INSTITUTO FEDERAL DE EDUCAÇÃO, CIÊNCIA E TECNOLOGIA DO CEARÁ – CAMPOS FORTALEZA
CURSO DE ENGENHARIA DA COMPUTAÇÃO

ANDRÉ VIEIRA DA SILVA

Relatório sobre algoritmo Raytrace implementação em linguagem python.

FORTALEZA,2018

ANDRE VIEIRA DA SILVA

Relatório sobre algoritmo Raytrace implementação em linguagem python.

Relatório técnico apresentado como requisito para aprovação na disciplina de Computação Gráfica.

Prof. Lucas Silva.

FORTALEZA,2018

RESUMO

Este trabalho apresenta as características exigíveis para a apresentação de um relatório técnico-científico, no que tange a abordagem acerca da implementação do algoritmo de renderização de objetos Raytrace contando com aspectos teóricos pertinentes à elaboração de todas as etapas pertinentes como as etapas de renderização de objetos, iluminação e sombreamentos, e transformações em dimensão 2D.

Palavras-chave: Raytrace, iluminação, sombreamento, transformações, renderização.

Sumário

1. INTRODUÇÃO

Dentro das vertentes da computação a simulação e a criação de ambientes simulados são propostas para diversas soluções desenvolvidas para incontáveis fins.

Criar elementos digitalizados que são representações com aproximação de alto nível de confiabilidade, e, por que não realidade, permitiu que os jogos, um exemplo de aplicação, evoluíssem ao longo dos anos de forma tão assombrosa em riqueza de detalhes que a imersão do usuário é uma experiência sensorial muita próxima da realidade.

Há casos específicos que o nível de imersão é tão elevado ,que o nosso cérebro, que interpreta o ambiente e interage com o mesmo, provoca ações dentro do ambiente simulado , semelhantes ao que faria em ambiente real.

Com a evolução de diversos aspectos computacionais outras aplicações foram se tornando possíveis, como é o caso das câmeras digitais presentes nas mãos que praticamente toda a população, e ainda com a capacidade de digitalização em tempo real as aplicações médicas de diagnostico visual.

Nisso nos segue o questionamento, como essas virtualizações do ambiente e principalmente dos elementos do ambiente são transformados ou construídos de forma tão eficiente hoje, como resposta a esta evolução temos diversos algoritmos de renderização, contudo para o escopo deste trabalho nos atentaremos aos conceitos do algoritmo RayTracing.

2. DESENVOLVIMENTO

2.1. Objetivo Geral

Entender como funciona o processo de renderização de elementos de uma dada cena e a relação entre eles através dos conhecimentos matemáticos aplicados.

2.2. Objetivo Específico

Desenvolver em linguagem python o algoritmo RayTracing obtendo como resultado objetos descritos matematicamente renderizados na dimensão 2D em imagens.

2.3. Metodologia

Serão abordados os aspectos teóricos pertinentes a etapa que segue bem como uma breve apresentação dos resultados esperados e logo imediatamente seguirão os resultados obtidos e quando for o caso simulações adicionais poderão ser admitidas afim de tornar evidenciado algum fator que seja relevante.

Como o processo apresenta etapas cujas considerações podem ser demasiado repetitivas, estas serão apenas citadas referencialmente.

3. RAYTRACE

O algoritmo Raytrace se baseia na simulação da refração ou reflexão provocada pelos objetos quando os raios de luz tocam os mesmos, em outras palavras entendendo a luz como uma composição vetorial de vários feixes de espectros da luz, podemos dizer que o algoritmo simula o desvio dos vetores resultantes da decomposição da luz pela presença dos objetos ,que são captados por um dado ponto de visão, ou seja ,esses vetores são desviados em infinitas direções de forma que havendo a presença de um elemento sensor, ponto de visão, há a capacidade de obter como resultado a percepção dos objetos através das transformações sofridas pelos raios de luz.

Assim o algoritmo tem seu foco nos raios que são desviados e projetados na direção do elemento sensor restringindo o poder computacional necessário, a somente os raios decompostos que atingem o necessariamente o objeto e o elemento sensor, ponto de visão.

Nesse comportamento apresentado pelo direcionamento dos raios desviados duas situações são extremamente distintas no que tange a forma como os raios são apresentados de forma que existem basicamente duas formas de projeção: a ortográfica e obliqua.

3.1. Projeção de Raios

Projetar raios, no contexto deste trabalho, é a atividade de produzir retas que partem de um determinado ponto em direção a outro de forma que podemos descrever essa reta como:

$$p(t) = e + t(s - e) \qquad 1.$$

Assumindo que e é um ponto qualquer no espaço cartesiano, tido como ponto de visão para o escopo deste trabalho , e s um ponto pertencente a um plano qualquer situado a uma distância t , podemos calcular o vetor que partindo do ponto e toca um determinado plano passando pelo ponto s pertencente a este plano.

Assim podemos obter um base ortonormal através da definição de três vetores perpendiculares entre si ,partindo de e na direção $\vec{w}$ de forma que:

$$\vec{w} = \frac{e}{|e|} \qquad 2.$$

De posse de $\vec{w}$ podemos estimar mais dois vetores $\vec{v}, \vec{u}$ perpendiculares entre si de forma que para o obtenção de $\vec{v}$ seguiremos a estratégia de estimar um vetor $\vec{t}$,que receberá o valor do vetor $\vec{w}$ tomando o cuidado de substituir o menor valor absoluto de coordenada por 1 assim poderemos aplicar a seguinte formula e obter $\vec{v}$.

$$\vec{v} = \frac{\vec{w} \times \vec{t}}{\left|\vec{w} \times \vec{t}\right|} \qquad 3.$$

E por último podemos obter o vetor $\vec{u}$ reaplicando a equação 3 entre os vetores $\vec{w}$ e $\vec{v}$ obtendo os vetores que definem um plano ortonormal a partir do ponto de visão .

$$\vec{u} = \frac{\vec{w} \times \vec{v}}{|\vec{w} \times \vec{v}|} \qquad 4.$$

Com os vetores $\vec{w}, \vec{v}$ e $\vec{u}$ obtidos podemos alcançar qualquer ponto no espaço ortogonal obtido como um combinação linear dos vetores $\vec{w}, \vec{v}$ e $\vec{u}$ então se p é um ponto pertencente a este citado espaço podemos escrever este ponto p como:

$$p(t) = d * \vec{w} + \mathrm{v}' * \vec{v} + \mathrm{u}' * \vec{u} \quad 5.$$

Os escalares d, v' e u' presentes na equação 5 são respectivamente a distância entre o ponto de visão e o plano obtido ,a quantidade de deslocamento na direção do vetor $\vec{v}$ e a quantidade de deslocamento na direção do vetor $\vec{u}$.

