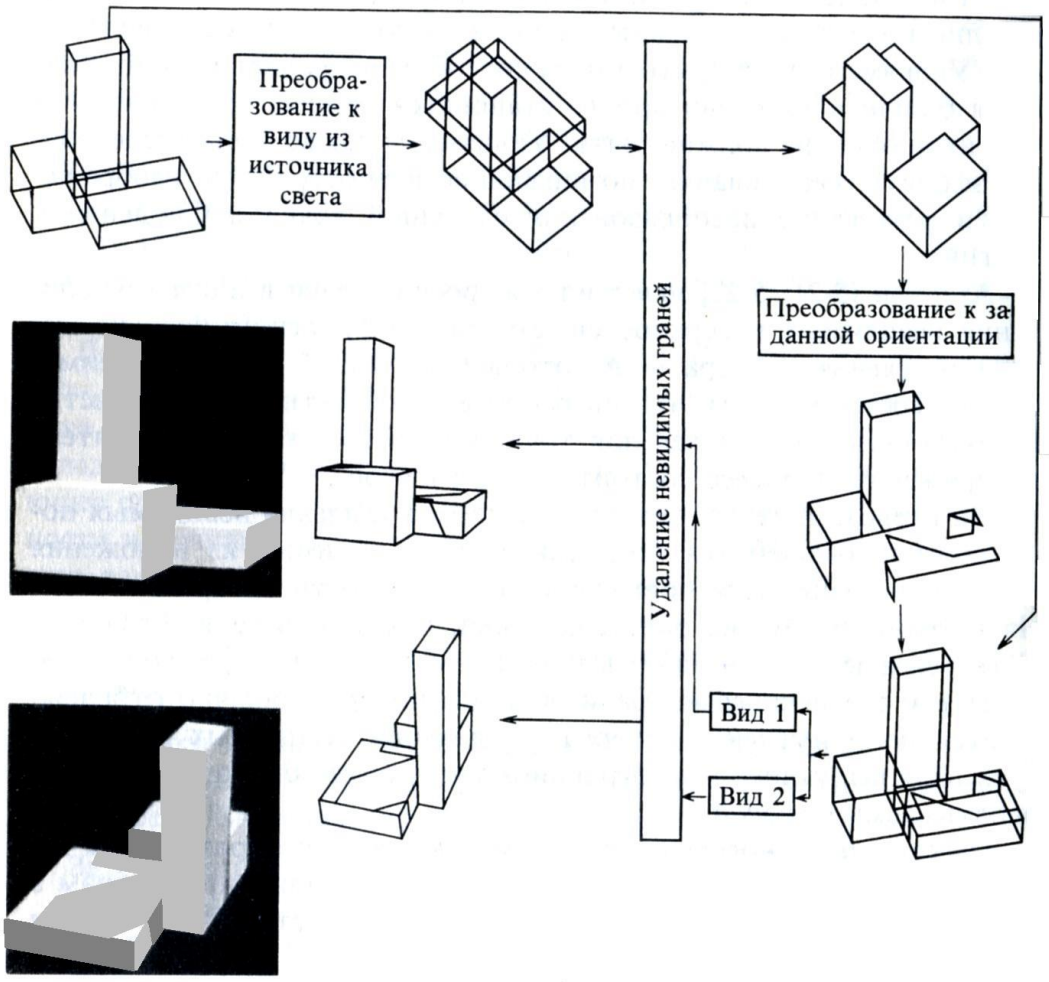


Использование алгоритма z-буфера для построения теней

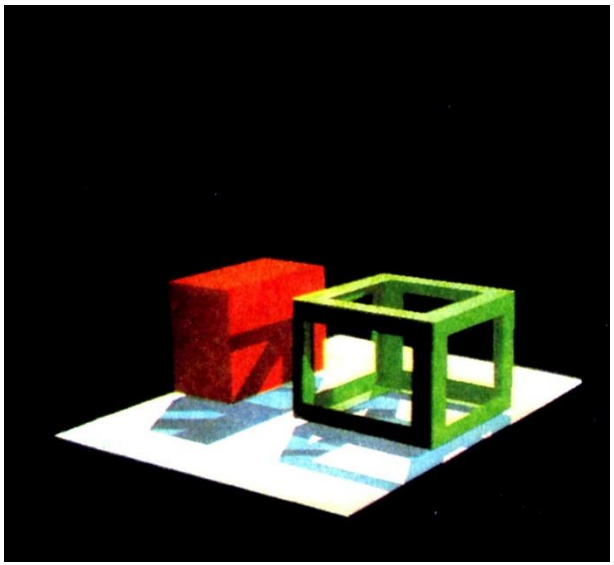
- построение теневых z-буферов (*shadow maps*) из положения источников света:
  - интенсивность не рассматривается;
  - результаты не зависят от положения наблюдателя;
- построение z-буфера из точки наблюдения.



Использование алгоритма удаления невидимых граней Вейлера-Азертона для построения теней

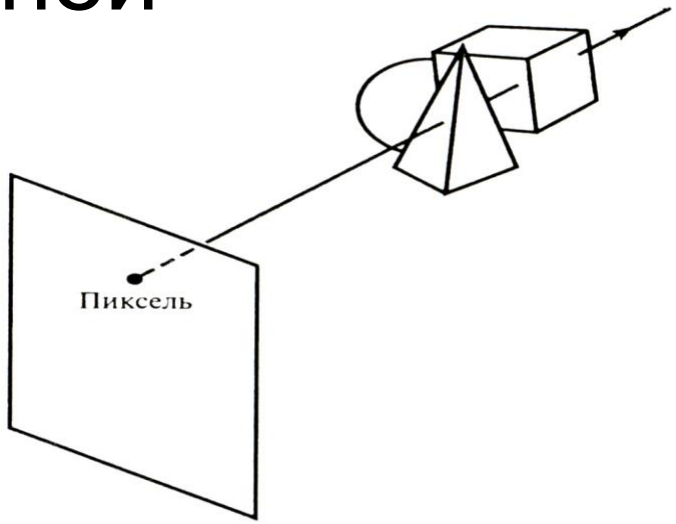


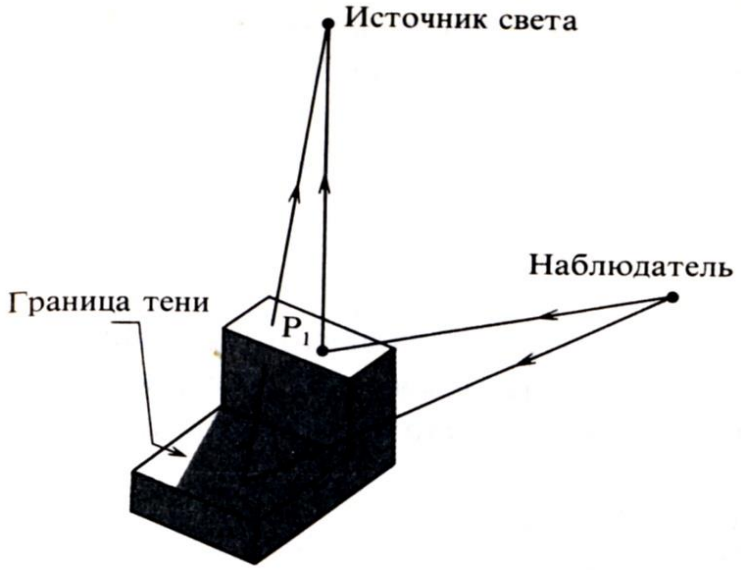
- построение наборов видимых граней для положения каждого из источников света;
- построение набора видимых граней для точки наблюдения.



