NOTAS SOBRE VISUALIZAÇÃO

ERNESTO P. LOPES, MARCELA S. M. DA SILVA, RAFAEL V. CARVALHO,
AND ELIANA P. L. AUDE

Resumo. Neste trabalho é apresentado uma exposição dos conceitos teóricos da computação gráfica referentes às transformações geométricas, a projeção e a iluminação que permitem a visualização das cenas. É dada uma breve explanação de como a biblioteca gráfica OpenGL implementa esses conceitos.

1. Introdução

A percepção de espacialidade de uma imagem pode ser entendida como a capacidade do ser humano de distinguir as formas, as cores, as texturas, e a relação espacial entre objetos pertencentes à uma cena através desta imagem. Considera-se cena uma porção finita qualquer do mundo real, para obter uma representação de uma cena, na visão humana, o cérebro leva em conta diversas informações obtidas por ambos os olhos. Os movimentos do globo ocular, as cores e as sombras dos objetos contribuem decisivamente para a construção desta representação. Cada olho forma uma imagem bidimensional, que é uma fotografia temporária dos objetos na retina. Essas duas imagens bidimensionais são ligeiramente diferentes devido a separação dos dois olhos que recebem as imagens em dois ângulos distintos, conforme a Figura 1. O cérebro combina estas imagens para produzir um único quadro composto 3D.

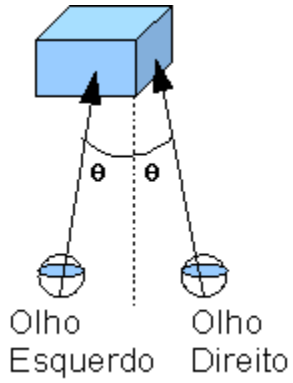

Figura 1. Visão 3D

A computação gráfica procura criar, para as cenas, imagens bidimensionais com boa percepção de espacialidade. Os sistemas gráficos devem transformar formas

Date: 27 de novembro de 2006.

3D em imagens bidimensionais para serem apresentadas em uma tela plana. Elas deverão dar a ilusão de profundidade, ou de tridimensão. Isto é feito através da projeção. Além disso, esses sistemas devem procurar facilitar o acesso do observador às informações contidas na cena e possibilitar a criação de novas cenas, isso é feito através das transformações geométricas e da iluminação. Inicialmente será discutido as transformações geométricas e em seguida as projeções e iluminação. E, por fim, será apresentado uma prévia dos recurso da biblioteca gráfica OpenGL.

2. Transformações Geométricas

As transformações geométricas são ferramentas importantes na manipulação das cenas tridimensionais. Elas são utilizadas, por exemplo, para transladar, rotacionar, escalar e espelhar os objetos, atuando no conjunto dos vértices ou pontos que definem esses objetos, transformando-os em um novo conjunto de vértices ou pontos. Nesta seção será apresentado primeiro as *coordenadas homogêneas*, que são a representação adequada para pontos e vetores e para as suas transformações geométricas em um sistema gráfico. Em seguida, explanar-se-á as transformações geométricas.

2.1. **Coordenadas Homogêneas.** A representação de pontos e vetores na álgebra linear ordinária é ambígua, uma vez fixado um sistema de coordenadas retangulares, um elemento do R^3 representa ao mesmo tempo um ponto e um vetor. O uso das *coordenadas homogêneas* permite resolver este problema. Em coordenadas homogêneas, pontos de um espaço afim de dimensão 3 e vetores do espaço vetorial associado são representados univocamente por elementos do R^4. As transformações geométricas de rotação, translação e de escala assumem uma representação matricial como uma transformação linear em espaços vetoriais de dimensão 4. Em um espaço afim, um *marco* é definido por um ponto, chamado origem e pelos vetores de uma base do espaço vetorial associado, ou seja, no caso de dimensão 3, que será o único tratado neste trabalho, pela especificação: $(\mathbf{v}_1, \mathbf{v}_2, \mathbf{v}_3, \mathbf{P}_0)$[1], onde os três primeiros elementos são vetores da base e o quarto é a origem. Qualquer ponto $\mathbf{P}$ do espaço afim pode ser escrito de uma maneira única como [MdBS98]:

$$\mathbf{P} = \alpha_1\mathbf{v}_1 + \alpha_2\mathbf{v}_2 + \alpha_3\mathbf{v}_3 + \mathbf{P}_0$$

assim, podemos expressar esta relação, usando um produto formal de matrizes, como:

$$\mathbf{P} = \begin{bmatrix} \alpha_1 & \alpha_2 & \alpha_3 & 1 \end{bmatrix} \begin{bmatrix} \mathbf{v}_1 \\ \mathbf{v}_2 \\ \mathbf{v}_3 \\ \mathbf{P}_0 \end{bmatrix}.$$

[1]Convenção: foi definido neste trabalho que os vetores serão representados por letras minusculas e em negrito; os pontos serão representados em letras maiúsculas e também em negrito; e os numeros reais serão representados por letras minusculas e em itálico ou por letras do alfabeto grego.

A matriz linha de é a representação das coordenadas homogêneas do ponto $\mathbf{P}$ no marco determinado por $\mathbf{v}_1, \mathbf{v}_2, \mathbf{v}_3$ e $\mathbf{P}_0$. Equivalentemente, podemos dizer que $\mathbf{P}$ pode ser representado por:

$$\mathbf{P} = \begin{bmatrix} \alpha_1 \\ \alpha_2 \\ \alpha_3 \\ 1 \end{bmatrix}.$$

A representação de um vetor tridimensional segue uma lógica similar. No mesmo marco, qualquer vetor $\mathbf{w}$ pode ser escrito como:

$$\mathbf{w} = \delta_1\mathbf{v}_1 + \delta_2\mathbf{v}_2 + \delta_3\mathbf{v}_3$$

$$\mathbf{w} = \begin{bmatrix} \delta_1 & \delta_2 & \delta_3 & 0 \end{bmatrix} \begin{bmatrix} \mathbf{v}_1 \\ \mathbf{v}_2 \\ \mathbf{v}_3 \\ \mathbf{P}_0 \end{bmatrix}.$$

Equivalentemente, $\mathbf{w}$ pode ser representado por:

$$\mathbf{w} = \begin{bmatrix} \delta_1 \\ \delta_2 \\ \delta_3 \\ 0 \end{bmatrix}.$$

Existem numerosas formas de interpretar esta formulação geometricamente, entretanto esse trabalho irá ressaltar apenas a simplicidade das operações com pontos e vetores utilizando suas representações em coordenadas homogêneas e a álgebra ma-tri-cial ordinária. Considerando a operação de mudança de marco. Sejam $(\mathbf{v}_1, \mathbf{v}_2, \mathbf{v}_3, \mathbf{P}_0)$ e $(\mathbf{u}_1, \mathbf{u}_2, \mathbf{u}_3, \mathbf{Q}_0)$ dois marcos, a expressão do segundo marco em função do primeiro é:

$$\mathbf{u}_1 = \gamma_{11}\mathbf{v}_1 + \gamma_{12}\mathbf{v}_2 + \gamma_{13}\mathbf{v}_3$$

$$\mathbf{u}_2 = \gamma_{21}\mathbf{v}_1 + \gamma_{22}\mathbf{v}_2 + \gamma_{23}\mathbf{v}_3$$

$$\mathbf{u}_3 = \gamma_{31}\mathbf{v}_1 + \gamma_{32}\mathbf{v}_2 + \gamma_{33}\mathbf{v}_3$$

$$\mathbf{Q}_0 = \gamma_{41}\mathbf{v}_1 + \gamma_{42}\mathbf{v}_2 + \gamma_{43}\mathbf{v}_3 + \mathbf{P}_0$$

