	Disciplina	CGRAF
	Curso	SI
ALEXANDRE BELETTI FERREIRA	**AULA - 11**	

Nesta aula iremos explorar a edição de textos e seus efeitos, para tanto, devemos discutir alguns tópicos, a saber:

Nó Material

O nó Material define as propriedades do material que compõe a superfície de objetos descritos por nós geométricos associados (Material é sempre usado como parâmetro para um nó Shape). Os campos neste nó determinam a maneira como a luz reflete no objeto para criar suas cores.

Campos:

diffuseColor	especifica a cor difusa, que reflete todas as fontes de luz dependendo do ângulo da superfície com relação à fonte de luz. Quanto mais diretamente a superfície está voltada para a luz, mais luz difusa é refletida;
ambientIntensity	especifica quanta luz ambiente esta superfície deve refletir. A luz ambiente é "onidirecional" (ilumina em todas as direções) e depende apenas do número de fontes de luz. A cor ambiente é calculada por *ambientIntensity * diffuseColor*;
specularColor	especifica a cor das partes iluminadas do objeto;
emissiveColor	especifica a luz produzida por um objeto incandescente. A cor emissiva é útil para modelos para os quais os valores de luminosidade são explicitamente computados ou para visualização de dados científicos;
shininess	grau de brilho da superfície do objeto, variando de 0.0 (0%) para uma superfície difusa, sem nenhum brilho, até 1.0 (100%) para uma superfície muito polida;
transparency	grau de transparência do objeto, variando de 0.0 para uma superfície completamente opaca até 1.0 para uma superfície totalmente translúcida.

O nó geometry pode ser utilizado para gerar texto se sua especificação for do tipo Text, assim sendo:

Text desenha uma ou mais strings de texto de acordo com o estilo especificado em *fontStyle*. Se forem especificadas mais de uma string, cada uma é desenhada em uma nova linha, com espaço entre as linhas determinado pelo nó **FontStyle**.

Inicialmente, o browser determina a extensão de cada string na direção principal (determinada pelo campo *horizontal* do nó FontStyle). Se o comprimento da maior string é maior do que *maxExtent*, ela sofre a aplicação de uma escala negativa, de maneira que seu comprimento seja igual a *maxExtent*. A escala também é aplicada a todas as outras strings.

Cada string também tem um valor *lenght* correspondente, que indica o seu comprimento desejado. O browser expande ou comprime cada string para o tamanho desejado, também através da aplicação de uma escala.

Campos:

string	a string (ou as strings) de texto que serão mostradas, codificadas segundo o padrão UTF-8;
fontStyle	contém um nó FontStyle que descreve como o texto deve ser desenhado;
maxExtent	a extensão máxima na direção principal de qualque linha de texto neste nó; deve ser maior ou igual a 0. A direção principal é horizontal se o campo *horizontal* do nó FontStyle é TRUE, e vertical se ele é FALSE. Um valor 0 significa que as strings podem ser de qualquer comprimento;
lenght	a extensão desejada para cada string de texto. Zero para uma string significa que ela pode ter qualquer comprimento.

Vejamos um exemplo utilizando alguns campos que foram descritos, então digite:

```
#VRML V2.0 utf8
 Shape {
   appearance Appearance {
      material Material {
                  diffuseColor 1 .85 0
                  specularColor .87 .25 .25
                  ambientIntensity .157
                  shininess 1
               }
   }
   geometry Text {
      string "FALS"
   }
}
```

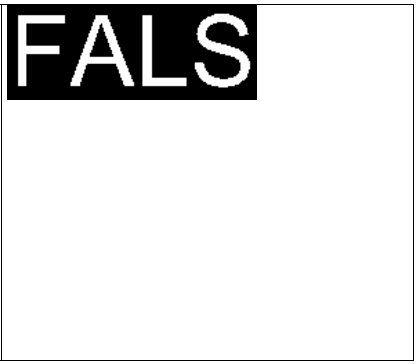
FontStyle aparece apenas como argumento para o nó Text, preenchendo o campo *fontStyle*. Ele define todas as propriedades que determinam como o texto é renderizado pelo browser.