Os valores para v' e u' podem ser obtidos com adaptações pertinentes ao tamanho e a quantidade dos pixels presentes na imagem ,assim:

$$v' = l + (r - l) * \frac{(i + 0.5)}{Nx} \quad 6.$$

$$u' = b + (t - b) * \frac{(j + 0.5)}{Ny} \quad 7.$$

Na equações 6 e 7 os termos $l, r, b,$ e t são termos que definem a janela de enquadramento da área de visão sobre o plano observado, os elementos que estiverem dentro desta janela serão renderizados. O termos i e j fazem referência a notação matricial adotada para a representação computacional da imagem ,sendo os valores das posições nas colunas e linhas respectivamente. Os valores Nx e Ny são as dimensões largura e altura da imagem definidas em quantidades de pixels. E por último os valores 0.5, presentes em ambas as equações, merecem certa atenção pois são fatores que centralizam o raio ao centro do pixel melhorando a precisão da computação dos valores na matriz da imagem gerada.

Todo esse procedimento descrito permite encontrar qualquer ponto no espaço a partir simplesmente do ponto de visão e estimar a direção d do raio , podendo reescrever a equação 1 como:

$$f(t) = e + td \quad 8.$$

O Raytrace tem como base os aspectos elucidados neste tópico sendo requisito exaustivamente utilizado para a definição da direção e da origem dos raios gerados.

3.2. Projeção Ortográfica

Na projeção ortográfica os raios são projetados em direção ortogonal ao plano do objeto ,como conseqüência disso são preservadas as dimensões do objeto de forma fiel ao real e há perda referencial das noções de profundidades entre os objetos e o espaço.

De certa forma podemos dizer que cada raio projetado do plano do objeto corresponde a um raio em um plano paralelo numa relação um para um.

Nesse tipo de projeção a distância entre o objeto e o ponto de visão não afeta o resultado obtido sendo freqüentemente usada nas simulações realizadas em ferramentas de desenho técnico.

Como na projeção ortográfica há um plano perpendicular ao plano onde o ponto de visão está inserido, podemos ,de posse dessa informação estimar o plano onde o objeto está. Ainda sabemos que a direção dos raios é da forma um para um entre os planos, ponto a ponto. Assim

podemos traçar raios que partem de um plano a outro ,apenas sabendo as coordenadas do ponto de visão aplicando manipulação algébrica ,desta forma obtemos a direção e a origem dos raios como :

$$direção = -\vec{w} \quad 9.$$

$$origem = e + v' * \vec{v} + u' * \vec{u} \quad 10.$$

Abaixo teremos o resultado visual da geração raios na projeção ortográfico.

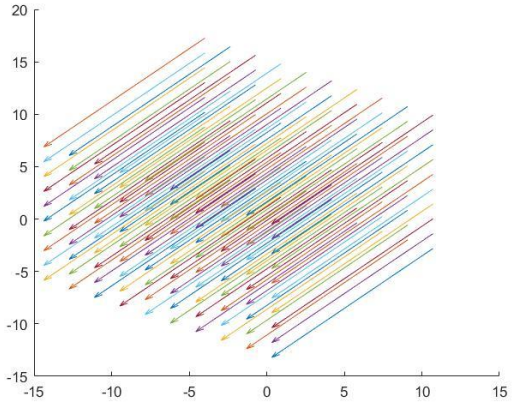

Figura 0-1 Resultado da geração de raio caso ortografico.

A imagem acima descreve os raios projetados a partir de um plano paralelo ao plano do ponto de visão situado nas coordenadas (5,5,5) com os parâmetros $Nx = 50$,$Ny = 50$, $l = -10$, $b = -10$,$r = 10$, $t = 10$.

3.3. Projeção Oblíqua

Na projeção obliqua os raios são projetados tendo como origem o ponto de visão, de forma que, partindo dele, os raios atinjam um plano ortogonal ao ponto de visão e em seguida um objeto que será computado de acordo com os pontos onde é interceptado pelos raios.

Neste tipo projeção os objetos renderizados sofrem variações nas suas dimensões em relação ao objeto real de acordo com a distância entre o ponto de visão e o objeto ,ainda também são mantidas as referências que permitem a identificação de profundidades em relação aos objetos e o ambiente em que estão inseridos.

Os raios são projetados com referência a um único ponto de visão *e* e da mesma forma que na projeção ortográfica podemos obter a direção e a sua origem ,assim:

$$direção = -d * \overrightarrow{w} + v' * \vec{v} + u' * \vec{u} \quad 11.$$

$$origem = e \quad 12.$$

Abaixo teremos o resultado visual da geração raios na projeção obliquo.

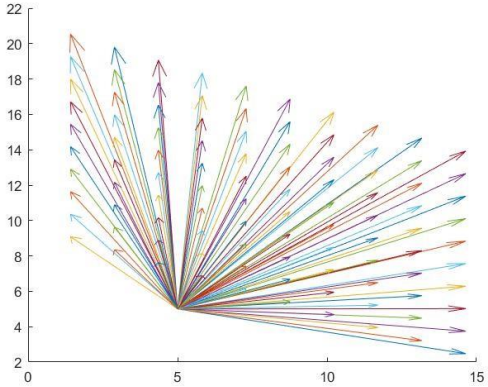

Figura 3.3-1Resultado da geração de raios caso obliquo.

A imagem acima descreve os raios projetados a partir de um plano paralelo ao plano do ponto de visão situado nas coordenadas $(5,5,5)$ com os parâmetros $Nx = 50$,$Ny = 50$, $l = -10$, $b = -10$,$r = 10$, $t = 10$.