Estas equações, na forma de matrizes, podem ser escritas como:

$$\begin{bmatrix} \mathbf{u}_1 \\ \mathbf{u}_2 \\ \mathbf{u}_3 \\ \mathbf{Q}_0 \end{bmatrix} = M \begin{bmatrix} \mathbf{v}_1 \\ \mathbf{v}_2 \\ \mathbf{v}_3 \\ \mathbf{P}_0 \end{bmatrix},$$

onde M é uma matriz 4×4

$$M = \begin{bmatrix} \gamma_{11} & \gamma_{12} & \gamma_{13} & 0 \\ \gamma_{21} & \gamma_{22} & \gamma_{23} & 0 \\ \gamma_{31} & \gamma_{32} & \gamma_{33} & 0 \\ \gamma_{41} & \gamma_{42} & \gamma_{43} & 1 \end{bmatrix}.$$

Considerando M' uma submatriz 3×3 da matriz M, cujo seus elementos são:

$$M' = \begin{bmatrix} \gamma_{11} & \gamma_{12} & \gamma_{13} \\ \gamma_{21} & \gamma_{22} & \gamma_{23} \\ \gamma_{31} & \gamma_{32} & \gamma_{33} \end{bmatrix}.$$

A matriz M' é a matriz de mudança de base no R^3, portanto, ela é não singular e seu determinante é não nulo. Isto é verdadeiro se e somente se a o determinante da matriz M 4×4 é não nulo. M é chamada de *matriz de representação*. Podemos utilizá-la para computar diretamente as alterações na representação de pontos e vetores. Supondo que $\mathbf{a}$ e $\mathbf{b}$ são representações em coordenadas homogêneas de um vetor em dois marcos distintos, Então:

$$\mathbf{b}^T \begin{bmatrix} \mathbf{u}_1 \\ \mathbf{u}_2 \\ \mathbf{u}_3 \\ \mathbf{Q}_0 \end{bmatrix} = \mathbf{b}^T M \begin{bmatrix} \mathbf{v}_1 \\ \mathbf{v}_2 \\ \mathbf{v}_3 \\ \mathbf{P}_0 \end{bmatrix} = \mathbf{a}^T \begin{bmatrix} \mathbf{v}_1 \\ \mathbf{v}_2 \\ \mathbf{v}_3 \\ \mathbf{P}_0 \end{bmatrix}.$$

$$\mathbf{a}^T = \mathbf{b}^T M$$
$$(\mathbf{a}^T)^T = (\mathbf{b}^T M)^T$$

Assim,

$$\mathbf{a} = M^T \mathbf{b}.$$

Claro que M^T, é

$$M^T = \begin{bmatrix} \gamma_{11} & \gamma_{21} & \gamma_{31} & \gamma_{41} \\ \gamma_{12} & \gamma_{22} & \gamma_{32} & \gamma_{42} \\ \gamma_{13} & \gamma_{23} & \gamma_{33} & \gamma_{43} \\ 0 & 0 & 0 & 1 \end{bmatrix},$$

que, assim como M, é determinada por 12 coeficientes.

É importante observar que a composição das transformações geométricas pode ser representada pela mul-ti-pli-ca-ção das representações matriciais dessas transformações. Embora se tenha que trabalhar em quatro dimensões para resolver problemas tridimensionais, o uso esse tipo de representação simplifica muito os cálculos e a programação.

2.2. **Translação.** A *translação* é uma operação que desloca um ponto, em uma dada direção, por uma distância fixada, como ilustrado na Figura 2. Se movermos o ponto $\mathbf{P}$, na direção do vetor $\mathbf{d}$, de uma distância $|d|$ então: $\mathbf{P'} = \mathbf{P} + \mathbf{d}$.

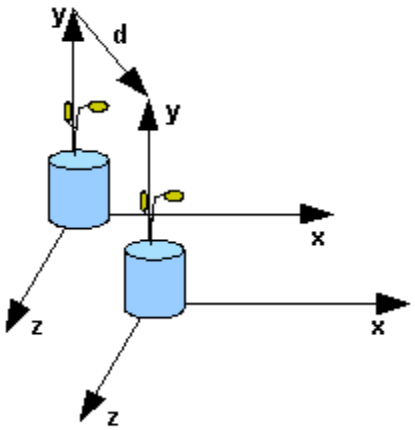

FIGURA 2. Translação de um objeto

Em coordenadas homogêneas temos:

$$\mathbf{P} = \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}, \mathbf{P'} = \begin{bmatrix} x' \\ y' \\ z' \\ 1 \end{bmatrix} \text{ e } \mathbf{d} = \begin{bmatrix} \alpha_x \\ \alpha_y \\ \alpha_x \\ 0 \end{bmatrix}$$

Logo temos:

$$x' = x + \alpha_x$$
$$y' = y + \alpha_y$$

$$z' = z + \alpha_z$$

Vemos que uma translação é obtida pela multiplicação de uma matriz por um vetor:

$$\mathbf{P}' = T\mathbf{P},$$

onde

$$T = \begin{bmatrix} 1 & 0 & 0 & \alpha_x \\ 0 & 1 & 0 & \alpha_y \\ 0 & 0 & 1 & \alpha_z \\ 0 & 0 & 0 & 1 \end{bmatrix}.$$

T é chamada *matriz de translação* e pode ser escrita como $T(\alpha_x, \alpha_y, \alpha_z)$ para enfatizar os três parâmetros de que ela depende, os demais estando fixados. É importante ressaltar que o ponto do marco que estiver sobre a origem do sistema de coordenadas será transferido, após a translação, para um novo ponto do universo. Esta característica em geral é utilizada na criação de modelos com o objetivo de posicioná-los no universo da melhor forma possível.

2.3. **Rotação.** A *rotação* define a orientação da imagem no universo. Sua especificação é mais complexa que a da translação, pois mais parâmetros estão envolvidos. A operação de rotação desloca um ponto em torno de um determinado eixo cartesiano de um ângulo θ. Essa operação inicia-se pela rotação de um ponto $\mathbf{P}$ em torno de outro ponto $\mathbf{Q}$ em um plano, especificando o ponto $\mathbf{Q}$ como origem e uma base ortogonal $(\mathbf{v}_1, \mathbf{v}_2, \mathbf{v}_3)$ e definindo um dos vetores, como por exemplo o vetor $\mathbf{v}_3$, como sendo um vetor perpendicular ao plano de rotação. Assim, fica definido um novo marco $(\mathbf{v}_1, \mathbf{v}_2, \mathbf{v}_3, \mathbf{Q})$. Supondo que o ponto $\mathbf{P}$, cuja representação neste marco é $(x, y, 0, \mathbf{P} + (\mathbf{Q} - \mathbf{P}))$, é rotacionado em torno de $\mathbf{v}_3$ de um ângulo θ para a posição $\mathbf{P}' = (x', y', 0, \mathbf{P} + (\mathbf{Q} - \mathbf{P}))$. As equações desta rotação no plano são obtidas como segue:

Escrevemos (x, y) e (x', y') na *forma polar*:

$$\begin{aligned} x &= \rho\cos\phi, \\ y &= \rho\sin\phi, \\ x' &= \rho\cos(\phi + \theta), \\ y' &= \rho\sin(\phi + \theta), \end{aligned}$$

Onde ρ é $||\mathbf{Q} - \mathbf{P}||$.

