Visibilidade

Prof. Antonio L. Apolinário Junior
Estagiária Docente: Rafaela Alcantara

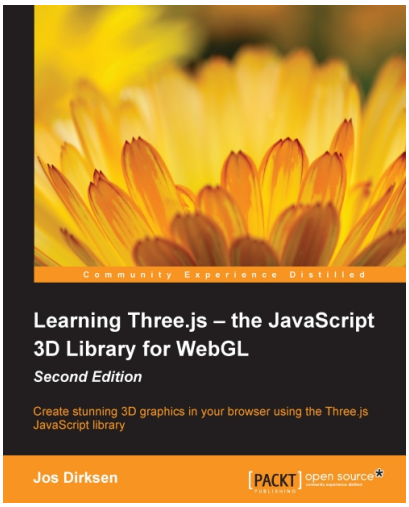
UFBA/IM/DCC/BCC - 2018.1

Roteiro

- Algoritmos de Visibilidade
- Aplicações utilizando Three.JS/WebGL

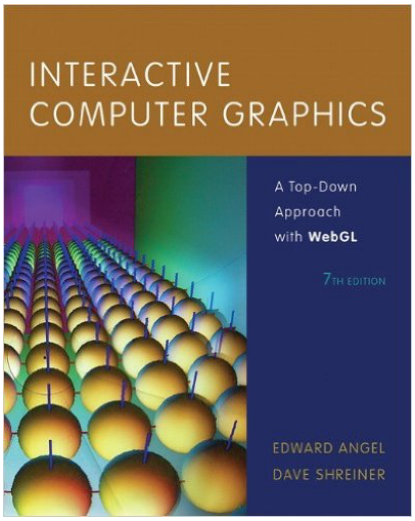
Leitura de referencia

- Capitulo 3
  Learning Three.js: The JavaScript 3D Library for WebGL
  Jos Dirksen
  2nd Edition.
  Packt Publishing - 2015.