4. RENDERIZAÇÃO DE OBJETOS

De posse dos conceitos de projeção ortográfico e obliquo pode-se estimar o plano aonde os objetos serão desenhados. Esse processo consiste em saber se os raios projetados atingem um dado objeto presente no espaço.

Os objetos desenhados podem ser representados por funções matemáticas que os descrevem, ou ainda, pelas coordenadas cartesianas dos pontos de interseção das suas arestas.

De posse das informações sobre o objeto ,seja função ou coordenadas dos vértices ,e do ponto de visão e é possível realizar a renderização dos elementos interceptados pelos raios gerados na projeção.

Matematicamente é obtido um modelo que recebendo o raio como parâmetro seja capaz de anular a função que descreve o objeto em caso positivo a cor do objeto é computado na matriz representante do plano, caso negativo é computado um valor predefinido.

$$f\big(p(t)\big) = f(e + t * d) = 0 \qquad 13.$$

No escopo deste trabalho nos deteremos a exemplificar a renderização de objetos clássicos como esferas, triângulos e polígonos.

4.1. Renderização de Esferas.

Esferas são elementos clássicos da geometria espacial, abstraindo sua representação matemática podemos escrever uma esfera c com coordenadas (Xc, Yc, Zc) implicitamente como :

$$(X - Xc)^2 + (Y - Yc)^2 + (Z - Zc)^2 - r^2 = 0 \qquad 14.$$

Já que (Xc, Yc, Zc) e (X, Y, Z) são coordenadas a equação 13 pode ser reescrita vetorialmente obtendo :

$$(p - c) \cdot (p - c) - r^2 = 0 \qquad 15.$$

Pelas equações 12 e 13 respectivamente:

$$f\big(p(t)\big) = (p - c) \cdot (p - c) - r^2 = 0 \qquad 16.$$

Aplicando como parâmetro em f a equação 8 chega-se:

$$f\big(p(t)\big) = (e + td - c) \cdot (e + td - c) - r^2 = 0 \qquad 17.$$

Desenvolvendo a equação 17 e organizando os termos.

$$f\big(p(t)\big) = (d \cdot d)t^2 + 2d(e - c)t + (e - c) \cdot (e - c) - r^2 = 0 \qquad 18.$$

Observando a equação 18 percebe-se que ela está escrita na forma quadrática e o único termo desconhecido é o termo t. Resumindo o processo, basta seguir os passos descritos abaixo para a verificação se um raio toca ou não a esfera.

1. Gere um raio obtendo sua direção (d) ;
2. Calcule o termo discriminante (Δ) da equação 18;

3. Se $\Delta < 0$,o raio não toca a esfera ;
4. Se $\Delta = 0$,raio toca a esfera em um único ponto;
    a. Calcule t;
    b. Marque o pixel referente ao raio com a cor da esfera;
5. Se $\Delta > 0$,raio atravessa a esfera em tocando em dois pontos;
    a. Calcule t;
    b. Marque o pixel referente ao raio com a cor da esfera onde t é o menor entre os valores de t obtidos;

Abaixo teremos os resultados obtidos com a implementação das projeções obliqua com os parâmetros $l = 10$, $r = -10$,$t = 10$, $b = -10$, $Nx = 250$,$Ny = 250$ e d com valores 0.5 ,0.5 ,0.15 respectivamente com centro da esfera $c = (0,0,0)$,raio $r = 5$ e ponto de visão e = (0 ,0 , -6).

Figura 4.1-2Figura 4.1 1 Esfera Obtida caso Obliquo com d = 0.5

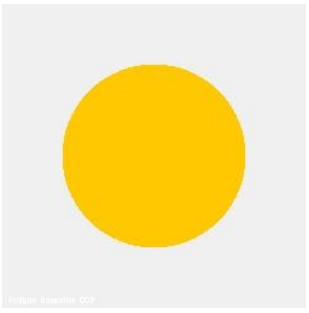

Figura 4.1-3 Esfera Obtida caso Obliquo com d = 0.25

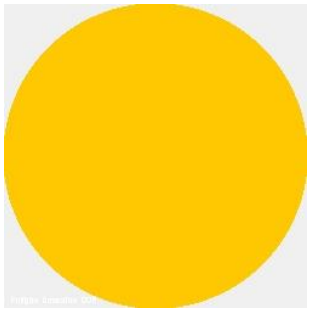

Figura 4.1-1Figura 4.1 2 Esfera Obtida caso Obliquo com d = 0.15

Variando a posição do ponto de visão e para os pontos (0,0,-5.2) ,(0,0,-6) e ()e a distância fixada em $d = 0.5$ obtemos os resultados abaixo.

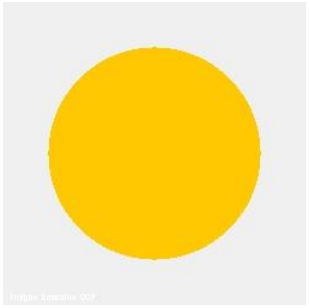

Figura 4.1-6 Esfera Obtida caso Obliquo com e = (0,0,-5.2)

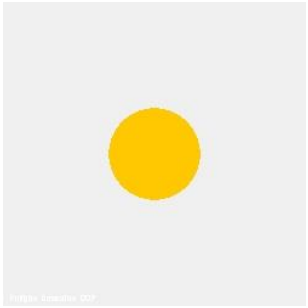

Figura 4.1-4Esfera Obtida caso Obliquo com e = (0,0,-6)

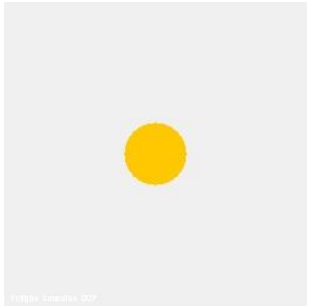

Figura 4.1-54 Esfera Obtida caso Obliquo com e = (0,0,-7)

Como é de se esperar, para o caso obliquo, o tamanho do objeto renderizado varia em relação a distância entre o ponto de visão e o objeto no espaço.

