# Laboratório de Aplicações com Interface Gráfica

**Aulas Práticas**

MIEIC – 2016/2017

## Trabalho 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 (ver secção 3 deste documento).

O ficheiro de texto deve respeitar uma "linguagem" própria, a que chamaremos DSX - *Description of Scenes in XML*, especificada na secção 3 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 ".DSX" 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 .dsx e devem poder ser alteradas por meio de controlos na interface gráfica 2D.

## 1.1 Componentes do Trabalho

- Preparar uma cena especificada num ficheiro .DSX 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.
- Implementar a componente de leitura e interpretação do ficheiro .DSX ( parsing), por recurso à biblioteca WebCGF (instruções e um exemplo estão disponíveis juntamente com este enunciado).
- Implementar uma estrutura de dados, que passa pelo grafo de cena apresentado na secção 1 deste documento.
- 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.

O trabalho deve ser desenvolvido de forma incremental:

- Versão inicial do *parser*, estrutura interna de dados e rotinas de visualização que permitam ler, armazenar e visualizar primitivas e transformações simples, ignorando inicialmente luzes, materiais e texturas.
- Versão progressivamente estendida do *parser*, estrutura e visualizador com as restantes funcionalidades.

---

### 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 .dsx, 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%.

Tendo em atenção as funcionalidades enunciadas abaixo, 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 |
| Fontes de Luz - descrição e ON/OFF | 3 valores |

---

### Datas Principais:

- Data limite de entrega do ficheiro .dsx: 17/10/2016
- Data limite de entrega do trabalho completo: 24/10/2016
- Por via eletrónica/moodle (instruções a divulgar proximamente).
- Avaliação do trabalho em aulas: semana com início em 24/10/2016
- Prova de avaliação individual: 26/10/2016

---

# 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 lhe corresponde no ficheiro .DSX. 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, cuja geometria é diretamente interpretada e desenhada por instruções WebGL. Deve por isso conter todos os parâmetros exigidos pela instrução WebGL.

## 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 a nós intermédios a declaração de eventuais transformações geométricas e propriedades de aspeto (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 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 da linguagem .DSX, na secção 3 deste documento.

# 2.3. Outras Entidades

Além de objetos, 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 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, a linguagem exige a sua declaração na parte inicial do ficheiro .DSX. 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.

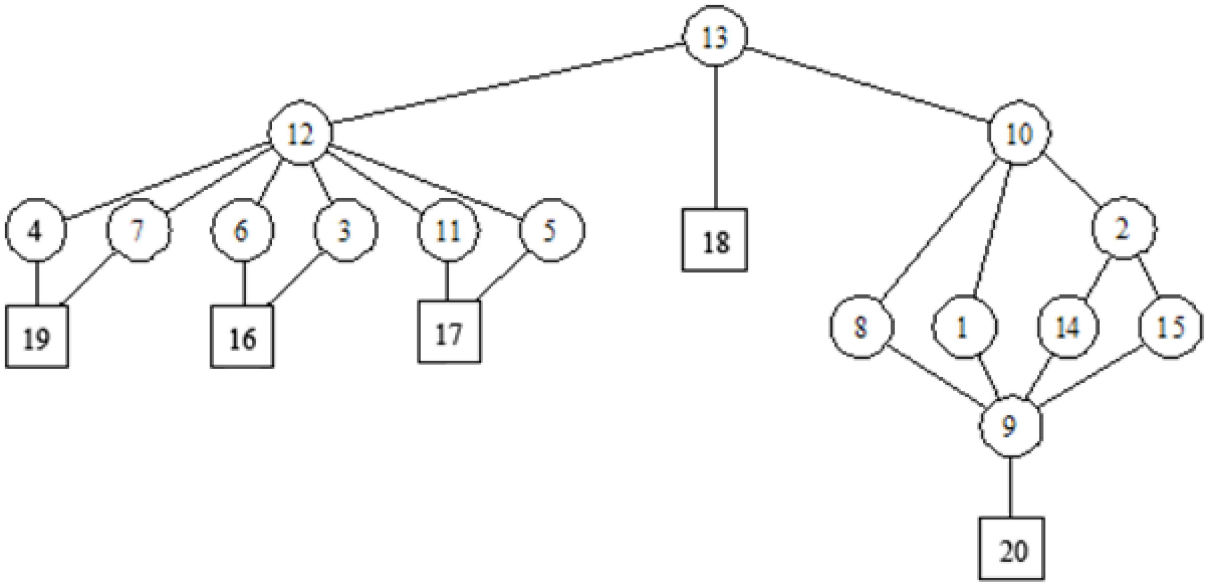


Figura 1 - Exemplo de um grafo de cena

Note-se que o número de descendentes 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 (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 a sua transformação geométrica, diferente das restantes.

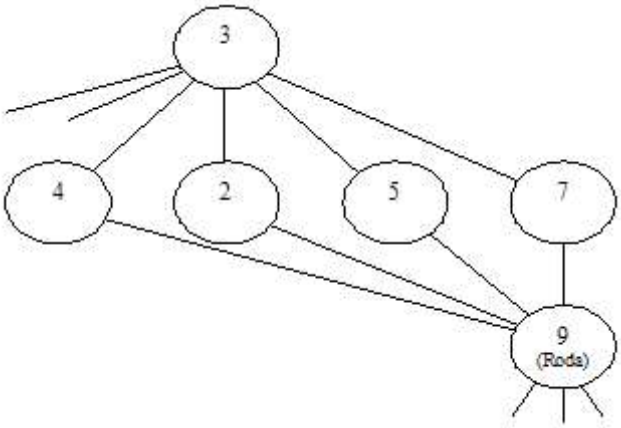


Figura 2 - Excerto de um grafo de cena com instanciação múltipla

# 3. Linguagem DSX

A linguagem DSX- *Description of Scene in XML* constitui um formato de especificação de cenas 3D de uma forma muito simples e fácil de interpretar. Um documento em linguagem DSX 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 DSX deve ser verificado em termos de sintaxe, devendo a aplicação gerar mensagens de erro ou avisos, identificando eventuais irregularidadesencontradas ou situações anómalas ou indesejáveis.
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.
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 DSX divide-se em blocos, cada um iniciando-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 terminando 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 -->
illumination         <!-- 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 -->
```

Os últimos seis blocos anteriores contêm definições de várias 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).

Uma listagem representativa da sintaxe pretendida pode ser encontrada no ficheiro anexo.

---