Usando as identidades trigonométricas para o seno e o cosseno da soma de dois ângulos, obtém-se:

$$x' = \rho\cos\phi\cos\theta - \rho\sin\phi\sin\theta = x\cos\theta - y\sin\theta,$$

$$y' = \rho\cos\phi\sin\theta + \rho\sin\phi\cos\theta = x\sin\theta + y\cos\theta.$$

Estas equações podem ser escritas na forma matricial como:

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix}$$

Essa matriz é a da rotação num plano em torno de um ponto. Para expressão tridimensional da rotação do ponto $\mathbf{P}$ do ângulo θ em torno do eixo perpendicular ao plano, no marco $(\mathbf{v}_1, \mathbf{v}_2, \mathbf{v}_3, Q)$, temos:

$$\begin{aligned} x' &= x\cos\theta - y\sin\theta, \\ y' &= x\sin\theta + y\cos\theta, \\ z' &= z; \end{aligned}$$

e na forma matricial

$$\mathbf{P}' = (x', y', 0, \mathbf{Q}) = M_z(\theta)\mathbf{P}$$

onde:

$$M_z(\theta) = \begin{bmatrix} \cos\theta & -\sin\theta & 0 & 0 \\ \sin\theta & \cos\theta & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

A rotação pode ser feita em torno dos demais eixos ortogonais com uma argumentação idêntica. As demais matrizes para os outros eixos são:

$$M_x(\theta) = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos\theta & -\sin\theta & 0 \\ 0 & \sin\theta & \cos\theta & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix},$$

$$M_y(\theta) = \begin{bmatrix} \cos\theta & 0 & \sin\theta & 0 \\ 0 & 1 & 0 & 0 \\ -\sin\theta & 0 & \cos\theta & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}.$$

A matriz de rotação em coordenadas homogêneas, como a de todas as transformações lineares no espaço tridimensional, tem a forma:

$$\begin{bmatrix} \alpha_{11} & \alpha_{21} & \alpha_{31} & 0 \\ \alpha_{12} & \alpha_{22} & \alpha_{32} & 0 \\ \alpha_{13} & \alpha_{23} & \alpha_{33} & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix},$$

Quando:

$$\begin{bmatrix} \alpha_{11} & \alpha_{21} & \alpha_{31} \\ \alpha_{12} & \alpha_{22} & \alpha_{32} \\ \alpha_{13} & \alpha_{23} & \alpha_{33} \end{bmatrix},$$

é M_x, M_y ou M_z, denota-se R_x, R_y ou R_z as respectivas matrizes de rotação em coordenadas homogêneas. A notação para a rotação em torno de um eixo genérico é R, e é obtida pela composição das rotações anteriormente descritas.

Uma rotação R de um ângulo θ pode ser desfeita através da mesma rotação de ângulo $-\theta$, então

$$R^{-1}(\theta) = R(-\theta).$$

Além disso, em M como todos os termos com cosseno estão na diagonal e os termos com seno não estão na diagonal, pode-se utilizar as identidades trigonométricas $\cos(-\theta) = \cos\theta$ e $\sin(-\theta) = -\sin\theta$ para encontrar que:

$$R^{-1}(\theta) = R^T(\theta).$$

Para construir a matriz de uma rotação qualquer, em torno de um ponto fixado como origem, basta fazer o produto das rotações individuais sobre os três eixos

$$R = R_z R_y R_x$$

Como a transposta do produto de matrizes é o produto das transpostas na ordem inversa para uma rotação qualquer, obtém-se também:

$$R^{-1} = R^T.$$

Uma matriz cuja inversa é igual a sua transposta é chamada de *matriz ortogonal*. A vantagem desta propriedade é que não é necessário nenhum cálculo adicional para se obter a matriz inversa, bastando apenas alterar os índices de linhas e colunas da matriz.

A Figura 3 ilustra uma rotação feita em torno do eixo z. Neste caso, os pontos originais são multiplicados por uma matriz de rotação ao redor do eixo definido pelo vetor z no sentido anti-horário.

No caso da rotação em torno de um eixo genérico (e_x, e_y, e_z), basta transladar a figura a ser rotacionada para a origem (calcula-se, para isto, a matriz de translação

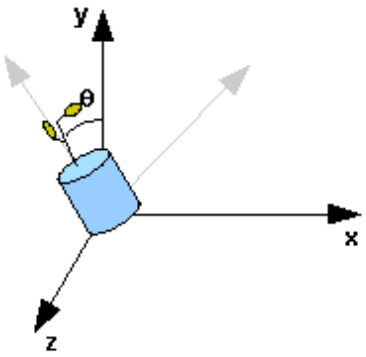

FIGURA 3. Rotação de um objeto em torno do eixo z

em relação ao eixo (e_x, e_y, e_z), aplicar as rotações necessárias e, finalmente, transladar a figura para a sua posição inicial, aplicando a inversa da matriz de translação calculada inicialmente.

2.4. **Escala.** O *escalamento* altera o tamanho de um objeto, podendo torná-lo maior ou menor em relação ao seu tamanho original. As transformações com escala possuem um ponto fixo, e para especificar uma escala, precisaremos deste ponto fixo, de uma direção na qual desejamos fazer a escala e um fator de escala (α). Para $\alpha > 1$, o objeto "cresce" em uma direção específica; para $0 \leq \alpha < 1$, o objeto "encolhe" nesta direção. Valores negativos de α nos fornece a *reflexão* sobre um ponto fixo, na direção de escalamento, como podemos observar na Figura 4. Nesta figura o ponto fixo é a origem, $\alpha = -1$ e a direção de escalamento é $(0, 1, 0)$.

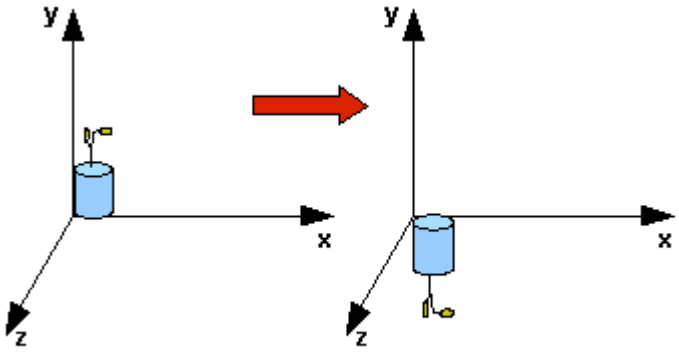

FIGURA 4. O escalamento de um objeto

Considerando o ponto fixo como sendo a origem, uma matriz de escalamento com um ponto fixo nessa posição permite o escalamento independente sobre os eixos

das coordenadas. Isso quer dizer que todo o ponto que tiverem sobre a origem, permanecerá nesta posição após a escala e os pontos que estão fora, irão sofrer um deslocamento. As três equações são

$$x' = \beta_x x,\ y' = \beta_y y,\ z' = \beta_z z.$$

Estas três equações podem ser combinadas na forma homogênea como $\mathbf{P'} = \mathbf{SP}$, onde

$$\mathbf{S} = \mathbf{S}(\beta_x, \beta_y, \beta_z) = \begin{bmatrix} \beta_x & 0 & 0 & 0 \\ 0 & \beta_y & 0 & 0 \\ 0 & 0 & \beta_z & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}.$$

Note que aqui, como para toda a transformação linear expressa em coordenadas homogêneas, a última linha da matriz força o quarto componente do ponto transformado a reter o seu valor.

Obter-se-á o inverso da matriz de escala aplicando os recíprocos dos fatores de escala:

$$\mathbf{S}^{-1}(\beta_x, \beta_y, \beta_z) = \mathbf{S}\left(\frac{1}{\beta_x}, \frac{1}{\beta_y}, \frac{1}{\beta_z}\right).$$

Para o escalamento de um objeto em um ponto $\mathbf{P'} = (x', y', z')$ arbitrário, é necessário utilizar a transformação na seguinte ordem:

(1) Translada-se o ponto $\mathbf{P'}$ para o ponto de origem do marco;
(2) Faz-se o escalamento do objeto em relação à origem e
(3) Translada-se o objeto escalonado da origem para P'.