Алгоритм трассировки лучей и построение теней

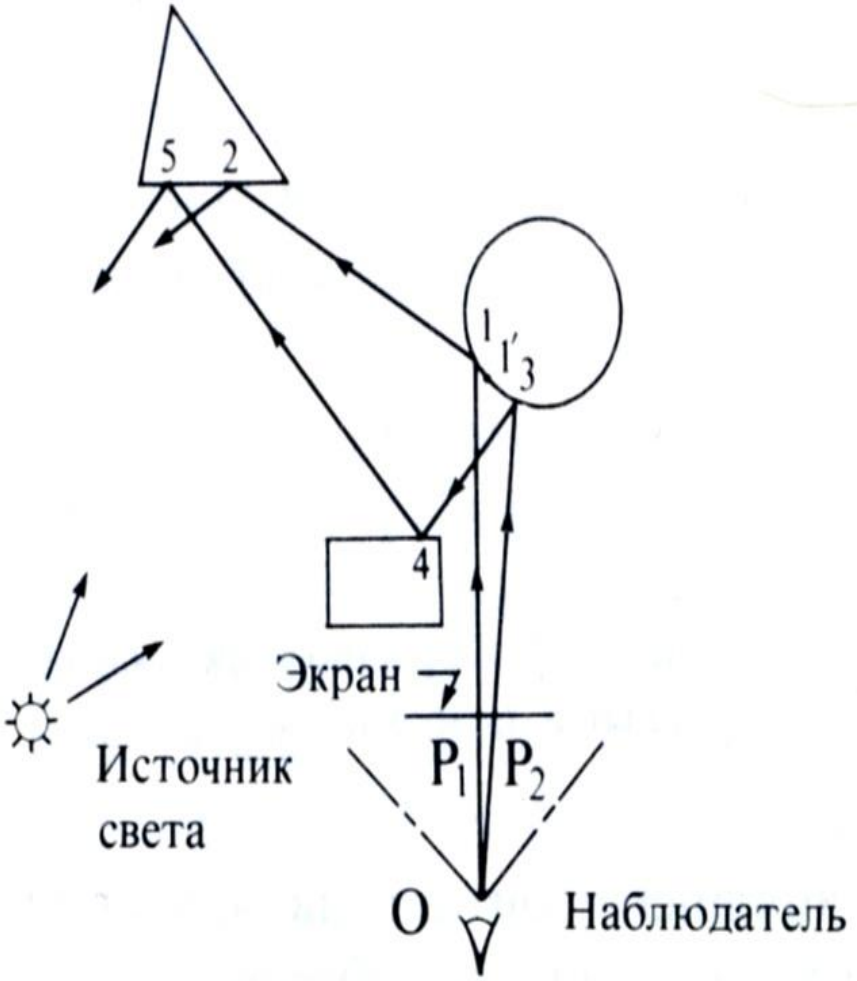
- двухшаговый алгоритм построения изображения:
  - обратная трассировка луча (построение луча из точки наблюдения через каждую точку экрана);
  - для каждой видимой точки поверхности – построение лучей к источникам света (теневых зондов);
- оптимизация вычислений:
  - использование прямоугольных / сферических / иерархических оболочек;
  - отложенное вычисление пересечений (после определения множества объектов и предварительной сортировки).





Глобальная модель освещения

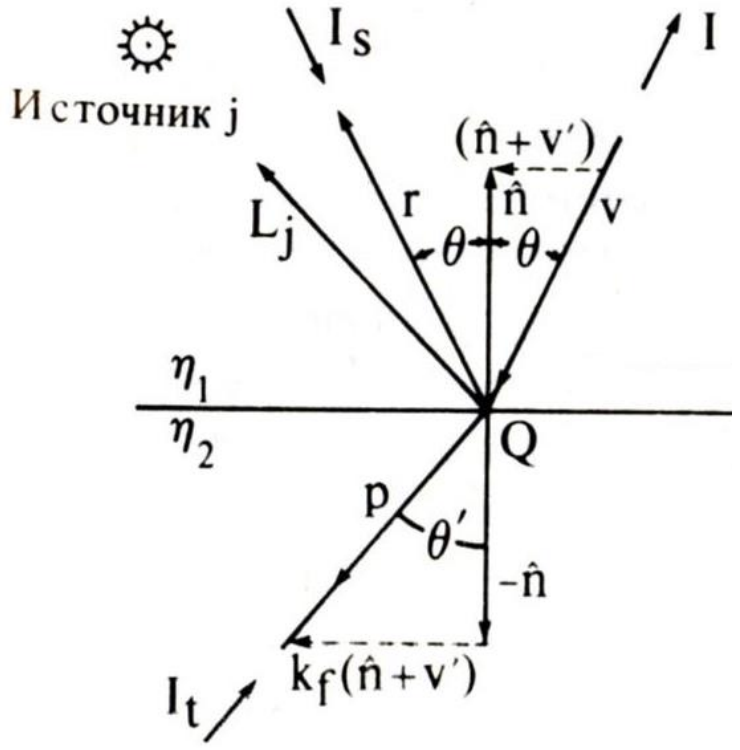
- модели освещения:
  - *локальная модель освещения*: учитывается только свет, падающий от источника (-ов), и ориентация поверхности;
  - *глобальная модель освещения*: дополнительно учитывается свет, отраженный от других объектов сцены или пропущенный сквозь них.



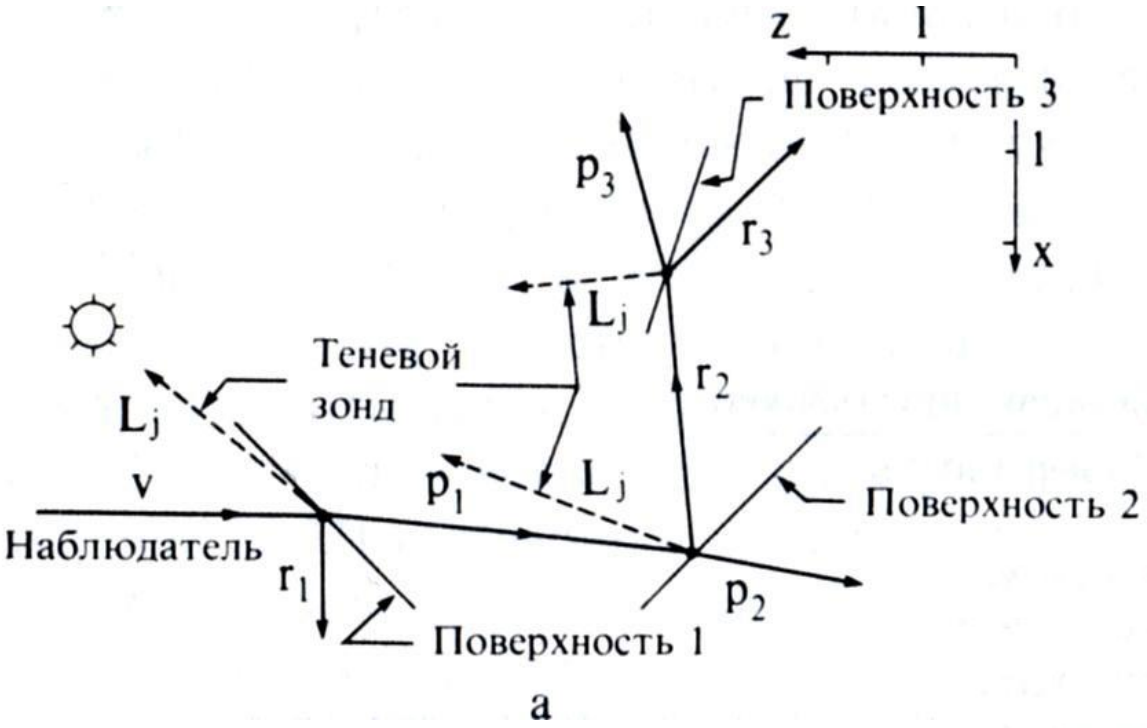
Глобальная модель освещения с трассировкой лучей (Whitted)

$$I = k_a I_a + k_d \sum_j I_{l_j}(\hat{\mathbf{n}} \cdot \hat{\mathbf{L}}_j) + k_s \sum_j I_{l_j}(\hat{\mathbf{S}} \cdot \hat{\mathbf{R}}_j)^n + k_s I_s + k_t I_t$$

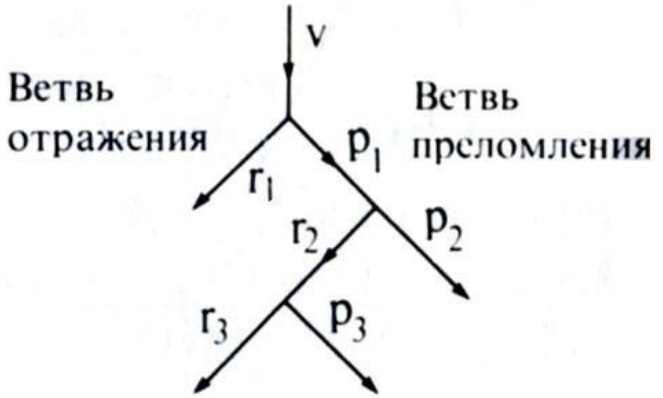
- при построении изображения учитываются:
  - локальная модель освещения (включая построение теневых зондов для учета теней);
  - интенсивность, поступающая в направлении зеркального отражения;
  - интенсивность, поступающая в направлении зеркального пропускания;
- процедура трассировки лучей является рекурсивной, которая прерывается в одном из следующих случаев:
  - луч покидает сцену;
  - интенсивность вдоль отраженного и преломленного лучей незначительна (учитывается только локальная модель освещения и расстояния);
  - превышена глубина рекурсии.

