Computação Gráfica

Aula 29: Visibilidade

Vicente Helano Feitosa Batista Sobrinho
Faculdade Paraíso do Ceará
Sistemas de Informação
1o. semestre de 2011

Faculdade Paraíso - CE

O problema de visibilidade

- As projeções nos fornecem um indicativo de profundidade
- Muitas vezes, é necessário adicionar um indicativo de oclusão entre objetos

[Azevedo e Conci, 2003]

O problema de visibilidade

- As projeções nos fornecem um indicativo de profundidade
- Muitas vezes, é necessário adicionar um indicativo de oclusão entre objetos

[Azevedo e Conci, 2003]

[Azevedo e Conci, 2003]

Espaços de definição

- São algoritmos de ordenação parcial em Z

- Podem ser definidos em dois espaços:
  1. Espaço da câmera: operações em ponto flutuante
  2. Espaço da imagem: operações em *pixels*

Backface Culling

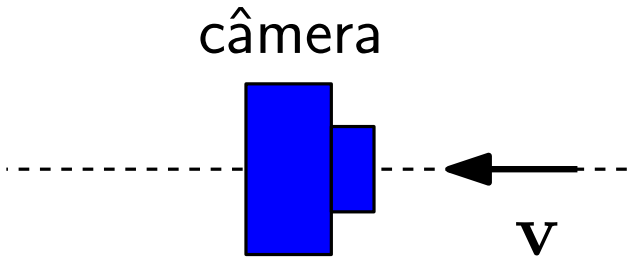
- Algoritmo simples para descarte de faces não visíveis

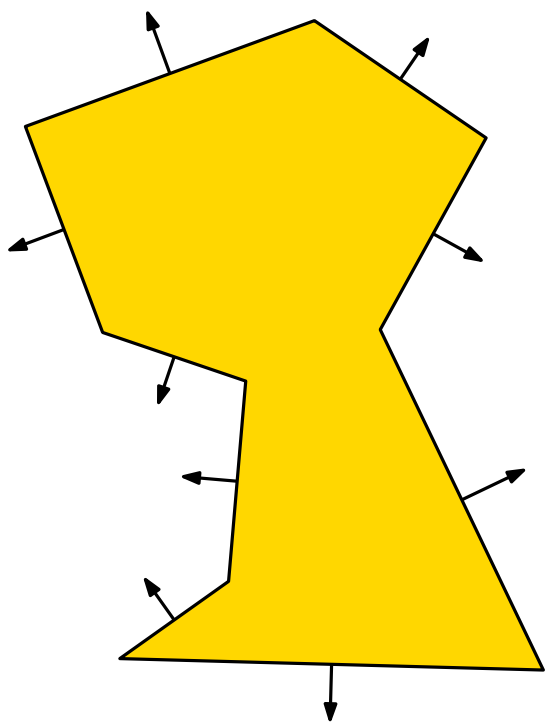
Backface Culling

- Algoritmo simples para descarte de faces não visíveis
- Dada a normal $\mathbf{n}$ de uma face F, temos:
  - Se $\langle \mathbf{n}, \mathbf{v} \rangle > 0$, então F é visível
  - Senão F está escondida, logo podemos descartá-la

Backface Culling

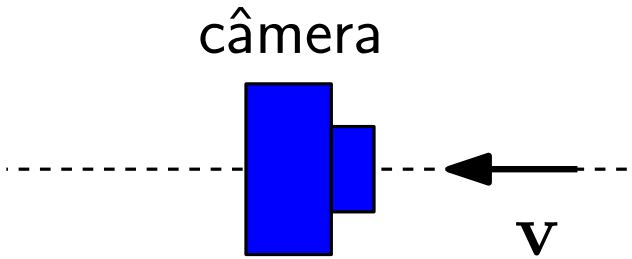
- Algoritmo simples para descarte de faces não visíveis
- Dada a normal $\mathbf{n}$ de uma face F, temos:
  - Se $\langle \mathbf{n}, \mathbf{v} \rangle > 0$, então F é visível
  - Senão F está escondida, logo podemos descartá-la

