LAIG1920.Enunciado.T1

Laboratório de Aplicações com Interface Gráfica
Aulas Práticas

MIEIC – 2019/2020

Trabalho Prático 1
Desenvolvimento de um Motor Gráfico em WebGL

1. Introdução

Pretende-se, com este trabalho, constituir uma aplicação dotada de um pequeno motor gráfico 3D. A aplicação deve ser capaz de produzir imagens de qualquer cena, sendo esta especificada através de um ficheiro de texto a ser lido pela aplicação.

O ficheiro de texto deve respeitar um esquema próprio, a que chamaremos LXS – LAIG XML Scene graph - especificado na secção 3 deste documento, e que obedece a um conceito muito vulgar em computação gráfica: o Grafo de Cena (Scene Graph, secção 2). A sintaxe obedece ao formato de tags do XML (*Extensible Markup Language*).

A aplicação deve, através de um *parser*, efetuar a leitura de um documento LXS (cuja extensão do ficheiro deve ser .xml) que descreve a cena, construindo simultaneamente a estrutura de dados correspondente ao grafo de cena. Só depois deve realizar a geração da imagem respectiva. As fontes de luz devem iniciar-se (on/off) de acordo com a especificação LXS e devem poder ser alteradas por meio de controlos na interface gráfica 2D. A seleção da vista ativa é também feita através de uma lista disponível na interface gráfica 2D.

1.1 Componentes do trabalho

1. **Preparação de uma cena de teste**

   Preparar uma cena especificada no esquema LXS e num ficheiro com extensão .xml, de acordo com as instruções nas secções seguintes do presente documento. Todos os ficheiros serão posteriormente divulgados e partilhados, constituindo-se assim um acervo de cenas de teste.

2. **Construção do parser e estrutura de dados**

   a. Implementar a componente de leitura e interpretação do ficheiro .XML (parsing), utilizando para esse efeito a biblioteca WebCGF (um exemplo é fornecido com a componente de leitura parcialmente implementada, juntamente com este enunciado).
   b. Implementar uma estrutura de dados capaz de armazenar o grafo de cena apresentado na secção 2 deste documento.

3. **Desenho da cena**

   Implementar um conjunto de funcionalidades que efetue a visita da estrutura de dados (e, portanto, do grafo) e que, com recurso à biblioteca WebCGF, construa a imagem correspondente usando a tecnologia WebGL.

O trabalho deve ser desenvolvido de forma incremental:

1. Versões iniciais, básicas, do parser, da estrutura interna de dados e das rotinas de visualização que permitam respetivamente ler, armazenar e visualizar primitivas e transformações simples, ignorando inicialmente luzes, materiais e texturas.

2. Versões progressivamente estendidas do parser, da estrutura de dados e visualizador com as restantes funcionalidades enunciadas.

1.2 Notas sobre a avaliação do trabalho

Composição dos Grupos: Os trabalhos devem ser efetuados em grupos de dois estudantes. Em caso de impossibilidade (por exemplo por falta de paridade numa turma), deve ser discutida com o docente a melhor alternativa.

Avaliação do Trabalho de Grupo: O código resultante do trabalho de grupo será apresentado e defendido em sala de aula, perante o docente respetivo. O trabalho poderá ser sujeito a uma bateria de testes, com origem em cenas representadas por ficheiros .xml, sendo a classificação atribuída dependente da adequação da resposta dada.

Tendo em atenção as funcionalidades enunciadas, serão considerados os seguintes critérios para efeitos de Avaliação do Trabalho de Grupo:

Estruturação e Documentação do código	2 valores
Primitivas e Geometria	4 valores
Transformações Geométricas - descrição e herança	4 valores
Materiais - descrição e herança	3.5 valores
Texturas - descrição, dimensões e herança	3.5 valores
Interface gráfica 2D: a) fontes de Luz - descrição e ON/OFF; b) seleção da vista ativa	3 valores

Avaliação Individual: Não aplicável no presente trabalho.
Avaliação Global do Trabalho: Igual à avaliação do trabalho de grupo.

