Renderização não fotorealista

INF1339 – Computação Gráfica Tridimensional

Waldemar Celes
celes@inf.puc-rio.br – sala 505 RDC

Tecgraf, DI/PUC-Rio

7 de Outubro de 2015

Non-photorealistic rendering

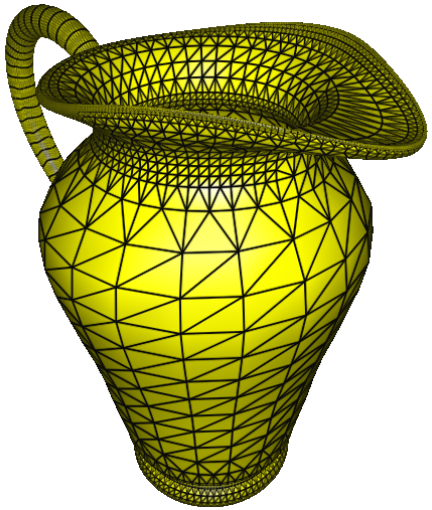
- Ilustração técnica
- *Cartoon*
- Arte

Demo from ATI

www.ign.com

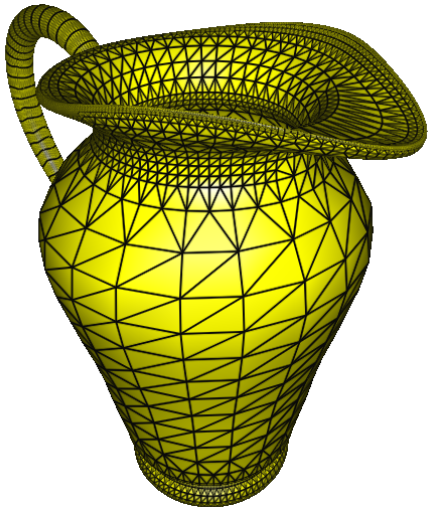
Renderização de linhas

Renderização da malha



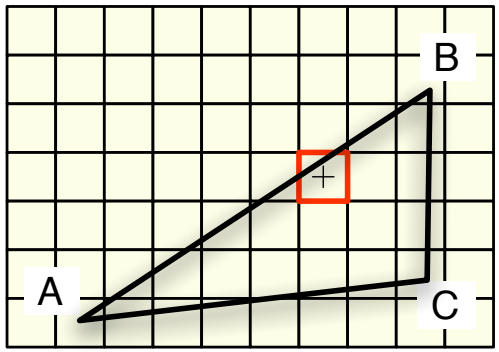
Renderização de linhas

Renderização da malha



Problema: conflito no valor de z

- Rasterização de linhas $\neq$ rasterização de polígonos



Renderização de linhas

Uso de *polygon offset*

1. Desenha preenchimento com *offset* positivo

```
glPolygonOffset(1.0, 1.0);
glEnable(GL_POLYGON_OFFSET_FILL);
```

2. Desenha linhas sem *offset*

Renderização de linhas

Uso de *polygon offset*

1. Desenha preenchimento com *offset* positivo

```
glPolygonOffset(1.0, 1.0);
glEnable(GL_POLYGON_OFFSET_FILL);
```

2. Desenha linhas sem *offset*

- Simples
- Exige duas passadas
- Permite *highlight*
- Pode falhar

Renderização de linhas

Uso do *stencil buffer*

- Para cada polígono
  1. Renderiza linha marcando *stencil*

```
glStencilFunc(GL_ALWAYS, 1, 1);
glStencilOp(GL_KEEP, GL_KEEP, GL_REPLACE);
glEnable(GL_STENCIL_TEST);
```

  2. Renderiza preenchimento sem fragmentos marcados no *stencil*

```
glStencilFunc(GL_NOTEQUAL, 1, 1);
glStencilOp(GL_KEEP, GL_KEEP, GL_KEEP);
glEnable(GL_STENCIL_TEST);
```

Renderização de linhas

Uso do *stencil buffer*

- Para cada polígono
  1. Renderiza linha marcando *stencil*

```
glStencilFunc(GL_ALWAYS, 1, 1);
glStencilOp(GL_KEEP, GL_KEEP, GL_REPLACE);
glEnable(GL_STENCIL_TEST);
```