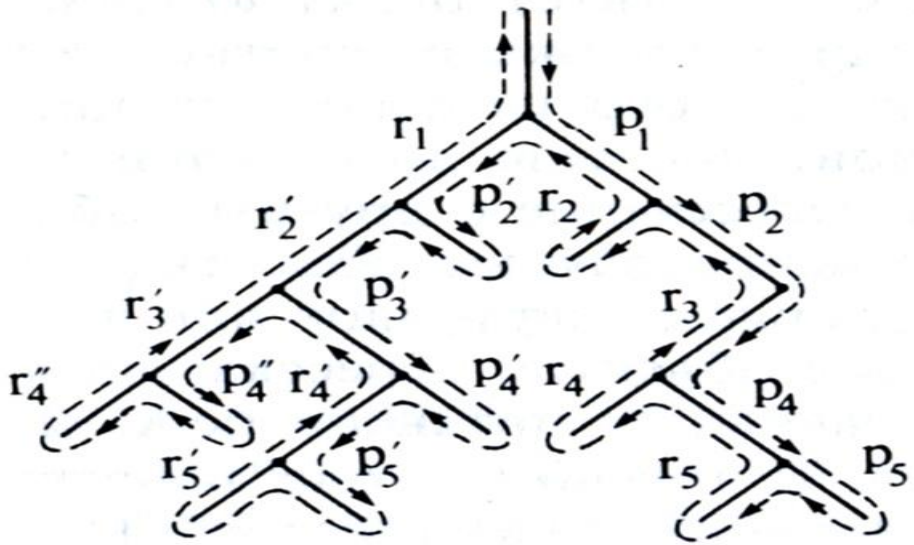


Глобальная модель освещения с трассировкой лучей: пример



а



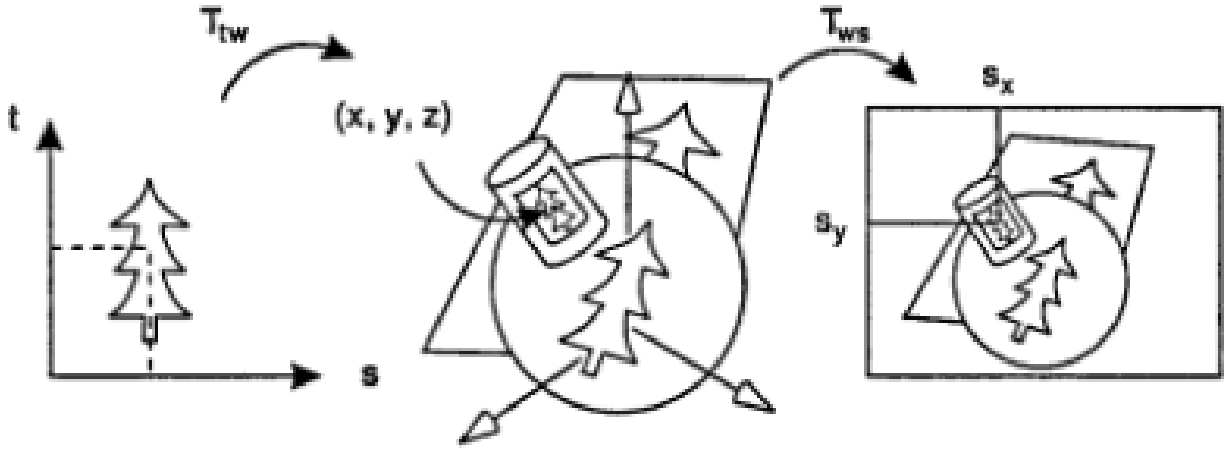


Фактуры и текстуры

- два вида детализации поверхности:
  - нанесение заданного узора на гладкую поверхность (текстурирование);
  - создание (визуальных) неровностей на поверхности путем внесения возмущения в параметры, задающие поверхность – bump mapping (Blinn);
- фактуры и текстуры могут быть заданы:
  - программно (с помощью процедуры генерации) – procedural texture;
  - растровым шаблоном или изображением (texels, texture elements);
- при наложении текстуры могут использоваться различные способы проецирования текстуры на поверхность (сферическая, цилиндрическая, плоская проекции);
- при наложении необходимо также предусмотреть операции устранения лестничного эффекта.

Текстуры

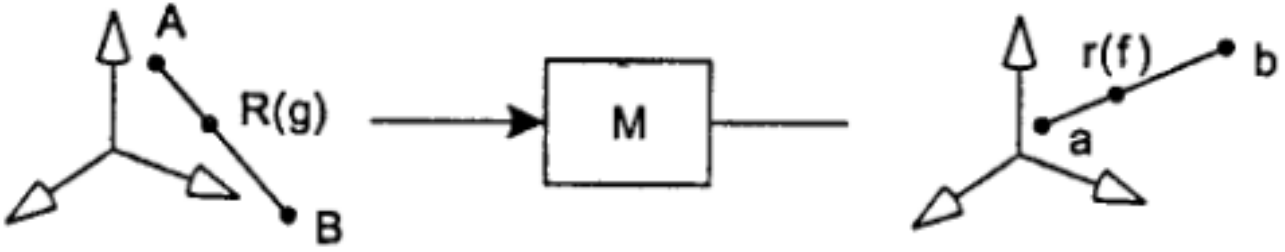
- наложение текстуры выполняется с помощью функции отображения текстурного и объектного координатных пространств:
  - задается текстурная функция (texture map) *texture(s,t)* в так называемом *текстурном пространстве* (texture space), которая генерирует значения цвета или яркости для каждого значения s,t $\in$ [0,1];
  - производится отображение на требуемую поверхность, заданную в объектном пространстве (s,t)→(x,y,z);
  - производится стандартное преобразование для вывода на экран;
- на практике может решаться обратная задача: определение текстурных координат, соответствующих заданному значению экранных;
- значения текстурной функции могут использоваться:
  - для задания интенсивности грани;
  - для определения коэффициентов отражения.

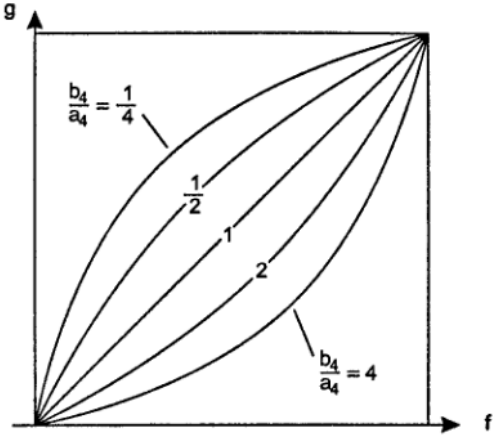


Текстуры и интерполяция

- при растровой развертке линейная интерполяция текстурных координат с равномерным шагом, в общем случае, приводит к некорректным результатам (равные шаги по спроецированной грани не соответствуют равным шагам по трехмерной грани);
- гиперболическая интерполяция – учет перспективного преобразования при отображении текстурных координат в объектное пространство (обеспечивается конвейером).

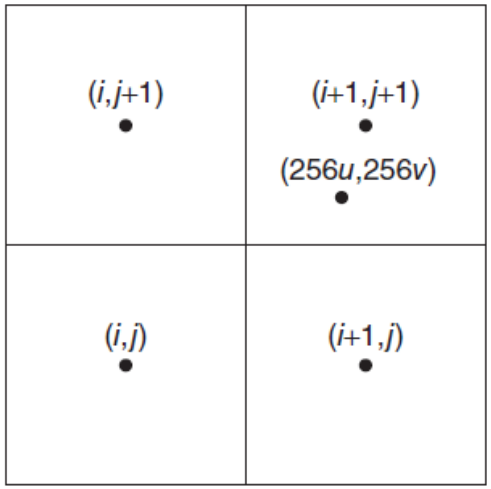
$$\frac{1}{z} = (1-\alpha)\frac{1}{z_1} + \alpha\frac{1}{z_2} \qquad \frac{u}{z} = (1-\alpha)\frac{u_1}{z_1} + \alpha\frac{u_2}{z_2}$$





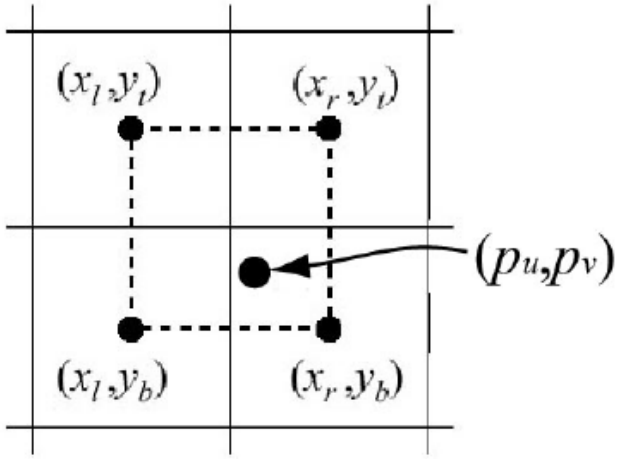
Текстурная фильтрация (texture filtering)

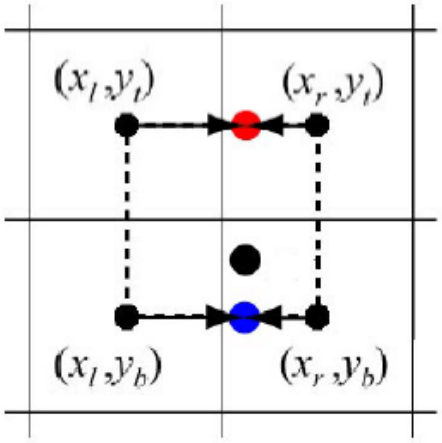
- в зависимости от относительного размера текселей и пикселей, необходимы операции масштабирования текстуры (texture magnification / minification – увеличение(растяжение) / сжатие);
- фильтрация в обоих случаях масштабирования позволяет вычислить корректное усреднённое значение цвета текселя:
  - при увеличении (тексель больше пикселя) имеет смысл учитывать один или четыре ближайших пикселя;
  - при уменьшении (тексель меньше пикселя) необходимы специальные процедуры усреднения, т.к. один пиксель может покрывать множество текселей.

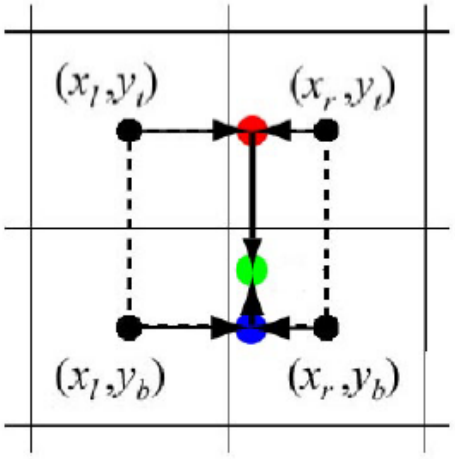


Увеличение текстур (texture magnification)

- методы:
  - ближайший сосед (nearest neighbour filtering);
  - билинейная интерполяция (bilinear filtering) – взвешенное усреднение;
  - бикубическая интерполяция (bicubic filtering);
  - …
- выбор метода влияет на:
  - производительность;
  - появление эффектов (ступенчатость, размытость границ и т.п.).

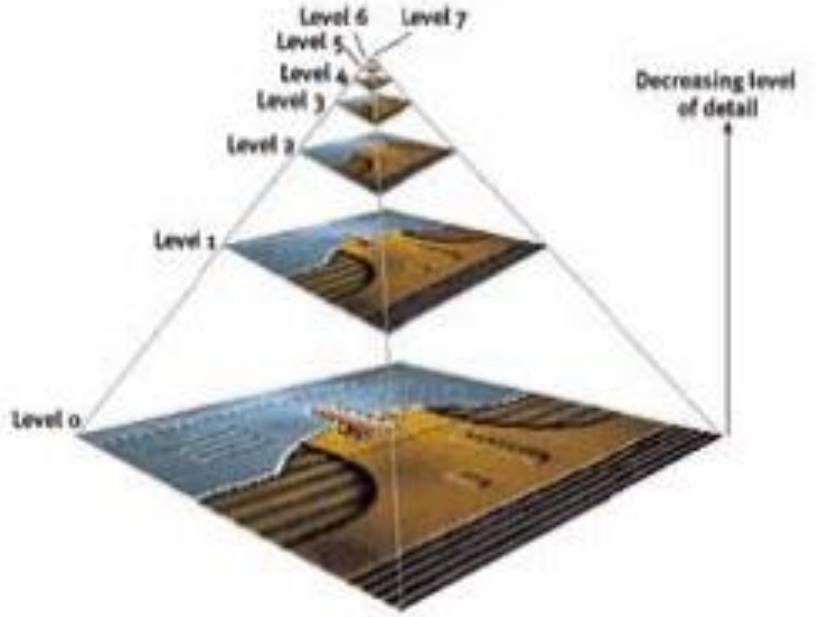


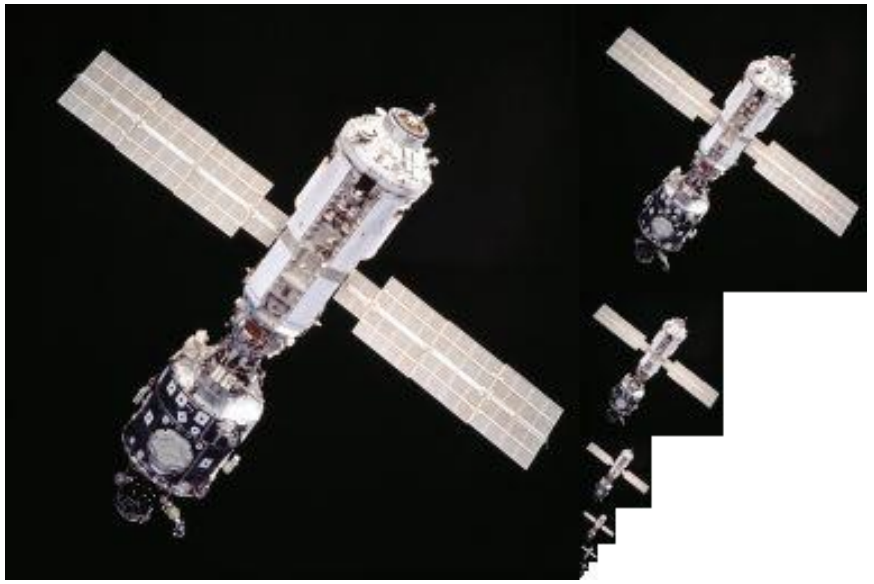




Уровни детализации текстуры (mipmaps)

