Visibilidade em Computação Gráfica

Universidade Estadual de Campinas
Faculdade de Engenharia Elétrica e de Computação
Disciplina IA 725 – Computação Gráfica

Elias Yauri
Natasha Nakashima

Campinas/2011

Conteúdo

- Motivação
- Objetivos
- Evolução histórica
  - Classificação dos algoritmos
- Pré-processamento
- Algoritmos
  - Z-Buffer
  - Pintor
  - Warnock
  - Ray-Tracing
- Experimentos
  - OpenGL
  - PovRay
- Conclusão

Motivação

- Determinar a visibilidade de linhas e superfícies foi uns dos principais problemas a ser resolvido pela nascente computação gráfica, dos anos 70; atualmente, com o incremento da complexidade da cena (cem, mil e milhões de segmentos e polígonos), a visibilidade ainda continua sendo crucial na síntese de imagens de uma cena 3D.

Objetivos

- Objetivo geral
  - Fazer um estudo e classificação dos algoritmos de visibilidade em computação gráfica.
- Objetivos específicos
  - Apresentar as técnicas de programação utilizadas em OpenGL para determinar a visibilidade de uma cena (Z-Buffer e Back-Culling).
  - Apresentar o traçado de raio (Ray-tracing) no PovRay para determinação de visibilidade.

Evolução Histórica

- Anos 60: primeiros algoritmos de visibilidade
  - Determinação de linhas e superfícies visíveis
- 1968: Appel apresenta o primeiro trabalho referente ao algoritmo de Ray-Tracing
- 1974: Sutherland apresenta vantagens na utlização de Coerência nos algoritmos de visibilidade
- 1975: Catmull apresenta o algoritmo Z-Buffer

Classificação dos Algoritmos

Visibilidade orientada a imagem

- Determinar os objetos visíveis em cada pixel da cena
- Custo computacional: np
  - n = número de objetos
  - p = número de pixels
- Z-Buffer
- Ray-tracing

Visibilidade orientada ao objeto

- Comparação entre cada objeto na cena
- Custo computacional: n^2
  - n = número de objetos
- Pintor
- BSP (Binary Spatial Partition)

Pré-processamento

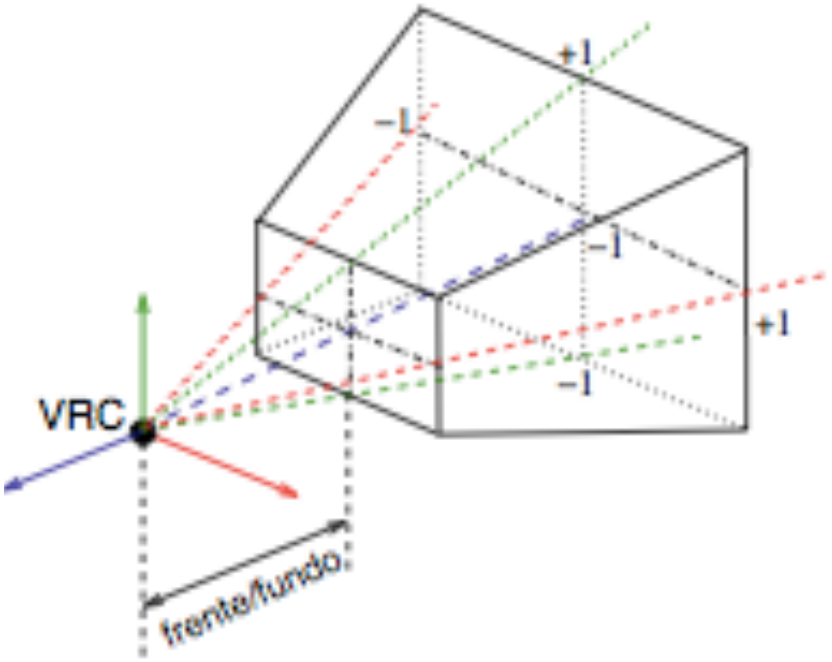
- Minimizar o tempo gasto para a geração de imagens
  - Coerência
  - Transformação de perspectiva
  - Bounding Volumes
  - Back-face Culling
  - Particionamento espacial
  - Hierarquia

Pré-processamento
Coerência

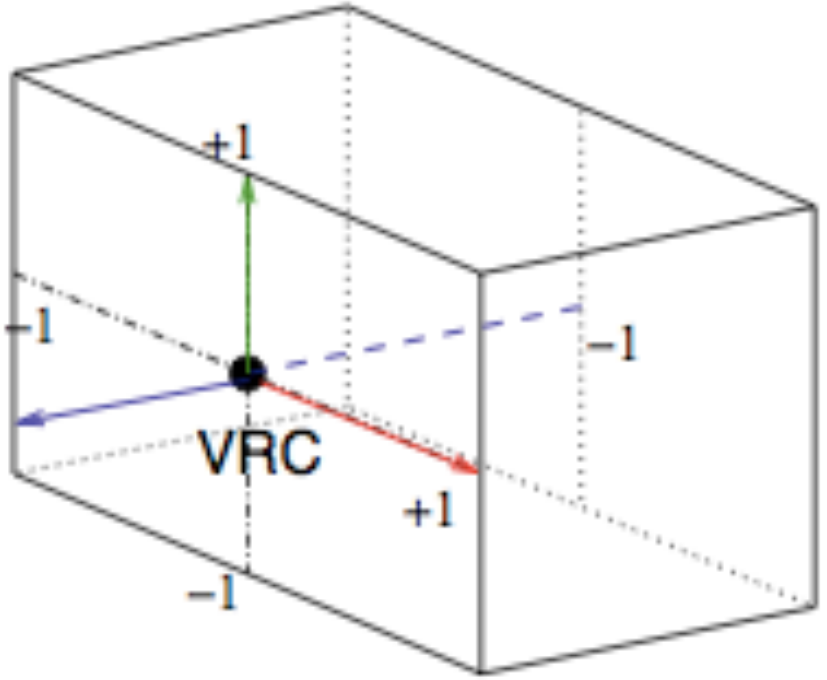
- Coerência de objetos
- Coerência das linhas de varredura
- Coerência na área de cobertura
- Coerência na profundidade