  2. Renderiza preenchimento sem fragmentos marcados no *stencil*

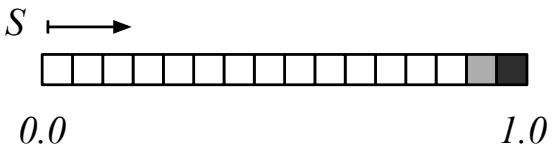
```
glStencilFunc(GL_NOTEQUAL, 1, 1);
glStencilOp(GL_KEEP, GL_KEEP, GL_KEEP);
glEnable(GL_STENCIL_TEST);
```

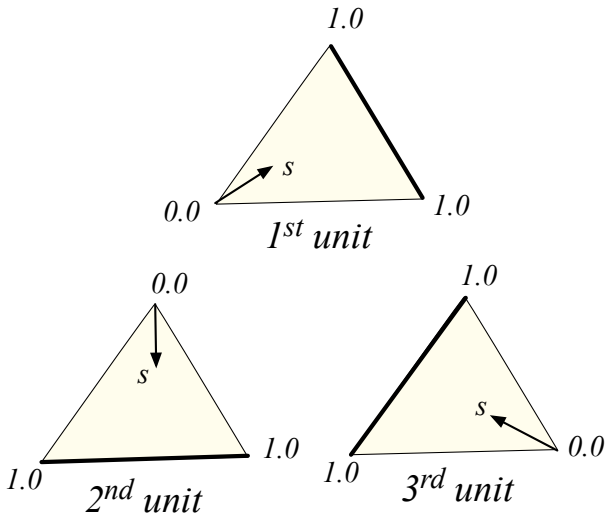
- Baixo desempenho
  - Deve-se ainda restaurar *stencil* após renderizar cada polígono