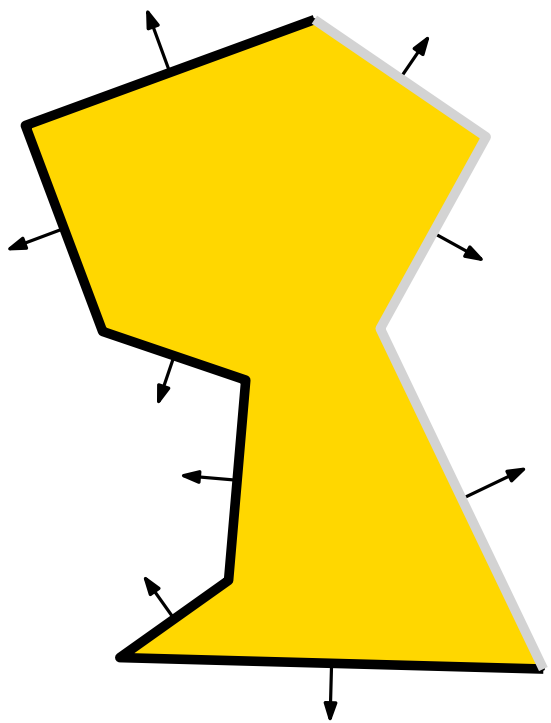




Backface Culling

- Algoritmo simples para descarte de faces não visíveis
- Dada a normal $\mathbf{n}$ de uma face F, temos:
  - Se $\langle \mathbf{n}, \mathbf{v} \rangle > 0$, então F é visível
  - Senão F está escondida, logo podemos descartá-la





Backface Culling

- Algoritmo simples para descarte de faces não visíveis
- Dada a normal $\mathbf{n}$ de uma face F, temos:
  - Se $\langle \mathbf{n}, \mathbf{v} \rangle > 0$, então F é visível
  - Senão F está escondida, logo podemos descartá-la
- Configuração de *backface culling* com OpenGL:
  - Escolhemos o lado a ser eliminado com `glCullFace(side)`, onde `side` pode ser: `GL_FRONT`, `GL_BACK` ou `GL_FRONT_AND_BACK`
  - Habilitamos a eliminação com `glEnable(GL_CULL_FACE)`

Backface Culling

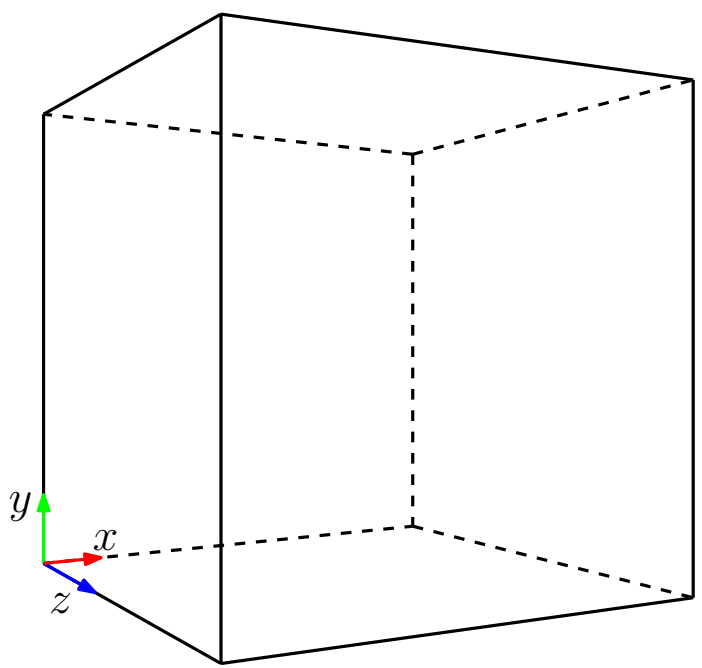
- Algoritmo simples para descarte de faces não visíveis
- Dada a normal $\mathbf{n}$ de uma face F, temos:
  - Se $\langle \mathbf{n}, \mathbf{v} \rangle > 0$, então F é visível
  - Senão F está escondida, logo podemos descartá-la
- Configuração de *backface culling* com OpenGL:
  - Escolhemos o lado a ser eliminado com `glCullFace(side)`, onde `side` pode ser: `GL_FRONT`, `GL_BACK` ou `GL_FRONT_AND_BACK`
  - Habilitamos a eliminação com `glEnable(GL_CULL_FACE)`
- Aproximadamente, metade das faces serão eliminadas