De acordo com a formulação constante na ficha de unidade curricular, a avaliação deste trabalho conta para a classificação final com um peso:

80% * 25% = 20%

Datas Principais:

- Data limite de entrega do ficheiro XML no esquema LXS: 06/10/2019
- Data limite de entrega do trabalho completo: 20/10/2019
- Avaliação do trabalho em aulas: semana com início em 21/10/2019
- Prova de avaliação individual: (não aplicável)

Plano de trabalhos sugerido:

- **Semana de 23/09:** Início do trabalho; criação e carregamento da estrutura de dados; Início da definição da cena de teste.
- **Semana de 30/09:** Desenho de primitivas base; travessia do grafo de cena com transformações; criação da cena de teste e submissão.
- **Semana de 07/10:** Aplicação de materiais e texturas; garantia de herança; iluminação
- **Semana de 14/10:** Verificações finais

2. Grafo de Cena

Um grafo de cena pode ser visitado como uma árvore que especifica, de forma hierárquica, uma cena 3D. Cada nó corresponde a um objeto, simples (folha) ou composto (nó intermédio).

Todo e qualquer nó deve ter um identificador do tipo *string* que é definido no ficheiro .XML. Um nó pode ser instanciado várias vezes, ou seja, referenciado por vários nós seus ascendentes; por exemplo, um nó pode representar a roda de um automóvel e, por isso, ser referenciado quatro vezes diferentes.

2.1. Nós tipo Folha

Cada folha refere-se exclusivamente a um objeto simples ou primitiva, cuja geometria é interpretada e imediatamente desenhada. Deve por isso conter todos os parâmetros exigidos pela instrução respetiva. Um nó folha não contém transformações geométricas nem propriedades de aspeto (materiais, etc.).

2.2. Nós Intermédios

Subindo na hierarquia, um nó intermédio da árvore possui um ou mais nós como seus descendentes diretos, sendo que estes poderão ser folhas ou outros nós intermédios. Está reservada aos nós intermédios a declaração de eventuais transformações geométricas e propriedades de aspeto.

Transformações Geométricas: Um nó recebe do seu antecessor uma matriz de transformações geométricas *Ma*. Sendo um nó intermédio, possui as suas próprias transformações geométricas, representadas por uma matriz única *Mp*. A matriz a aplicar ao objeto, assim como a passar aos seus eventuais descendentes, é calculada pela expressão *M=Ma*Mp*.

Propriedades de aspeto: Cada nó recebe propriedades de aspeto do seu antecessor (devem ser definidos valores de "default" para o nó raiz) e pode ter definidas as suas próprias propriedades de aspeto. As regras de desambiguação a usar neste caso são definidas na especificação do esquema LXS, na secção 3 deste documento.
Textura: Cada nó recebe uma textura do seu antecessor e pode ter definida a sua própria textura, que pode ser “nula”. As regras de desambiguação a usar neste caso são definidas na especificação do esquema LXS, na secção 3 deste documento.

2.3. Outras Entidades

Além de objetos, a esquema LXS pressupõe a existência de outras entidades, como sejam as câmaras de visualização, as fontes de luz, as texturas e os materiais. As entidades de visualização e de iluminação, tais como as fontes de luz, interferem com todo o grafo e, por isso, não devem ser ligadas a qualquer dos nós do grafo. Por isso, o esquema exige a sua declaração na parte inicial do ficheiro .xml. As texturas e os materiais podem ser usadas nos nós intermédios, desde que tenham sido preparadas anteriormente, pelo que também é previsto declararem-se no início do ficheiro. Ao declarar-se uma textura, a respetiva imagem deve ser lida para memória a partir do ficheiro .jpg ou .png correspondente (atenção: devido a limitações de algumas placas gráficas, sugere-se que o comprimento e largura de cada textura, medidos em número de pixels, sejam potências de 2).

A figura 1 apresenta um exemplo de um grafo de cena.

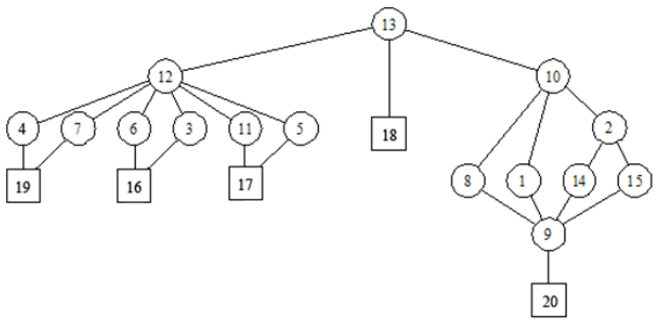

Figura 1 - Exemplo de um grafo de cena.

Note-se que o número de descendentes diretos de um nó intermédio é indeterminado, mas tem de existir pelo menos um descendente.