Campos:

size	A altura de cada linha de texto horizontal, ou a largura de cada linha de texto vertical;
family	o tipo genérico de fonte;
style	Indica se o texto deve estar em modo normal, negrito, itálico ou negrito-itálico;
horizontal	Indica a direção do texto. Quando o valor de *horizontal* é FALSE, o texto é mostrado na vertical;
leftToRight	Indica o sentido do texto na horizontal. Para texto horizontal, determina se cada caractere é mostrado à direita (TRUE) ou à esquerda do caractere anterior. Para texto vertical, determina se cada linha é mostrada à direita ou à esquerda da linha anterior;
topToBottom	Indica o sentido do texto na vertical. Para texto horizontal, determina se cada linha é mostrado abaixo (TRUE) ou acima da linha anterior. Para texto vertical, determina se cada caractere é mostrado abaixo ou acima do anterior;
language	A língua em que as strings estão escritas. Especificado como um código de dois caracteres para a língua (como "en" para inglês), seguidos de um sublinhado e dois caracteres opcionais que indicam uma região do planeta (como TW para Taiwan ou BR para Brasil).
justify	Indica como o texto deve ser alinhado. A primeira string se refere ao alinhamento entre as linhas de texto, e a segunda ao alinhamento do bloco de linhas já alinhadas de acordo com a primeira string. "BEGIN" é usado para alinhar o texto à esquerda, "END" para alinhá-lo à direita e "MIDDLE" para centralizá-lo;
spacing	Determina a distância, em linhas, entre linhas consecutivas. A altura de uma linha é determinada por *size*.

Vejamos construção de alguns exemplos:

```
#VRML V2.0 utf8
Shape {
        geometry Text {
        string ["FALS"]
                fontStyle FontStyle  {
                                justify
        "Center"
                                size 1
                                style "bold"
                                family "Arial"
                }
        }
}
```



Imagine que precisássemos escrever duas ou mais palavras, cada qual numa linha, então digite as linhas abaixo:


```
#VRML V2.0 utf8
Shape {
   appearance Appearance {
      material Material { }
   }
   geometry Text {
      string [ "FALS", "Praia Grande" ]
   }
}
```

Caso a necessidade indique que devemos ajustar o comprimento da string, desta forma veja a linha de código abaixo:


```
#VRML V2.0 utf8
Shape {
   appearance Appearance {
      material Material { }
```

```
    }
    geometry Text {
        string "FALS"
        length 2.0
    }
}
```

Agora discutiremos como ajustar o comprimento de textos escritos em linhas diferentes, para tanto, digite o seguinte código:


```
#VRML V2.0 utf8
Shape {
    appearance Appearance {
        material Material { }
    }
    geometry Text {
        string [ "FALS", "Praia Grande" ]
        length  [ 3.0, 4.0 ]
    }
}
```

Como definir espaço entre linhas editadas, digite as linhas de códigos abaixo:


```
#VRML V2.0 utf8
Shape {
    appearance Appearance {
        material Material { }
    }
    geometry Text {
        string [ "FALS", "Praia Grande" ]
        fontStyle FontStyle {
            spacing 2.0
        }
    }
}
```

Caso se precise mesclar tipos de letras e tamanhos diferentes numa mesma linha de texto. Para resolver este tipo de problema veja o exemplo a seguir:


```
#VRML V2.0 utf8
Group {
    children [
        Shape {
            appearance DEF White Appearance {
                material Material { }
            }
            geometry Text {
                string "FALS"
                fontStyle FontStyle {
                    family  "SERIF"
                    style   "ITALIC"
                    justify "END"
                    size    2.0
                }
            }
        },
        Shape {
            appearance USE White
            geometry Text {
                string "Praia Grande"
                fontStyle FontStyle {
                    family  "ARIAL"
                    style   "BOLD"
                    justify "BEGIN"
                    size    1.0
                }
            }
        }
    ]
}
```

Para finalizar, combinaremos texto com formas básicas, promovendo assim uma composição flexível aonde a criatividade e a imaginação de quem faz o curso será o limite. Então execute as linhas de código a seguir:

```
#VRML V2.0 utf8
Group {
   children [
      Shape {
         appearance DEF Tipo Appearance {
            material Material {
               diffuseColor 1 .45 .4
               specularColor .69 .4 .42
               ambientIntensity .153
               shininess .9
               }
         }
         geometry Text {
            string [ "FALS", "Praia Grande" ]
            fontStyle FontStyle {
               justify "MIDDLE"
            }
         }
      },
      Shape {
         appearance USE Tipo
         geometry Box {
            size 5.0 0.01 2.0
         }
      }
   ]
}
```

Ter-se-á como resultado o seguinte efeito: