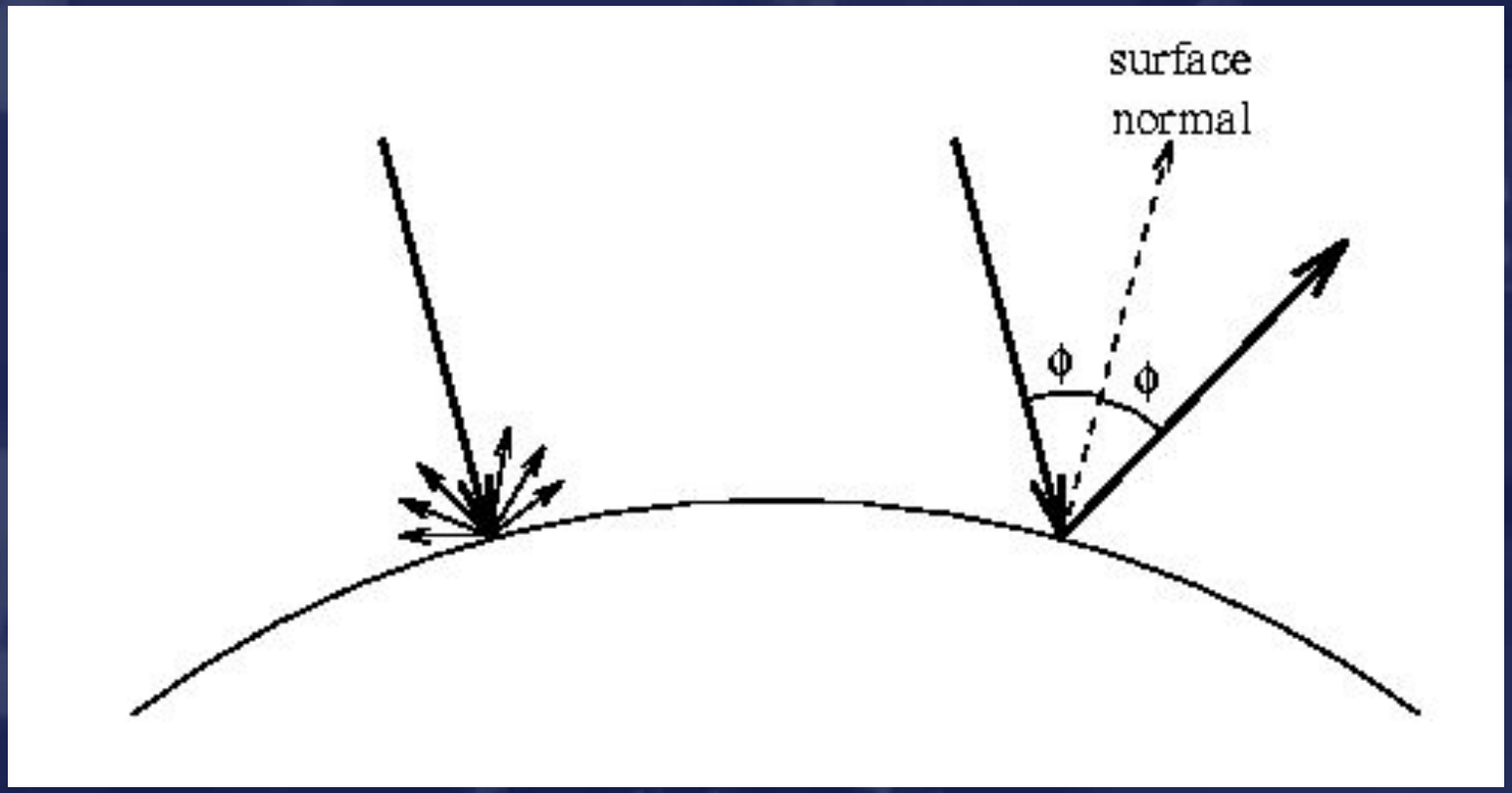
Освещение и текстурирование в OpenGL

Лекция 10

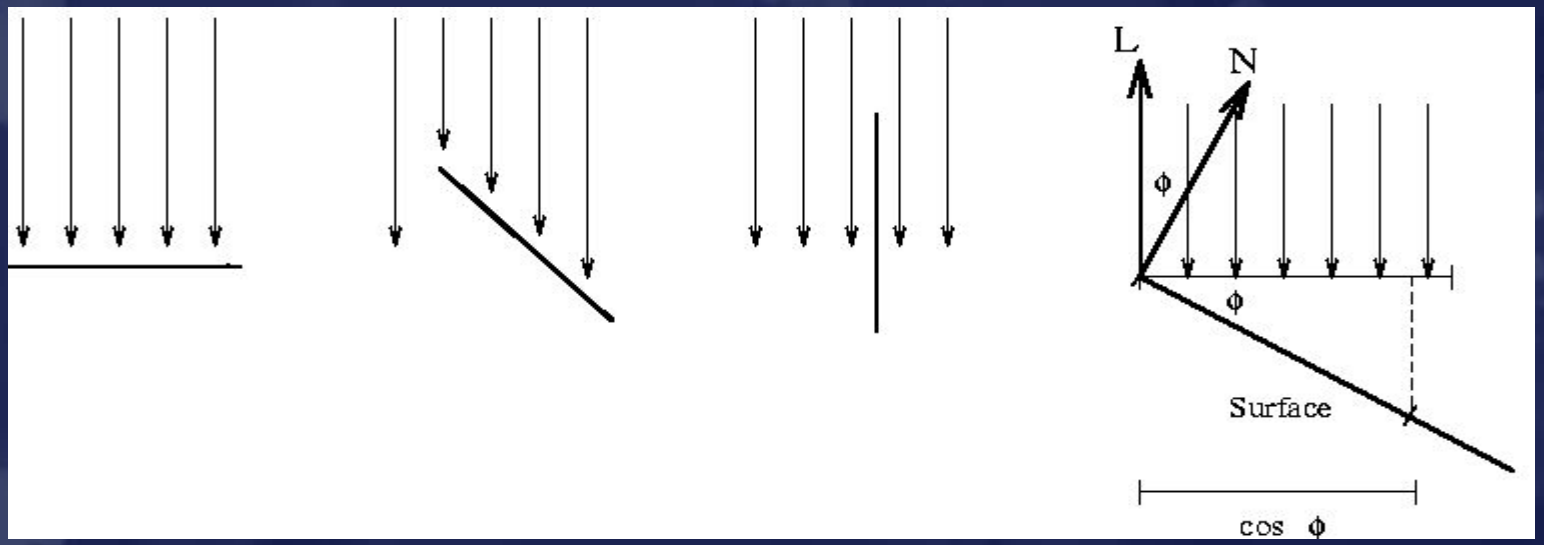
Астана 2004

Диффузное и зерк альное отражение



Диффузное отражение

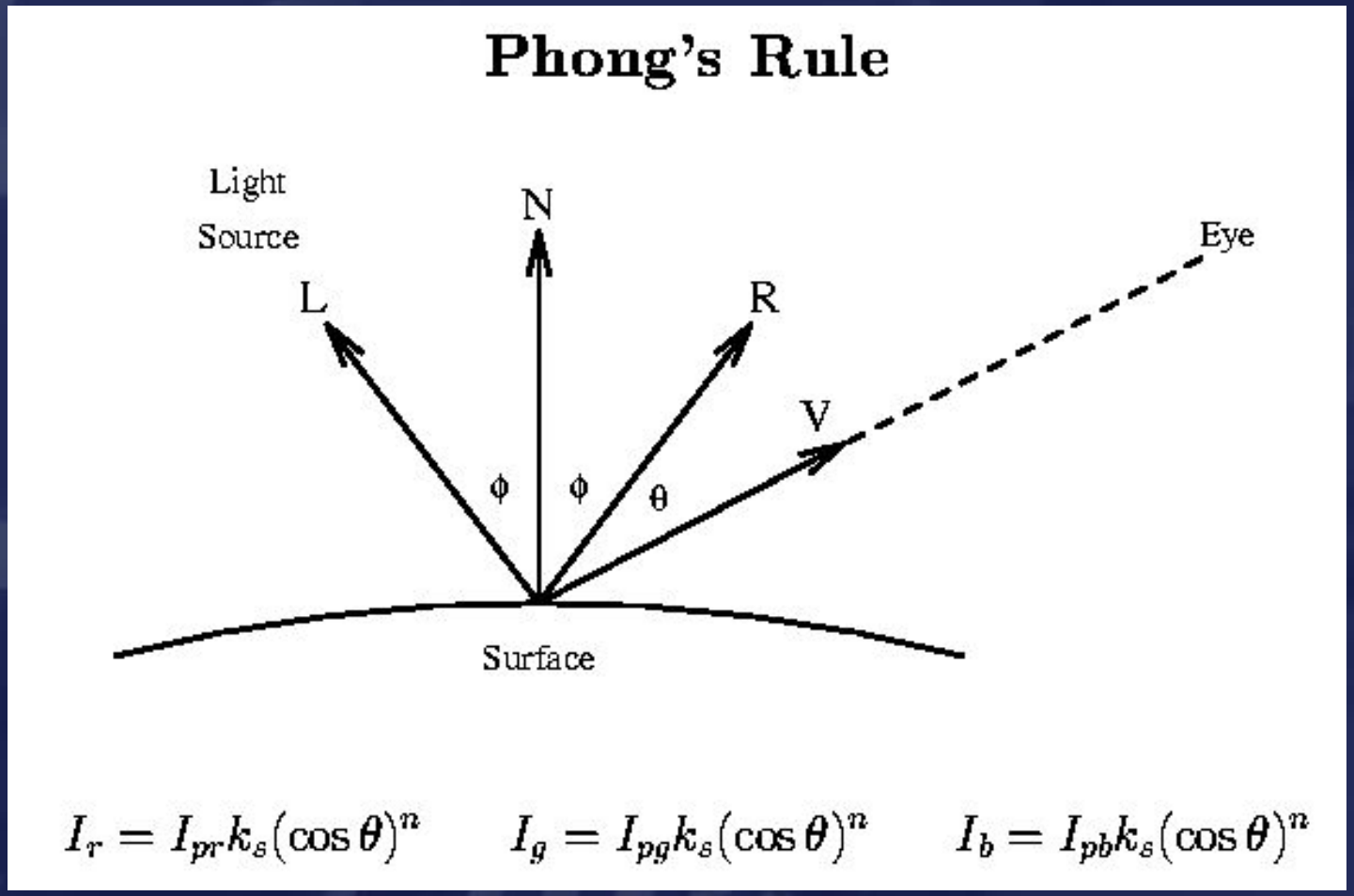
- Интенсивность освещения рассчитывается отдельно для каждой компонент R, G и B.



$$I_r = k_{dr}\frac{I_{pr}}{d}\cos\phi \qquad I_g = k_{dg}\frac{I_{pg}}{d}\cos\phi \qquad I_b = k_{db}\frac{I_{pb}}{d}\cos\phi$$

Зеркальное отражение

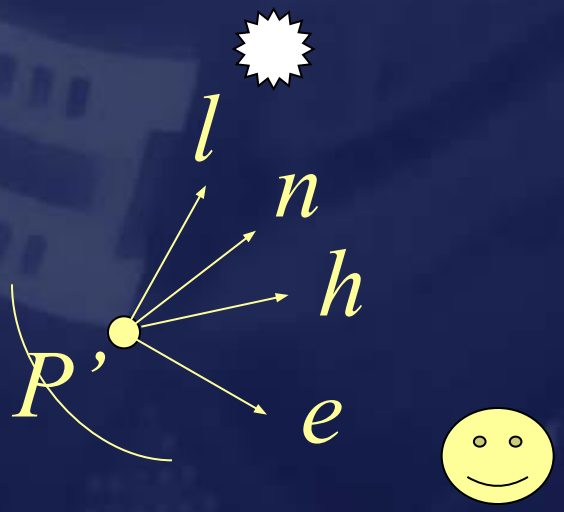
- Зеркальное отражение рассчитывается по закону Фонга