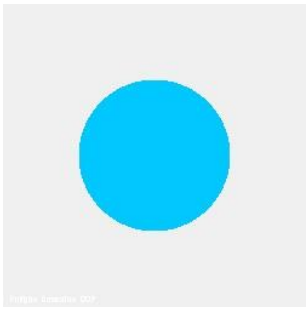

Figura 4.1-9 Esfera Obtida caso Ortográfico com e = (0,0,-7) e d = 1

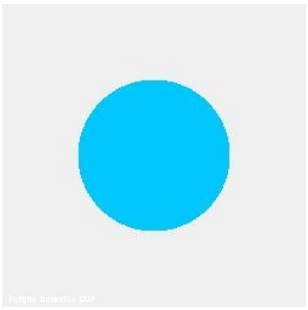

Figura 4.1-8 Esfera Obtida caso Ortográfico com e = (0,0,-100) e d = 100

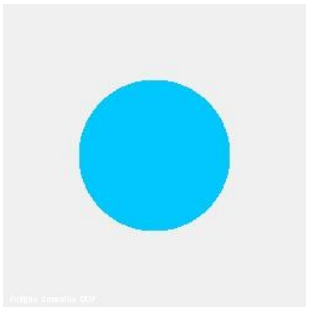

Figura 4.1-7 Esfera Obtida caso Ortográfico com e = (0,0,-50) e d = 0.0001

Acima os resultados obtidos com a implementação das projeções ortográficas para a esfera com os parâmetros $l = 10$, $r = -10$,$t = 10$, $b = -10$, $Nx = 250$,$Ny = 250$ e d com valores 1,1000,0.0001 respectivamente com centro da esfera $c = (0,0,0)$,raio $r = 5$ e ponto de visão e = (0 ,0 , -7), (0 ,0 , -100) e (0 ,0 , -7) .

O comportamento na renderização está dentro do esperado, a imagem manteve sua dimensão independente da variação dos parâmetros que inferissem variação da distância.

4.2. Renderização de Triângulos.

O processo para a renderização também inclui manipulação algébrica o que será abordado em seguida.

Será necessário, agora, detectar quando um determinado raio toca a superfície de um triangulo para isso precisaremos fazer uso das coordenadas do baricentro do objeto informado. Com as coordenadas do triângulo é possível estabelecer uma relação entre elas e o raio que para verificar se o mesmo o intercepta o resultado dessa relação é:

$$e + td = f(u,v) \rightarrow f(u,v) = \begin{cases} Xe + tXd = f(u,v) \\ Ye + tYd = g(u,v) \\ Ze + tZd = h(u,v) \end{cases} \quad 19.$$

Na equação 19 há três variáveis desconhecidas t, u e v, ainda também, sabendo que os vértices a, b e c definem um triângulo a interseção é válida quando:

$$e + td = a + \beta(b - a) + \gamma(c - a) \quad 20.$$

Para um raio atingir um ponto dentro do triângulo é necessário que as abaixo sejam válidas as condições:

$$\beta > 0 \, , \gamma > 0 \quad 21.$$

$$\beta + \gamma < 1 \quad 22.$$

Como $e \, , d, a, b, c$ são coordenadas podemos reescrever vetorialmente a equação 20 obtendo as equações:

$$\begin{aligned} Xe + tXd &= Xa + \beta(Xb - Xa) + \gamma(Xc - Xa) \\ Ye + tYd &= Ya + \beta(Yb - Ya) + \gamma(Yc - Ya) \\ Ze + tZd &= Za + \beta(Zb - Za) + \gamma(Zc - Za) \end{aligned} \qquad 23.$$

E por fim escrever o conjunto de equação 23 como um sistema linear obtendo:

$$\begin{bmatrix} Xa - Xb & Xa - Xc & Xd \\ Ya - Yb & Ya - Yc & Yd \\ Za - Zb & Za - Zc & Zd \end{bmatrix} \begin{bmatrix} \beta \\ \gamma \\ t \end{bmatrix} = \begin{bmatrix} Xa - Xe \\ Ya - Ye \\ Za - Ze \end{bmatrix} \qquad 24.$$

Podemos resolver facilmente o sistema linear 24 e obter os valores de β, γ e t aplicando a Regra de Kramer ,como é um método muito conhecido não se faz necessário detalhá-lo.

Com o auxílio do procedimento matemático acima é possível chegar a um algoritmo que nos devolva se um dado raio toca a superfície de um triângulo a saber:

1. Calcule t;
2. Se $t < t0$ ou t > 1 raio não toca o triangulo ;
3. Senão calcule γ;
4. Se $\gamma < 0$ ou $\gamma > 1$,raio não toca o triângulo;
5. Senão calcule β;
6. Se $\beta < 0$ ou $\beta > 1 - \gamma$, raio não toca o triângulo;
7. Senão o raio toca o triângulo;
8.

Os valores $t0$ e $t1$ são estimados empiricamente apenas pra limitar o espaço máximo de projeção do raio ,isso previne que o raio seja propagado ao infinito.

De posse do algoritmo e de todo o conhecimento algébrico chegamos a uma implementação capaz de renderizar triângulos presentes no espaço.

Assim para o caso obliquo com os parâmetros $l = 10$, $r = -10$,$t = 10$, $b = -10$, $Nx = 250$,$Ny = 250$ e d com valores 0.2,0.1,, respectivamente com triangulo com vértices localizados nos pontos a $= (5, -2{,}0)$, b $= (0{,}3{,}0)$ e, c $= (-5, -2{,}0)$, e ponto de visão e = (0 ,0 , -6).