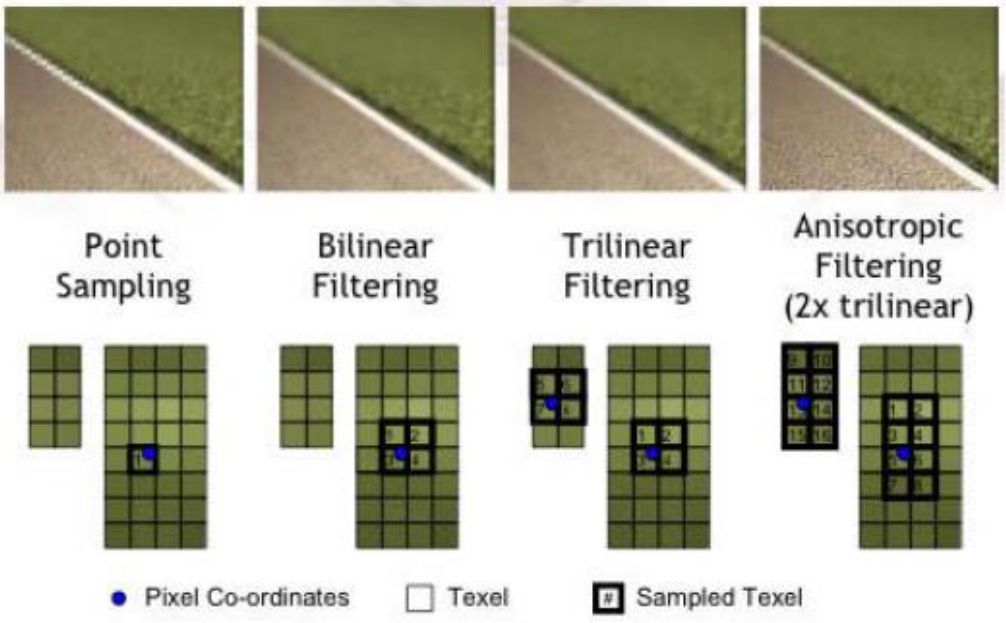
- mip = lat.“multum in parvo” (многое в малом), Lance Williams, 1983;
- последовательность уменьшенных вдвое по каждому измерению изображений (пирамида текстур / изображений, texture / image pyramid);
- построение путём усреднения изображения предыдущего уровня (ближайший сосед, билинейная фильтрация, свёртка (фильтр Гаусса) и т.п.).





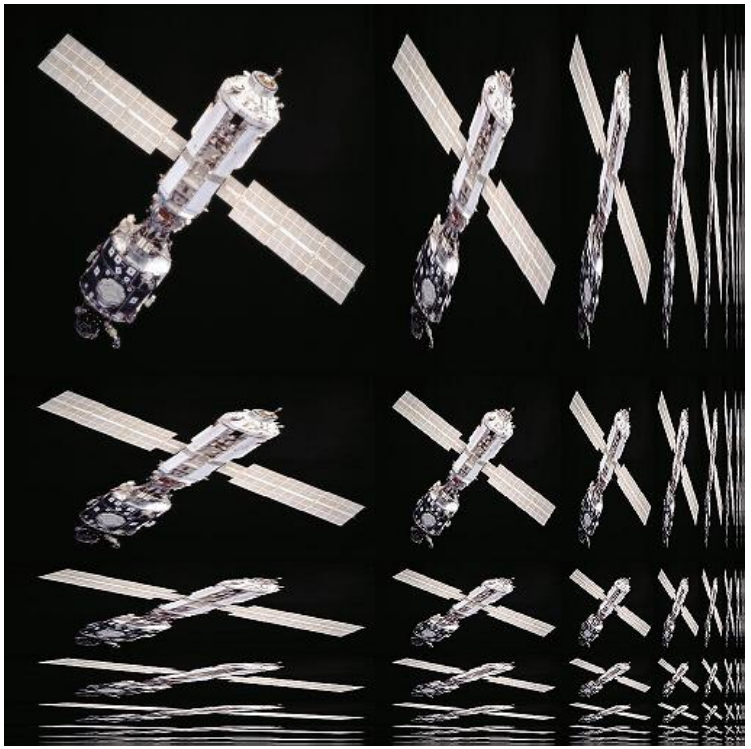
Сжатие текстур (texture minification)

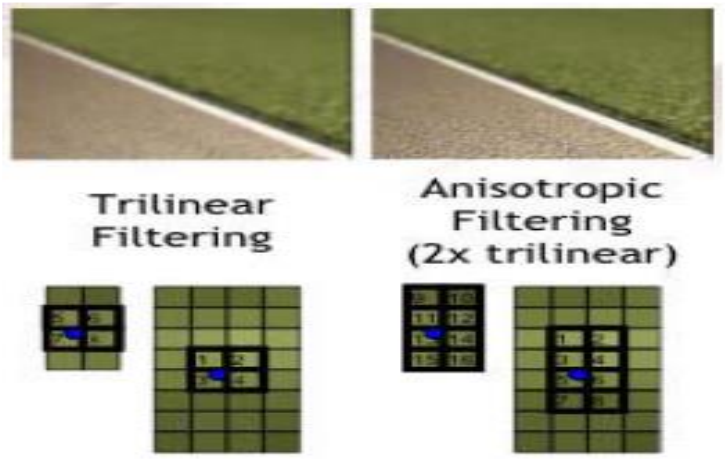
- ближайший сосед (nearest neighbour filtering);
- билинейная фильтрация (bilinear filtering);
- выбор ближайшего подходящего mipmap’а и поиск ближайшего соседа / билинейная фильтрация;
- выбор двух ближайших подходящих mipmap’ов и линейная интерполяция найденных в каждом из них ближайших соседей;
- трилинейная фильтрация (trilinear filtering) - выбор двух ближайших подходящих mipmap’ов и линейная интерполяция найденных в каждом из них с помощью билинейной фильтрации значений;
- анизотропная фильтрация (anisotropic filtering) – неоднородная фильтрация по различным направлениям.



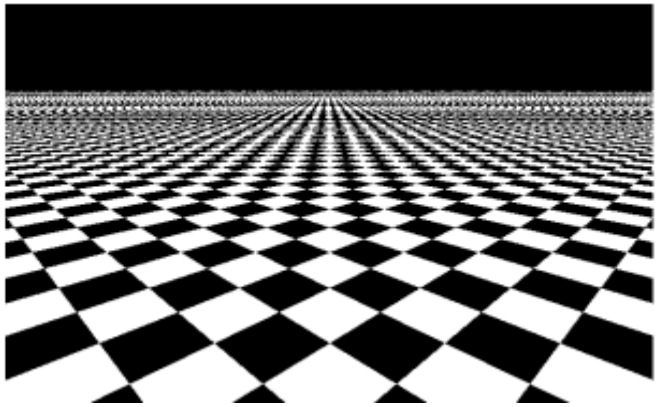
Анизотропная фильтрация

- анизотропная фильтрация (anisotropic filtering) – неоднородная фильтрация по различным направлениям
  - расширение идеи mipmap'ов → ripmap;
  - summed area table.

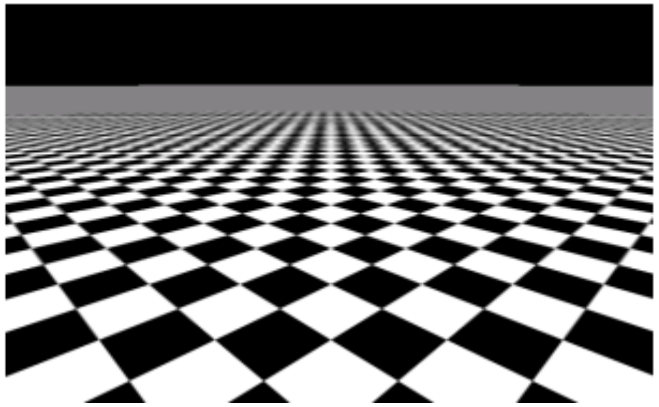




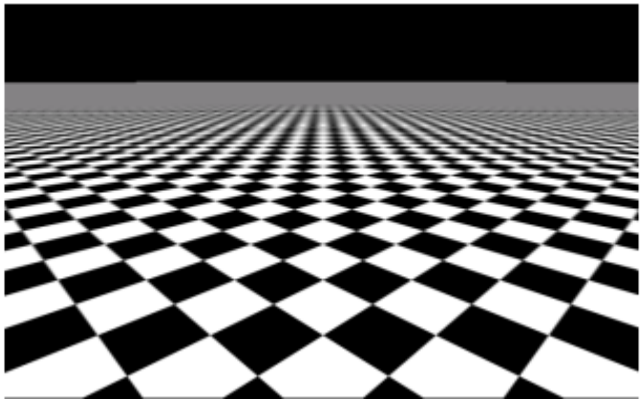
Сравнение фильтраций при текстурировании



