

Ray Tracing: O Mundo Através De Raios de Luz

XXXVI Jornada Giulio Massarani de Iniciação Científica, Tecnológica,
Artística e Cultural

Thiago Barroso Perrotta
Prof.^o Ricardo G. marroquim

Universidade Federal do Rio de Janeiro

10 de outubro de 2014



Agenda

- 1 Ray-tracer
 - O algoritmo
- 2 Extração de primitivas em nuvens de pontos
- 3 Resultados
- 4 Referências

O quê?

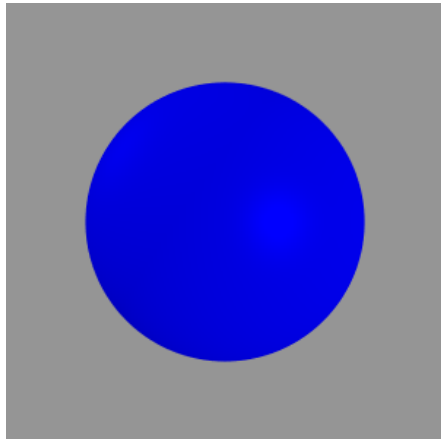
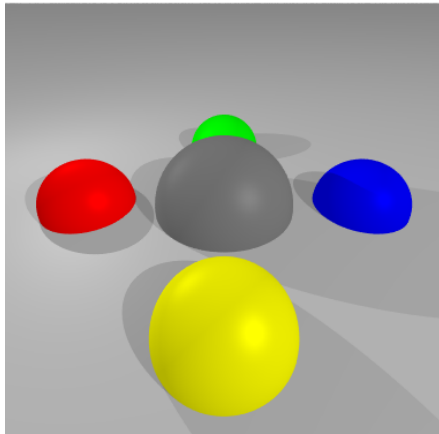
- Renderizar imagens
- Realismo
-

O algoritmo

- Defina alguns objetos
- Especifique um material para cada objeto
- Defina algumas fontes de luz
- Defina uma janela cuja superfície seja coberta com pixels
- Para cada *pixel*
 - Atire um raio, a partir do centro do *pixel*, na direção dos objetos
 - Compute, dentre os pontos atingidos em cada objeto, o que está mais próximo
 - Se o raio atingiu um objeto
 - Use o material do objeto e as fontes de luz para computar a cor do pixel
 - Senão
 - Defina a cor do *pixel* como preta

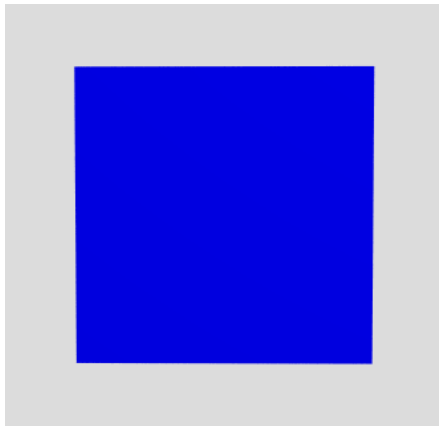
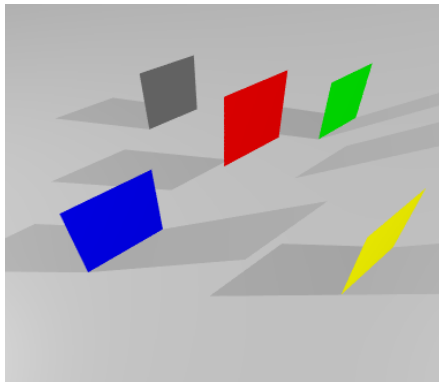
Objetos

Esferas



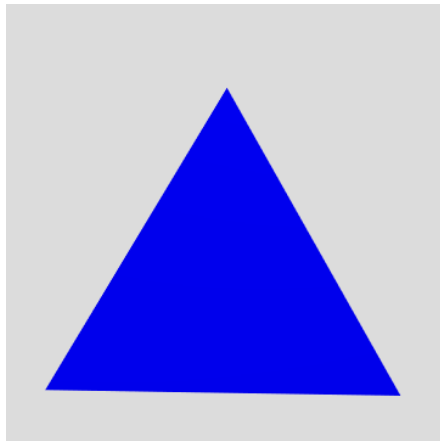
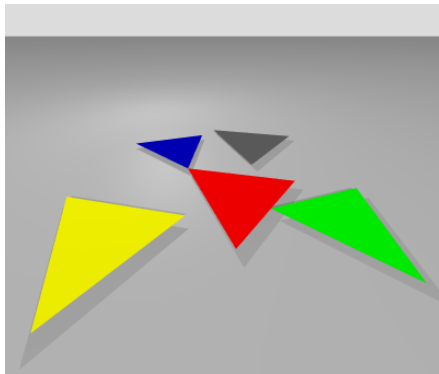
Objetos

