Instituto Federal de Educação, Ciência e Tecnologia do Ceará

Engenharia da Computação

Computação Gráfica

Prof. Lucas Silva de Sousa

­­

André Vieira da Silva

Relatório 04:Introdução ao Raytrace

Fortaleza

Agosto de 2018

Sumário

[RayTrace 3](#_Toc523616255)

[Tipos de RayTrace 3](#_Toc523616256)

[Modo Ortográfico 3](#_Toc523616257)

[Modo Obliquo 3](#_Toc523616258)

[Base Teórica 3](#_Toc523616259)

[Implementação 4](#_Toc523616260)

# RayTrace

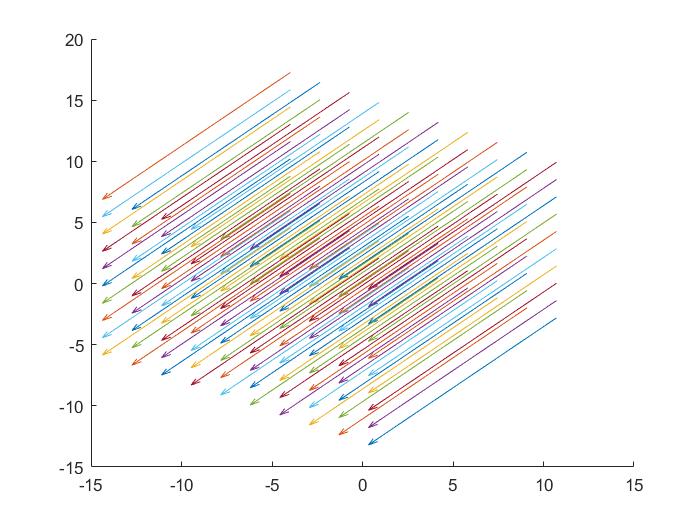
O Raytrace é um algoritmo usado na renderização de imagens do plano 3D. Pretende simular os raios de luz capturados pelo olho humano afim de transcrever em dados computacionais os objetos presentes em um determinado ambiente incluindo características como luminosidade, sobreposição de objetos e outros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

# Tipos de RayTrace

Basicamente o algoritmo do Raytrace trabalha de dois modos:

## Modo Ortográfico

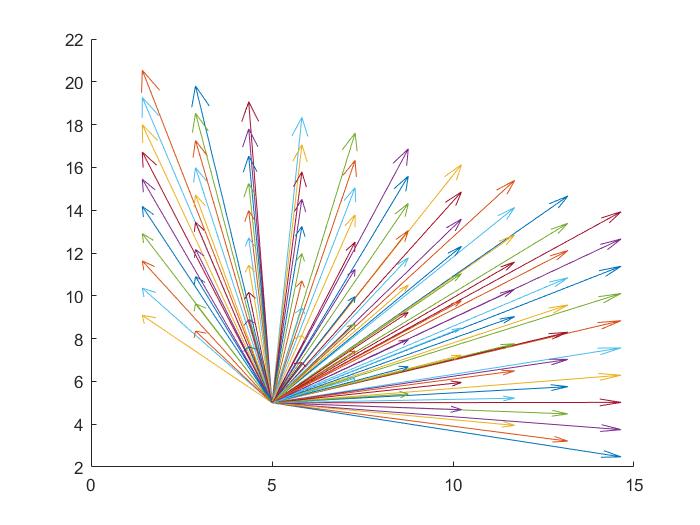
Permite uma vista sem perda das dimensões dos objetos da imagem, de implementação simplificada, contudo a noção de espaço, sombreamentos e outros elementos fica comprometida. Os raios partem em um ângulo de noventa graus ao plano de origem.



## 

Figura 1 Os raios partem em um ângulo de noventa graus ao plano de origem.

## 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.

# 

Figura 2 Raios partem de um ponto de visão em direção angular ao plano de origem

# 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.

# Implementação

Para a implementação foi usado o software MATLAB no que foram implementados os modelos matemáticos para obtenção sendo o código de ambas a implementações apresentados abaixo:

nx = 50 ny = 50 l = -10 b = -10 r = 10 t = 10

origem = [5 5 5]

w = origem / norm(origem)

tempV = w

minIndex = find(abs(min(tempV)))

if (tempV(minIndex) == 1)

tempV(minIndex) = 0

else

tempV(minIndex) = 1

end

u = cross(w,tempV) / norm(cross(w,tempV))

figure

hold on

for i=1:1:nx

for j = 1:1:ny

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

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

raio = origem + ul \* u + vl \* v

quiver3(origem(1),origem(2),origem(3),raio(1),raio(2),raio(3))

end

end

Figura 3Implementação Modo Obliquo

Figura 4 Implementação Modo ortográfico

nx = 50 ny = 50 l = -10 b = -10 r = 10 t = 10 k = 20

origem = [5 5 5]

w = origem / norm(origem)

tempV = w

minIndex = find(abs(min(tempV)))

if (tempV(minIndex) == 1)

tempV(minIndex) = 0

else

tempV(minIndex) = 1

end

u = cross(w,tempV) / norm(cross(w,tempV))

figure

hold on

for i=1:1:nx

for j = 1:1:ny

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

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

raio = origem + ul \* u + vl \* v

quiver3(raio(1),raio(2),raio(3),-k \* w(1),-k \* w(2),-k \* w(3))

end

end