Уравнение освещенности

$$I = a_m a_l + d_m d_l (n \cdot l) + s_m s_l (n \cdot h)^{h_s}$$

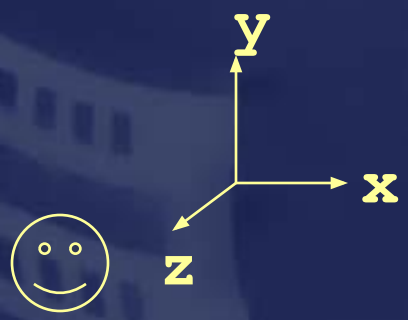
- Фоновое освещение не имеет источника и зависит только от сцены
- При диффузном освещении свет от источника равномерно рассеивается во всех направлениях.
- При зеркальном освещении свет от источника отражается от повехности.в одном направлении. Зеркальная освещенность дополнительно зависит от положения наблюдателя.



$$(a \cdot b) = \begin{cases} (a,b), & (a,b) \geq 0 \\ 0, & (a,b) < 0 \end{cases}$$

$$h = \frac{l + e}{\|l + e\|}$$

Рисуем тор

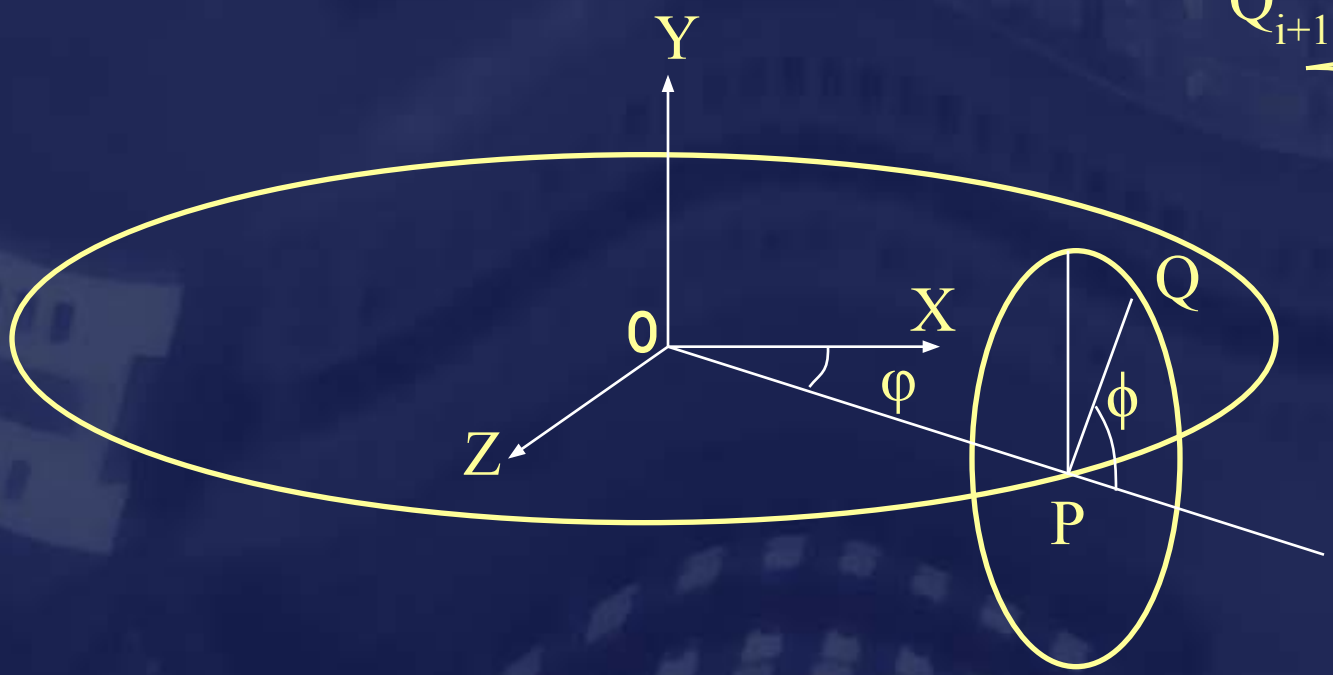


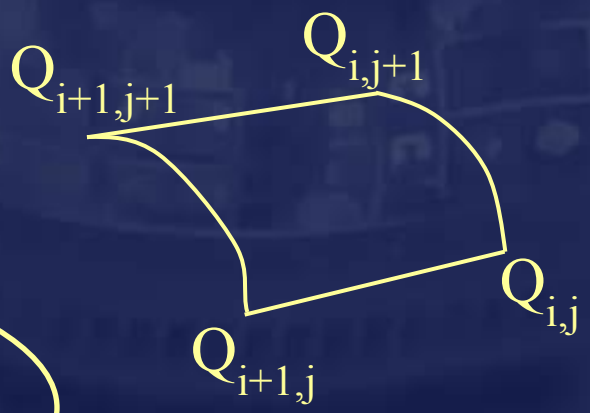
$$P(\phi) = (R_1 \cos\phi, 0, R_1 \sin\phi)$$

$$\bar{n}(\phi,\varphi) = (\cos\phi\cos\varphi, \sin\varphi, \sin\phi\cos\varphi)$$

$$Q(\phi,\varphi) = P(\phi) + R_2\bar{n}(\phi,\varphi)$$

$$\phi,\varphi \in [0,2\pi]$$





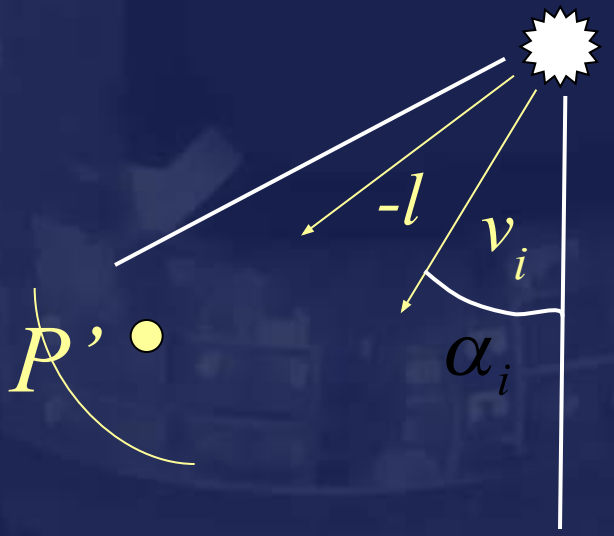
$$Q_{i,j} = Q(2\pi i / N_1, 2\pi j / N_2),\ i = 0,1,\ldots,N_1,\ j = 0,1,\ldots,N_2.$$

Уравнение освещенности OpenGL

$$c = e_m + a_m a_s + \sum_{i=0}^{n-1} att_i \cdot spot_i \cdot (a_m a_i + d_m d_i (n \cdot l) + s_m s_i (n \cdot h)^{h_m})$$