Algoritmo do Pintor

- Simples, definido no espaço de câmera
- Ideia utilizada por artistas: desenhar de trás pra frente
- *Pixels* mais ao fundo são sobrescritos ou combinados por *pixels* mais próximos ao plano de projeção

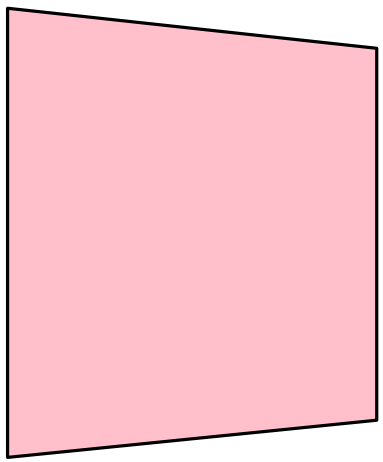
Algoritmo do Pintor

- Simples, definido no espaço de câmera
- Ideia utilizada por artistas: desenhar de trás pra frente
- *Pixels* mais ao fundo são sobrescritos ou combinados por *pixels* mais próximos ao plano de projeção



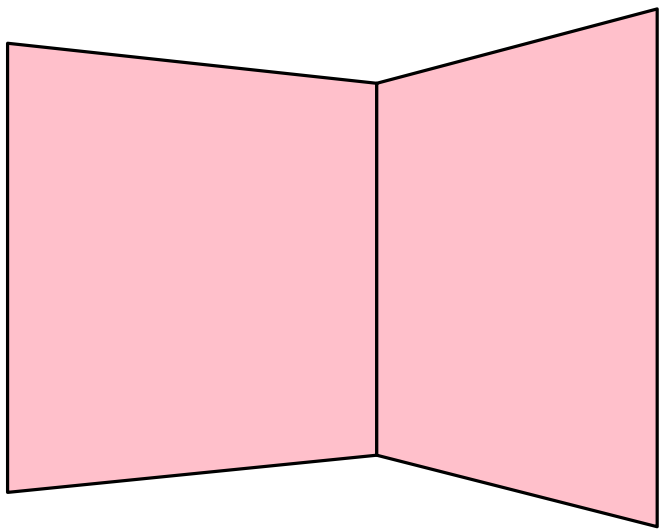
Algoritmo do Pintor

- Simples, definido no espaço de câmera
- Ideia utilizada por artistas: desenhar de trás pra frente
- *Pixels* mais ao fundo são sobrescritos ou combinados por *pixels* mais próximos ao plano de projeção



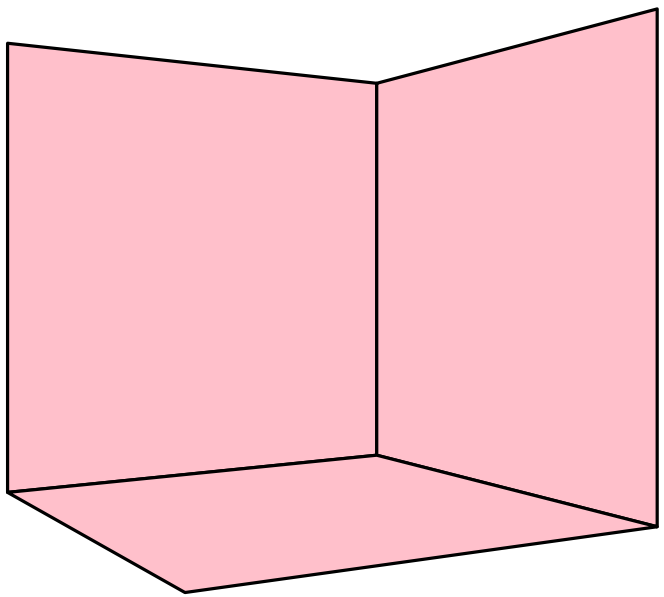
Algoritmo do Pintor

- Simples, definido no espaço de câmera
- Ideia utilizada por artistas: desenhar de trás pra frente
- *Pixels* mais ao fundo são sobrescritos ou combinados por *pixels* mais próximos ao plano de projeção