3. Projeção

Após a criação de cenas e objetos tridimensionais o próximo passo é efetuar a sua apresentação. Conforme mencionado anteriormente, existe o problema de apresentar uma entidade tridimensional bidimensionalmente. Esse processo de transformação de objetos tridimensionais em um *plano de projeção* bidimensional é denominado *projeção*.

Esse processo tem sido tratado exaustivamente por desenhistas, artistas, arquitetos, que buscaram técnicas e artifícios para sistematizar e solucionar este problema. Pode-se dizer que o olho do observador coloca-se no *centro de projeção*, o plano, que deve conter o objeto ou cena projetada, é chamado de plano de projeção. Os segmentos de reta que saem do centro de projeção e atingem o objeto projetado no plano de projeção são chamadas de *projetantes* (ou *projetores*), a projeção de uma imagem pode ser vista na Figura 5.

A classe das projeções é conhecida como *projeção geométrica planar*, uma vez que a projeção é feita sobre um plano e não sobre uma superfície curva. Projeções geométricas planares, também referidas simplesmente como projeções, podem ser divididas em duas classes básicas: *perspectiva* e *paralela*. A diferença entre essas

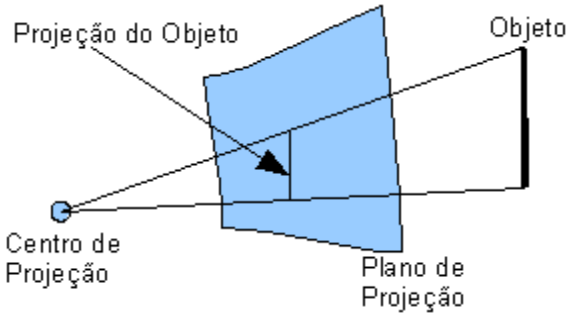

FIGURA 5. Projeção de uma imagem

duas classes está na relação entre o centro de projeção e o plano de projeção. Se a distância entre eles for finita então a projeção é perspectiva, entretanto, se essa distância for infinita então a projeção é paralela. As Figuras 6 e 7 ilustram estes dois casos. Quando definimos uma projeção perspectiva, o centro de projeção é explicitado, enquanto que na projeção paralela deve-se informar apenas a direção da projeção.

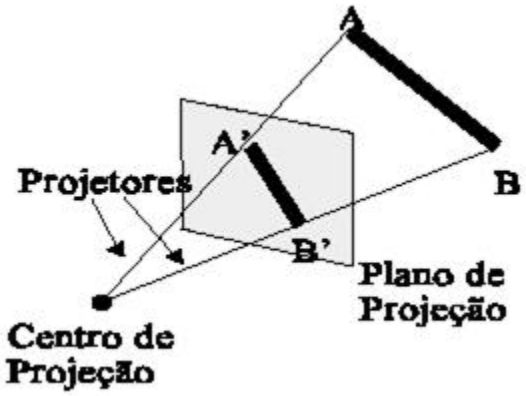

FIGURA 6. A projeção perspectiva

As projeções perspectiva e paralela podem ser definidas através de matrizes 4×4, o que é interessante para a composição de transformações juntamente com a projeção. A seguir, vamos detalhar cada uma destas projeções e suas matrizes.

3.1. **Projeção Perspectiva.** A projeção em perspectiva cria um efeito visual similar ao de sistemas fotográficos e ao da visualização humana. É utilizado quando se deseja um certo grau de realismo, uma vez que é possível se ter a percepção tridimensional da cena [FD84].

Os desenhos em perspectiva são caracterizados pelo *encurtamento perspectivo* e pelos *pontos de fuga*. O encurtamento perspectivo é a ilusão de que os objetos e comprimentos são cada vez menores à medida que sua distância ao centro de

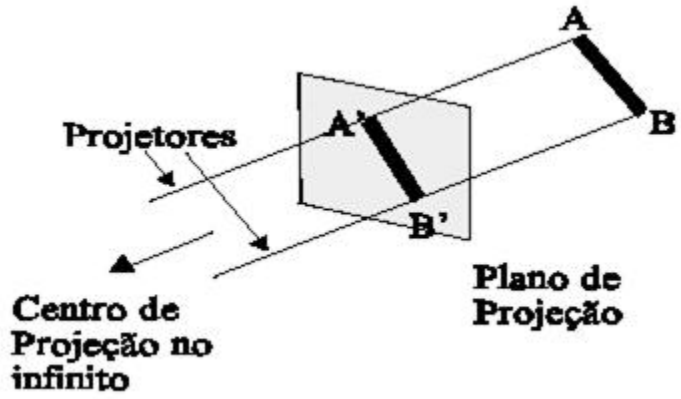

FIGURA 7. A projeção paralela

projeção aumenta. A projeção em perspectiva de qualquer conjunto de linhas que não são paralelas ao plano de projeção convergem para um ponto de fuga.

Projeções perspectivas são categorizadas pelo seu número de pontos de fuga principais, ou seja, o número de eixos que o plano de projeção intercepta. A Figura 8 mostra uma projeção perspectiva de um cubo vista por dois ângulos diferentes. Observe que esta projeção possui apenas um ponto de fuga, pois somente as linhas paralelas ao eixo z converge, e as linhas paralelas aos eixos x e y continuam paralelas.

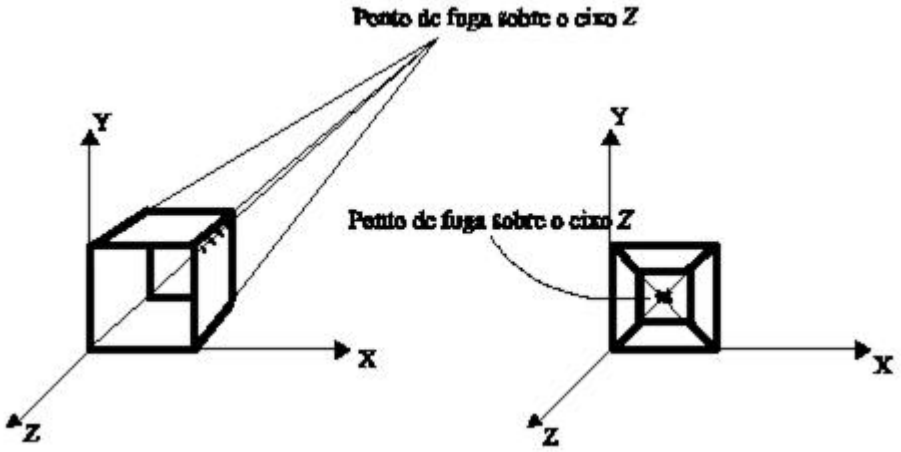

FIGURA 8. Projeções de um cubo com um ponto de fuga

Projeções perspectivas com dois pontos de fuga (quando dois eixos principais são interceptados pelo plano de projeção) são mais comumente usadas em arquitetura, engenharia, desenho publicitário e projeto industrial. Já as projeções perspectivas com três pontos de fuga são bem menos utilizadas, pois adicionam muito pouco em

termos de realismo comparativamente às projeções com dois pontos de fuga, e são mais difíceis de serem construídas.

Vamos supor agora que, no caso de projeção perspectiva, o plano de projeção é normal ao eixo z, em $z = d$. Assim, seja o plano de projeção que se encontra a uma distância d da origem, e $\mathbf{P}$ o ponto que será projetado sobre ele. Iremos calcular o ponto $\mathbf{P}_p = (x_p, y_p, z_p)$, que é a projeção perspectiva de $\mathbf{P} = (x, y, z)$ sobre o plano de projeção em $z = d$.

