Instituto Federal de Educação, Ciência e Tecnologia do Ceará

Engenharia da Computação

Computação Gráfica

Prof. Lucas Silva de Sousa

­­

André Vieira da Silva

Relatório 05 : Introdução ao Raytrace com Interseção de uma esfera.

Fortaleza

Setembro de 2018

Sumário

[RayTrace 3](#_Toc523859530)

[Tipos de RayTrace 3](#_Toc523859531)

[Modo Ortográfico 3](#_Toc523859532)

[Modo Obliquo 3](#_Toc523859533)

[Base Teórica 3](#_Toc523859534)

[Resultados 3](#_Toc523859535)

[Interceptação Esfera Caso Obliquo 4](#_Toc523859536)

[Interceptação Esfera Caso Ortográfico. 4](#_Toc523859537)

[Implementação 6](#_Toc523859538)

# RayTrace

O Raytrace é um algoritmo usado na renderização de imagens do plano 3D. Pretende simular os raios de luz capturados pelo olho humano afim de transcrever em dados computacionais os objetos presentes em um determinado ambiente incluindo características como luminosidade, sobreposição de objetos e outros objetos.

# Tipos de RayTrace

Basicamente o algoritmo do Raytrace trabalha de dois modos:

## Modo Ortográfico

Permite uma vista sem perda das dimensões dos objetos da imagem, de implementação simplificada, contudo a noção de espaço, sombreamentos e outros elementos fica comprometida.

## Modo Obliquo

Permite a visualização dos objetos através de pontos de visão o que preserva a influência percepção da variação luminosa que permite a noção mais aguçada do posicionamento dos objetos entre si.

# Base Teórica

A formulação teórica para o algoritmo do Raytrace se dá através da aplicação da decomposição de vetores e na obtenção de planos através de modelos matemáticos conhecidos fornecidos para álgebra linear de modo geral a base da teoria que envolve vetores e planos necessariamente.

A intersecção entre objetos é obtida através de modelos matemáticos que derivam também de modelos algébricos lineares. Basicamente é possível representear qualquer objeto através de uma função, mesmo que se tem que apelar a modelos polinomiais, diante disso podemos relaciona-los às suas devidas coordenadas no espaço.

No nosso exemplo faremos a inclusão no espaço de mais um elemento, o ponto de visão, de forma a relaciona-lo linearmente com o objeto presente no plano no caso específico uma esfera estabelecendo relações que nos permitem escrever o objeto sobre um plano e assim geram imagem do mesmo.

# Resultados

Os resultados foram obtidos tendo como parâmetros de entrada:

distancia da esquerda

## Interceptação Esfera Caso Obliquo

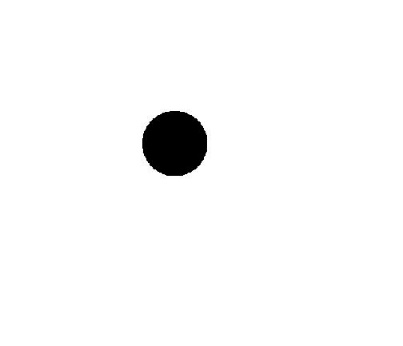
 A relação entre os raios gerados do caso obliquo permite capturar uma imagem similar a visão humana no qual os objetos se tornam menores conforme a distância entre o ponto de visão e os objetos alvos.

Figura 2 Interceptação com distancia = 1

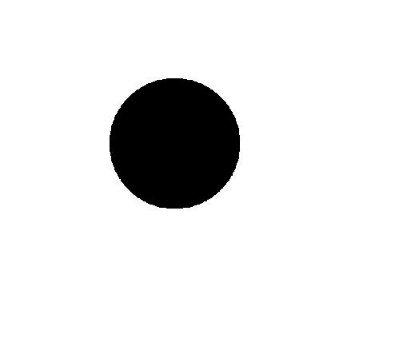
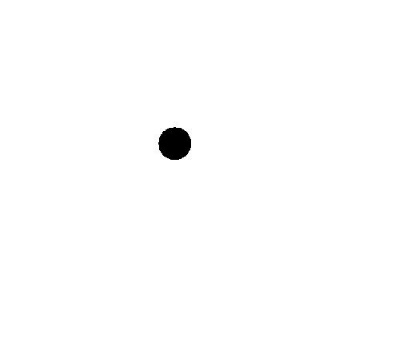