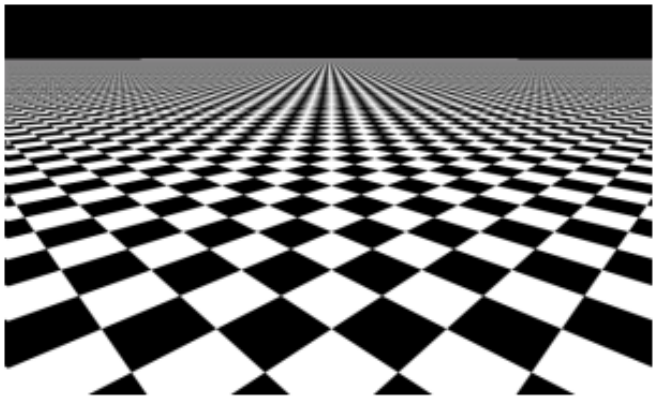
Ближайший сосед



Билинейная



Трилинейная



Анизотропная

Текстуры и OpenGL

- загрузка текстуры в память:
  - процедурная генерация;
  - загрузка из файла формата *.bmp;
  - загрузка из файлов других графических форматов (с использованием сторонних библиотек);
- текстуры должны иметь размеры, кратные степеням двойки
  - можно использовать функцию **gluScaleImage(…)**;
- генерация текстур меньших уровней детализации (mipmaps)

  gluBuild2DMipmaps(…)
- создание идентификатора и привязка (выбор в качестве текущей) текстуры:

  glGenTextures(…)

  glBindTexture(…)
- инициализация режима наложения поверхностных текстур:

  glTexImage2D(GLenum target, GLint level, GLint internalformat,
  GLsizei width, GLsizei height, GLint border,
  GLenum format, GLenum type, const GLvoid *pixels)

  glEnable(GL_TEXTURE_2D)

Текстуры и OpenGL (2)

- определение соответствия текстурных координат для каждой вершины объекта:

  glTexCoord2*(*s*, *t*)

  glTexGen(...)

- определение параметров текстуры, в частности:
  - определение режима интерпретации значений текстурных координат вне интервала [0;1] (по умолчанию – оборачивание: GL_REPEAT):

    glTexParameter*(GL_TEXTURE_2D, GL_TEX_WRAP_S | GL_TEX_WRAP_T, GL_REPEAT | GL_CLAMP)
  - определение режима интерполяции (по умолчанию – GL_LINEAR)

    glTexParameter*(GL_TEXTURE_2D, GL_TEXTURE_MIN_FILTER | GL_TEXTURE_MAG_FILTER, GL_NEAREST | GL_LINEAR | GL_NEAREST_MIPMAP_NEAREST | …)
- определение режима использования текстурных значений:
  - в качестве функции интенсивности I = texture(s,t)

    glTexEnv*(GL_TEXTURE_ENV, GL_TEXTURE_ENV_MODE, GL_REPLACE [GL_DECAL])
  - в качестве функции коэффициента диффузного отражения - «модулирование коэффициента отражения»:

    glTexEnv*(GL_TEXTURE_ENV, GL_TEXTURE_ENV_MODE, GL_MODULATE)

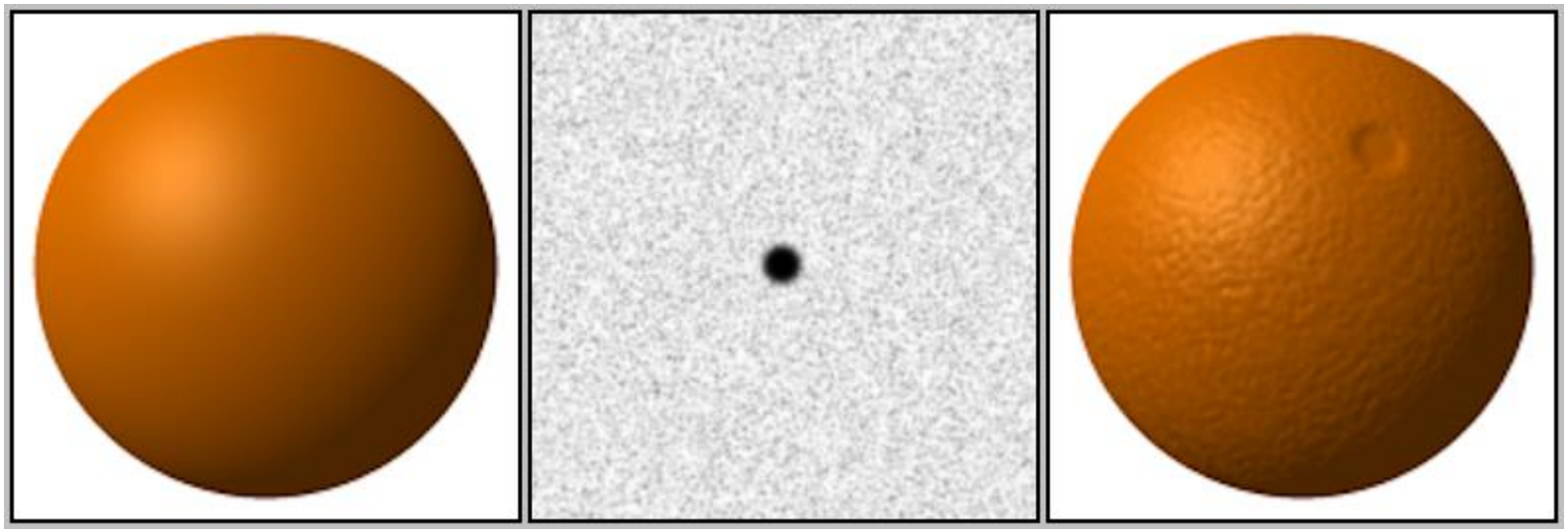
Фактура: метод возмущения нормали

- bump mapping (J. Blinn, 1978)
- displacement map (карта смещений)

$$P^*(s,t) = P(s,t) + d(s,t)\vec{n}(s,t)$$

$$\frac{\partial P^*}{\partial s} = \frac{\partial P}{\partial s} + \frac{\partial d}{\partial s}\vec{n} + d\frac{\partial \vec{n}}{\partial s} \qquad \frac{\partial P^*}{\partial t} = \frac{\partial P}{\partial t} + \frac{\partial d}{\partial t}\vec{n} + d\frac{\partial \vec{n}}{\partial t}$$

$$\vec{n}_{new} \approx \left(\frac{\partial P}{\partial s} + \frac{\partial d}{\partial s}\vec{n}\right) \times \left(\frac{\partial P}{\partial t} + \frac{\partial d}{\partial t}\vec{n}\right)$$



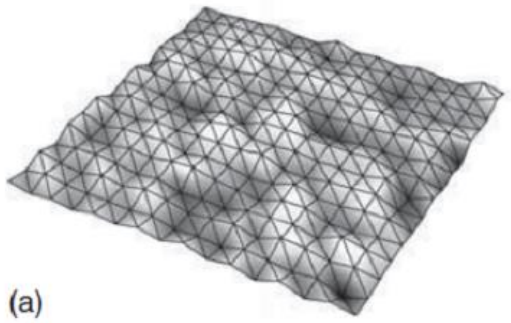
Модулирование параметров модели

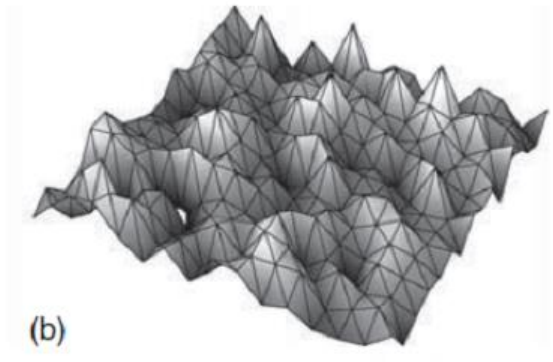
$$I = I_a k_a + I_l k_d \cos\theta + I_l k_s \cos^n \alpha$$