Pré-processamento
Transformação em Perspectiva

Volume de visão em perspectiva

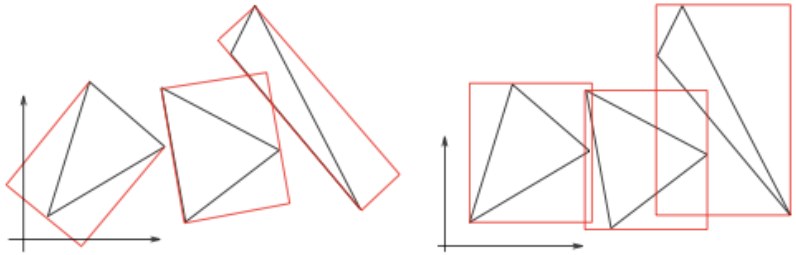


Volume de visão paralelo

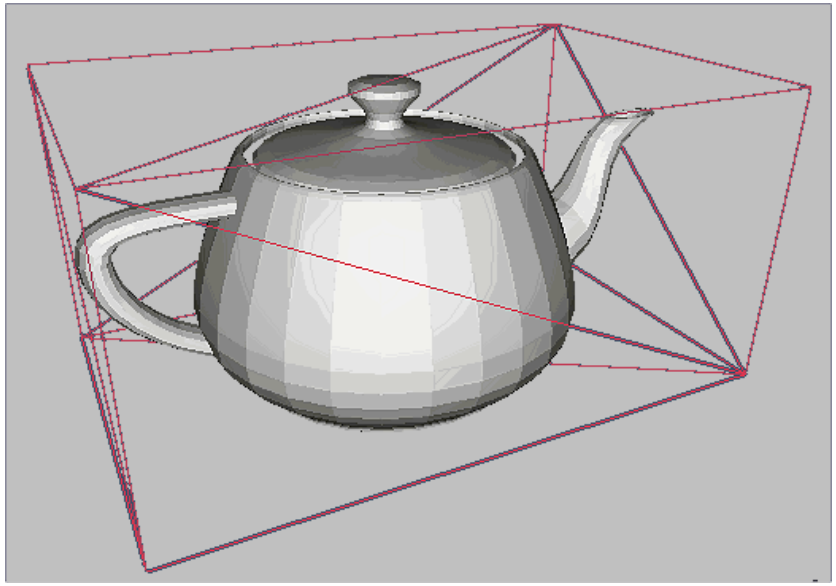