Cada nó, intermédio ou folha, pode ser instanciado várias vezes, ou seja, pode ser referenciado por mais do que um nó ascendente. Por exemplo, um nó pode representar a roda de um automóvel e, por isso, ser referenciado quatro vezes diferentes como se vê na figura 2: o objeto "roda" (numeração 9) tem a sua subárvore (sugerida mas não representada) e tem as suas transformações geométricas particulares. No entanto, para que as quatro rodas tenham distintas posições no espaço, é necessário que possuam diferentes transformações geométricas. Assim, são criados os nós intermédios de instanciação 4, 2, 5 e 7, todos referindo serem compostos pelo nó 9, mas cada um dos quatro com a sua transformação geométrica, diferente das restantes.

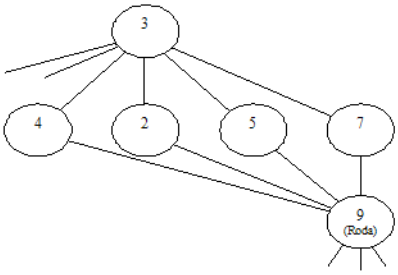

Figura 2 - Excerto de um grafo de cena com instanciação múltipla.

3. Esquema LXS

O esquema LXS – LAIG XML Scene graph - definido na linguagem XML, constitui um formato de especificação de cenas 3D de uma forma muito simples e fácil de interpretar. Um documento no esquema LXS pode ser escrito em qualquer editor de texto e obedece a regras de XML, baseadas em *tags*.

Ao ser lido e interpretado por uma aplicação gráfica, um ficheiro no esquema LXS deve ser verificado em termos de sintaxe e de consistência da estrutura, devendo a aplicação gerar mensagens de erro ou avisos, identificando eventuais irregularidades encontradas ou situações anómalas ou indesejáveis (por exemplo a referência a um nó inexistente).

Cada comando representa-se por um ou mais *tags*, contendo os parâmetros respetivos (se existirem). Um grupo de caracteres ou mesmo linhas limitado por <!-- e --> é considerado um comentário. Todo o ficheiro deve ser escrito em minúsculas.

Um documento escrito segundo o esquema LXS estende-se por nove blocos. Cada bloco inicia-se com um termo identificador de bloco, implementado em XML na forma de uma *tag*. A referência a uma *tag* inicia-se com um identificador alfanumérico entre os dois caracteres "< >" (por exemplo <lights>) e termina com o mesmo identificador antecedido de uma barra de divisão (no mesmo exemplo, </lights>). Entre as duas ocorrências, descreve-se o conteúdo do elemento identificado pela tag. A sequência de blocos é a seguinte:

```
scene               <!-- global values -->
views               <!-- specification of all views -->
ambient             <!-- illumination parameters -->

lights              <!-- specification of all light sources -->
textures            <!-- specification of all textures -->
materials           <!-- specification of all materials -->
transformations     <!-- specification of geometric transformations →
                    <!-- for use in different components -->
primitives          <!-- specification of all primitives -->
components          <!-- specification of all components: objects  -->
                          <!-- composed of primitives and other components -->
```

Nota: a ordem de blocos não pode ser desrespeitada; por exemplo, se um bloco "materials" aparecer antes de um bloco "textures", a aplicação deverá produzir uma mensagem de erro ou de aviso.

O bloco views e os últimos seis blocos anteriores contêm definições de várias entidades do tipo correspondente (várias views, luzes, várias texturas, etc...). Cada uma dessas entidades contém um identificador do tipo *string*. Cada identificador deve ser único dentro de cada bloco (por exemplo, não podem existir duas fontes de luz com o mesmo identificador).

3.1. Novas primitivas

O esquema LXS contém a definição de cinco primitivas, que correspondem aos nós folha no grafo de cena. As primitivas a implementar são ***rectangle, triangle, cylinder, sphere***, e ***torus***, sendo que as classes que representam ***rectangle*** e ***cylinder*** terão sido implementadas na unidade curricular de Computação Gráfica (***MyRectangle*** e ***MyCylinder***, respetivamente).
A classe MyCylinder deverá ser atualizada e as restantes classes deverão ser implementadas como descrito nas próximas secções.

Cilindro atualizado (*MyCylinder*)

Atualize a classe ***MyCylinder*** correspondente a um cilindro com os seguintes parâmetros: raio na base (); raio no topo (); altura (height); número de divisões em rotação (slices); número de divisões em altura (stacks).
O cilindro deve ser desenhado sem tampas, com a base centrada na origem, e o seu eixo central deve ser coincidente com o eixo de Z, como mostra a figura 1.