Renderização de linhas

Uso de textura 1D

- Com multi-textura

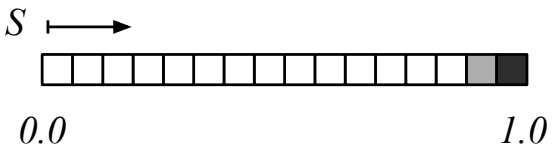


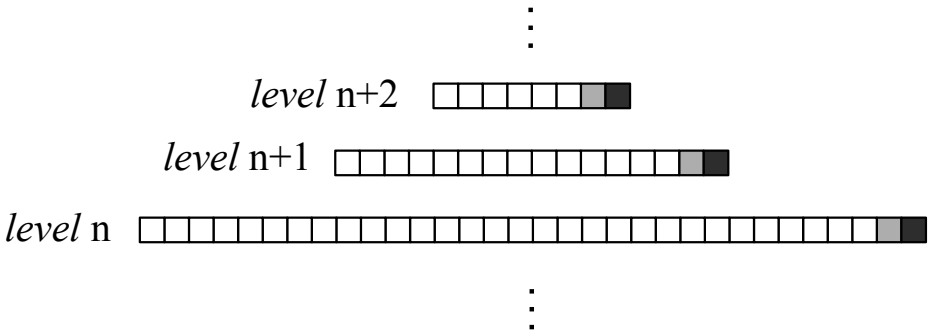


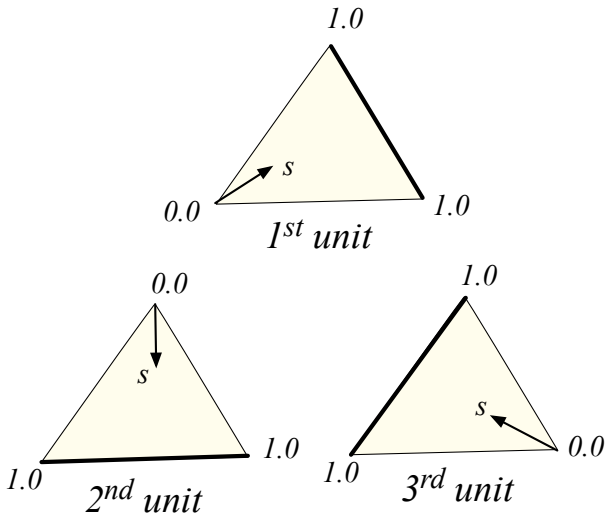
Renderização de linhas

Uso de textura 1D

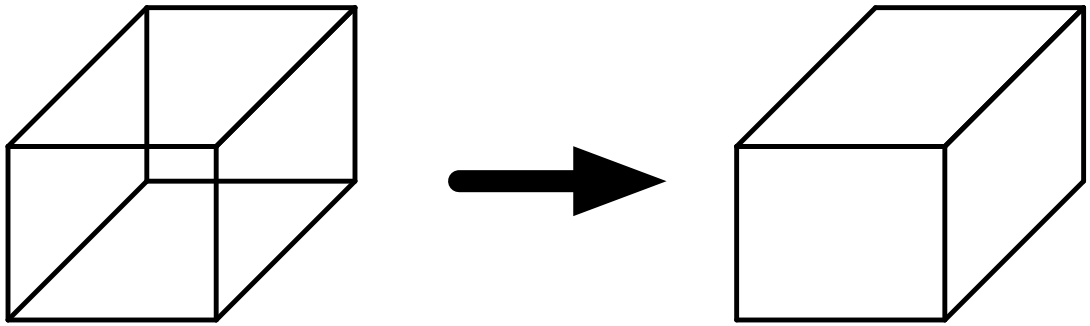
- Com multi-textura



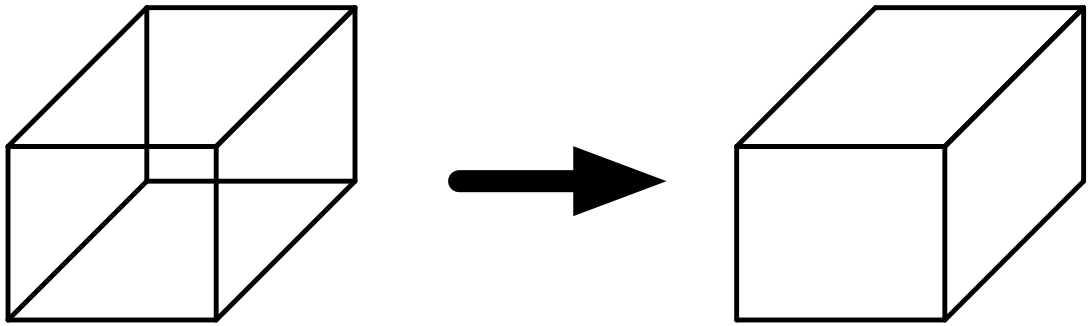




Hidden line



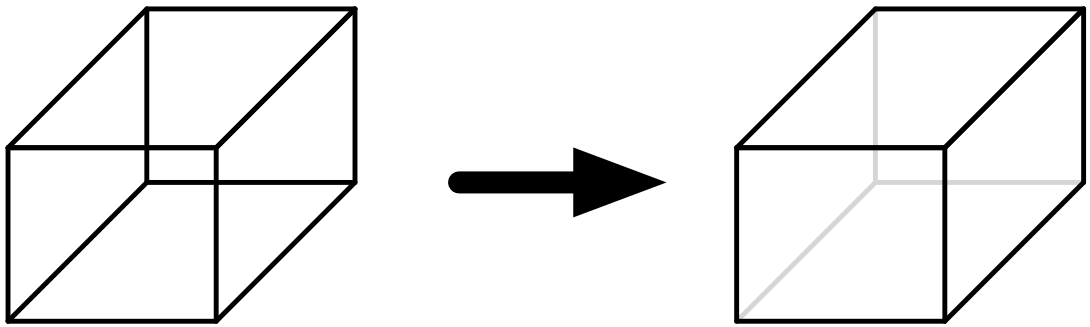
Hidden line



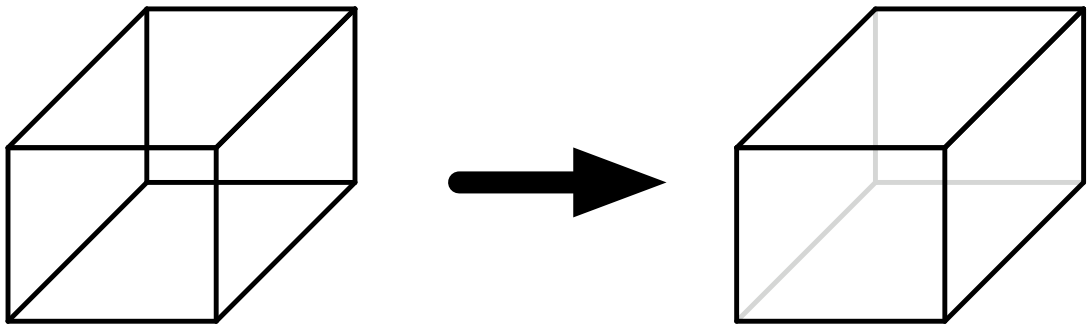
Algoritmo