Retângulos



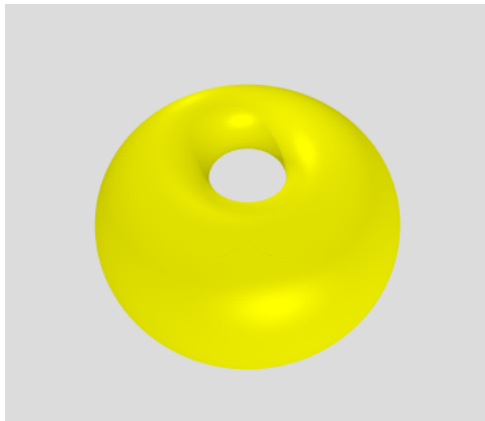
Objetos

Triângulos



Objetos

Toros



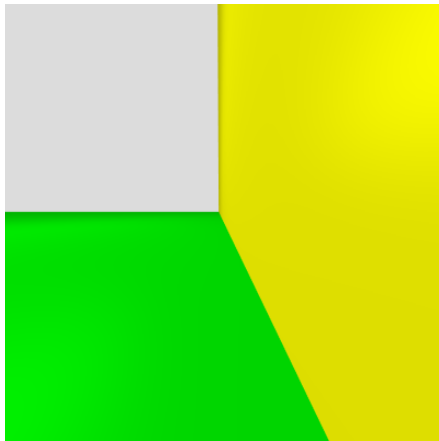
Objetos

Cilindros



Objetos

Planos



O algoritmo

- Defina alguns objetos
- Especifique um material para cada objeto
- Defina algumas fontes de luz
- Defina uma janela cuja superfície seja coberta com pixels
- Para cada *pixel*
 - Atire um raio, a partir do centro do *pixel*, na direção dos objetos
 - Compute, dentre os pontos atingidos em cada objeto, o que está mais próximo
 - Se o raio atingiu um objeto
 - Use o material do objeto e as fontes de luz para computar a cor do pixel
 - Senão
 - Defina a cor do *pixel* como preta

Materiais

Iluminação

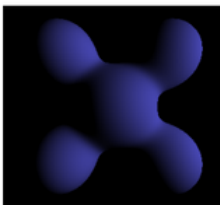
Tipos

- Ambiente
- Difusa
- Especular



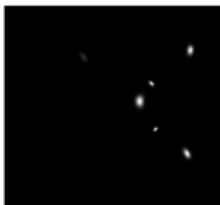
Ambient

+



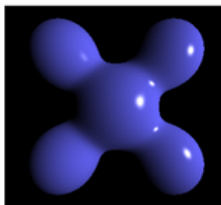
Diffuse

+



Specular

=

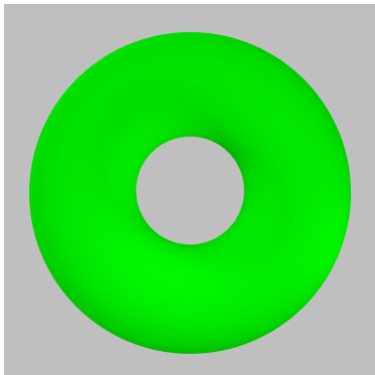


Phong Reflection

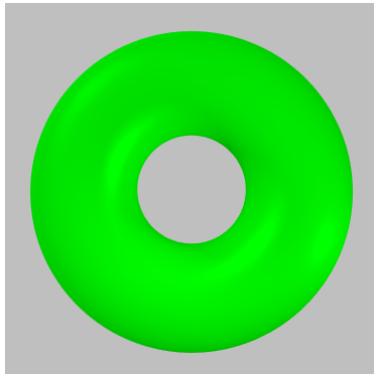
Materiais

Modelos

- *Matte* = Ambiente + Difusa
- *Phong* = Ambiente + Difusa + Especular



Matte



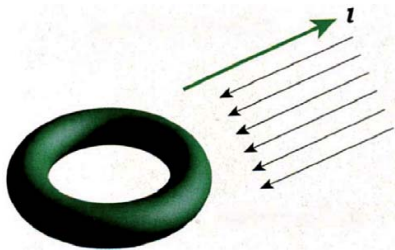
Phong

O algoritmo

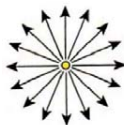
- Defina alguns objetos
- Especifique um material para cada objeto
- Defina algumas fontes de luz
- Defina uma janela cuja superfície seja coberta com pixels
- Para cada *pixel*
 - Atire um raio, a partir do centro do *pixel*, na direção dos objetos
 - Compute, dentre os pontos atingidos em cada objeto, o que está mais próximo
 - Se o raio atingiu um objeto
 - Use o material do objeto e as fontes de luz para computar a cor do pixel
 - Senão
 - Defina a cor do *pixel* como preta