$$att_i = \frac{1}{k_{c,i} + k_{l,i} r + k_{q,i} r^2},$$

$$spot_i = \begin{cases} 1, \alpha_i = \pi, \\ 0, (v_i, -l) < \cos(\alpha_i), \\ (v_i, -l), (v_i, -l) \geq \cos(\alpha_i). \end{cases}$$



$spot_i$ – коэффициент направленности

att_i – коэффициент затухания

a_s – фоновое освещение

a_i, s_i, d_i – свойства i-го источника освещения

e_m, a_m, s_m, d_m, h_m – свойства материала

Установка параметров освещения в OpenGL

- Задаем параметры материала:

```
void glMaterialfv(GLenum face,GLenum param,GLfloat *value);
    face = {GL_FRONT|GL_BACK}
    param = {GL_AMBIENT|GL_DIFFUSE|GL_EMISSIVE|GL_SPECULAR}
    value = float[4] // RGBA

void glMaterialf(GLenum face,GL_SHININESS,GLfloat value);
```

- Задаем цвет фонового освещения:

```
void glLightModelfv(GLenum param,GLfloat *value);
    param = LIGHT_MODEL_AMBIENT
    value = float[4] // RGBA
```

- Задаем цвет источника освещения:

```
void glLightfv(GLenum light,GLenum param,GLfloat *value);
    face = {GL_LIGHT0|GL_LIGHT1|…}
    param = {GL_AMBIENT|GL_DIFFUSE|GL_SPECULAR}
    value = float[4] // RGBA
```

Установка параметров освещения. Часть 2.