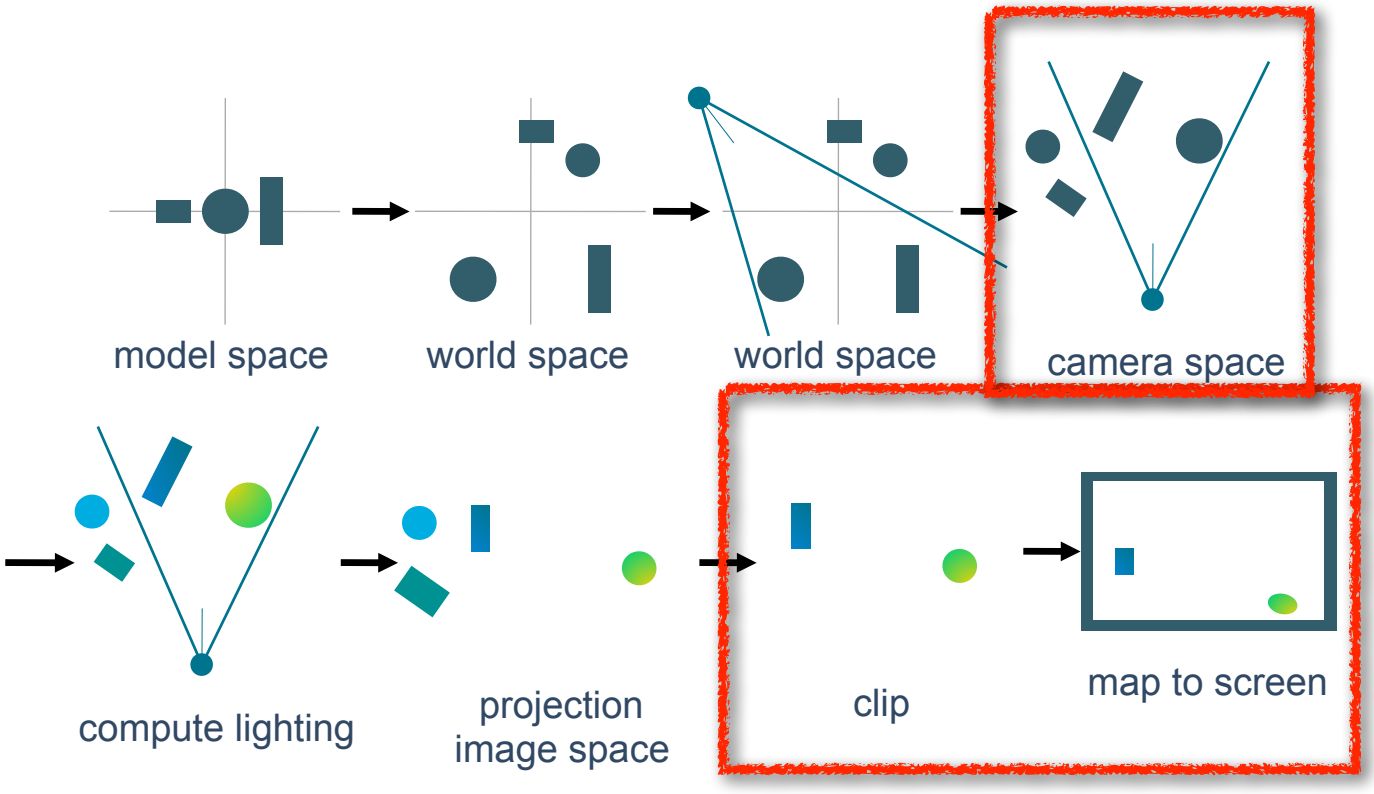
Leitura de referencia

- Capitulo 4
  Interactive Computer Graphics - A top-down approach with OpenGL
  7th Edition
  Angel, Edward.
  Addison-Wesley. 2014.