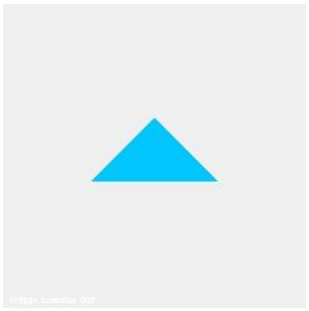

Figura 4.2-1 Triangulo renderizado no caso obliquo com d = 0.2

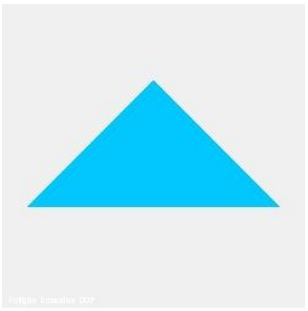

Figura 4.2-3 Triangulo renderizado no caso obliquo com d = 0.1

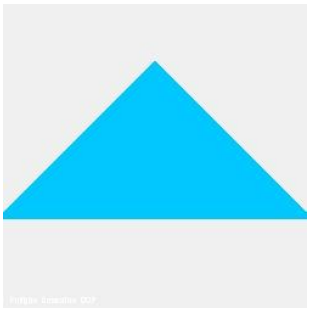

Figura 4.2-2Figura 4.2 2 Triangulo renderizado no caso obliquo com d = 0.08

Ainda fixando a distância em $d = 0.1$ e variando o ponto de visão e com os valores (0,0,-12) , (0,0,-9), (0,0,-6) respectivamente obtemos aos resultados abaixo.

Figura 4.2-6 Triangulo renderizado no caso obliquo com e = (0,0,-12)

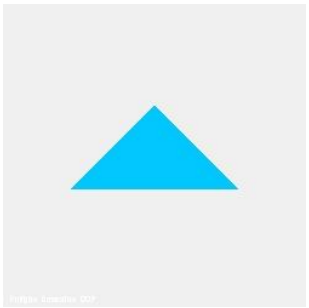

Figura 4.2-5 Triangulo renderizado no caso obliquo com e = (0,0,-9)

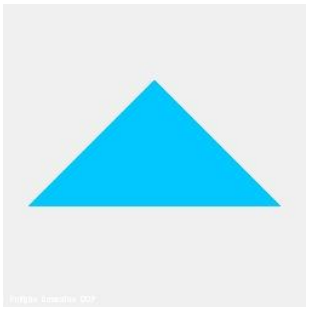

Figura 4.2-4Triangulo renderizado no caso obliquo com e = (0,0,-6)

Renderizando o triangulo usando projeção ortográfica com os parâmetros $l = 10$, $r = -10, t = 10,\ b = -10,\ Nx = 250, Ny = 250$ com o triangulo com vértices localizados nos pontos a $= (5, -2,0), \mathrm{b} = (0,3,0)$ e, c $= (-5, -2,0)$, e ponto de visão e = (0 ,0 , -6).

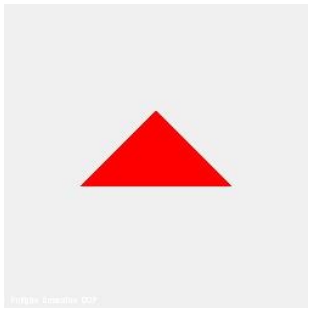

Figura 4.2-9 Triangulo renderizado no caso ortográfico com e = (0,0,-6) e d = 1.

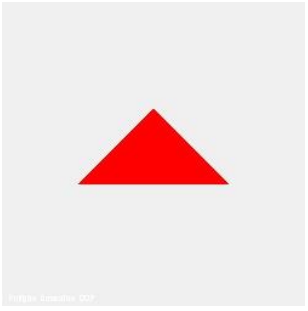

Figura 4.2-7 Triangulo renderizado no caso ortográfico com e = (0,0,-60) e d = 10.

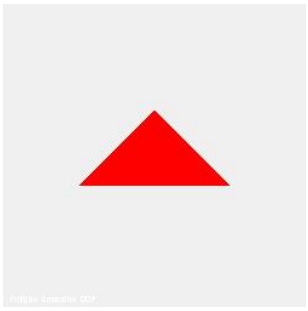

Figura 4.2-8 Triangulo renderizado no caso ortográfico com e = (0,0,-600) e d = 100.

Os resultados acima foram obtidos com ponto de visão e com valores (0,0,-6) , (0,0,-60) e (0,0,-600) e distancia d com os valores 1,10,100.

Assim como no caso da esfera a renderização na projeção ortográfica não provoca a variação das dimensões de tal forma que a distância e as coordenadas do ponto de visão se tornam irrelevantes.

4.3. Renderização de Polígonos.

A renderização de polígonos pode ser realizada usando os mesmos princípios matemáticos utilizados na renderização de triângulos.

Agrupando três a três os pontos do polígono que se deseja efetuar o processo podemos delimitar triângulos e assim realizar o restante do processo triangularizando o polígono em questão.

Podemos escrever um passo a passo com as seguintes etapas:

1. Para o polígono fixe as coordenadas de um vértice qualquer;
2. Percorra os vértices restantes de dois em dois e agrupe-os com o vértice fixado na etapa 1;
3. A cada conjunto de três vértices realize a verificação do triangulo formado conforme descrito na renderização de triângulos;

Aplicando o processo acima para o caso obliquo com os parâmetros $l = 10$, $r = -10$,$t = 10$, $b = -10$, $Nx = 250$,$Ny = 250$, com polígono de vértices localizados nos pontos a $= (8,3,0)$, b $= (0,8,0)$, c $= (-8,3,0)$,$d = (-5,-5,0)$ e $e = (5,-5,0)$, ponto de visão e = (0 ,0 , -6) e distância $d = 0.3,0.2,0.15$ respectivamente.

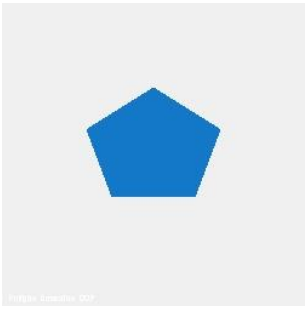

Figura 4.3-1 Pentágono renderizado no caso obliquo com e = (0,0,-6) e d = 0.3.

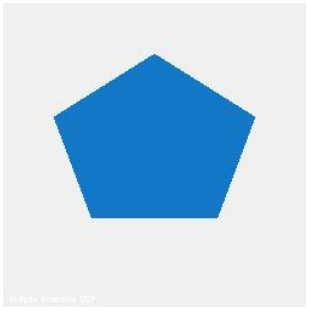

Figura 4.3-Pentágono renderizado no caso obliquo com e = (0,0,-6) e d = 0.2.

