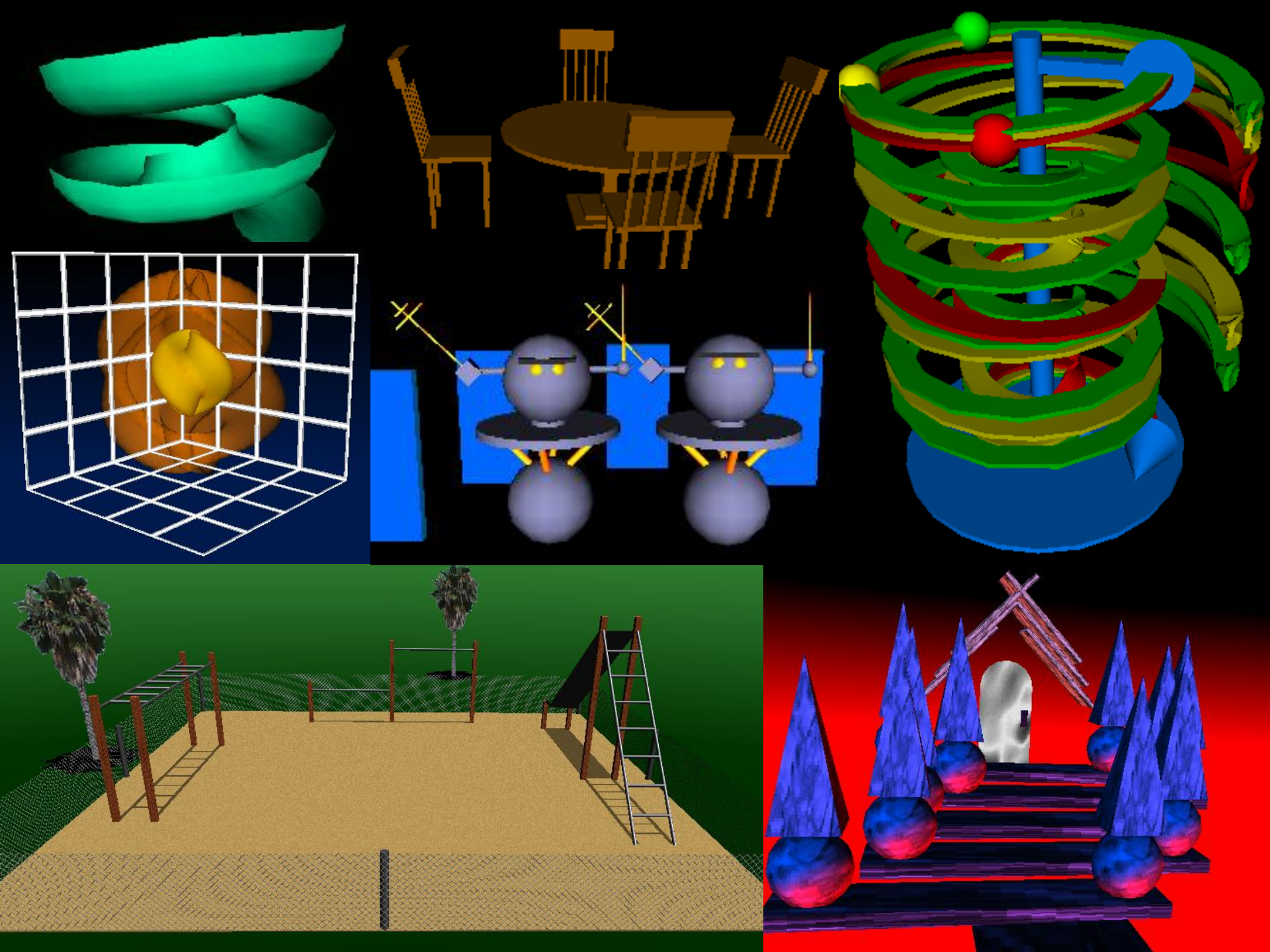
Cor, Iluminação e Sombreamento no VRML

Computação Gráfica

Baseado no tutorial “Introduction to VRML 97” de David R. Nadeau



Agenda

- Formas VRML
  - `geometry`
  - `appearance`

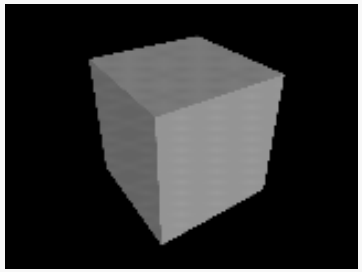
- Tipo de Luzes

Formas em VRML

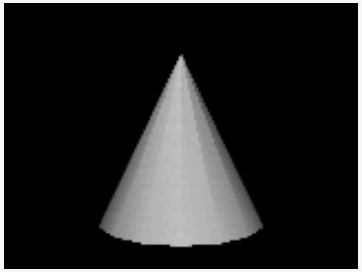
- Os objectos existentes numa cena VRML são construídos a partir de formas base (nó Shape)

- O nó Shape tem dois filhos
  - Para definição da sua **geometria** (`geometry ...`)
    - `Box`, `Cone`, `Cylinder`, `ElevationGrid`, `Extrusion`, `IndexedFaceSet`, `IndexedLineSet`, `PointSet`, `Sphere` e `Text`