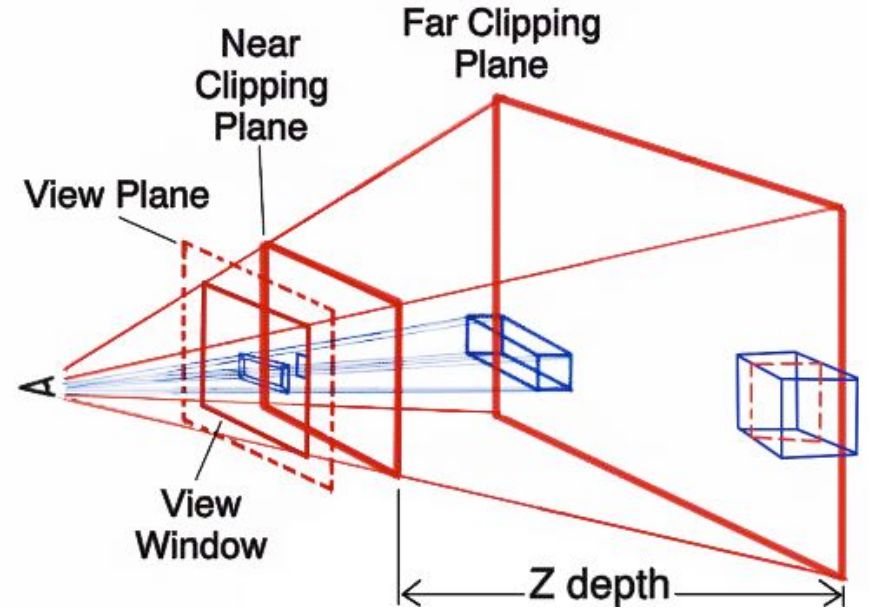
Algoritmos de Visibilidade

Pipeline Gráfico



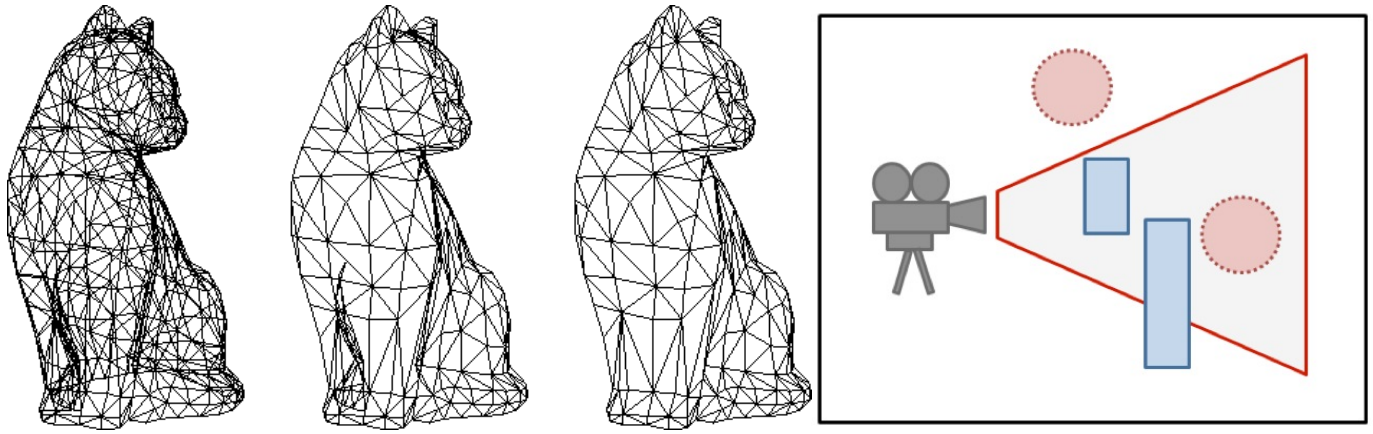
Visibilidade

- Sistema de Visualização resolve quais objetos são potencialmente visíveis...



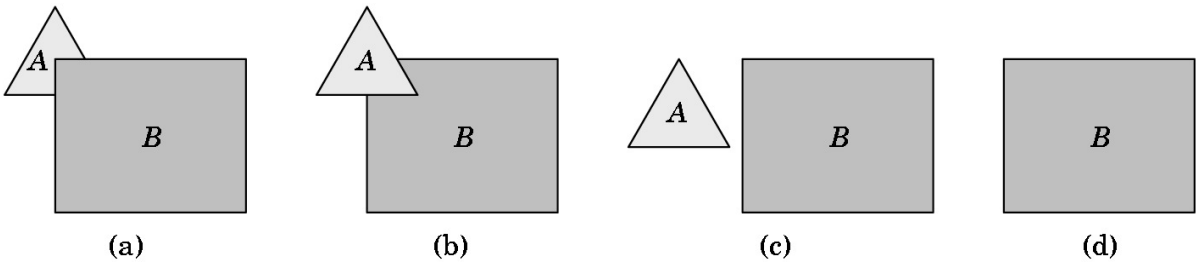
Visibilidade

- … mas quando consideramos a visibilidade geral da cena pode haver oclusão:
  - pelo próprio objeto
  - entre objetos



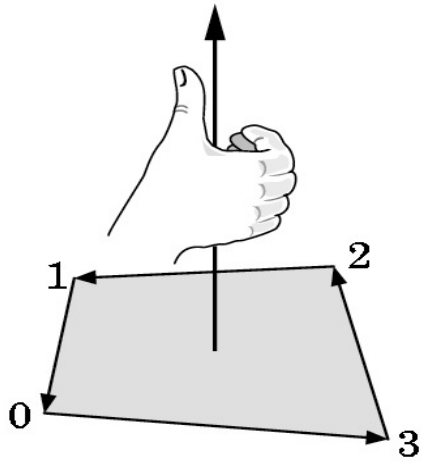
Visibilidade

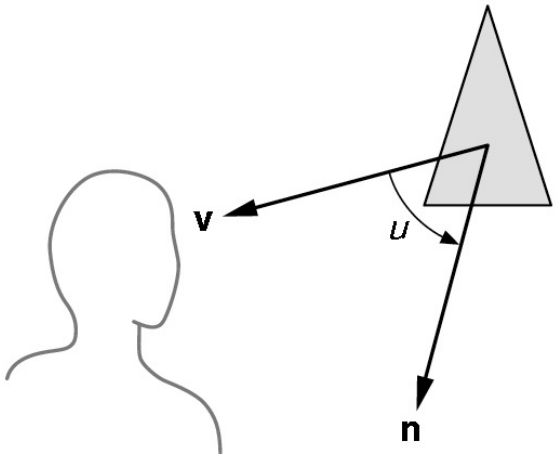
- Objetos que passam pela etapa de recorte (clipping) do *pipeline* podem não formar imagem ao final do processo
  - Objetos potencialmente visíveis
- Deve-se levar em conta a oclusão entre objetos
  - Oclusão por outros objetos da cena
  - “auto oclusão”