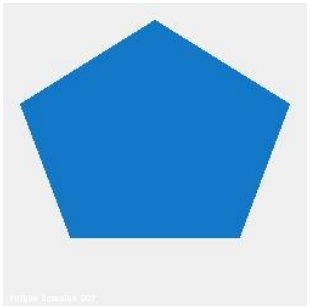

Figura 4.3-Pentágono renderizado no caso obliquo com e = (0,0,-6) e d = 0.15.

Abaixo a renderização obtida para o caso ortográfico.

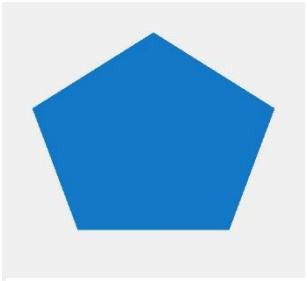

Figura 4.3-2 Pentágono renderizado no caso ortográfico com e = (0,0,-6)

Para evitar a repetição de idéias, a essa altura deste trabalho, os comportamentos de ambas as projeções seguem os mesmos já descritos anteriormente para os casos de renderização de triângulos, dado que são uma adaptação deste.

5. ILUMINAÇÃO.

Até agora nos preocupamos apenas com o processo de renderização dos objetos. Assim, o interesse estava voltado somente para a localização e captura da cor do objeto dentro do espaço.

Nessa etapa serão incluídos elementos dentro do espaço que interferirão na coloração dos objetos renderizados como ponto de luz que possuirão coordenadas, cor e intensidades com atributos.

Serão renderizados múltiplos objetos dentro do espaço usando projeção obliqua. Não serão considerados mais a variação decorrentes do posicionamento do ponto de visão e do valor de distância já que foram bem explicitadas em tópicos anteriores.

Esse tópico especificamente apresentará resultados sobre as implementações dos modelos de iluminação e sombreamento de Lambert e Bling-Phong considerando a cor do ambiente e variadas fontes de iluminação.

Para a criação dos resultados serão usados inicialmente uma fonte de luz localizado nos pontos (40,10, -30), com cor (240,0,0) em codificação RGB e intensidade 3, o ponto de visão está situado no ponto (0,0, -10), mais parâmetros serão adicionados conforme necessidade.

5.1. Modelo de Lambert

O sombreamento baseado no modelo de Lambert considera a influência da intensidade da luz sobre os objetos presentes no espaço.

O modelo relaciona o ângulo entre os vetores formados pelas coordenadas cartesianas do ponto de visão, a fonte de luz e o ponto de interseção do raio e o objeto

De posse dos parâmetros descritos e de conhecimentos algébricos relativamente simples podemos, apenas, substituir os valores na equação e o obter a cor com o modelo de Lambert.

$$L = Kd * i * \max(0, n \cdot l) \qquad 25.$$

Descrevendo a equação 25 ,L é cor obtida pelo modelo,Kd é o coeficiente de difusão ou cor da superfície tocado pelo raio, i a intensidade da fonte de iluminação, o termo n é o vetor normal da diferença entre os pontos de visão e da interseção do objeto , l é o vetor unitário da diferença entre os pontos de iluminação e da interseção do objeto.

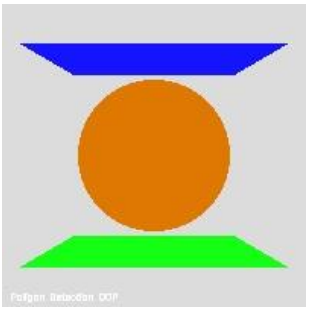

Figura 5.1-1 Objetos Renderizados sem Iluminação

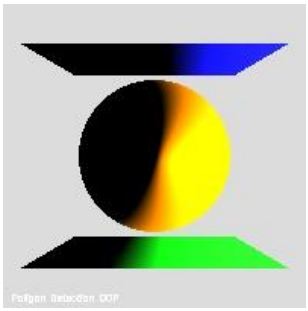

Figura 5.1-2Objetos Renderizados com Iluminação modelo de Lambert

Como se pode observar, há um decaimento muito rápido dos valores das zonas iluminadas até às não iluminadas, a conseqüência disso é a chegada muito abrupta ao escuro absoluto. Veremos no próximo tópico um a maneira de evitar isso o modelo de Bling-Phong .

5.2. Modelo de Bling-Phong

O modelo de Bling-Phong permite um decaimento menos abrupto entre as zonas iluminadas e não iluminadas em relação ao modelo de Lambert.

O modelo segue os mesmos princípios do modelo de Lambert, contudo, é adicionado um vetor h que divide o ângulo entre os vetores v e l exatamente ao meio. A iluminação passar a ser uma relação entre o vetor normal n e o vetor h no que diz respeito ao distância de n a h.

O valor do coeficiente de decaimento p deve ser maior que 0 e o vetor h pode ser obtido pela equação abaixo :

$$h = \frac{(v + l)}{||v + l||} \qquad 26.$$

E assim podemos através da equação abaixo computar o valor do pixel a aplicar o modelo com Ks sendo coeficiente especular ou cor da superfície tocada pelo raio .

$$L = Ks * i * \max(0, n \cdot h)^p \qquad 27.$$

Os resultados da implementação do modelo de Bling-Phong podem ser visualizados abaixo e uma comparação visual com o modelo de Lambert pode ser aferida.

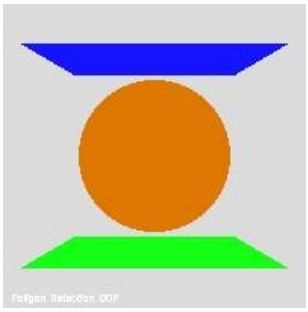

Figura 5.2-31 Objetos Renderizados sem Iluminação

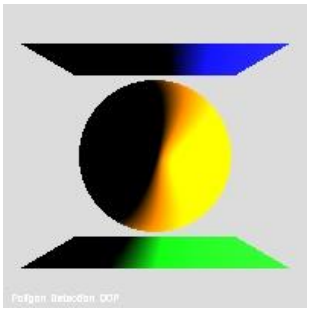

Figura 5.2-2Redenrização com modelo de Lambert

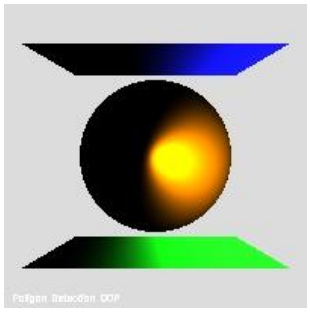

Figura 5.2-1Redenrização com modelo de Bling-Phong

Como se pode visualmente perceber, o modelo de Bling-Phong chegou a uma dispersão mais distribuída da iluminação no que diz respeito às zonas não iluminadas e mais concentrada das zonas iluminadas.