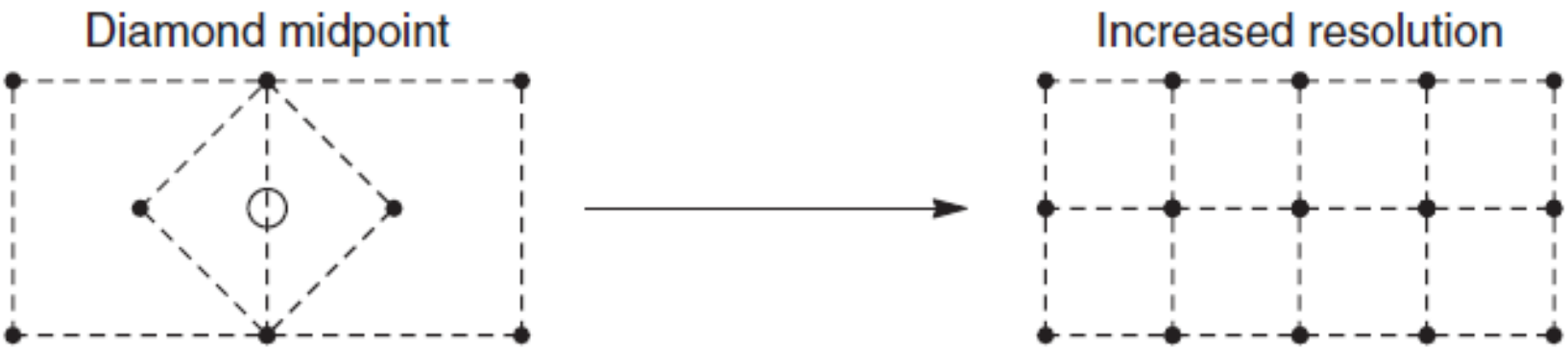
- отражение от поверхности – surface color texture;
- вектор нормали N – bump mapping;
- коэффициенты k_d, k_s, k_a, α – specularity mapping;
- падающий свет I_l – environment mapping;
- геометрия – displacement mapping
- прозрачность – transparency mapping;
- …

Генерация рельефа

- требования к алгоритму: естественный вид поверхности, построенной на основе случайных отклонений высоты, при различных уровнях детализации;
- алгоритмы на основе смещения средней точки (midpoint displacement techniques) – Diamond-Square algorithm
- http://www.vterrain.org



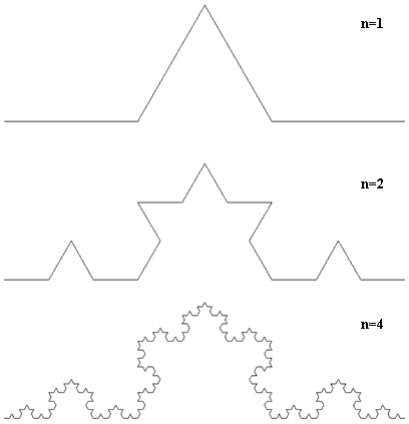


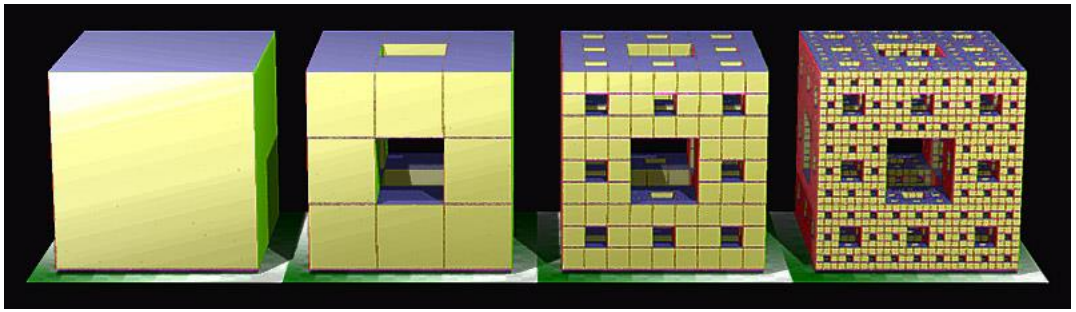


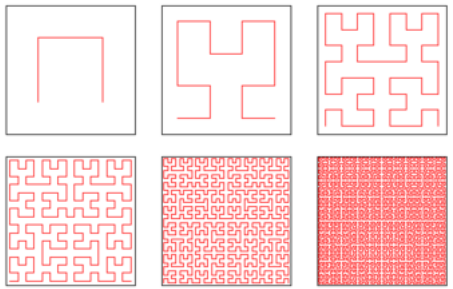
Фракталы

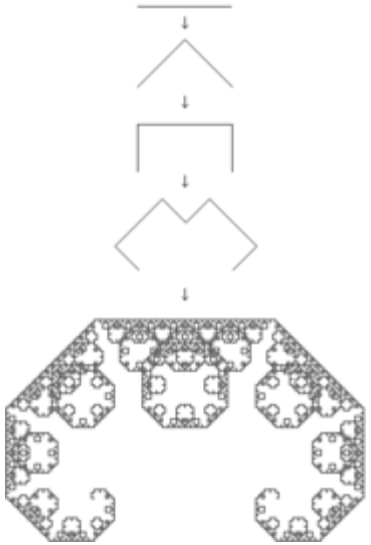
- Бенуа Мандельброт (Benoit Mandelbrot)*;*
- *фрактал* - бесконечно самоподобная геометрическая фигура, каждый фрагмент которой повторяется при уменьшении масштаба (масштабная инвариантость, наблюдаемая во фракталах, может быть либо точной, либо приближённой);
- процедура построения:
  - геометрические фракталы;
  - алгебраические фракталы (итерации нелинейных отображений, задаваемых простыми алгебраическими формулами);
  - стохастические фракталы;
- фракталы позволяют описывать целые классы изображений, для детального описания которых требуется относительно мало памяти;
- фракталы используются при создании изображений сложных, похожих на природные, объектов (облаков, снега, береговых линий, рельефа и др.).