1. Marca *z-buffer* com preenchimento
   - Com *polygon offset*
   - Sem afetar *color buffer*
2. Renderiza linhas
   - Com *z-test* ligado

Hidden line



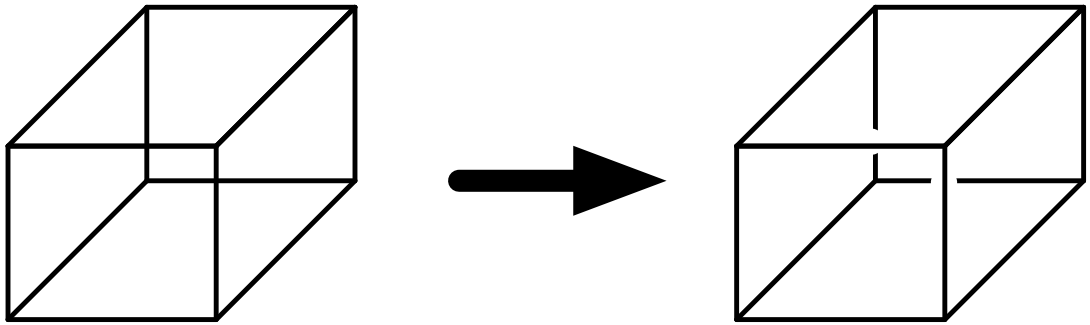
Hidden line



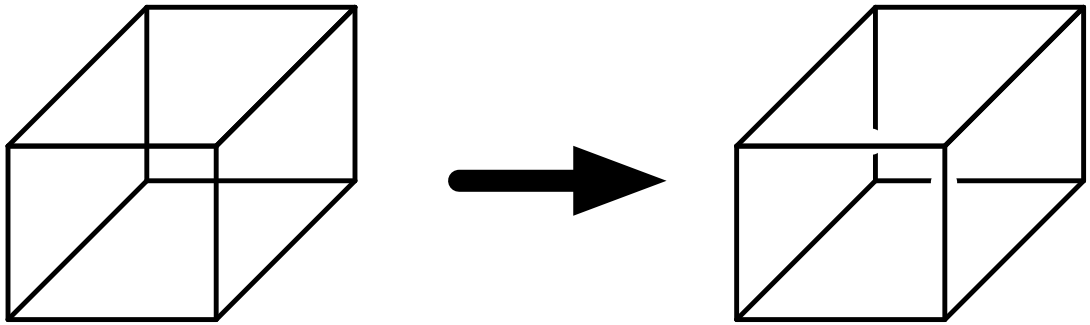
Algoritmo

1. Marca *z-buffer* com preenchimento
   - Com *polygon offset*
   - Sem afetar *color buffer*
2. Renderiza linhas visíveis
   - Com *z-test* ligado
3. Renderiza linhas não visíveis
   - Com *z-test not equal*