5.3. Modelo com Sombreamento do Ambiente

Os modelos de iluminação aplicados até o momento desconsideram os elementos presentes naturalmente como o ambiente e sua cor e aplicam às zonas não iluminadas valores muitos próximos do preto.

Adicionalmente aos modelos já implementados podemos adicionar um componente de cor referente a coloração do ambiente. Esse componente pode ser descrito como o produto entre o componente RGB da cor da variável de ambiente e da intensidade da cor do ambiente.

Assim chegamos a mais um modelo de iluminação descrito pela equação abaixo:

$$L = Ka * Ia + Kd * i * \max(0, n \cdot l) + Ks * i * \max(0, n \cdot h)^p \qquad 28.$$

Os termos Ka e Ia são os elementos adicionais referentes a coloração do ambiente ,respectivamente a componente cor RGB da cor do ambiente e a intensidade deste cor. E assim está pronto o modelo que permite a implementação para os resultados obtidos abaixo.

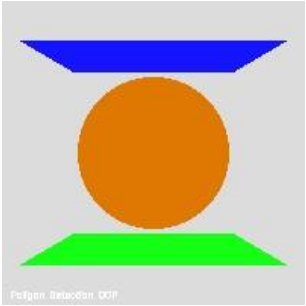

Figura 5.3-4 Objeto sem Iluminação

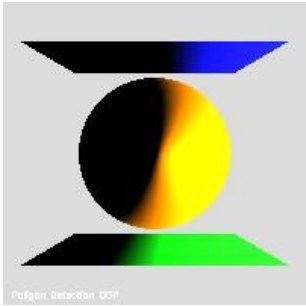

Figura 5.3-3Iluminação com modelo Lambert

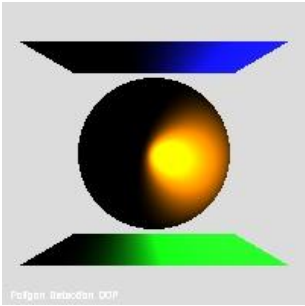

Figura 5.3-2Iluminação com modelo de Bling-Phong

Figura 5.3-1Iluminação com modelo sombreamento de ambiente

5.4. Modelo com múltiplas fontes de luz.

Os resultados apresentados até agora fizeram uso apenas de uma única fonte de luz. Podemos adaptar os modelos já apresentados para representar uma quantidade indefinida de fontes de iluminação o resultado.

O modelo gerado nessa situação de mais de uma fonte de iluminação está descrito abaixo.

$$L = Ka * Ia + \sum_{j=1}^{N} [Kd * i_j * \max(0, n \cdot l_j) + Ks * i_j * \max(0, n \cdot h)^p] \qquad 29.$$

Para esse modelo os termos com índices j enumeram os pontos de iluminação presentes no ambiente de forma que a cor resultante é a soma de todas as cores geradas pela interferência das diferentes fontes de iluminação j sobre os pontos tocados pelos raios.

Com a implementação do modelo podemos chegar aos resultados abaixo adicionando aos modelos anteriores por exemplo mais uma fonte de iluminação no ponto $(-40, -10, -10)$ com intensidade 1 e cor $(0{,}245{,}0)$ no modelo RGB.

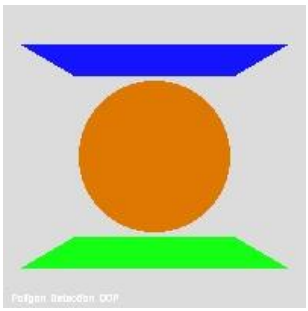

Figura 5.4-2 Objetos sem Iluminação

Figura 5.4-1Iluminação com uma fonte iluminação

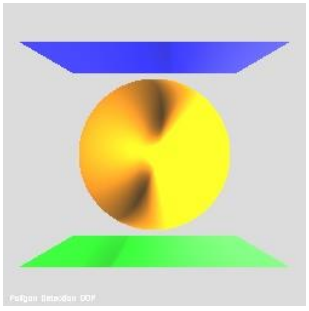

Figura 5.4-3Iluminação com 2 fontes de iluminação

6. TRANSFORMAÇÕES

As transformações dentro da renderização de objetos são maneiras de manipular os raios de forma a provocar alterações nos resultados das renderizações dos objetos. Essas transformações são definidas por matrizes de transformações pré-definidas.

As transformações são aplicadas simplesmente com o produto entre o resultado dos escalares gerados nas equações 6 e 7 e a matriz de transformação. Cada par de valores $u´$ e $v´$,que definem a direção do raio no plano ,deve ser multiplicado pela matriz de transformação.

O processo todo provoca a mudança nos pontos de interseção dos raios e por conseqüência a alteração nos objetos ao final do restante do processo.

6.1. Transformação Redimensionamento - Scale

A transformação scale provoca a variação nas dimensões dos objetos conforme a matriz utilizada para transformação.

As variações podem ocorrer no de acordo com a matriz abaixo:

$$Scale(Sx, Sy) = \begin{bmatrix} Sx & 0 \\ 0 & Sy \end{bmatrix} \quad 30.$$

Se considerarmos os valores $u´$e $v´$ e realizarmos o produto entre as matrizes teremos:

$$Scale(Sx, Sy) = \begin{bmatrix} Sx & 0 \\ 0 & Sy \end{bmatrix} x \begin{bmatrix} u´ \\ v´ \end{bmatrix} = \begin{bmatrix} Sxu´ \\ Syv´ \end{bmatrix} \quad 31.$$

O que demonstra o comportamento da direção dos raios na renderização dos objetos.