Visibilidade

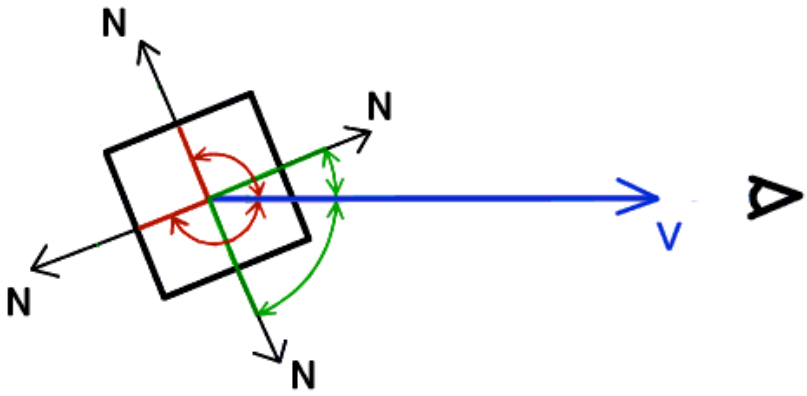
- Auto oclusão pode ser resolvida identificando-se a orientação das faces em relação ao observador
  - *Backface Culling*





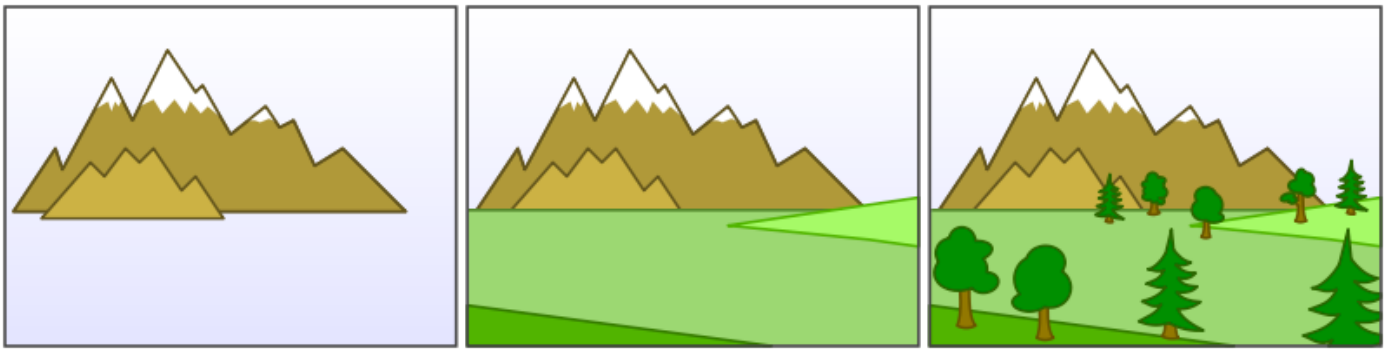
Visibilidade

- *Backface Culling*
  - Para cada face de cada objeto
    - computar sua visibilidade



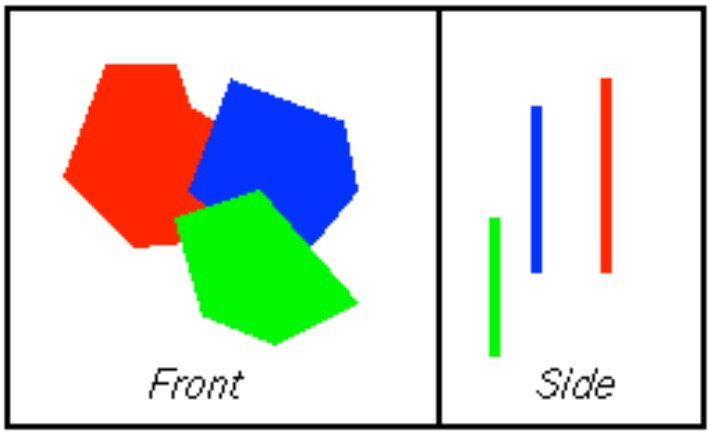
Algoritmos de visibilidade

- Oclusão entre objetos
  - Algoritmo do Pintor
    - Pintor utiliza uma ordenação para pintar
    - Primeiro fundo
    - Último objetos mais próximos