- Задаем положение источника освещения:

```
void glLightfv(GLenum light, GL_POSITION,GLfloat *value);
    face = {GL_LIGHT0|GL_LIGHT1|…}
    value = float[4] // x,y,z,w
```

Координаты источника освещения преобразуются текущей матрицей модельного преобразования!

- Включаем расчет освещенности

```
void glEnable(GLenum type); type = GL_LIGHTING;
```

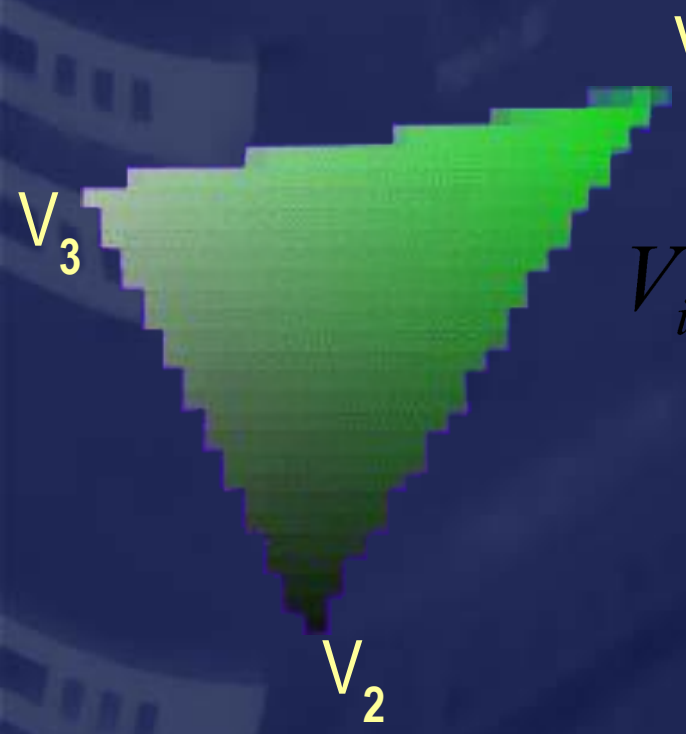
- Включаем требуемые источники освещения

```
void glEnable(GLenum type); type = GL_LIGHT0;
```

- Включаем требуемые источники освещения

```
void glShadeModel(GLenum type);
    type = GL_FLAT; - плоская закраска грани
    type = GL_SMOOTH - закраска по Гуро
```

Растеризация



$$V_i = \{x_i, y_i, z_i, RGBA_i, \ldots\}, i = 1,2,3$$

- Интерполяция координаты z

$$z(x, y) = L(x, y, z_i)$$

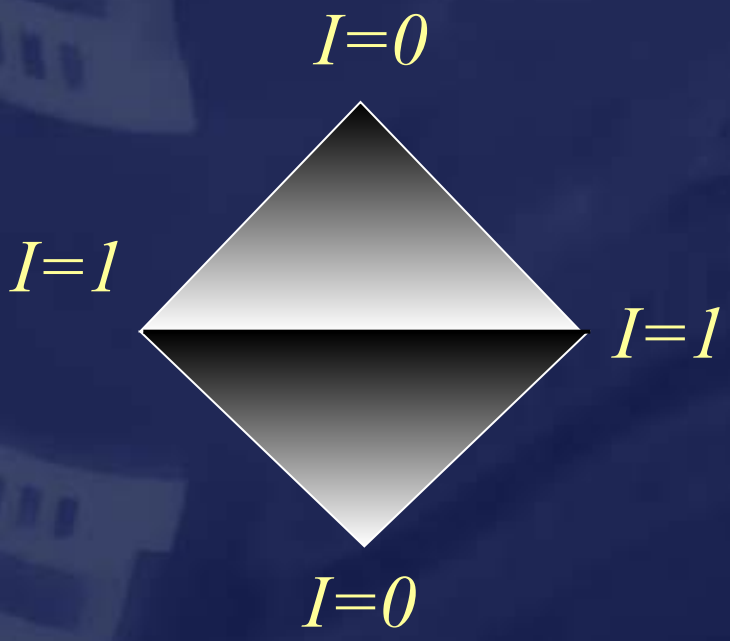
- Интерполяция цвета вдоль примитива - закраска по Гуро