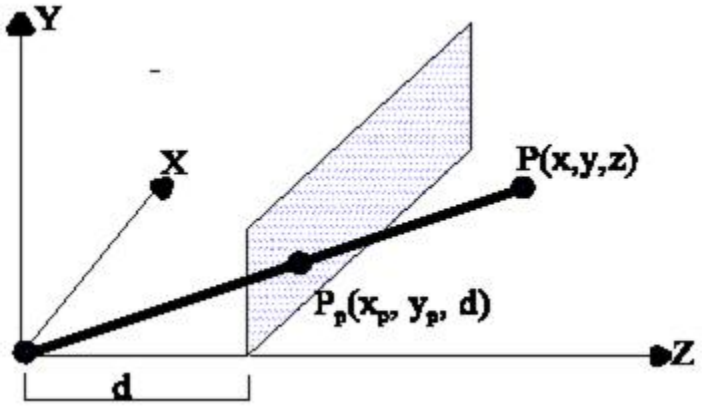

FIGURA 9. Cálculo da projeção perspectiva

Usando a semelhança de triângulos da Figura 9, temos:

$$\frac{x_p}{d} = \frac{x}{z}, \; x_p = \frac{dx}{z} = \frac{x}{z/d} \tag{3.1}$$

e

$$\frac{y_p}{d} = \frac{y}{z}, \; y_p = \frac{dy}{z} = \frac{y}{z/d} \tag{3.2}$$

A distância d pode ser vista como o fator de escala que se aplica a x_p e y_p. Na projeção perspectiva, a divisão por z faz com que os objetos mais distantes do plano de projeção pareçam menores do que os que estão mais próximos. Além disso, z pode assumir qualquer valor (com exceção de $z = 0$). Consideremos agora a seguinte matriz, que é a matriz da projeção perspectiva:

$$M = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 1/d & 0 \end{bmatrix}.$$

Multiplicando a matriz M pelo ponto **P** em coordenadas homogêneas, teremos um ponto $\mathbf{Q} = [x\ y\ z\ z/d]^T$:

$$\mathbf{Q} = \begin{bmatrix} x \\ y \\ z \\ z/d \end{bmatrix} = MP = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 1/d & 0 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}.$$

Dividindo **Q** por z/d iremos obter a representação equivalente do ponto **Q**:

$$\mathbf{Q}' = \begin{bmatrix} \frac{x}{z/d} \\ \frac{y}{z/d} \\ d \\ 1 \end{bmatrix} = \begin{bmatrix} x_p \\ y_p \\ z_p \\ 1 \end{bmatrix}.$$

Então, para obter as coordenadas cartesianas, temos:

$$(x_p, y_p, z_p) = \left(\frac{x}{z/d}, \frac{y}{z/d}, d \right)$$

Vemos que a formulação deriva diretamente do resultado das equações 3.1 e 3.2, e a coordenada $z_p = d$ resulta da posição do plano de projeção sobre o eixo z (normal ao eixo z).

3.2. **Projeção Paralela.** A projeção paralela é uma visualização menos realística comparada à projeção perspectiva, e a razão disto é que a projeção paralela não possui encurtamento perspectivo. No entanto, assim como na perspectiva, os ângulos são preservados somente nas faces do objeto que são paralelas ao plano de projeção.

As projeções paralelas são categorizadas em dois tipos, baseado na relação da direção de projeção e na *normal*[2] do plano de projeção. Nas projeções paralelas *ortográficas*, estas direções são as mesmas, enquanto que nas projeções paralelas *oblíquas* não são. Ou seja, na projeção ortográfica a direção da projeção é normal ao plano de projeção, enquanto que na projeção oblíqua, as projetantes são inclinadas em relação ao plano de projeção formando um ângulo α.

Para a projeção paralela, suponhamos que o plano de projeção é $z = 0$. Na projeção ortográfica, como a direção da projeção é a direção da normal ao plano de projeção, ela será o eixo z. Assim, o ponto **P** se projeta como

$$x_p = x,\ y_p = y,\ z_p = 0.$$

Sua projeção é expressa pela matriz:

[2]A normal é o vetor que é perpendicular a um determinado plano

$$M = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}.$$

Escreve-se este resultado, em coordenadas homogêneas, como:

$$\begin{bmatrix} x_p \\ y_p \\ z_p \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$$

Consideremos agora a projeção oblíqua. Na Figura 10 o cubo unitário é projetado no plano xy. O ponto $\mathbf{P}(0,0,1)$, que é um ponto atrás do cubo, é projetado no plano xy, como $\mathbf{P}'$.

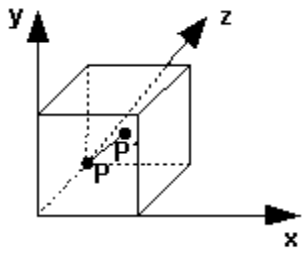

FIGURA 10. Projeção oblíqua de um cubo ($\mathbf{P}'$ é a projeção de $\mathbf{P}(0,0,1)$)

A reta projetante, que não é perpendicular ao plano de projeção, deve passar por $\mathbf{P}$ e P', conforme podemos observar na Figura 11. A direção desta reta projetante é dada por $\mathbf{P}' - \mathbf{P} = (l\cos\alpha, l\sin\alpha, -1)$ que faz um ângulo β com o plano xy.

Consideremos agora um ponto genérico (x, y, z) e a sua projeção oblíqua dada por $(x_p, y_p, 0)$. As equações para os valores de projeção de x e y em função de z são obtidas por semelhança de triângulos como descrito na Figura 12.

$$\frac{x - x_p}{l\cos\alpha} = \frac{y - y_p}{l\sin\alpha} = \frac{z}{-1}.$$

Desta relação, temos

$$x_p = x + zl\cos\alpha$$

e

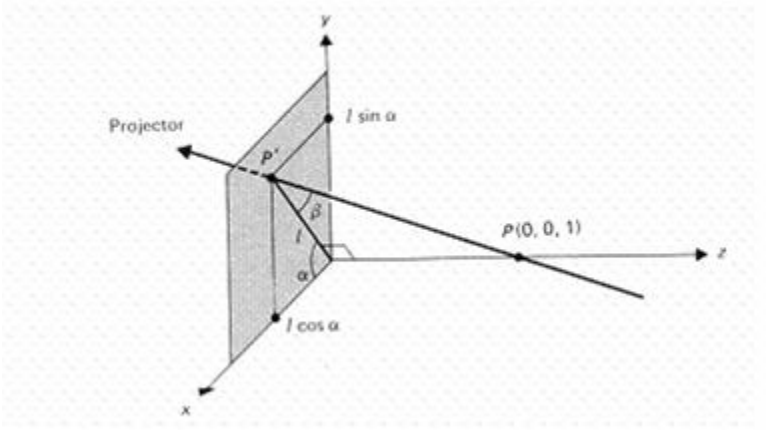

FIGURA 11. Projeção oblíqua de $\mathbf{P}(0,0,1)$ em $\mathbf{P}'(l\cos\alpha, l\sin\alpha, 0))$

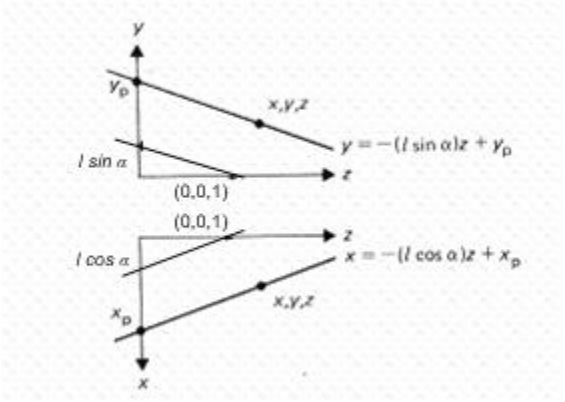

FIGURA 12. Projeção oblíqua de (x, y, z) em $(x_p, y_p, 0)$

$$y_p = y + zl\sin\alpha.$$

Chegamos, assim, à matriz de projeção oblíqua

$$M = \begin{bmatrix} 1 & 0 & l\cos\alpha & 0 \\ 0 & 1 & l\sin\alpha & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}.$$

4. Iluminação

A iluminação é um recurso gráfico essencial para produzir a sensação de tridimensão na visualização bidimensional da cena. O princípio da iluminação consiste em simular como os objetos refletem as luzes que incidem em sua superfície[MWS99].