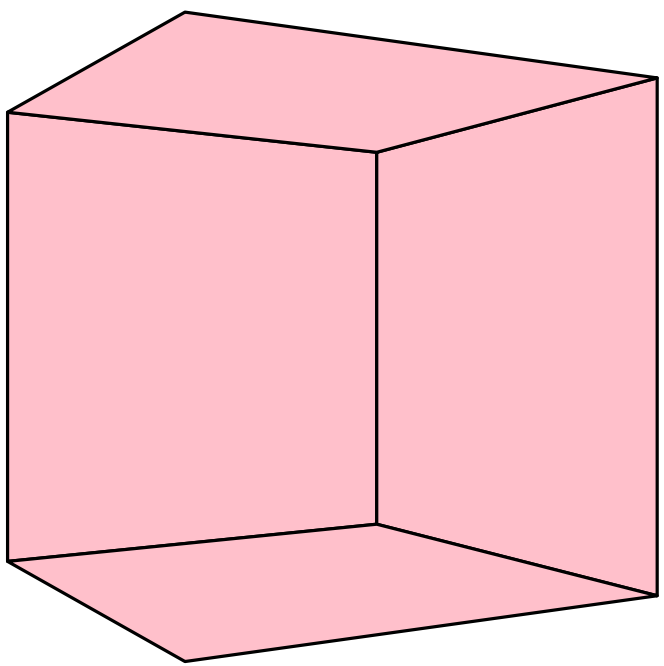
Algoritmo do Pintor

- Simples, definido no espaço de câmera
- Ideia utilizada por artistas: desenhar de trás pra frente
- *Pixels* mais ao fundo são sobrescritos ou combinados por *pixels* mais próximos ao plano de projeção



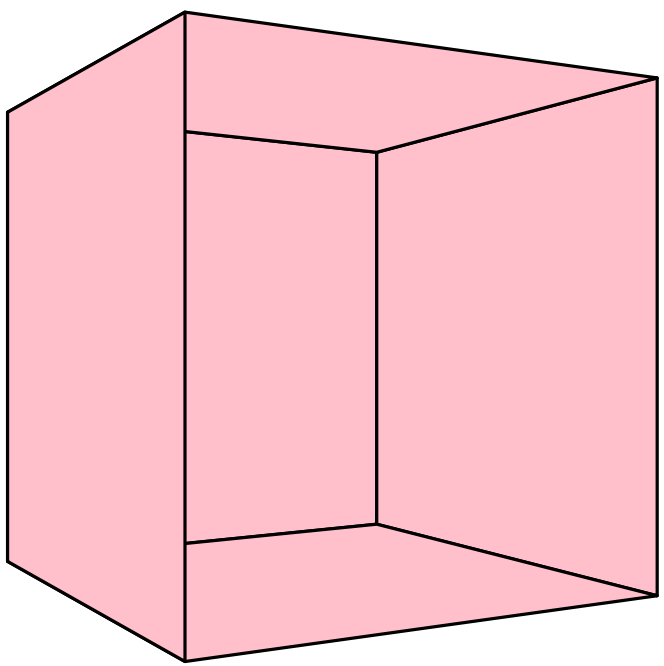
Algoritmo do Pintor

- Simples, definido no espaço de câmera
- Ideia utilizada por artistas: desenhar de trás pra frente
- *Pixels* mais ao fundo são sobrescritos ou combinados por *pixels* mais próximos ao plano de projeção