Figura 3 Interceptação com distancia = 2

Figura1 Interceptação com distancia = 0,5

## Interceptação Esfera Caso Ortográfico.

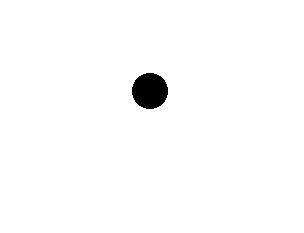
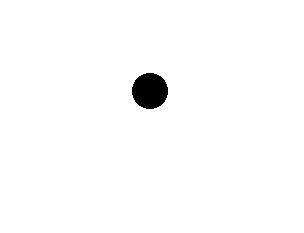
A relação entre os raios gerados do caso ortográfico permite capturar uma imagem preservando o tamanho original dos objetos presentes nesta, contudo coma relação entre os raios e pixels da imagem é feita de forma perpendicular a noção de profundidade pode ser facilmente comprometida dado que a distância do ponto de visão até o objeto no caso ortográfico perde importância significativa.

Figura 5 Interceptação com distancia = 1

Figura 5 Interceptação com distancia = 2

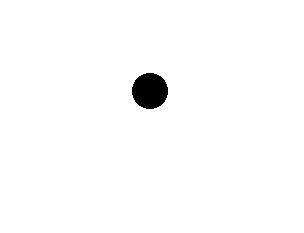


Figura 4 Interceptação com distancia = 0,5

# Implementação

Adotamos como ferramentas para este trabalho o programa MATHLAB da Patchworks versão 2018b licença estudante sobre com modularização de funções aplicadas sobre paradigma estruturado. Abaixo as funções implementadas.

function vetor\_w = calcular\_vetor\_w(vetor\_origem)

%calcular\_vetor\_w

%calcula um vetor unitario na direção da origem

vetor\_w = (vetor\_origem / norm(vetor\_origem));

end

function vetor\_u = calcula\_vetor\_u(vetor\_w)

%calcula\_vetor\_u

%calcula um vetor perpendicular a vetor\_w

temp\_vet = vetor\_w;

minIndex = find(abs(min(temp\_vet)));

if(temp\_vet(minIndex) == 1)

temp\_vet(minIndex) = 0;

else

temp\_vet(minIndex) = 1;

vetor\_u = cross(vetor\_w,temp\_vet) / norm(cross(vetor\_w,temp\_vet));

end

function vetor\_v = calcular\_vetor\_v(vetor\_w,vetor\_u)

%calcular\_vetor\_v

%calcula um vetor perpendicula a vetor\_w e vetor\_u

vetor\_v = cross(vetor\_w,vetor\_u);

end

function delta = calcular\_delta(vetor\_w,origem,centro\_esfera,raio\_esfera)

%intercepta\_esfera

%verifica se o raio intercepta uma esfera através do delta

emenosc = origem - centro\_esfera;

var\_A = (2 \* dot(vetor\_w,emenosc)) ^ 2;

var\_B = - 4 \* dot(vetor\_w,vetor\_w) \* (dot(emenosc,emenosc) - (raio\_esfera ^ 2));

delta = var\_A + var\_B;

end

function raio = gerar\_raio(i,j,l,b,r,t,nx,ny,distancia,vetor\_origem,vetor\_u,vetor\_v)

%gerar\_raio

%retorna um raio a partir da origem

ul = l + (r - l) \* (0.5 + i) / nx;

vl = b + (t - b) \* (0.5 + j) / ny;

raio = distancia \* vetor\_origem + ul \* vetor\_u + vl \* vetor\_v;

end

function img = raytrace\_obliquo(l,b,r,t,nx,ny,distancia,vetor\_origem,vetor\_w,vetor\_u,vetor\_v)

%raytrace

%gera raios a partir de um ponto de origem

img = zeros(nx,ny);

figure;

hold on;

raios = [];

for i=1:1:nx

for j = 1:1:ny

raio = gerar\_raio(i,j,l,b,r,t,nx,ny,1/distancia,vetor\_origem,vetor\_u,vetor\_v);

%raios = [raios,raio];

delta = calcular\_delta(raio,vetor\_origem,[0 0 0 ],3);

if(delta < 0)

img(i,j) = 255;

else

img(i,j) = 100;

end

end

end

imshow(img)

end

function img = raytrace\_ortografico(l,b,r,t,nx,ny,distancia,vetor\_origem,vetor\_w,vetor\_u,vetor\_v)

%raytrace

%gera raios a partir de um ponto de origem

img = zeros(nx,ny);

figure;

hold on;

for i=1:1:nx

for j = 1:1:ny

raio = gerar\_raio(i,j,l,b,r,t,nx,ny,distancia,vetor\_origem,vetor\_u,vetor\_v);

%raios = [raios,raio];

delta = calcular\_delta(vetor\_origem,raio,[0 0 0 ],3);

if(delta < 0)

img(i,j) = 255;

else

img(i,j) = 0;

end

end

imshow(img)

end

end