Laboratório de Aplicações com Interface Gráfica

Aulas Práticas

MIEIC – 2015/2016

Trabalho 1 – Desenvolvimento de um Motor Gráfico em WebGL

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 (ver secção 2 deste documento).

O ficheiro de texto deve respeitar uma "linguagem" própria, a que chamaremos LSX - *Language of Scenes in XML*, especificada na secção 2 deste documento, que obedece a um conceito muito vulgar em computação gráfica: o Grafo de Cena (*Scene Graph*). A sintaxe obedece ao formato de *tags* do XML

A aplicação deve efectuar a leitura do ficheiro ".lsx" que descreve a cena, construindo simultaneamente a estrutura de dados correspondente ao grafo de cena. Só depois deve efectuar a afixação da imagem respectiva. As fontes de luz devem iniciar-se (*on*/*off*) de acordo com o especificado no ficheiro .lsx e devem poder ser alteradas por meio de controlos na interface gráfica 2D.

Componentes do Trabalho:

1. Preparar uma cena especificada num ficheiro .lsx original, de acordo com as instruções das secções seguintes do presente documento. Todos os ficheiros serão posteriormente divulgados e partilhados, constituindo-se assim um acervo de cenas de teste.
2. Implementar a componente de leitura e interpretação do ficheiro .lsx (*parsing*), por recurso à biblioteca WebCGF (instruções e um exemplo estão disponíveis juntamente com este enunciado).
3. Implementar uma estrutura de dados, que passa pelo grafo de cena apresentado na secção 1 deste documento.
4. Implementar um conjunto de funcionalidades que efectue a visita do grafo anterior e construa a imagem correspondente usando a tecnologia WebGL e com recurso à biblioteca WebCGF.

Notas sobre a avaliação do trabalho:

Composição dos Grupos : Os trabalhos devem ser efetuados em grupos de dois estudantes. Em caso de impossibilidade (p.ex. 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 sujeito a uma bateria de testes, com origem em várias cenas representadas por ficheiros .lsx, sendo a classificação atribuída dependente da adequação da resposta dada.

Avaliação Individual : Na prova de avaliação individual, serão pedidas várias funcionalidades adicionais, a implementar sobre o código original desenvolvido em trabalho de grupo. Cada alínea será avaliada, à semelhança do trabalho de grupo, pelas respostas dadas pelo software a uma bateria de testes.

Avaliação do Trabalho : Média aritmética das duas avaliações anteriores.

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

> 80% * 30% = 24%.

Entrega e avaliação:

- Data limite de entrega do ficheiro .lsx: 12/10/2015
- Data limite de entrega do trabalho completo: 19/10/2015
    - Por via eletrónica/moodle (instruções a divulgar proximamente).
- Avaliação do trabalho em aulas: semana com início em 19/10/2015
- Prova de avaliação individual: 21/10/2015

---

1. 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 objecto, simples (folha) ou composto (nó intermédio).

Todo e qualquer nó deve ter um identificador numérico único, assim como manter o identificador do tipo *string* que lhe corresponde no ficheiro .lsx. 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.

1.1. Nós tipo Folha

Cada folha refere-se exclusivamente a um objecto simples, cuja geometria é directamente interpretada e desenhada por instruções WebGL. Deve por isso conter todos os parâmetros exigidos pela instrução WebGL.

1.2. Nós Intermédios

Subindo na hierarquia, um nó intermédio da árvore possui um ou mais nós como seus descendentes directos, sendo que estes poderão ser folhas ou outros nós intermédios. Está reservada a nós intermédios a declaração de eventuais:

- Propriedades de aspecto (materiais, etc.);
- 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 objecto, assim como a passar aos seus eventuais descendentes, é calculada pela expressão ***M=Ma*Mp***.

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 da linguagen .lsx, na secção 2 deste documento.

1.3. Outras Entidades

Além de objectos, a linguagem pressupõe a existência de outras entidades, como sejam as câmaras de visualização, as fontes de luz, as texturas e os materiais. As entidades de iluminação e de visualização, 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, a linguagem exige a sua declaração na parte inicial do ficheiro .lsx. As texturas e os materiais servem para utilização nos nós intermédios, pelo que também é previsto prepararem-se no início do ficheiro. Ao declarar-se uma textura esta deve ser lida, para memória, a partir do ficheiro .jpg ou .png correspondente (atenção: o comprimento e a largura de cada textura devem ser potências de 2).