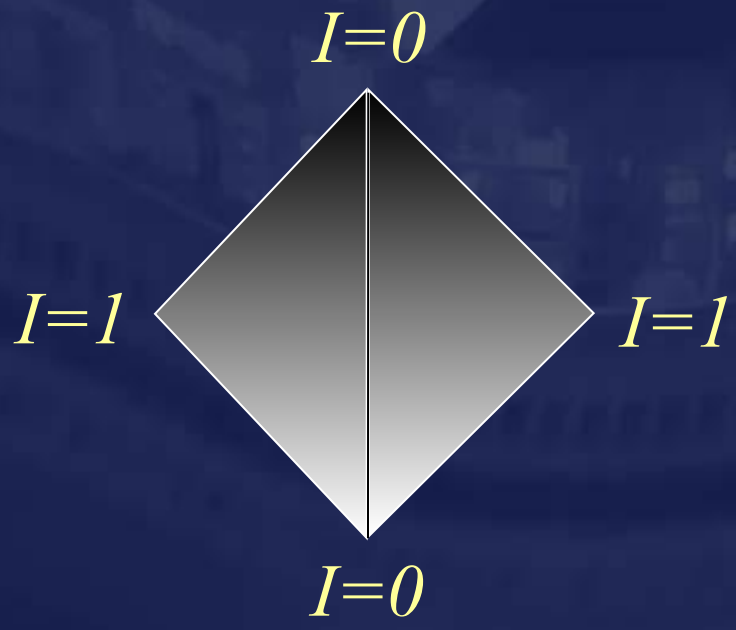
$$RGBA(x, y) = L(x, y, RGBA_i)$$

- Интерполяция цвета вдоль примитива - закраска по Фонгу

Фонг и Гуро - ошибки интерполяции

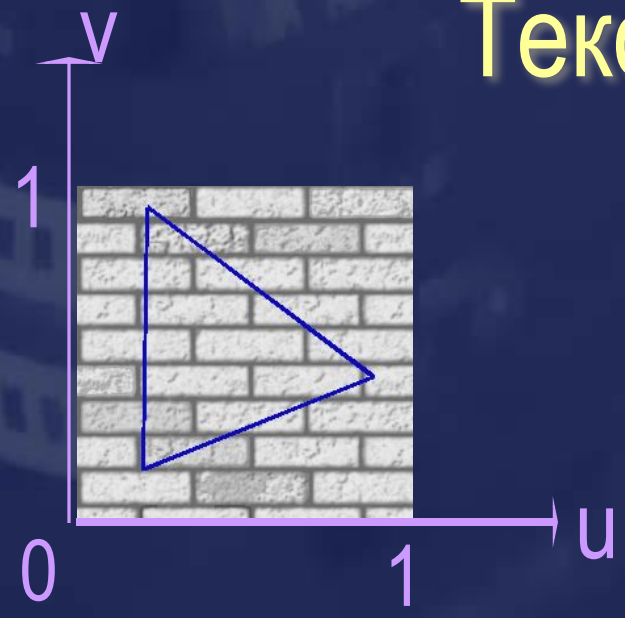
- Освещенность зависит от способа разбиения на примитивы





- Поле нормалей лучше задавать в виде текстуры!