Durante a interação da luz com um objeto, parte da energia é absorvida, parte é transmitida e parte é refletida na superfície do objeto. A componente refletida da energia luminosa incidente é que é a responsável pela sensação de cor produzida no cérebro de um ser humano. Quando a componente refletida perde a energia de forma aproximadamente igual em todas as frequências do espectro visível tem-se a cor cinza. Quando quase toda a energia é absorvida, tem-se a cor preta e quando quase toda a energia é refletida tem-se o branco.[**?**].

A quantidade de luz refletida depende da:

- Composição, direção e geometria da fonte de luz;
- Orientação da superfície do objeto em relação à fonte de luz;
- Propriedades da superfície do objeto.

Uma fonte de luz pode ser descrita de acordo com a função de iluminação

$$\mathbf{i} = \begin{bmatrix} i_r \\ i_g \\ i_b \end{bmatrix}$$

onde i_r, i_g e i_b são os componentes de intensidade das cores vermelha, verde e azul respectivamente. As fontes de luz também podem ser classificadas em:

- **Luz Ambiente**: geralmente oriunda de fontes largas as quais dissipam luz em todas as direções, possibilitando uma iluminação uniforme da cena. A iluminação ambiente é caracterizada pela intensidade

$$\mathbf{i}_a = \begin{bmatrix} i_{ar} \\ i_{ag} \\ i_{ab} \end{bmatrix}$$

  a qual, embora idêntica para todos os pontos da cena, cada superfície pode refletir esta luz diferentemente. As componentes de i_a representam as intensidades das freqüências vermelha, verde e azul da luz ambiente.

- **Fonte Pontual**: emite luz igualmente em todas as direções. Uma fonte pontual localizada em um ponto $\mathbf{P}_0$ pode ser escrita como

$$\mathbf{i}(\mathbf{P}_0) = \begin{bmatrix} i_r(\mathbf{P}_0) \\ i_g(\mathbf{P}_0) \\ i_b(\mathbf{P}_0) \end{bmatrix}$$

  onde $i(\mathbf{P}_0)$ denota qualquer um dos componentes de $\mathbf{i}(\mathbf{P}_0)$. A intensidade proveniente de uma fonte pontual é proporcional ao inverso do quadrado da distância entre a fonte e a superfície, ou seja, em um ponto $\mathbf{P}$ da superfície a intensidade de luz recebida de uma fonte pontual $\mathbf{P}_0$ é dada por:

$$i(\mathbf{P}, \mathbf{P}_0) = \frac{1}{|\mathbf{P} - \mathbf{P}_0|^2}\mathbf{i}(\mathbf{P}_0).$$

- **Spot de Luz**: é caracterizado por um feixe de luz emitido, como um cone de origem $\mathbf{P}_s$ apontando na direção l_s e cuja abertura é dada por um ângulo θ. Assim, a intensidade proveniente desta fonte em um ponto $\mathbf{P}$ de uma superfície é uma função do ângulo ϕ entre a direção da fonte (l_s) e a direção da reta que, partindo de $\mathbf{P}_s$ encontra a superfície em $\mathbf{P}$. Esta função é, usualmente, definida por $\cos^e \phi$ para $\phi \in (-\theta, \theta)$, onde e é uma constante maior que zero que determina o quão rapidamente a intensidade da luz decresce com $|\phi|$.
- **Fonte de Luz Distante**: neste caso, a fonte de luz é caracterizada por uma direção e uma intensidade constantes na cena. Assim, o ângulo de incidência desta luz nas superfícies planas é constante.

A criação de imagens realistas envolve, obrigatoriamente, a perfeita compreensão da iteração da luz com a matéria e a utilização de modelos que, de alguma forma, simulem esta iteração como por exemplo a aparência metálica de um objeto. Para implementar um modelo de iluminação é necessário o cálculo do vetor de reflexão. Este cálculo é bastante simples e pode ser obtido através da relação da Figura 13 que é dada por:

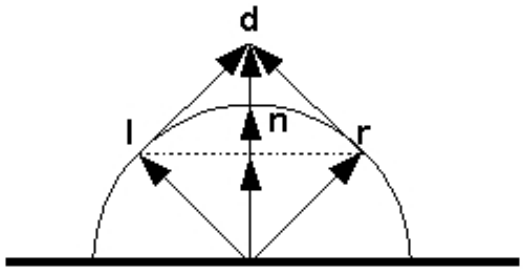

FIGURA 13. Cálculo do Vetor Reflexão

$$||\mathbf{l}|| = ||\mathbf{r}|| = ||\mathbf{n}|| = 1$$

$$\mathbf{d} = 2 < \mathbf{l}, \mathbf{n} > \mathbf{n}$$

$$cos(\alpha) =< \mathbf{l}, \mathbf{n} >=< \mathbf{r}, \mathbf{n} >$$

Logo temos que:

$$\mathbf{r} = 2 < \mathbf{l}, \mathbf{n} > \mathbf{n} - \mathbf{l}$$

Onde $\mathbf{n}$ é o vetor normal à superficie no ponto e $\mathbf{l}$ é o vetor que indica a direção da fonte de luz.

Existem diversos modelos de iluminação. Podemos citar o Modelo de Bouknight que foi um dos primeiros modelos de iluminação, e o modelo de Phong, o mais utilizado atualmente na computação gráfica.

4.1. **Modelo de Bouknight.** Bouknight introduziu um dos primeiros modelos de iluminação local de superfície. [Bou70] Ele considerou que a intensidade lumionosa em um ponto $\mathbf{P}$ é o resultado da soma das contribuições das intensidades luminsosa difusa i_d e ambiental i_a.

$$i(\mathbf{P}) = i_a(\mathbf{P}) + i_d(\mathbf{P})$$

A intensidade luminosa do ambiente é aquela que provêm de reflexões da luz em todas as direções. A reflexão ambiente é o resultado das reflexões indiretas nos objetos, que resultam em uma iluminação uniforme de todos os pontos da cena. A contribuição desta energia para a intensidade luminosa num ponto $\mathbf{P}$ é dada por:

$$i_a(\mathbf{P}) = l_a k_a(\mathbf{P})$$

Onde $l_a(\mathbf{P})$ é a intensidade da luz ambiente e $k_a(\mathbf{P})$ o coeficiente de reflexidade ambiente do ponto $\mathbf{P}$ na superfície, sendo $0 \leq k_a \leq 1$.

A luminosidade Difusa é provocada pela chamada reflexão *lambertiana*, esta é avaliada baseado no modelo de reflexão de Lambert. Neste modelo a intensidade da luz refletida é proporcional ao cosseno do anglo entre a direção da incidência da luz e a normal da superfície, como mostra a Figura 14.

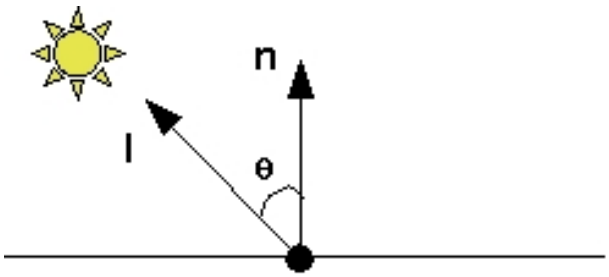

FIGURA 14. Lei de Lambert

Na Reflexão Difusa, a luz que vem de uma direção, atinge a superfície e é refletida em todas as direções; assim, parece possuir o mesmo brilho independente de onde a câmera esteja posicionada. É o caso das reflexões de objetos lisos foscos. A luminosidade aparente da superfície não depende do observador, mas sim do cosseno do ângulo de incidência da luz como mostra a Figura 15.