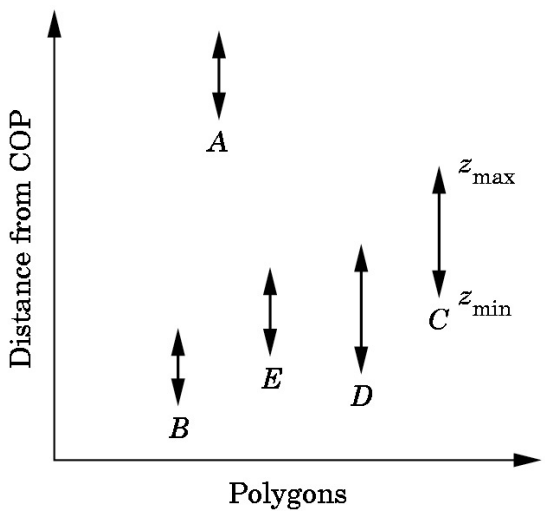
Algoritmos de visibilidade

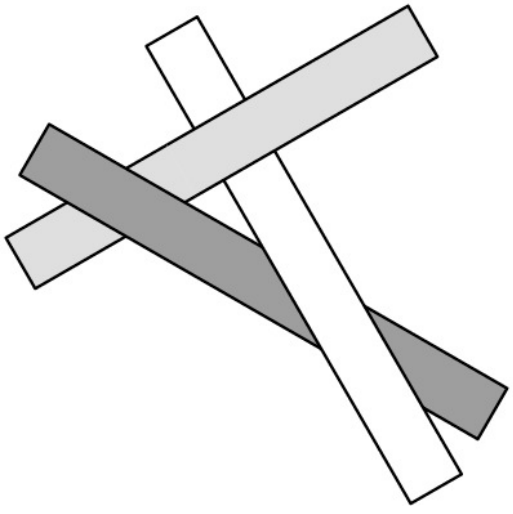
- Algoritmo do Pintor
  - Solução simples para casos simples



Algoritmos de visibilidade

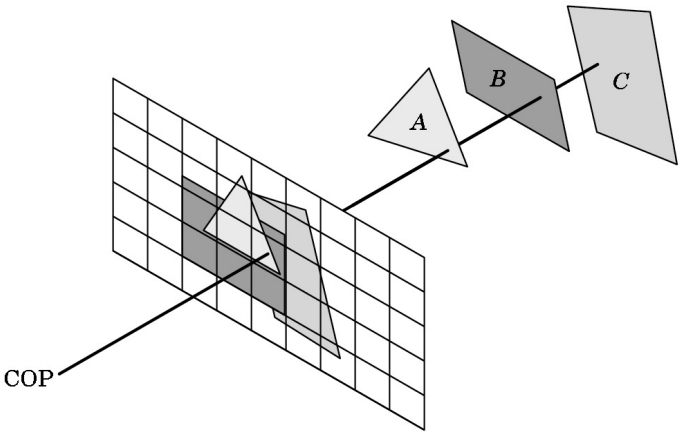
- Algoritmo do Pintor
  - Nem sempre é possível estabelecer um critério único de ordenação para todos os objetos

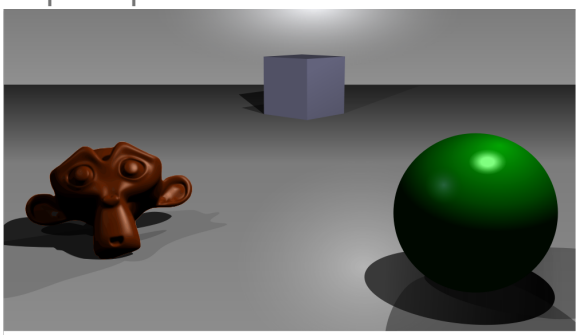




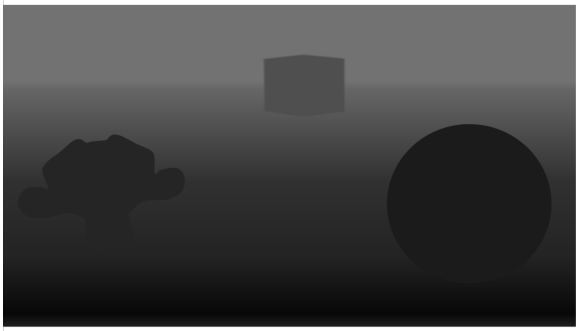
Algoritmos de visibilidade

- Visibilidade pode ser computada por pixel
  - Z-buffer ⇒ “copia” do frame buffer com informações sobre a profundidade (z)