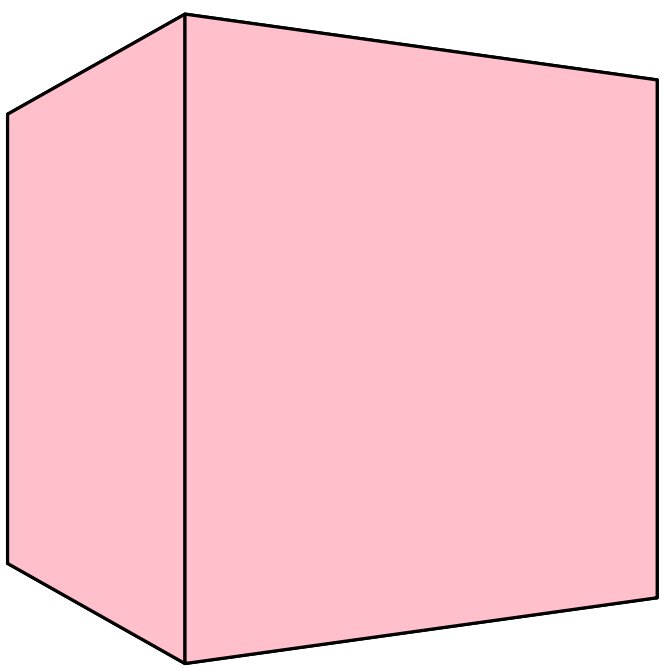
Algoritmo do Pintor

- Simples, definido no espaço de câmera
- Ideia utilizada por artistas: desenhar de trás pra frente
- *Pixels* mais ao fundo são sobrescritos ou combinados por *pixels* mais próximos ao plano de projeção



Algoritmo do Pintor

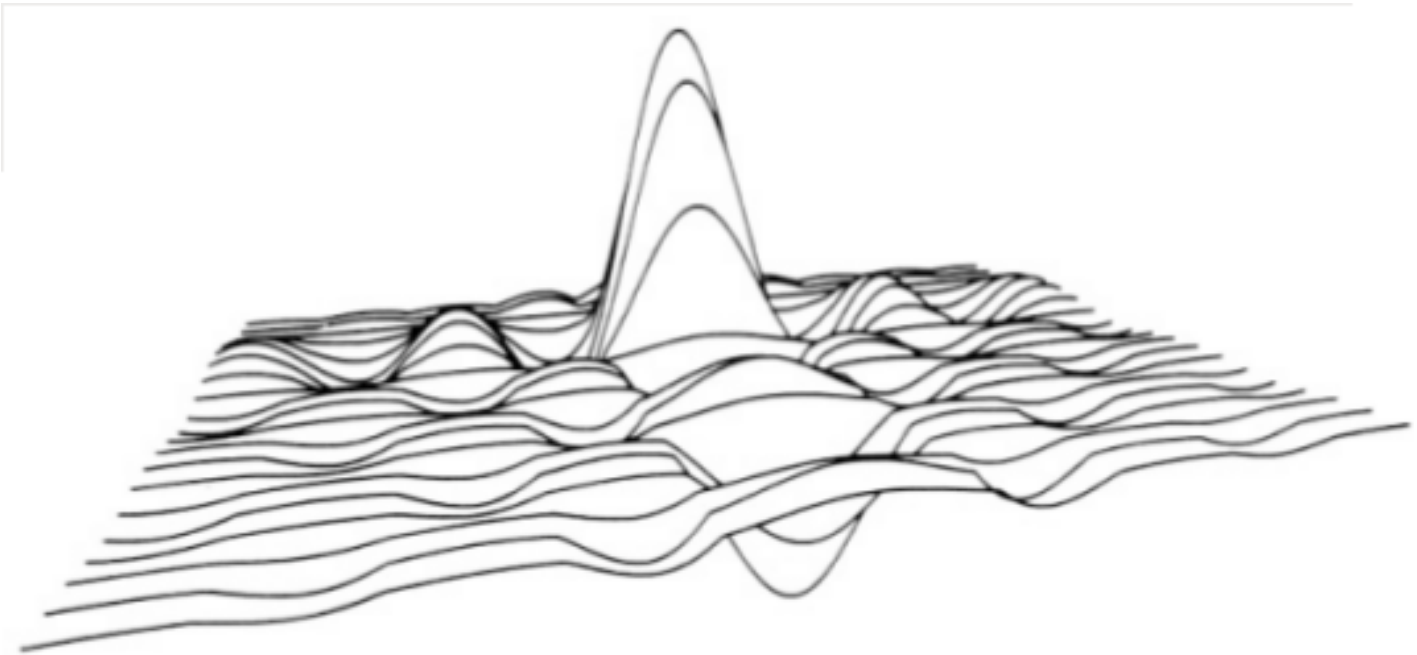
- Simples, definido no espaço de câmera
- Ideia utilizada por artistas: desenhar de trás pra frente
- *Pixels* mais ao fundo são sobrescritos ou combinados por *pixels* mais próximos ao plano de projeção



Algoritmo do Pintor

Vantagens

- Algoritmo simples
- Para objetos ordenados sem inconsistências
- Possibilita emprego de transparência