Pré-processamento
Caixas Limitantes

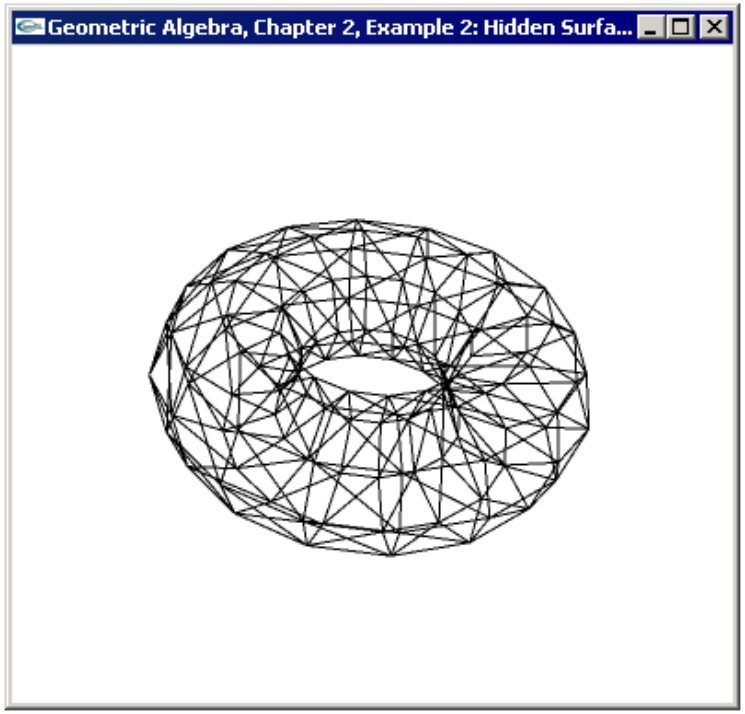
Caixas limitantes

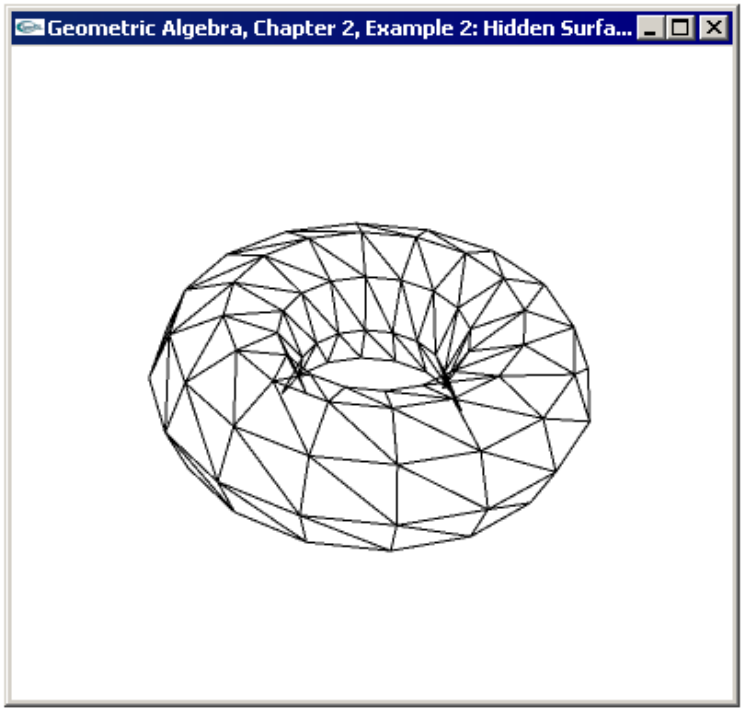


Volume limitante

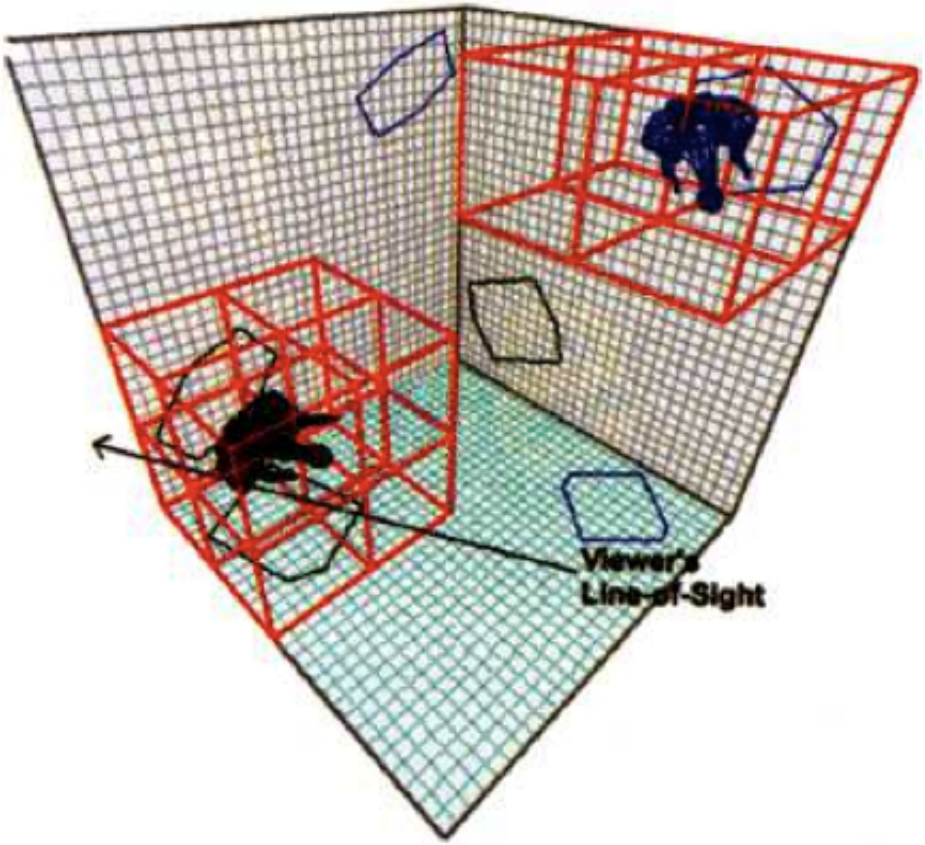