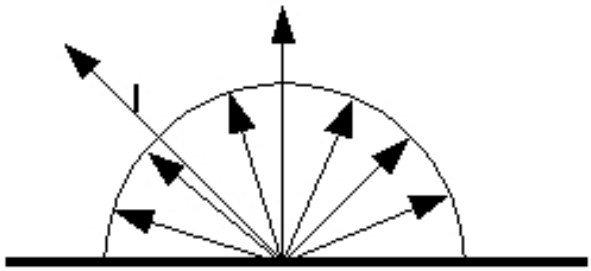

FIGURA 15. Reflexão Difusa

A contribuição dessa energia luminosa é dada por

$$i_d(\mathbf{P}) = \frac{k_d(\mathbf{P})}{a + bd + cd^2}(\mathbf{l} \cdot \mathbf{n})\mathbf{l}_d$$

Onde $k_d(\mathbf{P})$ representa a fração da luz difusa refletida, l_d a intensidade da luz difusa incidente em cada um dos pontos da superfície, $\mathbf{l}$ o vetor direção da fonte de luz normalizado, $\mathbf{n}$ o vetor normal à superfície, no ponto de interesse, normalizado e o termo quadrático expresso à atenuação da energia luminosa com a distância d entre a fonte de luz e o ponto $\mathbf{P}$.

4.2. **Modelo de Phong.** Este modelo é uma simplificação da equação integral que descreve a distribuição de energia luminosa nas superfícies de uma cena. O modelo de *Phong* usa quatro vetores para calcular a energia luminosa em cada freqüência (vermelha, verde e azul) para um ponto arbitrário $\mathbf{P}$ da superfície. Os vetores são: $\mathbf{n}$ (vetor normal à superfície em $\mathbf{P}$), $\mathbf{v}$ (vetor direção da reta com origem em $\mathbf{P}$ e passando pelo observador), $\mathbf{l}$ (vetor direção da reta com origem em $\mathbf{P}$ e na direção da fonte de luz) e $\mathbf{r}$ (vetor direção de reflexão de $\mathbf{l}$) como mostra a Figura16. Assume-se também que cada superfície é composta de um material com várias propriedades. O material pode emitir luz, refletir parte da luz incidente em todas as direções, ou refletir uma parte da luz incidente numa única direção, tal como um espelho.

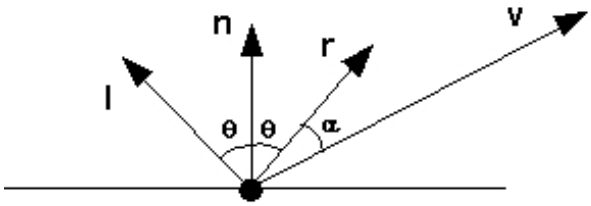

FIGURA 16. Modelo de Phong

Phong além de levar em consideração as outras componentes definidas pelo modelo Bouknight, definiu a componente de Reflexão Especular

Na Reflexão Especular, a luz vem de uma direção e tende a ser refletida numa única direção, que é o caso das reflexões de objetos lisos brilhantes (metálicos ou polidos). Esta reflexão depende da disposição entre observador, objeto e fonte de luz. Em um espelho perfeito, a reflexão se dá em ângulos iguais e o observador só enxergaria a reflexão de uma fonte pontual se estivesse na posição certa. Simula-se refletores imperfeitos assumindo que a luz é refletida segundo um cone cujo eixo passa pelo observador como mostra a Figura 17.

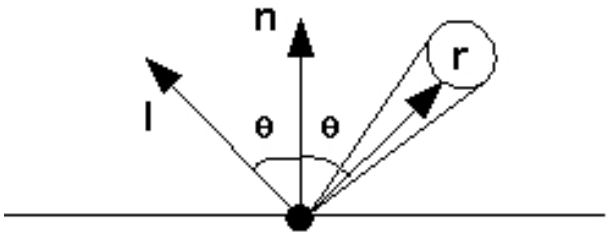

FIGURA 17. Reflexão Especular

A contribuição de Phong nessa componente é dada por:

$$i_s(\mathbf{P}) = k_s(\mathbf{P}) l_s (\mathbf{r} \cdot \mathbf{v})^{\alpha}$$

Onde $k_s(\mathbf{P})$ é o coeficiente de reflectividade especular, l_s a intensidade da fonte de luz, $\mathbf{r}$ o vetor direção de um refletor perfeito normalizado, $\mathbf{v}$ o vetor direção do observador normalizado e α o *coeficiente de especularidade*, que indica quão polida é a superfície. O espelho ideal tem o coeficiente de especularidade igual a infinito. Na prática, usam-se valores entre 5 e 100.

Obtêm-se, finalmente, o modelo de *Phong* para o cálculo da intensidade luminosa de um ponto visível da cena, considerando os componentes de luz ambiente, difusa e especular:

$$i = i_a + i_d + i_s = \frac{1}{a + bd + cd^2}(k_d l_d \mathbf{l} \cdot \mathbf{n} + k_s l_s (\mathbf{r} \cdot \mathbf{v})^{\alpha}) + k_a l_a$$

Este modelo assume que cada fonte de luz na cena pode ter as componentes difusa, ambiente e especular separadamente, para cada uma das três cores primárias (vermelho, verde e azul). Quando existem m fontes de luz na cena as regiões que estão na sombra não podem ser determinadas a partir do modelo de iluminação, obrigando que sejam executados algoritmos para a determinação explícita das regiões sombrias da cena.

5. OpenGL

OpenGL é uma biblioteca de rotinas gráficas de modelagem, manipulação de objetos e exibição tridimensional que permite a criação de aplicações que usam computação gráfica. Seus recursos permitem ao usuário criar objetos gráficos com qualidade, de modo rápido, além de incluir recursos avançados de animação, tratamento de imagens e texturas, onde é possível ter visualização em vários ângulos.

O OpenGL também pode ser definido como uma interface de software para dispositivos de hardware. Esta interface consiste em cerca de 150 comandos distintos usados para especificar os objetos e operações necessárias para produzir aplicativos tridimensionais interativos. O OpenGL foi desenvolvido independente de interface de hardware para ser implementado em múltiplas plataformas.

A biblioteca OpenGL foi introduzida em 1992 pela *Silicon Graphics*, no intuito de conceber uma API (Interface de Programação de Aplicação) gráfica independente de dispositivos de exibição. Com isto, seria estabelecida uma ponte entre o processo de modelagem geométrica de objetos, situadas em um nível de abstração mais elevado, e as rotinas de exibição e de processamento de imagens implementadas em dispositivos (hardware) e sistemas operacionais específicos. A função utilizada pelo OpenGL para desenhar um ponto na tela, por exemplo, possui o mesmo nome e parâmetros em todos os sistemas operacionais nos quais OpenGL foi implementada, e produz o mesmo efeito de exibição em cada um destes sistemas.

Diante das funcionalidades providas pelo OpenGL, tal biblioteca tem se tornado um padrão amplamente utilizado na indústria de desenvolvimento de aplicações. Este fato tem sido adotado também pela facilidade de aprendizado, pela estabilidade das rotinas, pela boa documentação disponível e pelos resultados visuais consistentes para qualquer sistema de exibição concordante com este padrão. Diversos jogos, aplicações científicas e comerciais tem utilizado OpenGL como ferramenta de apresentação de recursos visuais, principalmente com a adoção deste padrão por parte dos fabricantes de placas de vídeo destinadas aos consumidores domésticos.