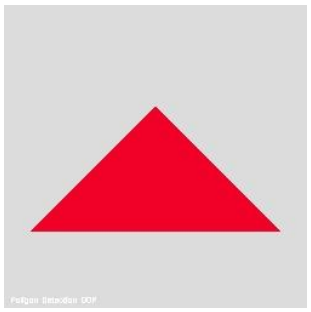

Figura 6.1-1Objeto sem transformação

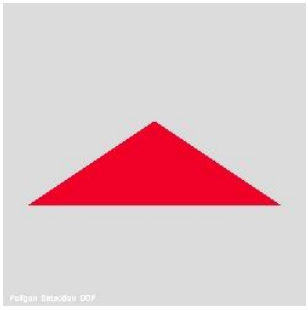

Figura 6.1-2Objeto renderizado com Scale(1.5,1)

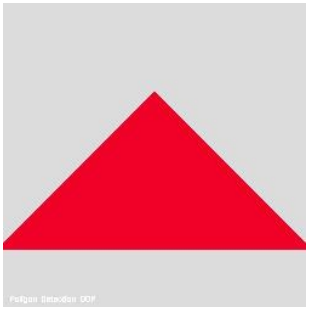

Figura 6.1-3Objeto renderizado com Scale(0.8,0.8)

6.2. Transformação Cisalhamento – Shear

A operação de cisalhamento provoca o deslocamento das arestas do objeto de acordo com a matriz de cisalhamento utilizada.

A matriz para o cisalhamento horizontal é realizada com uso da matriz:

$$ShearX(S) = \begin{bmatrix} 1 & S \\ 0 & 1 \end{bmatrix} \quad 32.$$

A matriz para o cisalhamento vertical é realizada com uso da matriz:

$$ShearY(S) = \begin{bmatrix} 1 & 0 \\ S & 1 \end{bmatrix} \quad 33.$$

De posse das matrizes podemos obter os resultados abaixo.

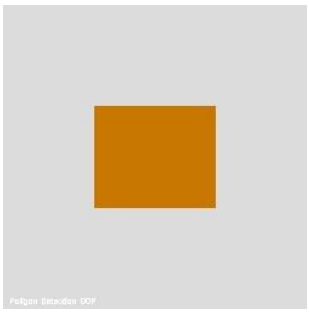

Figura 6.2-1Objeto original sem transformação

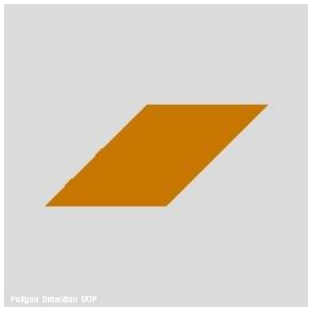

Figura 6.2-2Objeto com ShearX em 45graus

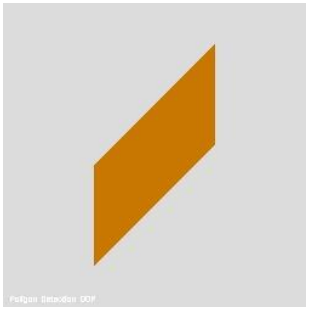

Figura 6.2-3Objeto com ShearY em 45 graus

6.3. Transformação Rotação – Rotate

Assim como as outras transformações a rotação é realizado com o auxílio de uma matriz de transformação essa matriz de é da forma:

$$rotate(s) = \begin{bmatrix} cos(s) & -\sin(s) \\ \sin(s) & \cos(s) \end{bmatrix} \quad 34.$$

O exemplo para a transformação de rotação pode ser visualizado abaixo.

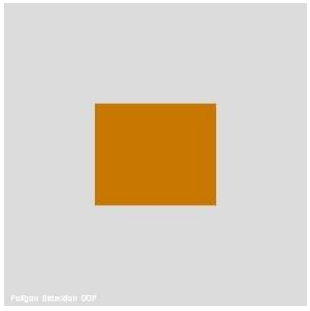

Figura 6.3-2Objeto original sem transformação

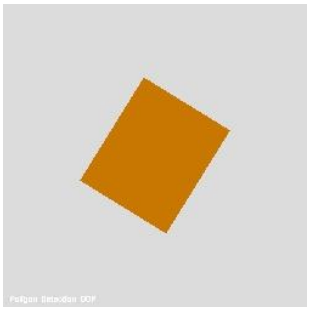

Figura 6.3-1Objeto com rotação em 45 graus

6.4. Transformação Reflexão – Reflect

A transformação de reflexão rebate o objeto na direção do eixo oposto de acordo com a matriz de transformação aplicada.

A matriz de transformação para a reflexão no eixo x é:

$$reflectX() = \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix} \quad 35.$$

A matriz de transformação para a reflexão no eixo y é:

$$reflectY() = \begin{bmatrix} -1 & 0 \\ 0 & 1 \end{bmatrix} \quad 36.$$

Abaixo a visualização das reflexões executadas.

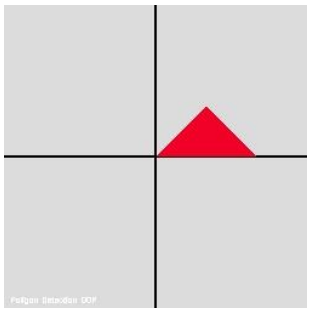

Figura 6.4-2 Objeto sem transformação

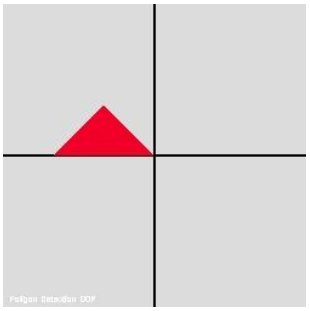

Figura 6.4-4Objeto refletido em Y

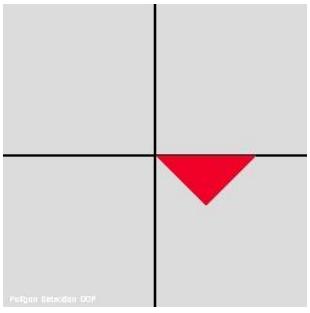

Figura 6.4-1Objeto refletido em X

Figura 6.4-3Objeto refletido em X