Fontes de luz

- Direcionais
- Pontuais



Direcional



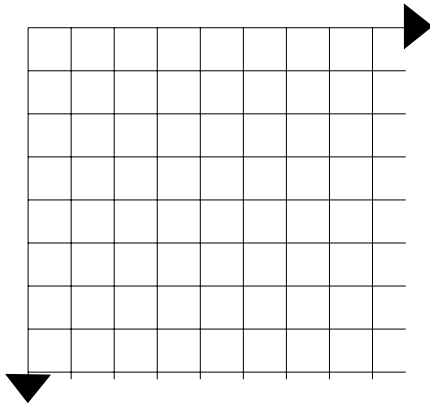
Pontual

O algoritmo

- Defina alguns objetos
- Especifique um material para cada objeto
- Defina algumas fontes de luz
- Defina uma janela cuja superfície seja coberta com pixels
- Para cada *pixel*
 - Atire um raio, a partir do centro do *pixel*, na direção dos objetos
 - Compute, dentre os pontos atingidos em cada objeto, o que está mais próximo
 - Se o raio atingiu um objeto
 - Use o material do objeto e as fontes de luz para computar a cor do pixel
 - Senão
 - Defina a cor do *pixel* como preta

Plano de visualização

- Número de *pixels* (ex.: 400x400)
 - Horizontal
 - Vertical
- Tamanho de cada *pixel* \Rightarrow *zoom*



O algoritmo

- Defina alguns objetos
- Especifique um material para cada objeto
- Defina algumas fontes de luz
- Defina uma janela cuja superfície seja coberta com pixels
- Para cada *pixel*
 - Atire um raio, a partir do centro do *pixel*, na direção dos objetos
 - Compute, dentre os pontos atingidos em cada objeto, o que está mais próximo
 - Se o raio atingiu um objeto
 - Use o material do objeto e as fontes de luz para computar a cor do pixel
 - Senão
 - Defina a cor do *pixel* como preta

Interseção entre raio e objetos

- Função *Hit* para cada objeto
-

O algoritmo

- Defina alguns objetos
- Especifique um material para cada objeto
- Defina algumas fontes de luz
- Defina uma janela cuja superfície seja coberta com pixels
- Para cada *pixel*
 - Atire um raio, a partir do centro do *pixel*, na direção dos objetos
 - **Compute, dentre os pontos atingidos em cada objeto, o que está mais próximo**
 - Se o raio atingiu um objeto
 - Use o material do objeto e as fontes de luz para computar a cor do pixel
 - Senão
 - Defina a cor do *pixel* como preta

ponto mais perto...

O algoritmo

- Defina alguns objetos
- Especifique um material para cada objeto
- Defina algumas fontes de luz
- Defina uma janela cuja superfície seja coberta com pixels
- Para cada *pixel*
 - Atire um raio, a partir do centro do *pixel*, na direção dos objetos
 - Compute, dentre os pontos atingidos em cada objeto, o que está mais próximo
 - Se o raio atingiu um objeto
 - Use o material do objeto e as fontes de luz para computar a cor do pixel
 - Senão
 - Defina a cor do *pixel* como preta

Computar cores

- RGB (Vermelho, Verde e Azul)
- Extras
 - *gamma*
 - *gamut*

Extração de primitivas

Conceituando

- Primitivas
 - Cilindros...

RANSAC

explicar, esquema, ...

Alguns resultados

Ray-tracing

Alguns resultados

Extração de primitivas

Implementação

Detalhes

Linguagem de Programação	C++, com orientação a objetos
<i>Kit</i> gráfico	Qt 5
Gerenciamento de <i>build</i>	CMake
<i>Framework</i> de testes	Google Test

Plataforma



- Ubuntu 14.04 LTS 64-bit
- Intel Core i7 950 @ 3.07 GHz x 8 cores
- 15,7 GB de RAM

Benchmarking

Ideias Futuras

- Ideias futuras
 - Ray-tracer
 - Serialização
 - Paralelização
 - Extração de Primitivas
 - Melhorar algoritmos
- Conclusão

Referências

-  Suffern, Kevin Geoffrey, and Suffern, Kevin. Ray Tracing from the Ground up. AK Peters, 2007.
-  Schnabel, Ruwen, Roland Wahl, and Reinhard Klein. "Efficient RANSAC for Point-Cloud Shape Detection." Computer graphics forum. Vol. 26. No. 2. Blackwell Publishing Ltd, 2007.