Pré-processamento
Back-face Culling

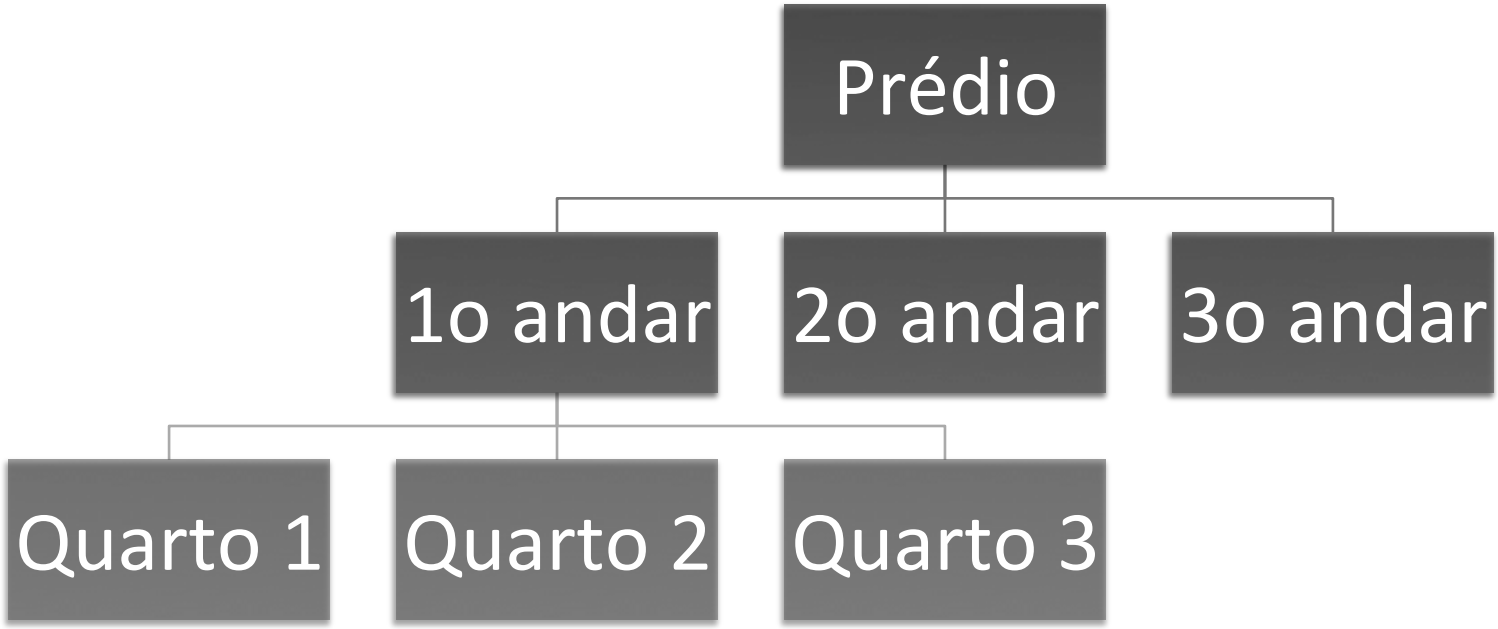




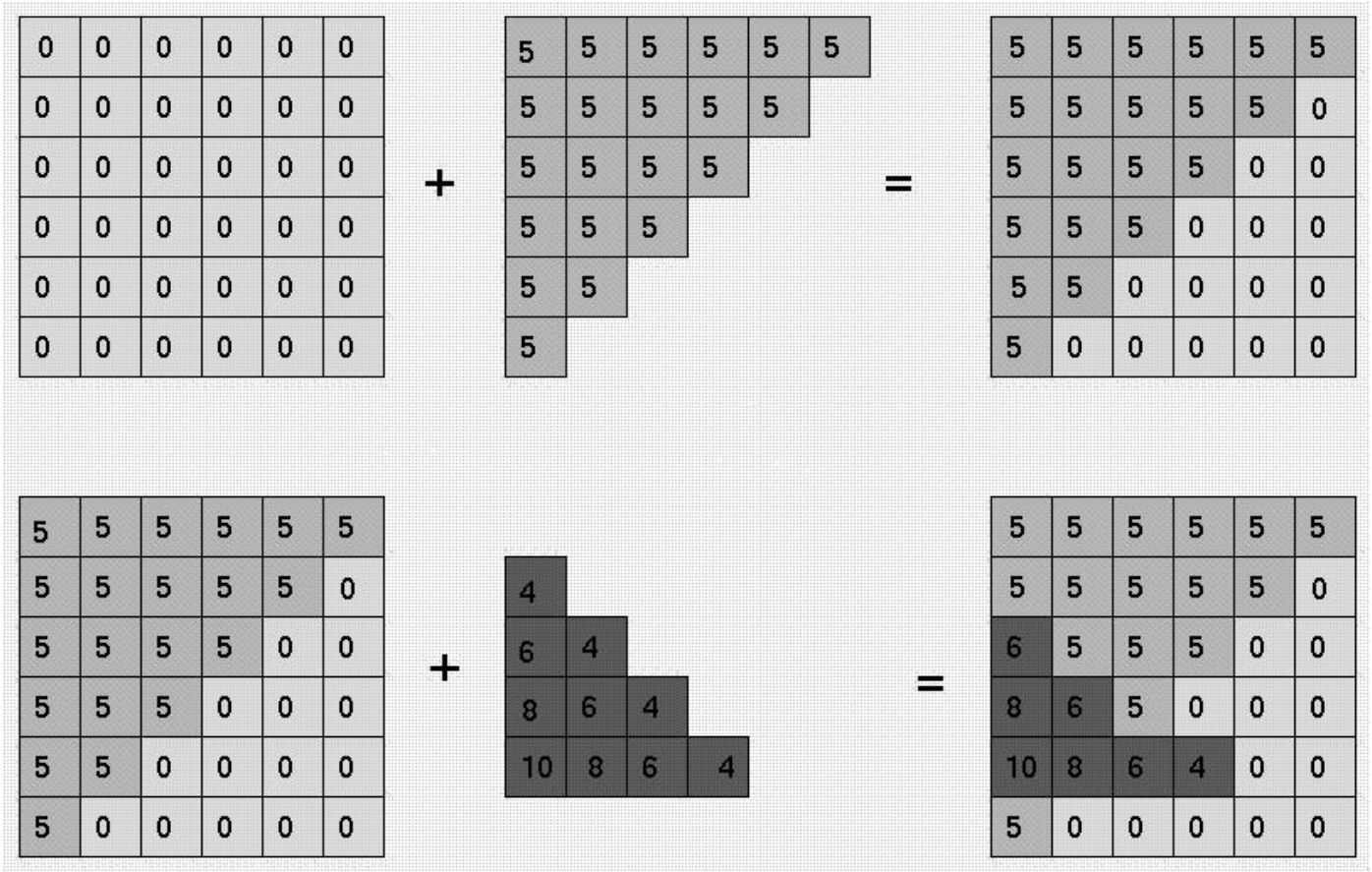
Pré-processamento
Particionamento Espacial