  - Para definição do seu **aspecto** (`appearance Appearance {...}`)
    - `material Material { }`
    - `texture ImageTexture { }`
    - `texture MovieTexture { }`
    - `texture PixelTexture { }`
    - `textureTransform TextureTransform {}`
    - Atenção que apenas pode estar definido um nó `texture`

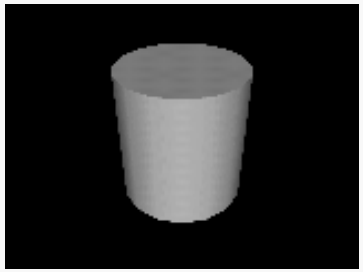
Nós geometry



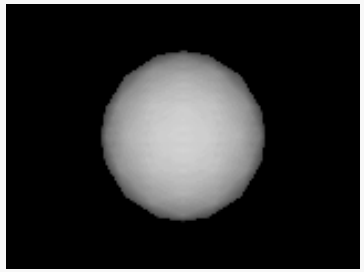
Box



Cone

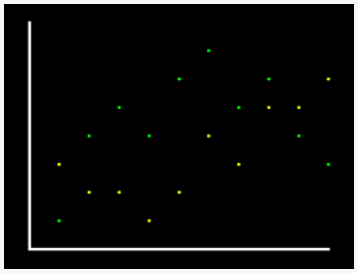


Cylinder

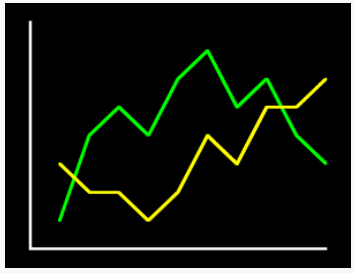


Sphere

Text

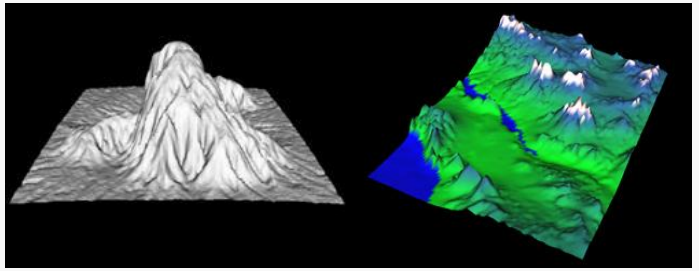


IndexedPointSet

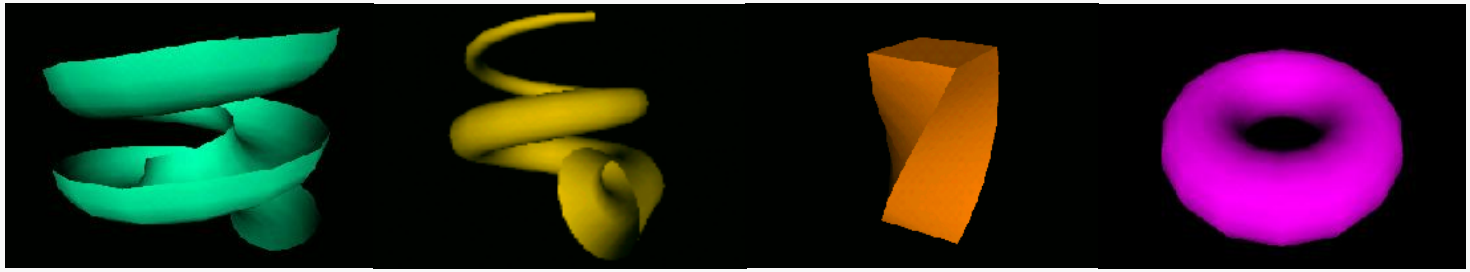


IndexedLineSet

IndexedFaceSet



ElevationGrid



Extrusion

Nó Extrusion (1)