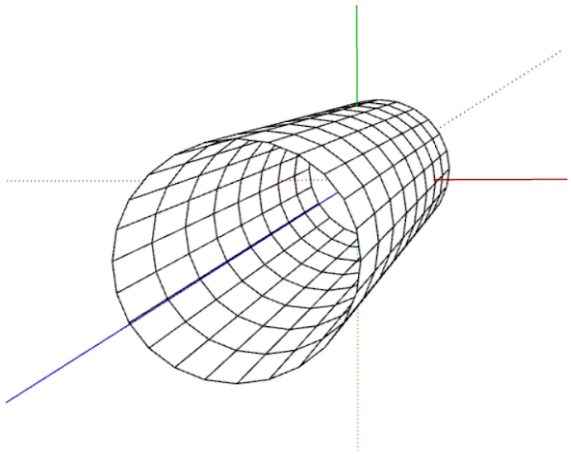

Figura 1: Imagem exemplo de um cilindro com base centrada na origem.

Triângulo (*MyTriangle*)

Crie uma nova classe ***MyTriangle*** correspondente a um triângulo caraterizado pelos três vértices definidos no ficheiro da cena. Defina as coordenadas de textura seguindo como referência o documento *"Triângulos e Coordenadas de Textura"* disponível no Moodle.

Nota: Esta classe difere da classe criada com o mesmo nome na unidade curricular de Computação Gráfica, que representava um triângulo rectângulo.

Esfera (*MySphere*)

Pretende-se que crie a classe ***MySphere*** correspondente a uma esfera com o centro na origem, com eixo central coincidente com o eixo Z e raio variável (radius). A esfera deverá ter um número variável de "lados" à volta do eixo Z (slices), e de “pilhas” (stacks), que corresponde ao número de divisões ao longo do eixo Z, desde o centro até a um dos "pólos" (ou seja, número de "fatias" da semi-esfera). A Figura 2 tem uma representação visual desta esfera, assim como o documento *“Geometria de Quádricas”* no Moodle.

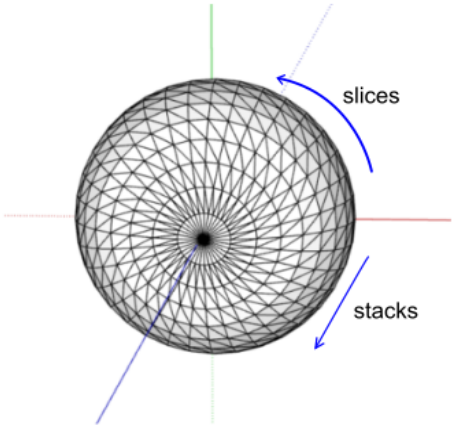

Figura 2: Imagem exemplo de uma esfera centrada na origem.

Torus (*MyTorus*)

Pretende-se que crie a classe ***MyTorus*** para que esta consiga desenhar um *torus* centrado na origem, com raios interior e exterior variáveis (inner/outer radius) à volta do eixo Z, como representado na Figura 3. O torus deverá ter um número de “lados” (slices) variável à volta do raio interior, e um número de “voltas” (loops) à volta do eixo circular.

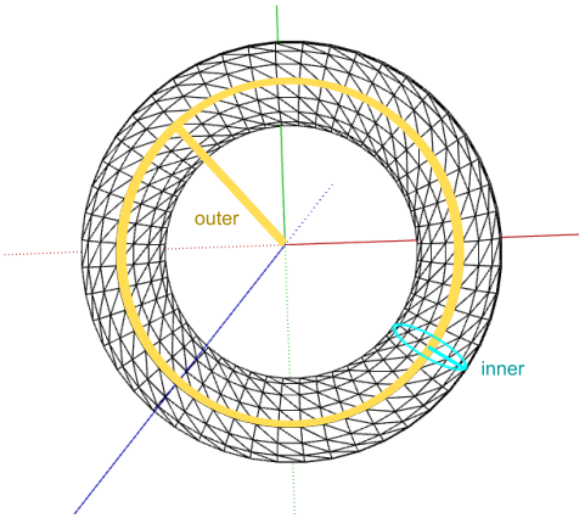

Figura 3: Imagem exemplo de *torus* centrado na origem à volta do eixo Z.