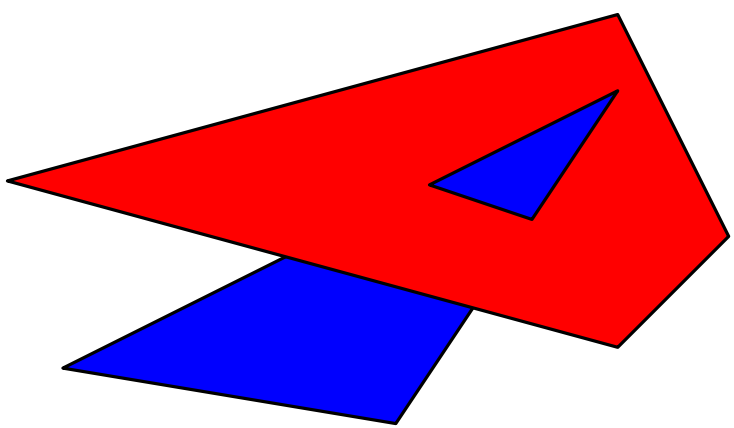


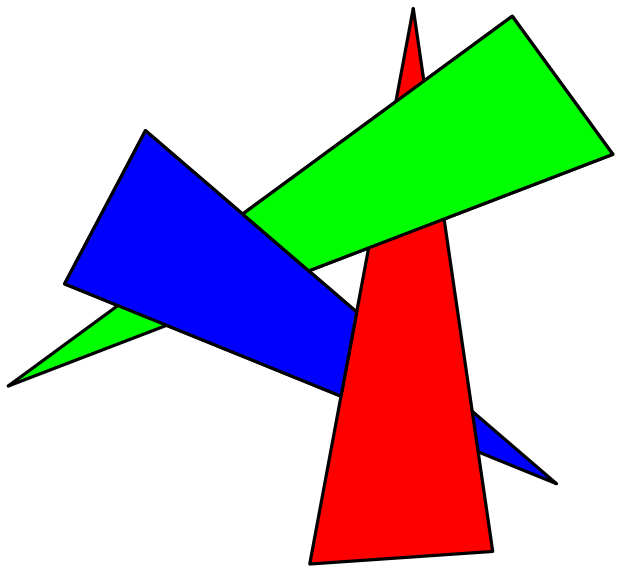
[Foley et al.]

Algoritmo do Pintor

Desvantagens

- Muitos *pixels* oclusos são rasterizados
- Manutenção da ordenação dos polígonos cenas com objetos móveis diminue drasticamente o desempenho
- Ciclos e penetração entre polígonos são um problema
  - Solução: recortar interseções





Z-Buffer

- É implementado no espaço da imagem
- Ideia básica: guarda-se a profundidade do *pixel* mais próximo ao plano de projeção
- Dois *buffers* são mantidos: *buffer de cor* (rascunho) e *buffer de profundidade* (*z-buffer*)

Z-Buffer

Algoritmo:

```
Para cada polígono P da cena
  Para cada pixel (x, y) de um polígono P
    computar z_depth na posição x, y
    se z_depth < z_buffer (x, y) então:
    defina_pixel (x, y, color)
    troque o valor : z_buffer (x, y) = z_depth
```

Inicializações:

- Inicializar o *buffer* de cor com a cor de fundo
- Inicializar o *z-buffer* com o valor da máxima profundidade

Z-Buffer

Vantagens:

- Simples implementação
- Objetos podem ser desenhados em qualquer ordem

Z-Buffer

Vantagens:

- Simples implementação
- Objetos podem ser desenhados em qualquer ordem

Maior desvantagem:

- Dificulta o uso de transparência e anti-serrilhado

Z-Buffer

Vantagens:

- Simples implementação
- Objetos podem ser desenhados em qualquer ordem

Maior desvantagem:

- Dificulta o uso de transparência e anti-serrilhado

Em OpenGL:

- Inicialização do GLUT: `glutInitDisplayMode ( ... | GLUT_DEPTH);`
- Inicialização antes de desenhar: `glClear( ... | GL_DEPTH_BUFFER_BIT)`
- Habilitar: `glEnable (GL_DEPTH_TEST);`