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

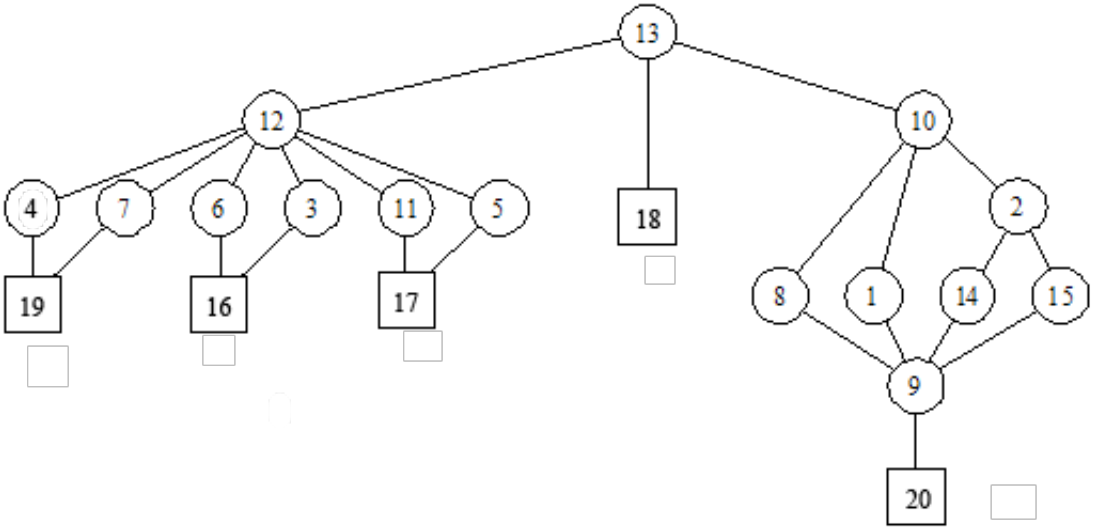

Figura 1 - Exemplo de um grafo de cena

Note-se que o número de descendentes directos de um nó intermédio é indeterminado, mas tem de existir pelo menos um descendente. Todo e qualquer nó deve ter um identificador numérico, além do identificador que é fornecido no ficheiro .lsx. O identificador numérico deve ser único para todo o grafo, incluindo nós intermédios e nós folha (isto é, a numeração é conjunta para os dois tipos de nós).

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 objecto "roda" tem a sua subárvore (não representada para não sobrecarregar o desenho) 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 transformações geométricas diferentes.

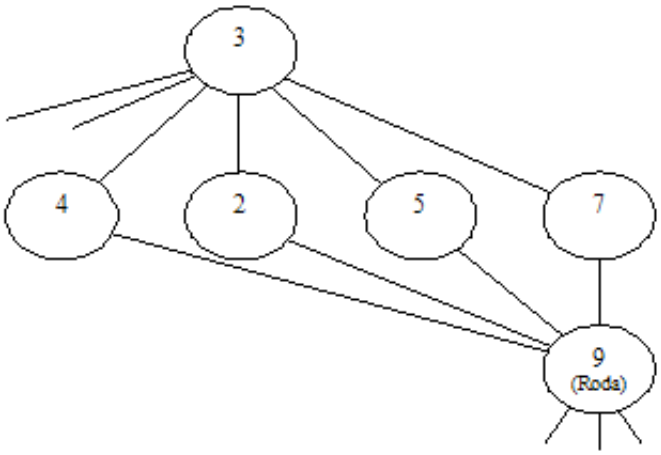

Figura 2 - Instanciação do objecto 9, com diferentes transformações geométricas.

2. Linguagem LSX

A linguagem LSX - *Language of Scenes* constitui uma forma de especificar cenas 3D de uma forma muito simples e fácil de interpretar. Um documento em linguagem LSX 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 em linguagem LSX deve ser verificado em termos de sintaxe, devendo a aplicação gerar algumas mensagens de erro, eventualmente identificando os eventuais erros encontrados.

Cada comando representa-se por um ou mais *tags*, contendo os parâmetros respectivos (se existirem). Um grupo de caracteres ou mesmo linhas limitado por `<!--` e `-->` é considerado um comentário.

A linguagem assume os seguintes valores por omissão (*defaults*) que, por isso, a aplicação deve inicializar:

```
front face = CCW
depth func = LEQUAL, enable
cull face  = back, enable
lighting   = enable
shading    = Gouraud
polygon mode = fill
```

Um documento LSX divide-se em blocos, cada um iniciando-se com um termo identificador de bloco, escrito entre " < >" (por exemplo `<LIGHTS>`) e terminando com o mesmo termo antecedido de uma barra de divisão (no mesmo exemplo, `</LIGHTS>`). A sequência de blocos é a seguinte:

```
INITIALS          <!--global values -->
ILLUMINATION      <!--illumination parameters -->
LIGHTS            <!--specification of all lights -->
TEXTURES          <!--specification of all textures -->
MATERIALS         <!-- specification of all materials -->
LEAVES            <!-- specification of all simple objects -->
NODES             <!--specification of all combined objects -->
```

Os últimos cinco blocos anteriores contêm definições de entidades do tipo correspondente (várias 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).