3.2. Sintaxe do Esquema LXS

A seguir apresenta-se uma listagem representativa da sintaxe pretendida em LXS:

```
<!-- Os comentarios devem ter espacos no inicio e no fim, a -->
<!-- separar dos hifens -->
<!-- Nao devem ser usados caracteres especiais (p.ex. acentos) -->
<!-- Todas as tags e atributos sao obrigatorios, exceto onde for -->
<!-- referido o contrario -->

<!-- Na descricao abaixo, os simbolos utilizados tem o seguinte significado: -->
    <!-- ii: valor inteiro -->
    <!-- ff: valor real. Para cor e alpha, o valor varia [0.0 e 1.0] -->
    <!-- ss: sequência de caracteres -->
    <!-- cc: caracter unico. Por exemplo: "x" or "y" or "z" -->
    <!-- tt: "0" or "1" valor booleano com significado falso e verdadeiro -->


<lxs>
    <!-- deve definir-se um objeto para raiz da arvore, assim -->
    <!-- como o comprimento dos tres eixos (cilindros) -->

    <scene root="ss" axis_length="ff" />


    <views default="ss" >
        <!-- tem de existir, pelo menos, uma vista de -->
        <!-- entre as seguintes (perspective ou ortho) -->

        <perspective id="ss" near="ff" far="ff" angle="ff">
            <from x="ff" y="ff" z="ff" />
            <to x="ff" y="ff" z="ff" />
        </perspective>

        <ortho id="ss"  near="ff" far="ff" left="ff" right="ff" top="ff" bottom="ff" >
            <from x="ff" y="ff" z="ff" />
            <to x="ff" y="ff" z="ff" />
            <up x="ff" y="ff" z="ff" /> <!-- opcional, default 0,1,0 -->
        </ortho>

    </views>


    <globals>
        <ambient r="ff" g="ff" b="ff" a="ff" />
        <background r="ff" g="ff" b="ff" a="ff" />
    </globals>


    <lights>
        <!-- Deve existir um ou mais blocos "omni" ou "spot" -->
        <!-- Os identificadores "id" nao podem ser repetidos -->

        <omni id="ss" enabled="tt" >
            <location x="ff" y="ff" z="ff" w="ff" />
```

```
            <ambient r="ff" g="ff" b="ff" a="ff" />
            <diffuse r="ff" g="ff" b="ff" a="ff" />
            <specular r="ff" g="ff" b="ff" a="ff" />
            <attenuation constant="ff" linear="ff" quadratic="ff" />
        </omni>

        <spot id="ss" enabled="tt" angle="ff" exponent="ff">
            <!-- atencao, "target" e' diferente de "direction" -->

            <location x="ff" y="ff" z="ff" w=”ff” />
            <target x="ff" y="ff" z="ff" />
            <ambient r="ff" g="ff" b="ff" a="ff" />
            <diffuse r="ff" g="ff" b="ff" a="ff" />
            <specular r="ff" g="ff" b="ff" a="ff" />
            <attenuation constant="ff" linear="ff" quadratic="ff" />
        </spot>

    </lights>


    <textures>
        <!-- Deve existir um ou mais blocos "texture" -->
        <!-- Os identificadores "id" nao podem ser repetidos -->
        <!-- o valor do campo "ficheiro" deve ser o caminho relativo para a imagem -->
        <!-- o ficheiro deve ter extensao .jpg ou .png -->
        <!-- preferencialmente as dimensoes do ficheiro devem ser potencia de 2 -->
        <texture id="ss" file="ss" />

    </textures>


    <materials>
        <!-- Deve existir um ou mais blocos "material" -->
        <!-- Os identificadores "id" nao podem ser repetidos -->

        <material id="ss" shininess = "ff" >
            <emission r="ff" g="ff" b="ff" a="ff" />
            <ambient r="ff" g="ff" b="ff" a="ff" />
            <diffuse r="ff" g="ff" b="ff" a="ff" />
            <specular r="ff" g="ff" b="ff" a="ff" />
        </material>

    </materials>


    <transformations>
        <!-- Deve existir um ou mais blocos "transformation" -->
        <!-- Os identificadores "id" nao podem ser repetidos -->
        <!-- Os angulos sao expressos em graus -->

        <transformation id="ss">
            <!-- instrucoes a usar sem limite nem ordem -->
            <!-- deve existir pelo menos uma transformacao -->

            <translate x="ff" y="ff" z="ff" />
            <rotate axis="cc" angle="ff" />
            <scale x="ff" y="ff" z="ff" />
      </transformation>

    </transformations>
```

```
    <primitives>
        <!-- Uma "primitive" corresponde à noção de "folha" -->
        <!-- Deve existir um ou mais blocos "primitive" -->
        <!-- Os identificadores "id" nao podem ser repetidos -->

        <primitive id="ss">