Текстурирование



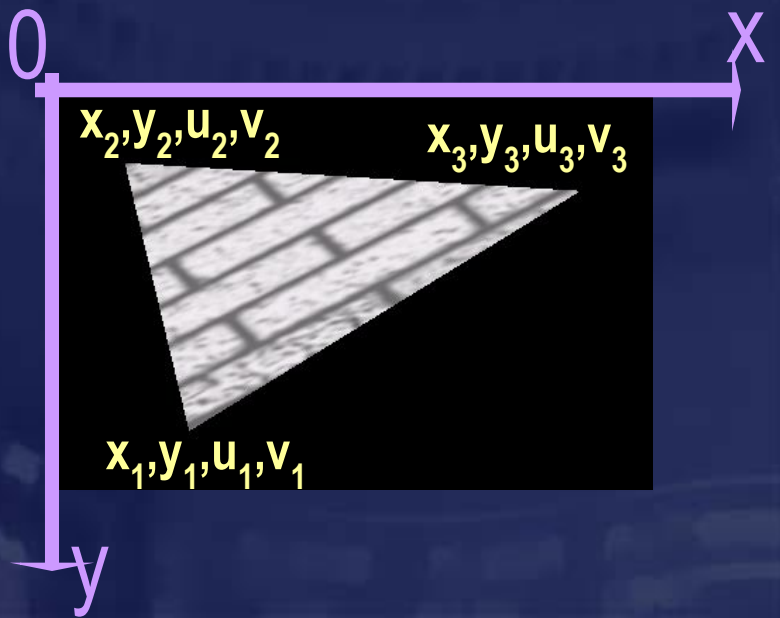
$$V_i = \{P_i, n_i, u_i, v_i, \ldots\}, i = 1,2,3$$

$$V_i' = \{x_i, y_i, u_i, v_i, \ldots\}, i = 1,2,3$$

“Перспективное” текстурирование:

$$u(x,y) = \frac{Ax + By + C}{Px + Qy + R}$$

$$v(x,y) = \frac{Dx + Ey + F}{Px + Qy + R}$$



Текстурирование в OpenGL

- Создаем текстуру - прямоугольный массив с цветами пикселов. Высота и ширина должны быть степенями двойки.

RGB_{00}	RGB_{10}	...	RGB_{N0}
RGB_{01}	RGB_{11}	...	RGB_{N1}
...	...	...	...
RGB_{0M}	RGB_{1M}	...	RGB_{NM}

$$N = 2^n - 1,$$

$$M = 2^m - 1.$$

- Получаем номер текстурного объекта

```
GLuint texture;
glGenTextures(1,&texture);
```

- Активизируем текстурный объект

```
glBindTexture(texture);
```

Текстурирование в OpenGL: часть 2

- Загружаем текстуру

```
glPixelStorei(GL_UNPACK_ALIGNMENT,1);
glTexImage2D(GL_TEXTURE_2D,
        0,                      // Mip-level
        GL_RGB,                 // Формат текстуры
        tex_width,tex_height,
0,                      // Ширина границы
            GL_RGB,                 // Формат исходных данных
            GL_UNSIGNED_BYTE,       // Тип данных
        tex_bits);              // Исходные данные
```

- Устанавливаем режимы текстурирования

```
glTexParameteri(GL_TEXTURE_2D,
                GL_TEXTURE_WRAP_S, GL_REPEAT);
glTexParameteri(GL_TEXTURE_2D,
                GL_TEXTURE_WRAP_T, GL_REPEAT);
glTexParameteri(GL_TEXTURE_2D,
                GL_TEXTURE_MAG_FILTER, GL_LINEAR);
glTexParameteri(GL_TEXTURE_2D,
                GL_TEXTURE_MIN_FILTER, GL_LINEAR);
```

Текстурирование в OpenGL: часть 3

- Разрешаем текстурирования

```
glEnable(GL_TEXTURE_2D);
```

- Задаем текстурные координаты

```
glTexCoord2d(u,v);
```

- Возможно, потребуется включить режим перспективного текстурирования

```
glHint(GL_PERSPECTIVE_CORRECTION_HINT,GL_NICEST);
```

- Возвращаем номер текстурного объекта в список свободных

```
glDeleteTextures(1,&texture);
```