Segue-se uma listagem representativa da sintaxe pretendida.

```
In the following, these symbols have the significance:

    ii: integer value
    ff: float value
    ss: string value
    cc: character "x" or "y" or "z"
    tt: "0" or "1" with Boolean significance

<SCENE>


<INITIALS>
    <frustum near="ff" far="ff"/>                    <!-- frustum planes-->
    <translate x="ff" y="ff" z="ff" />               <!-- initial translate -->
```

```
    <rotation axis="cc" angle="ff" />                  <!-- initial rotation 3 -->
    <rotation axis="cc" angle="ff" />                  <!-- initial rotation 2 -->
    <rotation axis="cc" angle="ff" />                  <!-- initial rotation 1 -->
    <scale sx="ff" sy="ff" sz="ff" />                  <!-- initial scaling -->
    <reference length="ff" />                          <!-- axis length; "0" means no
axis-->
</INITIALS>



<ILUMINATION>
    <ambient r="ff" g="ff" b="ff" a="ff" />            <!-- global ambient -->
    <doubleside value="tt" />                          <!-- double or single side
illum. -->
    < background r="ff" g="ff" b="ff" a="ff" />         <!-- background color -->
</ILUMINATION>


<LIGHTS>
    <LIGHT id="ss">                                    <!-- light identifier -->
        <enable value ="tt" />                         <!-- enable/disable -->
        <position x="ff" y="ff" z="ff" w="ff" />       <!-- light position -->
        <ambient r="ff" g="ff" b="ff" a="ff" />        <!-- ambient component -->
        <diffuse r="ff" g="ff" b="ff" a="ff" />        <!-- diffuse component -->
        <specular r="ff" g="ff" b="ff" a="ff" />       <!-- specular component -->
    </LIGHT>

    <!-- NOTE: this block "LIGHT" must be repeated as necessary with different "id" -->

</LIGHTS>


<TEXTURES>
    <TEXTURE id="ss">
        <file path="ss" />                             <!-- path to file -->
        <amplif_factor s="ff" t="ff" />                <!-- x/s, y/t -->
    </TEXTURE>

    <!-- NOTE: this block "TEXTURE" must be repeated as necessary with different "id" -
->

</TEXTURES>


<MATERIALS>

    <MATERIAL id="ss">
        <shininess value="ff" />
        <specular r="ff" g="ff" b="ff" a="ff" />       <!-- specular reflection -->
        <diffuse r="ff" g="ff" b="ff" a="ff" />        <!-- diffuse reflection -->
        <ambient r="ff" g="ff" b="ff" a="ff" />        <!-- ambient reflection -->
        <emission r="ff" g="ff" b="ff" a="ff" />       <!-- emission component -->
     </MATERIAL>

    <!-- NOTE: the "MATERIAL" block may be repeated as required. Each defined material
requires a distinct "id" -->

</MATERIALS>


<LEAVES>
    <!-- next lines define nodes of type leaf; they may be repeated, in any order, as
necessary -->
     <LEAF id="ss" type="rectangle" args="ff ff ff ff" />     <!-- 2D coordinates for left-
top and right-bottom vertices. -->
    <LEAF id="ss" type="cylinder" args="ff ff ff ii ii" />    <!-- height, bottom radius, top
```

```
radius, sections along height, parts per section -->
    <LEAF id="ss" type="sphere" args="ff ii ii" />           <!-- radius, parts along radius,
parts per section -->
    <LEAF id="ss" type="triangle" args="ff ff ff  ff ff ff  ff ff ff" />   <!-- coordinates of
each vertex -->
</LEAVES>


<NODES>

    <ROOT id="ss" />     <!-- identifier of root node of the scene graph; this node     -->
                         <!--   must be defined in one of the following NODE declarations -->

    <NODE id="ss">       <!--   defines one intermediate node; may be repeated as necessary --
>

        <!-- next two lines are mandatory -->
        <MATERIAL id="ss" />      <!-- declared material superimposes the material received
from parent node -->
                                  <!-- id="null" maintains material from parent node        --
>
        <TEXTURE id="ss" />       <!-- declared texture superimposes the texture received from
parent node -->
                                  <!-- id="null" maintains texture from parent node       -->
                                  <!-- id="clear" clears texture declaration received from
parent node      -->

        <!-- geom. transf. are optional and may be repeated, in any order, as necessary
-->
        <TRANSLATION x="ff" y="ff" z="ff" />
        <ROTATION axis="cc" angle="ff" />
        <SCALE sx="ff" sy="ff" sz="ff" />

        < !-- declaring descendents, ate least one node or one leaf must be present -->
        <DESCENDANTS>
            <DESCENDANT id="ss" />       <!-- "ss" is the identifier of a node or of
leaf; -->
                                          <!"--     may be repeatad as necessary. It
can refer an -->
                                          <!--      identifier of another node, later
defined in the lsx file. -->
            </DESCENDANTS>

    </NODE>

</NODES>


</SCENE>
```