As especificações do OpenGL não descrevem as interações entre OpenGL e o sistema de janelas utilizado (MS Windows, X Window etc). Assim, tarefas comuns em uma aplicação, tais como criar janelas gráficas, gerenciar eventos provenientes de mouse e teclado, e apresentação de menus ficam a cargo de bibliotecas próprias de cada sistema operacional. Neste trabalho será utilizada a biblioteca GLUT (OpenGL ToolKit) para gerenciamento de janelas.

Desde sua introdução, OpenGL transformou-se num padrão extensamente utilizado pelas indústrias. OpenGL promove a inovação e acelera o desenvolvimento de aplicações incorporando um grande conjunto de funções. Entre os recursos gráficos disponíveis pelo OpenGL, temos os modos de desenho de pontos, ajuste de largura de linhas, aplicação de transparência, atenuação de serrilhamento (anti-aliasing), mapeamento de superfícies com textura, seleção de janela de desenho, manipulação de fontes/tipos de iluminação e sombreamento, transformação de sistemas de coordenadas e transformações em perspectiva e combinação de imagens (blending).

5.1. **Transformações e Perspectiva em OpenGL.** Sistemas gráficos como o OpenGL trabalham com o conceito de matriz corrente para tratar as transformações. Desta forma, apenas uma matriz fica armazenada no sistema gráfico para esta função. Todos os vértices das primitivas que estão sendo definidas são transformados por ela. O sistema gráfico fornece funções para iniciar e alterar esta matriz. No OpenGL esta matriz é chamada de *matriz de modelagem e visualização (model view)* e as transformações são acumuladas à direita. Ou seja, ao fornecermos ao sistema uma nova matriz $\mathbf{M}$, ela é multiplicada pela esquerda pela matriz corrente $\mathbf{C}$ e a nova matriz corrente assume o valor $\mathbf{CM}$. Geometricamente isto significa que a transformação $\mathbf{M}$ ocorre antes das transformações acumuladas em $\mathbf{C}$.

O sistema OpenGL implementa um mecanismo de *pilha* para a matriz de transformação. Isto porque a idéia central das transformações em OpenGL é que elas

são cumulativas, ou seja, podem ser aplicadas umas sobre as outras. A cada transformação a matriz de transformação é alterada e usada para desenhar os objeto a partir daquele momento, até que seja novamente alterada. A pilha é uma estrutura muito utilizada em computação [SM94]. Ela é denominada por **LIFO** (*Last In First Out*), ou seja, o último elemento que entra é o primeiro que sai da pilha. Com ela o programador pode recuperar matrizes através de mecanismos de *push* (inserção na pilha) e *pop* (remoção da pilha). Com o mecanismo de *push* e *pop* na pilha podemos garantir que a função retorna sem alterar o estado corrente das transformações, ou seja, sem efeitos colaterais indesejados.

Já para as projeções basta definirmos uma matriz 4×4, apresentada na seção anterior, a qual deveremos aplicar depois da matriz de modelagem e visualização. Entretanto, no caso das projeções perspectivas, deve ser feita uma *divisão perspectiva* no final (que é a divisão pelo fator **w** apresentado na subseção referente à projeção perspectiva). Nas projeções ortogonais essa divisão não é necessária.

5.2. **Remoção de Superfícies Ocultas.** Dado um objeto tridimensional e uma especificação de visualização definindo o tipo de projeção, o plano de projeção, etc, deseja-se agora determinar quais as arestas e superfícies do objeto que são visíveis a partir do centro de projeção (para projeções em perspectiva) ou ao longo de uma direção de projeção (para projeções paralelas), de forma que apenas serão exibidas as arestas e as superfícies visíveis. Existem dois casos possíveis quando uma face (ou aresta) está oculta; está oculta pelo próprio objeto ou por outros objetos. Apesar da idéia fundamental ser relativamente simples, sua implementação é custosa, o que encorajou a construção cuidadosa de numerosos algoritmos [FD84]. Não se pode afirmar que uma técnica é melhor do que a outra pois irá depender da aplicação (complexidade da cena, tipos de objetos, equipamento disponível, entre outros). Alguns algoritmos fornecem soluções mais rápidas, outros fornecem soluções mais lentas mas que, em compensação, possuem mais realismo e detalhes (incluem sombras, transparência, etc).

O *OpenGL* possui um depth buffer que trabalha através da associação de uma profundidade, ou distância, do plano de visualização (geralmente o plano de corte mais próximo do observador) com cada pixel da window. Inicialmente, os valores de profundidade são especificados para serem o maior possível. Entretanto, antes de cada pixel ser desenhado pode ser feita feita uma comparação com o valor de profundidade já armazenado através de funções da propria OpenGL. Se o valor de profundidade for menor (está mais próximo) o pixel é desenhado e o valor de profundidade é atualizado. Caso contrário as informações do pixel são desprezadas.

Para remover faces ocultas de um mesmo objeto, a OpenGL fornece a opção de desenhar apenas as faces visíveis em relação ao observador. Para situações em que não é posível ver algumas faces do objeto, é pode-se reduzir o trabalho requerido em se remover faces ocultas apenas eliminando todas as faces invisíveis deste mesmo objeto (*back-face culling*) antes de ser aplicado qualquer outro algoritmo de remoção de superfícies ocultas.

Para se encontrar a face oculta de um polígono, basta analisar a normal de cada uma de suas faces. A face será frontal se a normal estiver apontando para o observador, como mostra a Figura 18. Considerando θ o ângulo entre a normal e o observador, então o polígono estará com a face visível se e somente se $-90^0 < \theta < 90^0$ ou, equivalentemente, $\cos\theta > 0$. Por causa desta segunda condição, ao invés

de calcular o cosseno, pode-se utilizar o *produto interno*, ou seja, $\mathbf{n}.\mathbf{v} > 0$, onde $\mathbf{n}$ é a normal da face a ser analisada e $\mathbf{v}$ é o vetor de visão do observador.

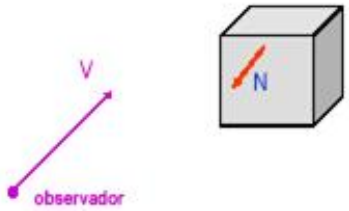

FIGURA 18. A visão do observador e a normal da face

Referências

[Bou70] W.J. Bouknight. A procedure for generation of three-dimensional half-tone computer graphics presentations. Technical report, Comunications of the ACM, 1970.

[FD84] J. D. Foley and A. Van Dam. *Fundamentals of Interactive Computer Graphics*. Addison Wesley, USA, 1st edition, 1984.

[MdBS98] M. Overmars M. de Berg, M. van Kreveld and O. Schwarzkopf. *Computational Geometry - Algorithms and Applications*. Springer, New York, USA, 1998.

[MWS99] T. Davis M. Woo, J. Neider and D. Shreiner. *OpenGL Programming Guide*. Addison Wesley, USA, 1999.

[SM94] J.L. Szwarcfiter and L. Markenzon. *Estruturas de Dados e Seus Algoritmos*. LTC Editora, Rio de Janeiro, Brasil, 1994.

(A. One) Institute of Mathematics, Federal University of Rio de Janeiro
P.O. Box 2324, Rio de Janeiro, RJ. 20001-970 BRAZIL
E-mail address, A. One: lopes@cos.ufrj.br
URL: http://www.nce.ufrj.br/controlab

(A. Two, A. Three and A. Four) Núcleo de Computação e Eletrônica - NCE, Federal University of Rio de Janeiro
P.O. Box 2324, Rio de Janeiro, RJ. 20001-970 BRAZIL
URL: http://www.nce.ufrj.br/controlab

E-mail address, A.Two: marcelasms@nce.ufrj.br
E-mail address, A.Three: rvcarvalho@ufrj.br
E-mail address, A.Four: elaude@nce.ufrj.br