Pré-processamento
Hierarquia

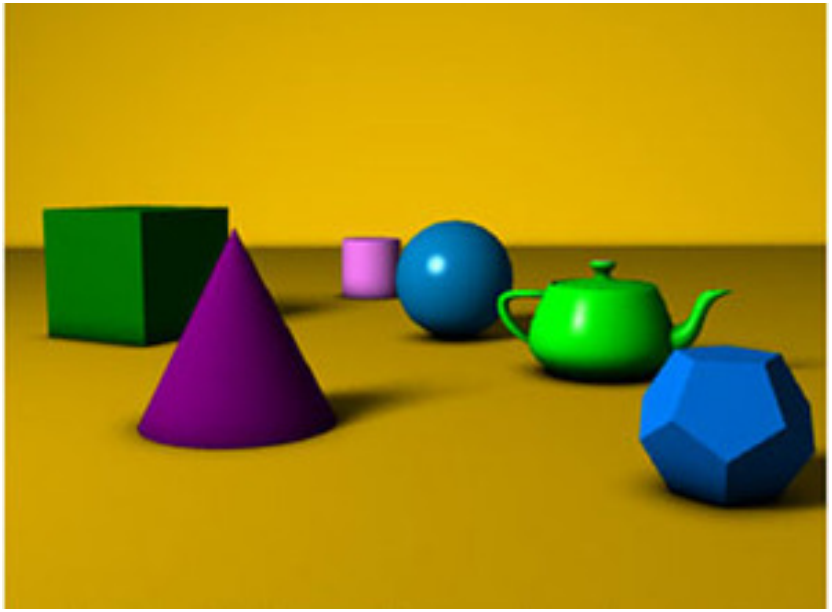


Principais Algoritmos
Z-Buffer



Principais Algoritmos
Z-Buffer

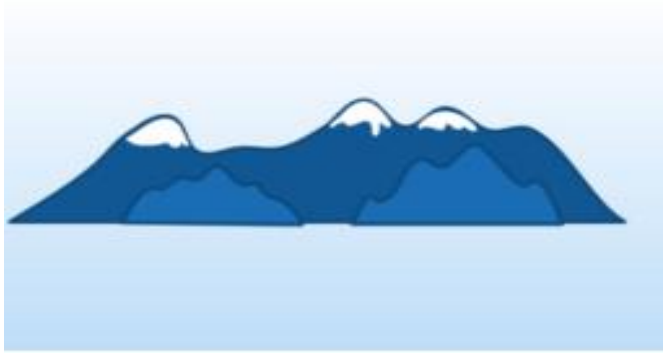
Cena 3D

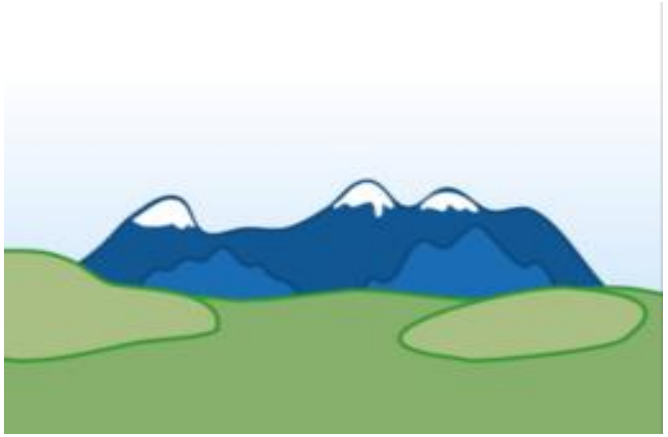


Mapa de profundidade

Principais Algoritmos Pintor

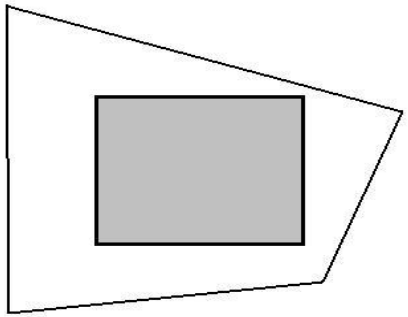
- Os objetos da cena são renderizados de acordo com sua profundidade