            <!-- apenas pode existir UMA das seguintes tags: -->
            <!--    rectangle, triangle, cylinder, sphere, torus -->
            <!-- os parametros devem ser interpretados, genericamente, -->
            <!-- como em WebGL; -->
            <!-- importante: consultar o tópico Geometria de Quadricas -->
            <!-- disponível no MOODLE relativamente 'a orientacao das quadricas -->
            <!-- assim como o documento sobre as coordenadas dos triangulos -->

            <rectangle x1="ff" y1="ff" x2="ff" y2="ff" />
            <triangle   x1="ff" y1="ff" z1="ff"
                        x2="ff" y2="ff" z2="ff"
                        x3="ff" y3="ff" z3="ff" />
            <cylinder base="ff" top="ff" height="ff" slices="ii" stacks="ii" />
            <sphere radius="ff" slices="ii" stacks="ii" />
            <torus inner="ff" outer="ff" slices="ii" loops="ii" />
        </primitive >
    </primitives >


    <components>

        <component id="ss">
            <!-- Uma "component" e' um objeto composto e pode ser -->
            <!--      usada em nos intermédios -->
            <!-- bloco "transformation" e' obrigatorio -->

            <transformation>
                <!-- deve conter uma referencia a uma das "transformation" -->
                <!-- declaradas anteriormente -->

                <transformationref id="ss" />

                <!-- ou, ALTERNATIVAMENTE, transformacoes explicitas, -->
                <!-- usando zero ou mais das instrucoes seguintes, sem -->
                <!--        limite nem ordem -->
                <!-- ex: bloco transformation pode ficar sem conteudo -->

                <translate x="ff" y="ff" z="ff" />
                <rotate axis="cc" angle="ff" />
                <scale x="ff" y="ff" z="ff" />
            </transformation>

            <!-- declaracao obrigatoria de pelo menos um material; -->
            <!-- o material id="inherit", mantem (herda) material do "pai" -->
            <!-- se varios materiais declarados, o default e' o -->
            <!-- primeiro material; de cada vez que se pressione a tecla m/M, -->
            <!-- o material muda para o proximo material da lista; do -->
            <!-- ultimo material da lista volta ao primeiro -->
            <!-- O comportamento despoletado pela tecla m/M deve ser aplicado -->
            <!-- simultaneamente a todos os nos do grafo de cena -->

            <materials>
                <material id="ss" />
            </materials>
```

```
            <!-- declaracao obrigatoria de texture -->
            <!-- id="inherit" mantem (herda) a textura do objeto "pai" -->
            <!-- id="none" remove a textura recebida do pai -->
            <!-- a textura declarada sobrepoe a textura recebida do -->
            <!-- objecto "pai" -->
            <!-- length_s e length_t sao fatores de escala de textura:-->
            <!-- Exemplo length_s=3.0: uma ocorrencia da textura, em -->
            <!-- comprimento, deve cobrir um comprimento igual -->
            <!-- a 3 unidades;  -->
            <!-- Exemplo length_t=0.4, uma ocorrencia da textura, em -->
            <!-- largura, deve cobrir uma largura igual a 0.4 unidades. -->
            <!-- Sempre que id="inherit" ou id="none", os parametros -->
            <!-- length_s e length_t não devem incluir-se na instrucao. -->
            <!-- No caso inherit, utilizam-se os valores de -->
            <!-- length_s e length_t recebidos do nó pai. -->
            <!-- E' permitido que objetos afetados por Transf. Geometr. -->
             <!-- do tipo escalamento violem esta regra.        -->
            <!-- Nao e' necessario aplicar fatores de escala em superficies -->
            <!-- quadricas (esfera, cilindro...) -->

            <texture id="ss" length_s="ff" length_t="ff" />

            <!-- bloco "children" obrigatorio num "component" -->
            <children>
                    <!-- deve existir uma ou mais tags "componentref" e/ou -->
                    <!-- "primitiveref", identificando outros -->
                    <!-- componentes ou primitivas -->

                    <componentref id="ss" />
                    <primitiveref id="ss" />
             </children>
        </component>
    </components>

</lxs>
```

Versão 20191010.1115