A simple three-dimensional scene

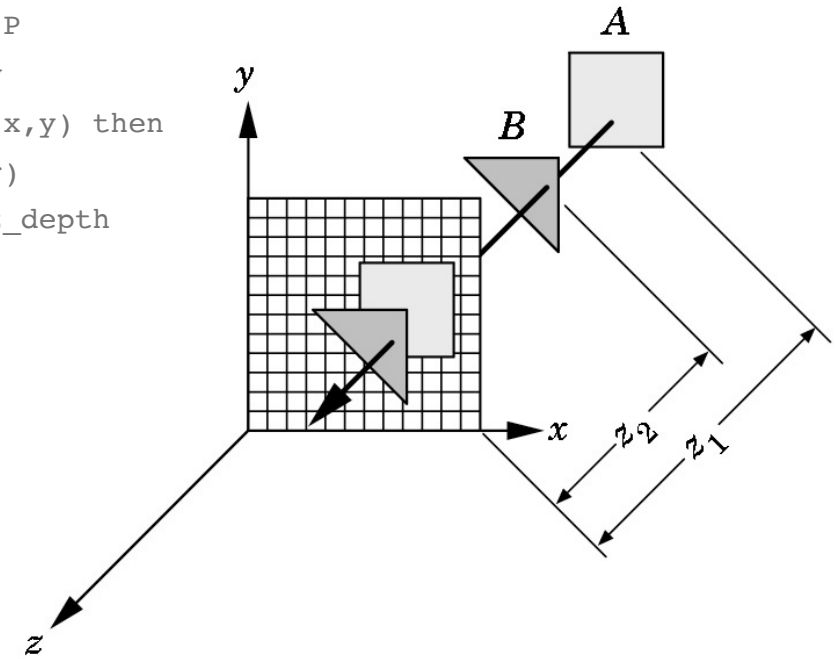


Z-buffer representation

Algoritmos de visibilidade

- Algoritmo Z-Buffer:

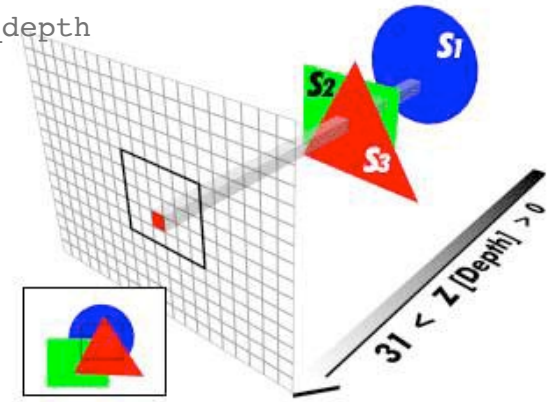
```
for each polygon P
 for each pixel (x,y) in P
   compute z_depth at x,y
   if z_depth < z_buffer(x,y) then
      set_pixel(x,y,color)
      z_buffer (x,y) <= z_depth
```

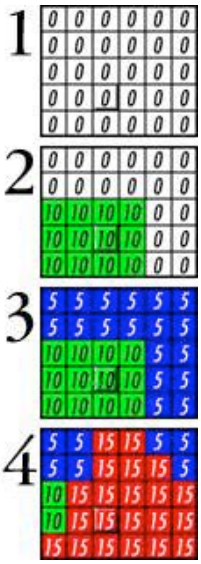


Algoritmos de visibilidade

- Algoritmo Z-Buffer:

```
for each polygon P
 for each pixel (x,y) in P
   compute z_depth at x,y
   if z_depth < z_buffer(x,y) then
      set_pixel(x,y,color)
      z_buffer (x,y) <= z_depth
```





Algoritmos de visibilidade

- Algoritmo Z-Buffer
  - Vantagens:
    - Simples ⇒ hardware
    - Não necessita ordenação
  - Desvantagens:
    - Precisão do z-buffer
    - *Alias ou serrilhado*
      - *Associado ao processo de discretização*

Aplicações utilizando Three.JS/WebGL

Visibilidade na Three.JS

- Propriedade do material dos objetos
  - z-buffer

```
var materialColor = new THREE.MeshBasicMaterial({depthTest:false});
```

  - Backface Culling

```
var materialColor = new THREE.MeshPhongMaterial({side:THREE.BackSide});
```

    - `THREE.FrontSide`
    - `THREE.BackSide`
    - `THREE.DoubleSide`

A seguir…
Primeira Avaliação