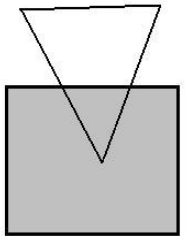


Principais Algoritmos Warnock

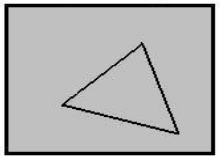
Polígono envolvente



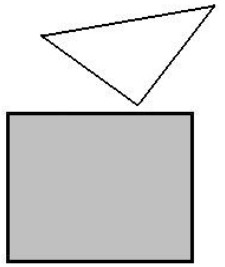
Polígono de intersecção



Polígono contido

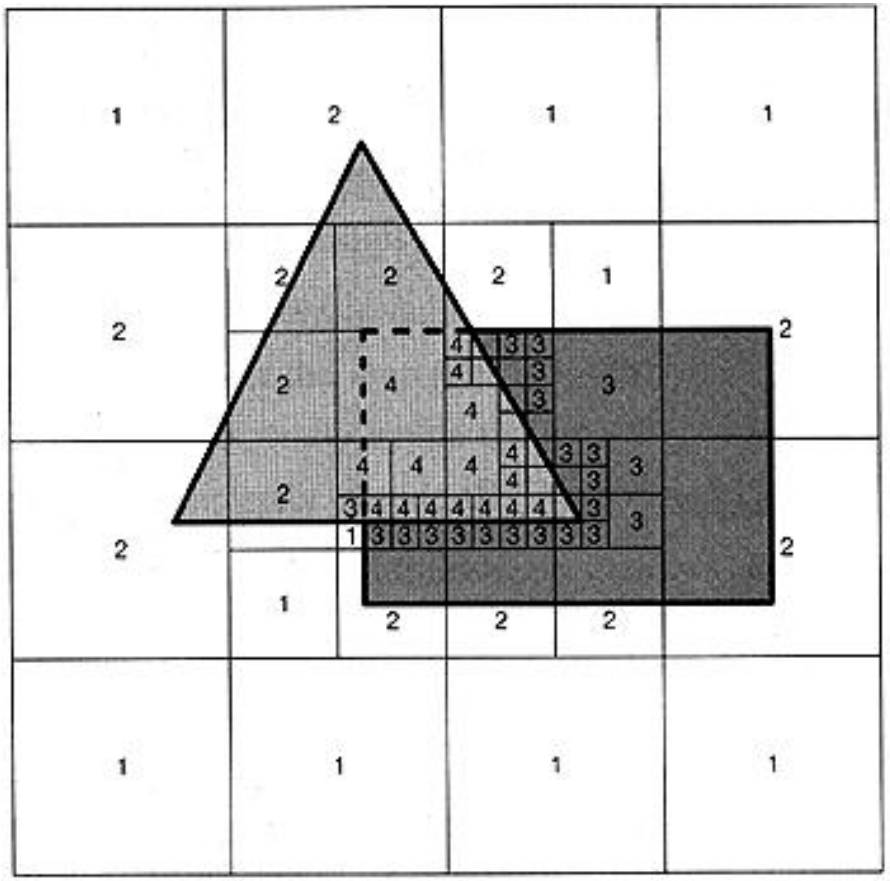


Polígono disjunto

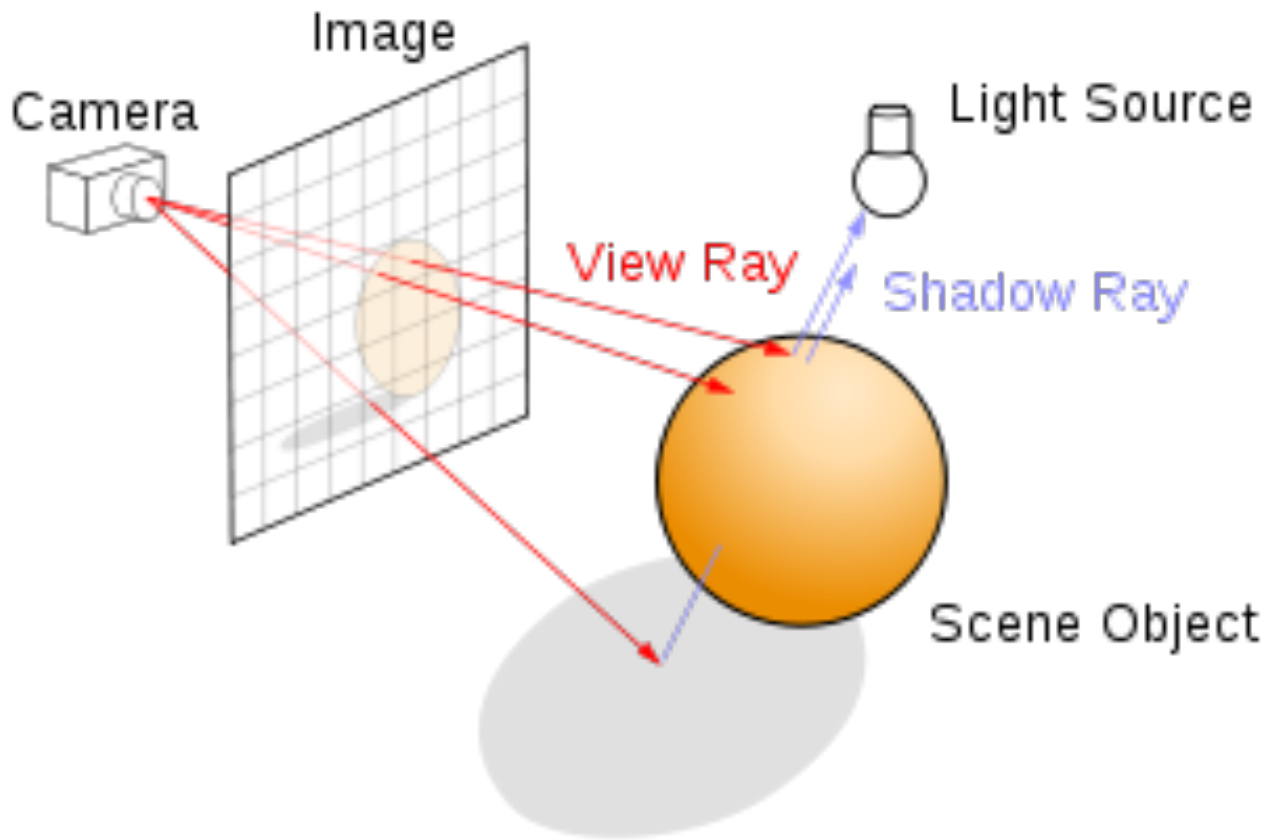


Principais Algoritmos Warnock

- Subdivisão recursiva de área em quadrados iguais.

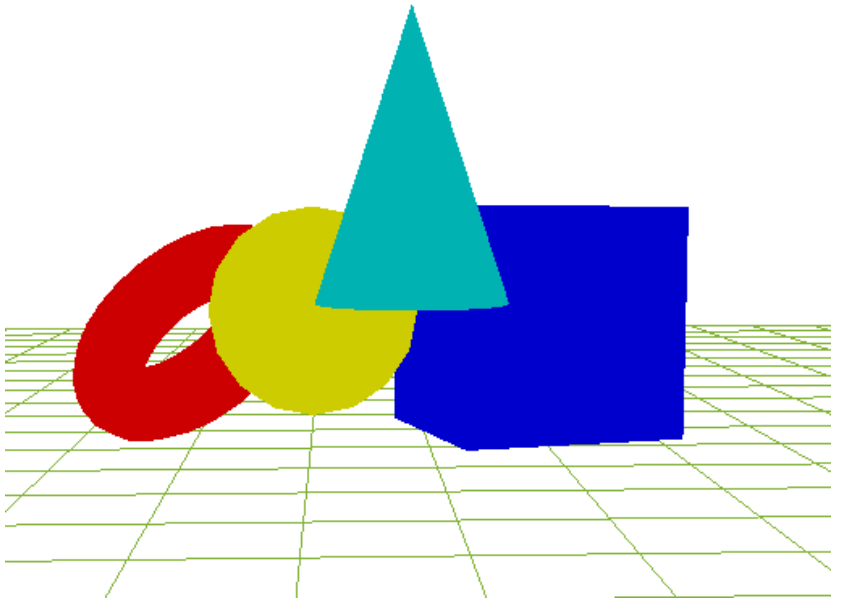


Principais Algoritmos Ray-tracing



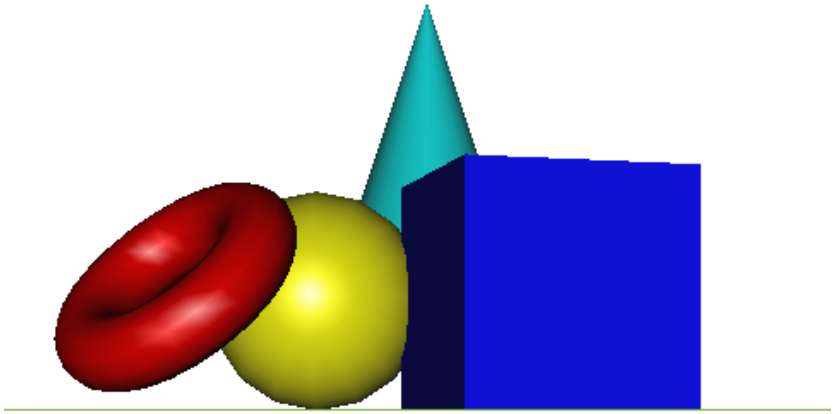
Visibilidade em OpenGL
Teste de profundidade

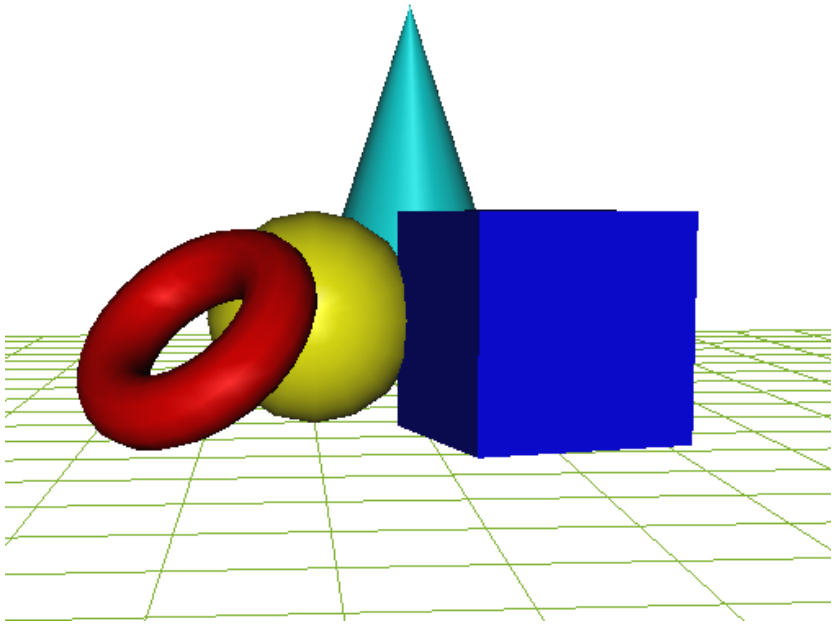
- Projeção em perspectiva, sem ativar **gl_depth_test**



Visibilidade em OpenGL
Teste de profundidade

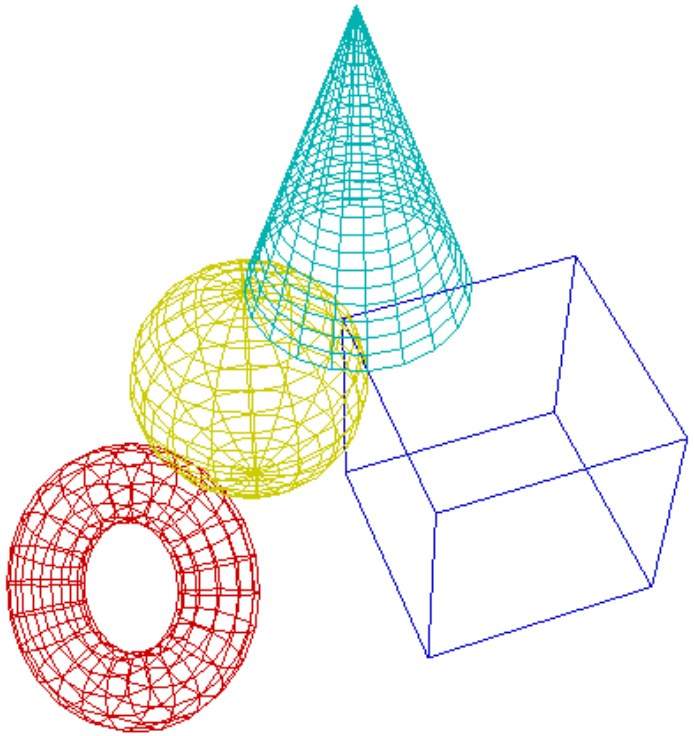
- Projeção em perspectiva, utilizando **gl_depth_test**





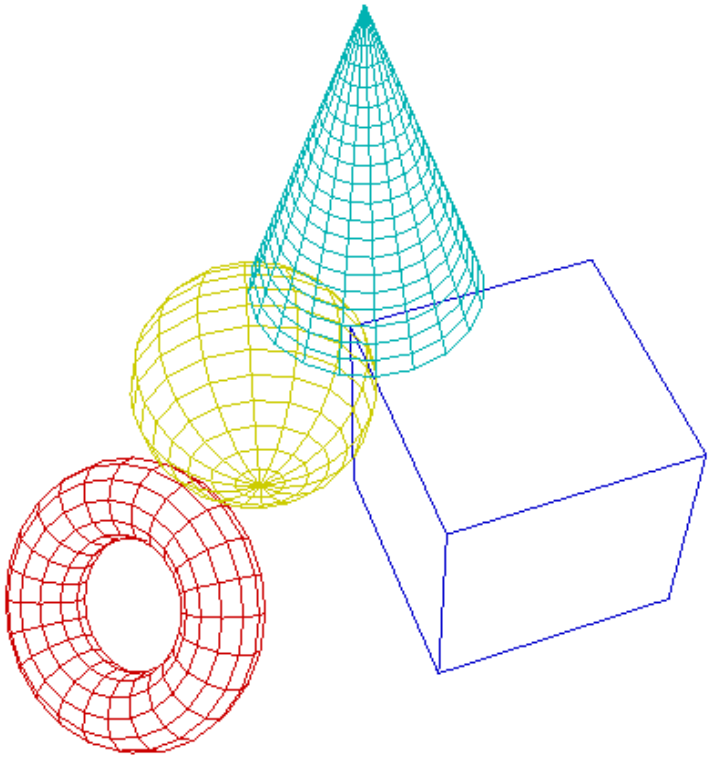
Visibilidade em OpenGL
Back-face Culling

- Rasterização de polígonos com linhas



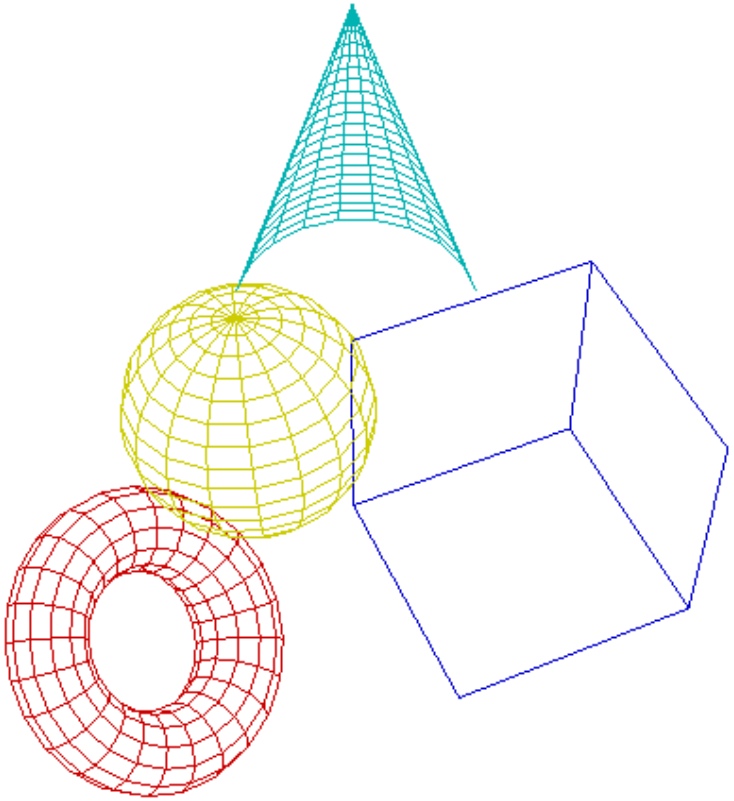
Visibilidade em OpenGL
Back-face Culling