- Nó bastante versátil
- Definido à custa de um **polígono planar** (`crossSection`) e uma **espinha** (`spine`) através da qual o polígono se expande de forma a gerar um sólido

- É ainda possível definir
  - A orientação (rotação) do polígono ao longo da espinha (`orientation`)
  - A escala a aplicar ao polígono ao longo da espinha (`scale`)

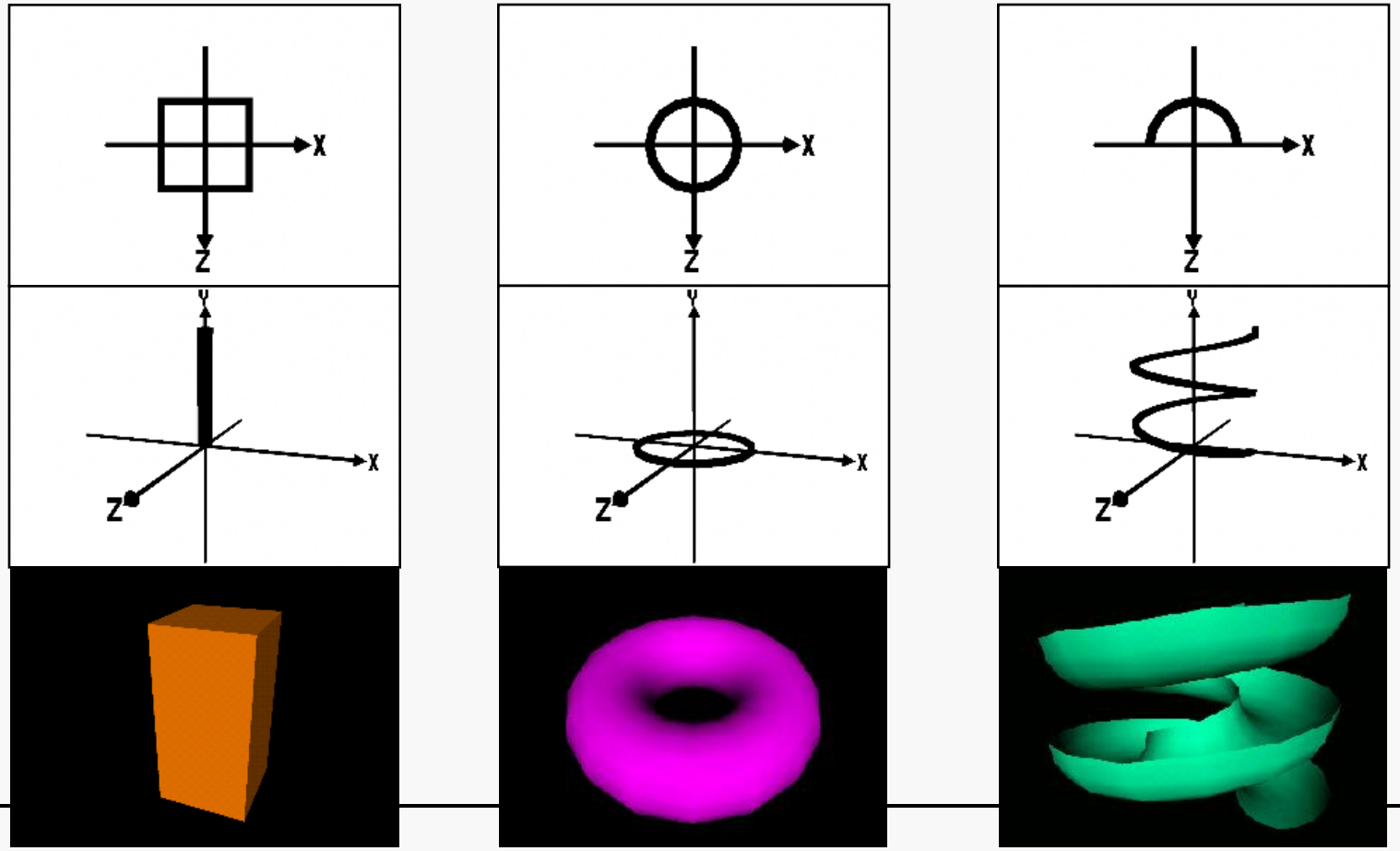
Nó Extrusion (2)