Haloing



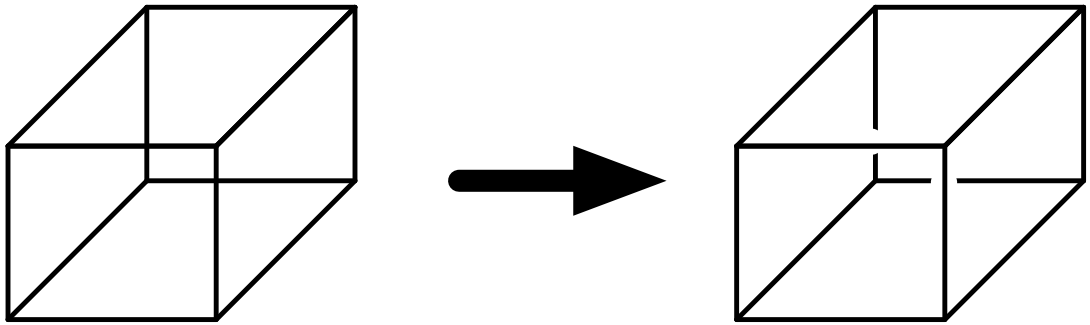
Haloing



Algoritmo

1. Marca *z-buffer* com linha grossa
   - Com *polygon offset*
   - Sem afetar *color buffer*
2. Renderiza linhas
   - Com *z-test* ligado

Haloing



Algoritmo

1. Marca *z-buffer* com linha grossa
   - Com *polygon offset*
   - Sem afetar *color buffer*
2. Renderiza linhas
   - Com *z-test* ligado

Linhas mais grossas podem ocultar linhas visíveis perto dos corners

- Uma solução é encurtar as linhas grossas

Toon shading

Características

- Iluminação difusa com poucos tons
- Iluminação especular, em geral, com um tom
- Arestas desenhadas em preto
- Silhueta desenhada em preto

Toon shading

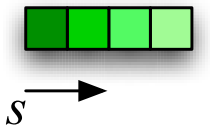
Iluminação difusa com poucos tons

Toon shading

Iluminação difusa com poucos tons

Uso de textura 1D

- Filtro de textura para *nearest*



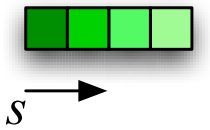
$$s = f(\mathbf{l} \cdot \mathbf{n})$$

Toon shading

Iluminação difusa com poucos tons

Uso de textura 1D

- Filtro de textura para *nearest*



$$s = f(\mathbf{l} \cdot \mathbf{n})$$

Para evitar *aliasing*

- Filtro de textura para *linear*

Toon shading

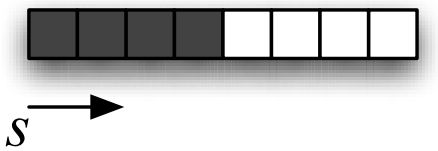
Iluminação especular com um tom

Toon shading

Iluminação especular com um tom

Uso de textura 1D

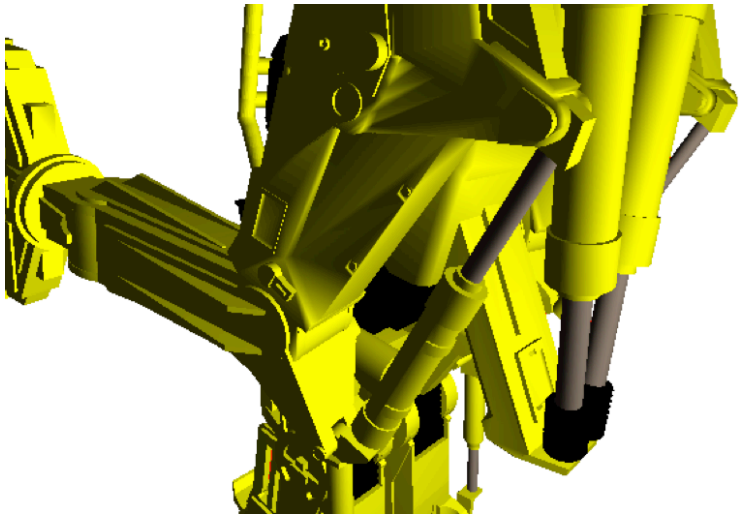
- Filtro de textura para *linear*

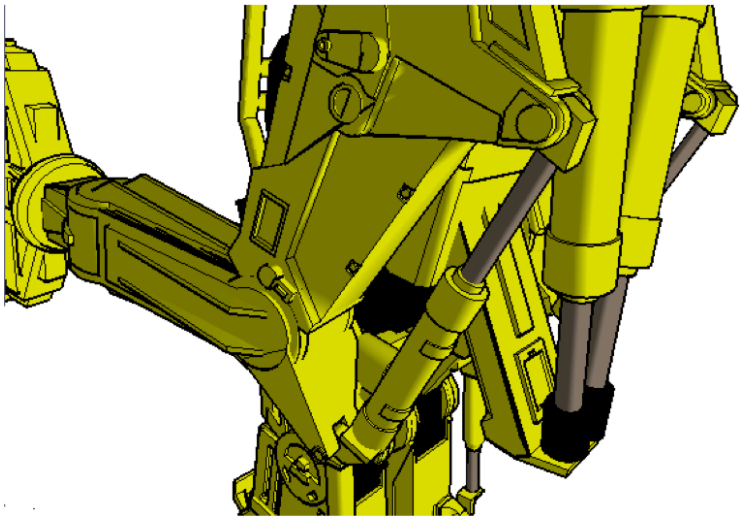


$$s = f(\mathbf{h} \cdot \mathbf{n})$$

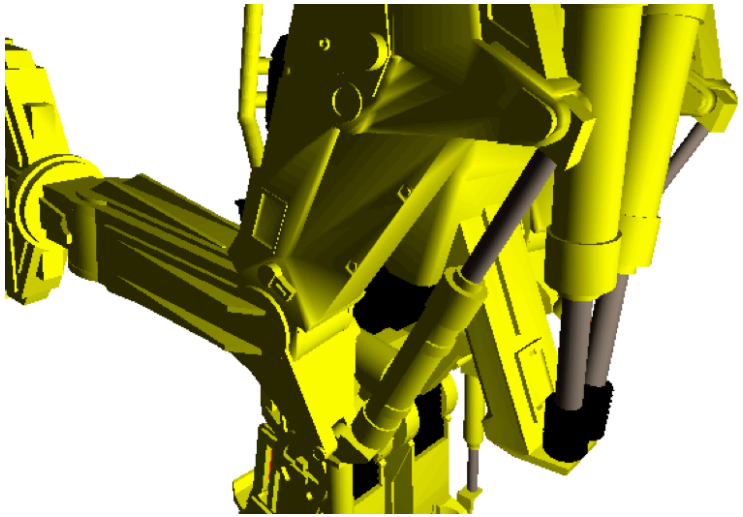
Toon shading

Renderização das arestas em preto





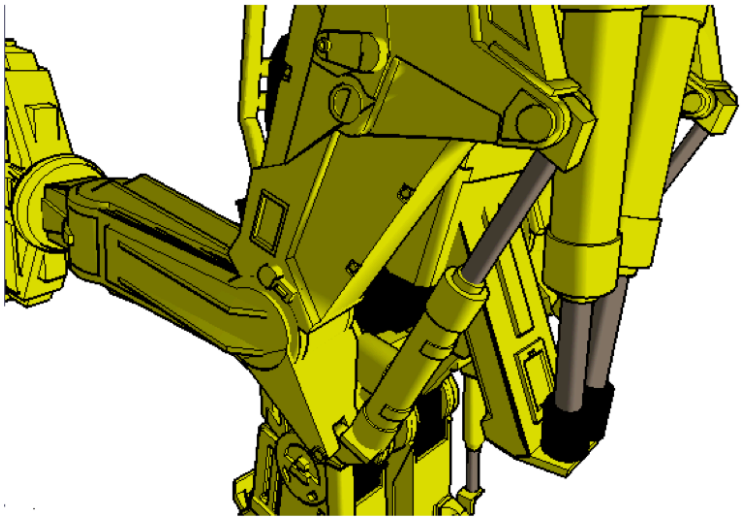
Toon shading



Renderização das arestas em preto

- Identificar arestas em pré-processamento
  - Para cada aresta
    (adjacente a dois triângulos)

$$\cos^{-1}(\mathbf{n}_0 \cdot \mathbf{n}_1) > 60^o$$



Toon shading

Renderização da silhueta

Toon shading

Renderização da silhueta

Métodos

- Processamento no espaço do objeto
  - Para cada aresta, em cada quadro
    - Arestas que tem triângulos adjacentes: *front* & *back*