- Rasterização de polígonos com linhas
- eliminação de linhas escondidas – **glEnable(GL_CULL_FACE)**



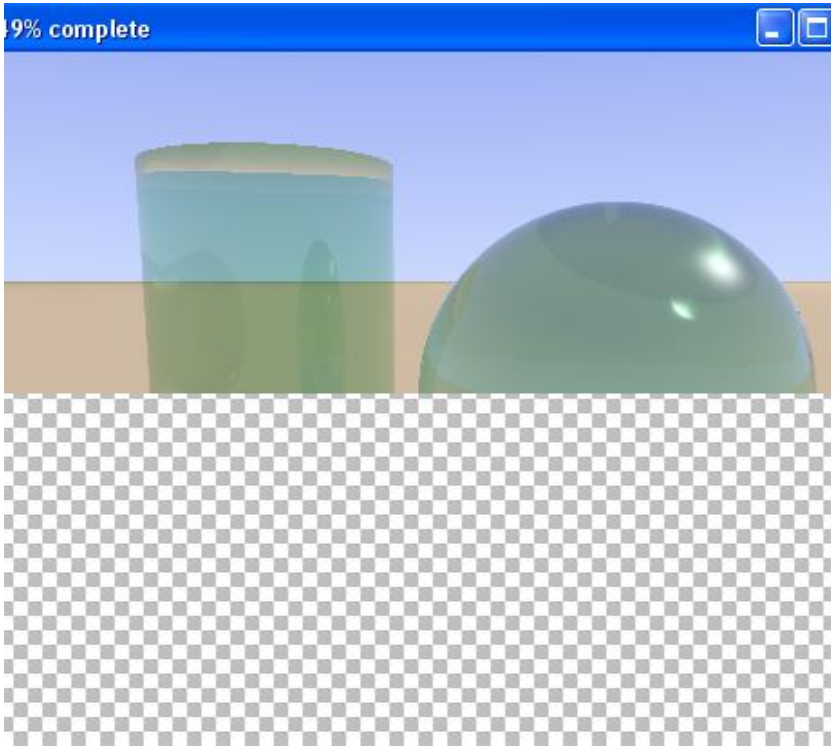
Visibilidade em OpenGL
Back-face Culling

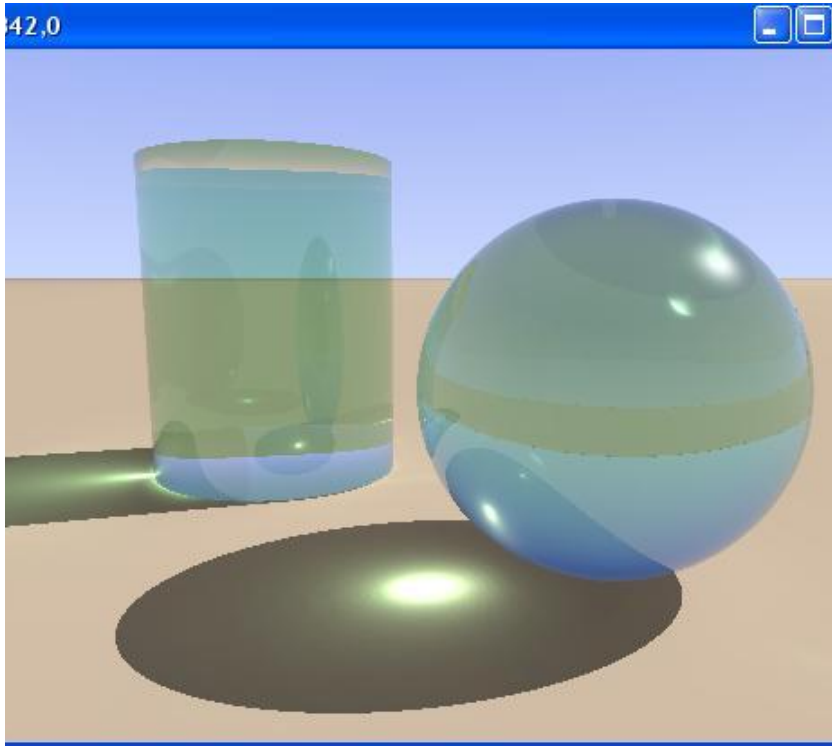
- Rasterização de polígonos com linhas
- eliminação de linhas escondidas – **glEnable(GL_CULL_FACE)**
- Não renderiza as faces frontais – **glCullFace(GL_FRONT)**



Visibilidade no PovRay

- Renderização de cena 3D utilizando PovRay





Conclusão

- Conseguiu-se fazer um estudo dos principais algoritmos de visibilidades, como Z-Buffer, Pintor, Warnock e Ray-Tracing
- Atualmente, os algoritmos mais utilizados são o Z-Buffer e o Ray-tracing, combinados com outros algoritmos existentes
- A visibilidade pode ser aplicada para determinação de objetos em regiões de sombra
- Consegue-se visualizar a diferença da utilização de técnicas de visibilidade em uma cena 3D em OpenGL através de suas funções.
- Ao renderizar uma cena no PovRay, conseguiu-se perceber a geração da imagem pixel a pixel

Obrigado!

- Dúvidas?