- A `crossSection` é definida no plano XZ (2D)
- A `spine` é definida no espaço (3D)

CrossSection

Spine

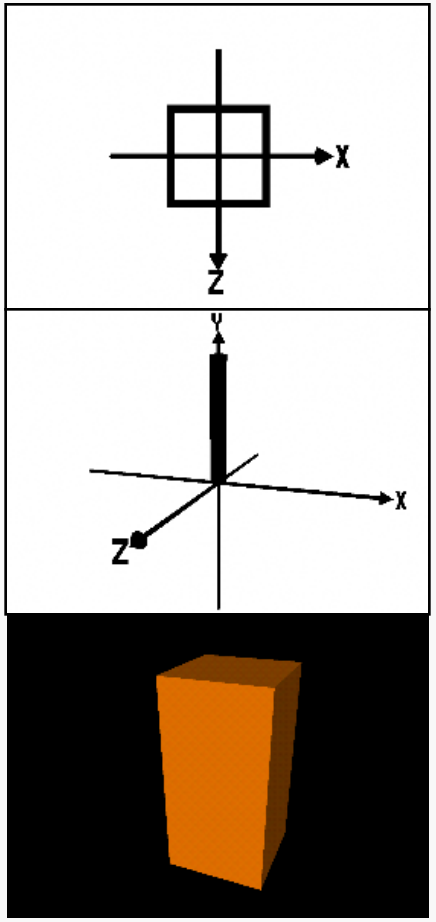
Resultado final



Nó Extrusion (3)

- Sintaxe

```
Shape {
  geometry Extrusion {
    crossSection [1 1, 1 -1, -1 -1, -1 1, 1 1]
    spine [0 0 0, 0 1 0]
    scale [1 1]
    orientation 0 0 1 0
    endCap TRUE
    beginCap TRUE
    solid TRUE
    ccw TRUE
    convex TRUE
    creaseAngle 0
  }
}
```



Nó appearance

- Definição do aspecto da geometria definida com o nó `geometry`

- Para definição do **aspecto** (`appearance Appearance {...}`)
  - `material Material { }`
  - `texture ImageTexture { }`
  - `texture MovieTexture { }`
  - `texture PixelTexture { }`
  - `textureTransform TextureTransform {}`

  - Atenção que apenas pode estar definido um nó `texture`

Cores

- Sistema RGB
- Cada cor básica é representada por número decimal
  - Varia de `0` a `1`
- A cor é obtida através da combinação das três cores básicas

Nome	R	G	B	Cor
vermelho	1	0	0	
verde	0	1	0	
azul	0	0	1	
preto	0	0	0	
branco	1	1	1	
amarelo	1	1	0	
magenta	1	0	1	
ciano	0	1	1	
cinza médio	0.5	0.5	0.5	

- Definição de **uma** cor
  - **`SFColor:`** cor representada em RGB (Ex: `0 0.7 0.2`)
- Definição de **um conjunto** de cores
  - **`MFColor:`** lista de valores do tipo `SFColor` (Ex: `[0 0.5 0, 1 0 0]`)

Definição da cor dos objectos

- Utiliza-se o nó VRML `Material` que é filho do nó `Appearance`

- Sintaxe (e valores por omissão):

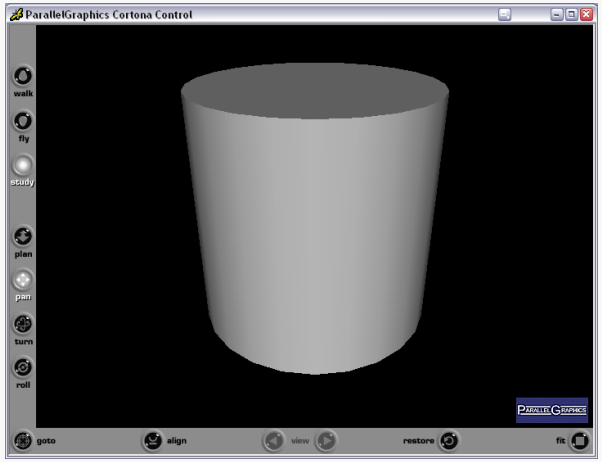
```
material Material {
    ambientIntensity  0.2
    diffuseColor      0.8 0.8 0.8
    emissiveColor     0 0 0
    shininess         0.2
    specularColor     0 0 0
    transparency      0
}
```

- Transparência do objecto: `0` – Opaco; `1` - Transparente

Exemplo: Cor (1)

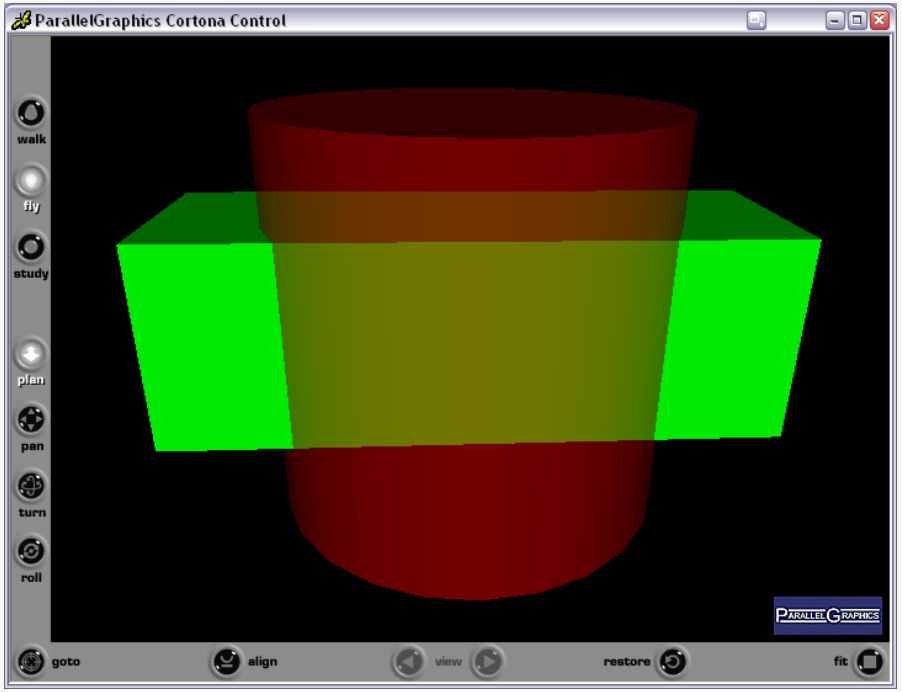
- Utilização dos valores por omissão

```
#VRML V2.0 utf8
Shape {
  geometry Cylinder {}
  appearance Appearance {
    material Material {
      ambientIntensity 0.2
      diffuseColor 0.8 0.8 0.8
      shininess 0.2
      specularColor 0 0 0
      transparency 0
      emissiveColor 0 0 0
    }
  }
}
```



Exemplo: Cor (2)

- Exemplo com transparência
  - Cubo verde dentro de um cilindro vermelho com transparência a 50%



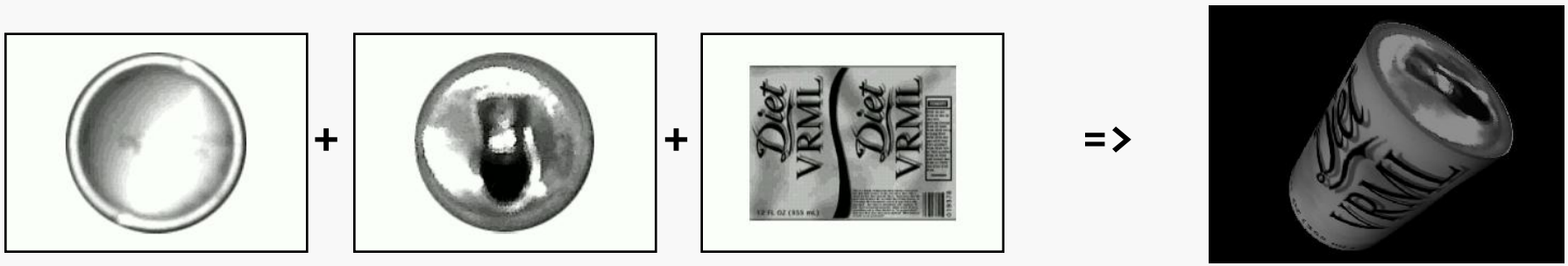

```
#VRML V2.0 utf8
Shape {
  geometry Box { size 3 1 1 }
  appearance Appearance {
    material Material {
      diffuseColor 0 1 0
    }
  }
}

Shape {
  geometry Cylinder { radius 1 }
  appearance Appearance {
    material Material {
      ambientIntensity 0.2
      diffuseColor 1 0 0
      shininess 0
      specularColor 0 0 0
      transparency 0.5
      emissiveColor 0 0 0
    }
  }
}
```

Aparência dos objectos: outros aspectos

- Além da cor, também é possível aplicar uma textura ao objecto
- Objectivo: Aumento do realismo da cena

- Utiliza-se um nó do tipo **`texture`**, filho do nó `Appearance`
  - `ImageTexture`
  - `MovieTexture`
  - `PixelTexture`



Formatos de imagem suportados

- GIF
  - imagem comprimida sem perdas (8-bit)
  - Uma cor transparente (*color key*)
  - Formato livre (GLD – *GIF liberation day*)
- JPEG
  - Imagem comprimida com perdas (24-bit)
  - Não suporta transparência
  - Bom para imagens naturais, mau para imagens sintéticas
  - Formato livre (norma ISO)
- PNG
  - Imagem comprimida sem perdas (24-bit)
  - 8-bit de transparência por *pixel*
  - Formato livre (norma W3C)

Relação entre a textura e a cor

- Ao utilizar texturas devem-se ter em atenção os seguintes aspectos
  - Texturas **coloridas** sobrepõem a cor definida no nó material
  - Texturas em **tons de cinzento** são multiplicadas pela cor definida no nó material
  - Se o nó **material não estiver definido** a textura é aplicada de forma emissiva

Imagens com transparência

Agenda

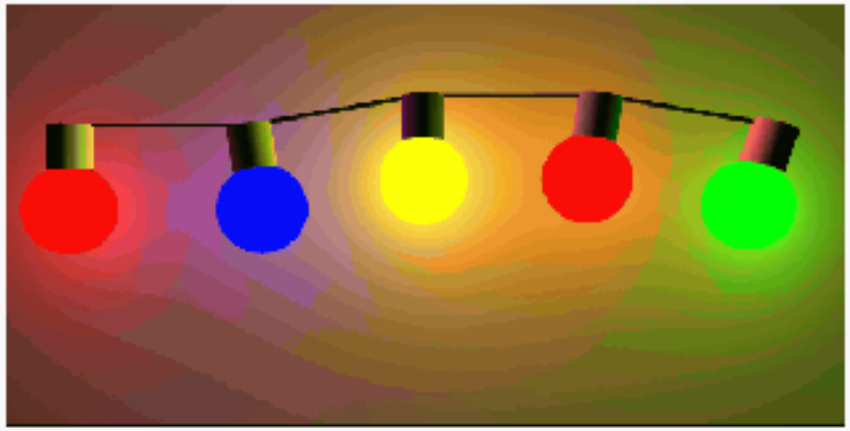
- Formas VRML
  - `geometry`
  - `appearance`

- Tipo de Luzes

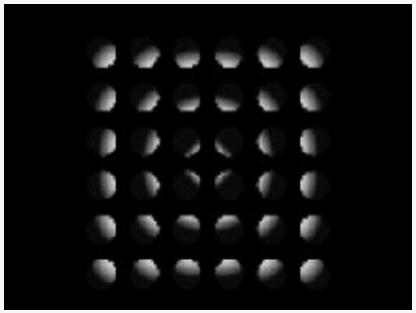
Faça-se luz.....