Toon shading

Renderização da silhueta

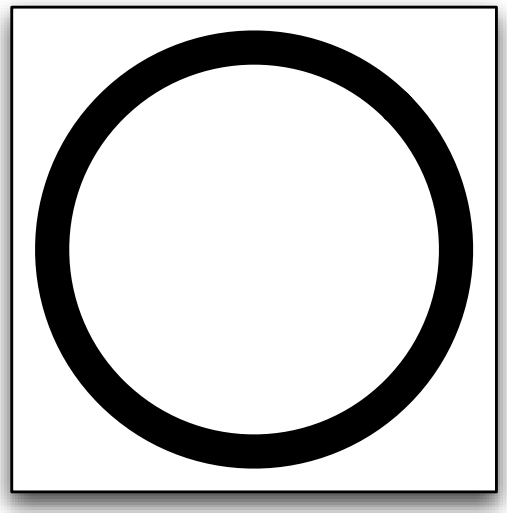
Métodos

- Processamento no espaço do objeto
  - Para cada aresta, em cada quadro
    - Arestas que tem triângulos adjacentes: *front* & *back*
- Renderização direta com textura
- Renderização direta com *back faces*

Toon shading

Renderização direta da silhueta com textura

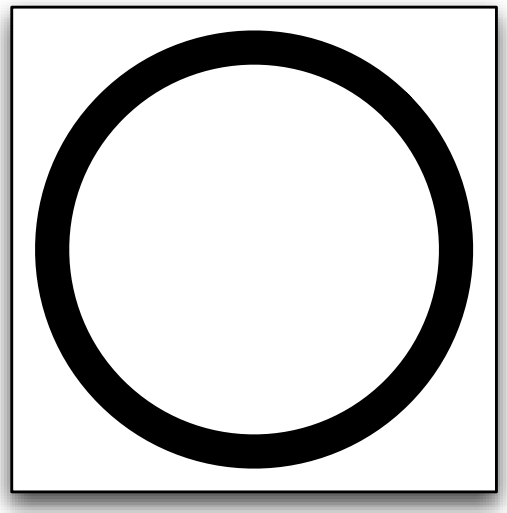
- Uso de *sphere mapping*



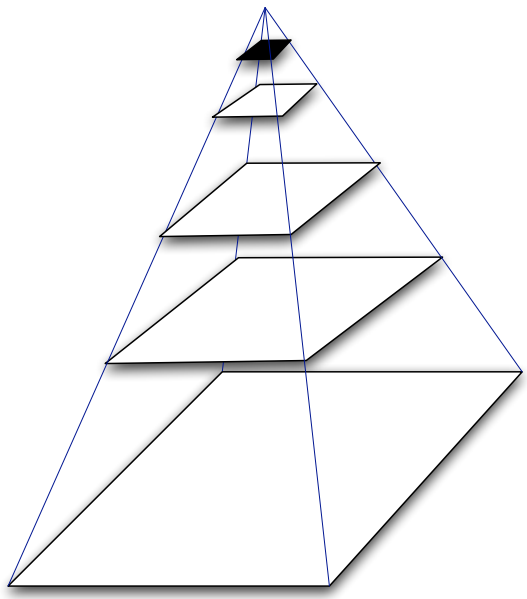
Toon shading

Renderização direta da silhueta com textura

- Uso de *sphere mapping*



- Uso de *mipmapping*



Toon shading

Renderização direta da silhueta com *back face*

1. Desenha *front faces*
2. Desenha silhueta

Toon shading

Renderização direta da silhueta com *back face*

1. Desenha *front faces*
2. Desenha silhueta
   - **Alternativa 1**: Desenha arestas das *back faces* em preto
     - Com uso de *polygon offset*
     - Pode variar espessura

Toon shading

Renderização direta da silhueta com *back face*

1. Desenha *front faces*
2. Desenha silhueta
   - **Alternativa 1**: Desenha arestas das *back faces* em preto
     - Com uso de *polygon offset*
     - Pode variar espessura
   - **Alternativa 2**: Desenha *back faces* em preto
     - Valor do *offset* controla espessura da silhueta