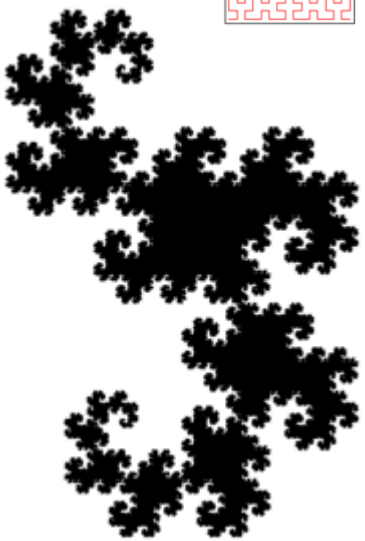
Геометрические фракталы

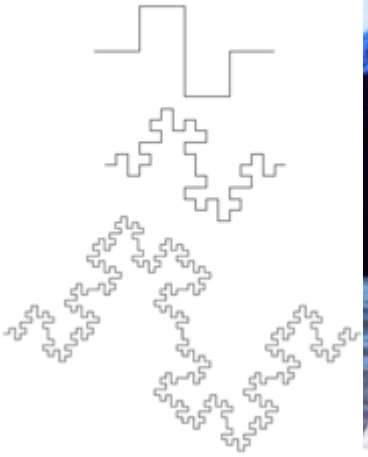


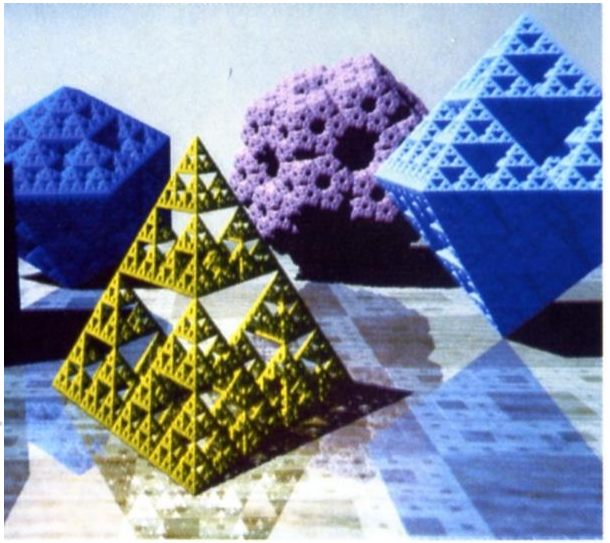






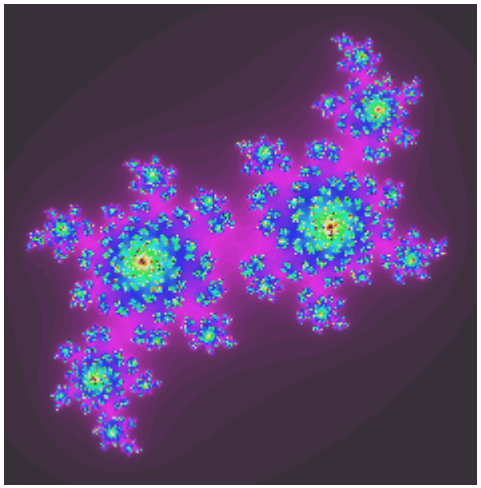




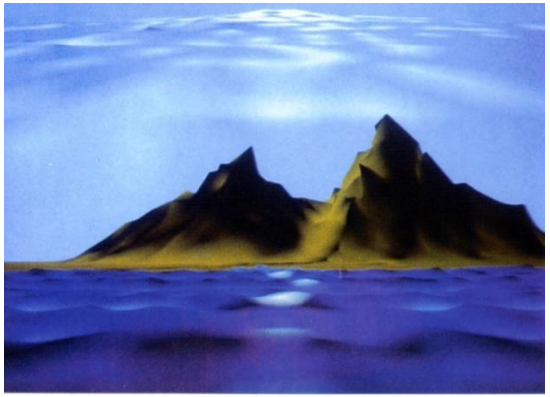


Алгебраические фракталы

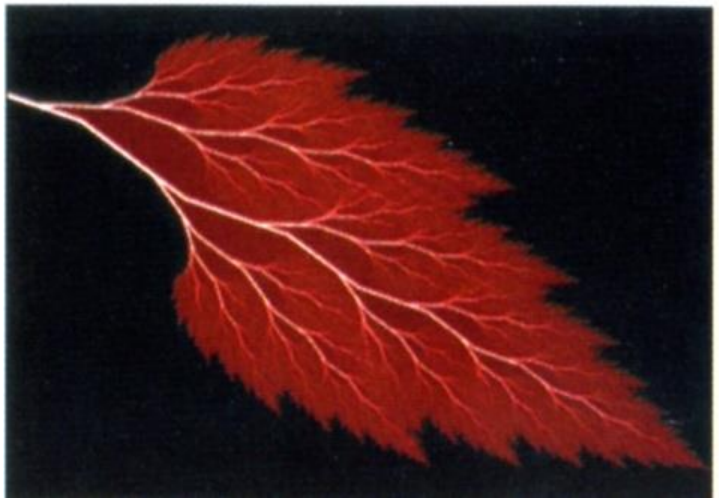


Стохастические фракталы: рельеф

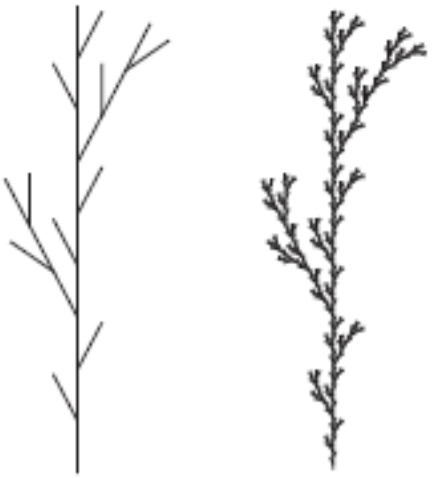


Стохастические фракталы: природные объекты



L-системы (L-systems)

- системы, описывающие рекурсивную природу роста растений (Aristid Lindenmayer, 1968)
- L-системой называется грамматика, в которой все возможные правила вывода (productions) применяются одновременно;
- детерминированная контекстно-свободная L-система – L-система, в которой для каждого символа существует единственное правило вывода, имеющее вид **λ→W** (λ – символ, W – слово).



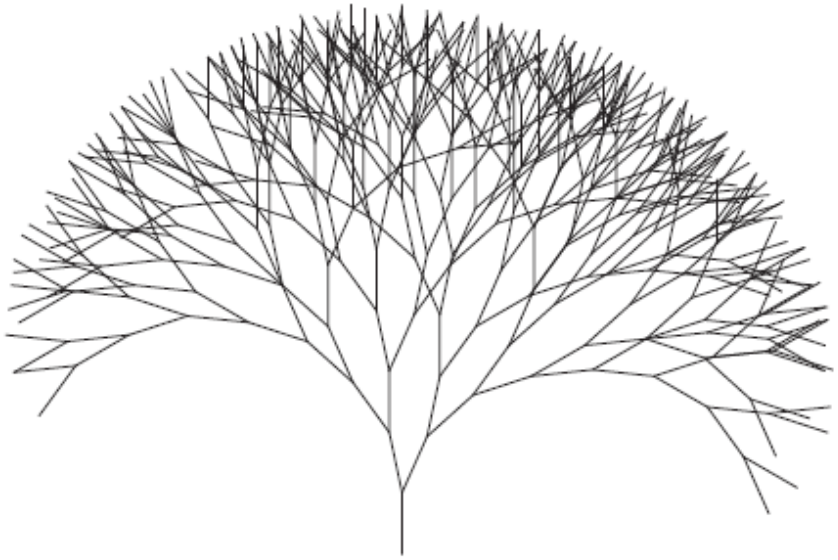
$$F_1 \rightarrow F_0[++F_1]-F_1$$

$$F_1 \rightarrow F_0[+F_1]-F_1$$

$$F_1 \rightarrow F_0[+F_1]-F_1$$

$$F_1 \rightarrow F_0[++F_1]-F_1$$

$$F_0 \rightarrow F_0$$



Лабораторная работа №6 - построение реалистичных изображений

- базовой лабораторной работой является лабораторная работа №3 (модельно-видовые преобразования и преобразования проецирования);
- определить параметры модели освещения OpenGL (свойства источника света, свойства материалов (поверхностей), характеристики глобальной модели освещения);
- исследовать один из методов повышения реалистичности получаемых изображений сцены (в соответствии с вариантом);
- реализовать один из алгоритмов анимации (в соответствии с вариантом);
- реализовать наложение текстуры (загрузка из файла *.bmp или процедурная генерация) с возможностью отключения (в соответствии с вариантом);
- разработать формат описания сцены и обеспечить сохранение и загрузку сцены из файла описания.

Вопросы к экзамену

- Анимация: определение, процедура. Спецификация движения. Твининг-анимация. Учёт физических законов при анимации. Обнаружение столкновений. Абсолютно упругие соударения.
- Простая модель освещения и её компоненты: диффузное отражение, зеркальное отражение, рассеянный свет. Вычисление вектора отражения и промежуточного вектора. Учёт расстояния от источника света до поверхности.
- Модель освещения OpenGL и ее параметры: общий вид модели освещения, определение характеристик источника света, определение параметров среды и свойств граней (материалов).
- Плоское и плавное закрашивание граней. Метод Гуро. Метод Фонга.
- Прозрачность и преломление. Вычисление преломлённого луча. Альфа-канал. Комбинирование цветов с учётом прозрачности. Прозрачность и смешивание цветов в OpenGL.
- Построение теней при создании реалистических изображений. Учет теней в алгоритмах удаления невидимых поверхностей.
- Алгоритм трассировки лучей с использованием глобальной модели освещения.
- Использование текстурных карт при создании реалистичных изображений. Процедурная генерация текстур. Метод возмущения нормали. Наложение текстур и гиперболическая интерполяция. Масштабирование текстур и тектурная фильтрация.
- Фракталы и L-системы.