- Por omissão, no VRML, o actor tem uma luz no seu capacete
  - Configurado através do nó `NavigationInfo { headlight TRUE ... }`

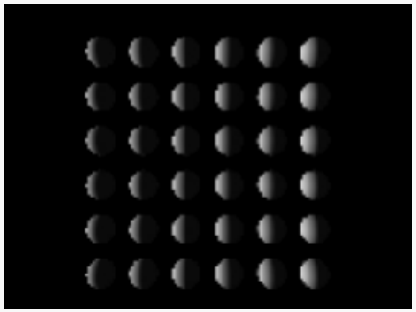
- Para aumentar o realismo da cena é possível adicionar luzes à cena
  - Suporte para vários tipos de luzes (sol, lâmpada, vela, candeeiro, ….)
  - Luzes podem ser posicionadas, orientadas e “coloridas”
  - Luzes não produzem sombras



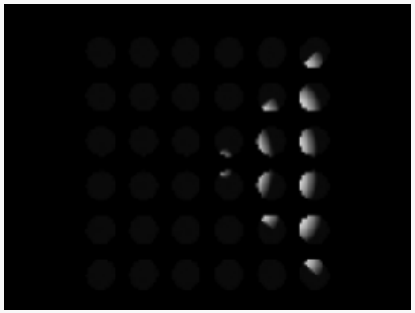
Tipo de luzes



- Luzes pontuais (`PointLight`)
  - Radiam em todas as direcções a partir de um ponto



- Luzes direccionais (`DirectionalLight`)
  - Radiam de acordo com determinada direcção a partir do infinito



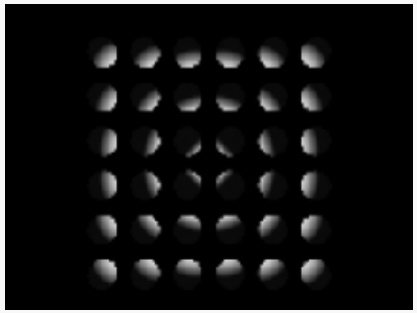
- Luz *SpotLight* (`SpotLight`)
  - Radiam a partir de um ponto numa direcção, com determinada abertura
  - Radia em efeito de cone

Valores comuns a todas as luzes

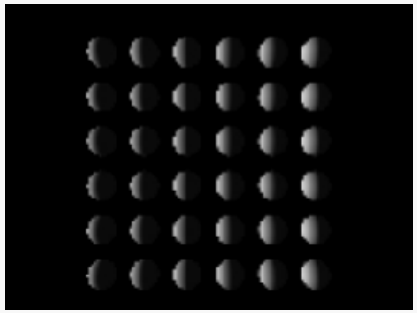
- on - liga e desliga a luz
- `intensity` - brilho da cor
- `ambientIntensity` - controla o efeito ambiente
- `color` - cor da luz

Valores específicos por tipo de luz

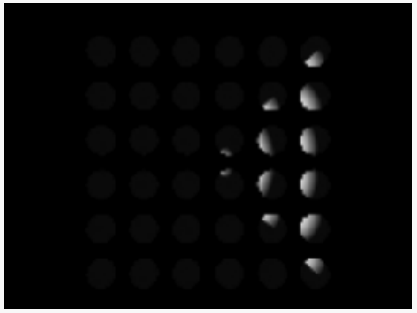
```
PointLight {
    location 0.0 0.0 0.0   # posição da luz
    intensity 1.0
    color 1.0 1.0 1.0
}
```



```
DirectionalLight {
    direction 1.0 0.0 0.0  # direcção da luz (posição no ∞)
    intensity 1.0
    color 1.0 1.0 1.0
}
```



```
SpotLight {
    location  0.0 0.0 0.0  # posição da luz
    direction 1.0 0.0 0.0  # direcção da luz
    cutOffAngle 0.785      # abertura máxima do cone de luz
    beamWidth   0.52       # cone com brilho constante
    intensity 1.0
    color 1.0 1.0 1.0
}
```



Referências

- Norma VRML97 (VRML2.0)
  - http://www.web3d.org/x3d/specifications/vrml/ISO-IEC-14772-VRML97/
- **Introduction to VRML 97 (David R. Nadeau)**
  - **http://www.siggraph.org/education/materials/siggraph_courses/S98/18/vrml97/slides/mt0000.htm**
- Floppy's VRML 97 Tutorial:
  - http://web3d.vapourtech.com/tutorials/vrml97/
- Web 3D Consortium - VRML Archives
  - http://www.web3d.org/x3d/vrml/index.html