

# Laboratório de Aplicações com Interface Gráfica

## Aulas Práticas

MIEIC – 2015/2016

### Trabalho Prático 2 – Aperfeiçoamento das Técnicas de Utilização de WebGL

#### 1.Introdução

O objetivo deste trabalho é introduzir novas técnicas gráficas, como superfícies 2D/3D, animação, e algumas mais avançadas, como sejam os *shaders* baseados em GLSL ES 1.0 (OpenGL for Embedded Systems' Shading Language). Propõe-se assim a implementação de algumas funcionalidades em código, que possam depois ser exploradas através de uma extensão à linguagem LSX, e à criação de uma cena que as utilize. Este documento descreve as funcionalidades pretendidas, bem como a extensão proposta. Apesar de não ser obrigatório, recomenda-se a utilização/extensão do parser LSX, realizado no TP1, para suportar as novas funcionalidades solicitadas neste enunciado.

#### Funcionalidades pretendidas

##### 1 Animação

- \* Implementar um conjunto de classes para o suporte a animações.
  - a) Implementar a classe **Animation** como classe base para aplicar animações a um objeto.
  - b) Criar a classe **LinearAnimation**, derivada de **Animation**, para trajetórias lineares, que permita definir uma animação caracterizada por um vetor de Pontos de Controlo e tempo de duração total em segundos.

Exemplo:

Pontos de Controlo = {(0,0,0), (1,0,0), (1,1,0)}

Tempo= 10 s

O objeto em movimento deve alterar a sua orientação horizontal (x,z), rodando em torno de um eixo vertical, de modo a corrigir a direção quando, de acordo com a trajetória, muda de segmento de reta (ou seja, um movimento de "helicóptero").

- c) Criar a classe **CircularAnimation**, derivada de **Animation**, para trajetórias circulares, que permita definir uma animação caracterizada pelo centro e raio de circunferência, ângulo inicial (medido em relação à direção positiva do eixo **XX**) e ângulo de rotação, e tempo de duração em segundos.

Exemplo:

Centro = (10, 10, 10)

Raio = 5

Ângulo Inicial = 40°

Ângulo de rotação = 20°

Tempo= 20 s

O objeto em movimento deve alterar a sua orientação horizontal (x,z) de acordo com a trajetória, tal como no caso anterior.

- \* Implementar (em código ou usando a extensão proposta à LSX) uma animação para o veículo que inclua, no seu trajeto, pelo menos dois segmentos de reta, recorrendo para tal à classe **LinearAnimation**, e um segmento circular usando **CircularAnimation**. O objeto mantém a sua horizontalidade (ou seja, mantém-se paralelo ao plano XZ), apenas podendo rodar em torno do seu eixo vertical, de forma a manter uma orientação coerente com a direção e sentido do seu movimento (como p.ex. um helicóptero).

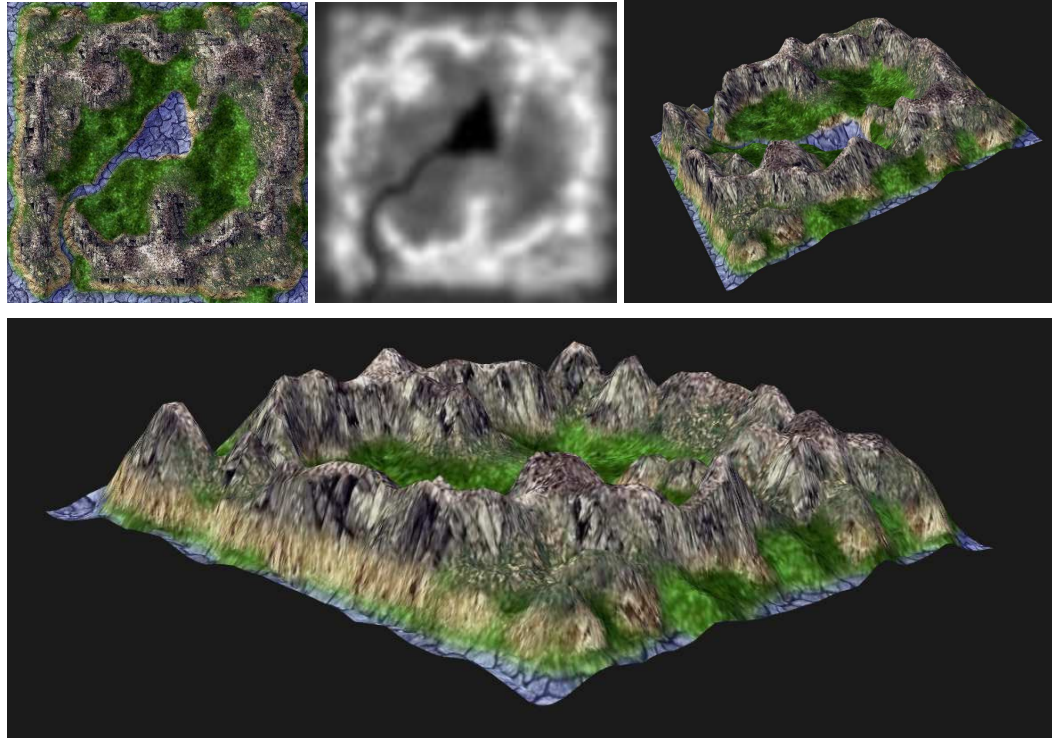
## 2 Superfícies 2D/3D

As superfícies vão ser modeladas através da representação *Non-uniform rational basis spline* (NURBS) ([https://en.wikipedia.org/wiki/Non-uniform\\_rational\\_B-spline](https://en.wikipedia.org/wiki/Non-uniform_rational_B-spline)).

- \* (Re)crie uma classe **Plane**, extensão de **CGFObject**, de forma a gerar, utilizando NURBS, um plano de dimensões 1 x 1 unidades, assente em XZ, centrado na origem e com a face visível apontando para +Y. Devem ser geradas também as coordenadas de textura para o plano, variando linearmente de (0,0) a (1,1). O número de divisões deve ser especificado no construtor da classe. Com esta classe, criar uma primitiva “**plane**” na linguagem LSX.
- \* Criar uma nova primitiva “patch” a incluir na linguagem LSX que possa representar superfícies de grau 1, 2 ou 3 nas duas direções U e V.
- \* Criar em código ou usando as extensões ao LSX, propostas abaixo, um novo objeto que corresponde a um veículo voador que inclua pelo menos uma superfície não-plana gerada utilizando a primitiva “patch”.

## 3 Shaders

Crie uma nova primitiva “**terrain**”, baseada na classe **Plane** criada anteriormente, para representar um terreno com elevações, recorrendo a uma textura de cor, um mapa de alturas e um par de shaders (vertex e fragment) - ver imagens abaixo.



(textura de cor, mapa de alturas e geometria gerada; origem das texturas: Outside of Society [http://oos.moxiecode.com/js\\_webgl/terrain/index.html](http://oos.moxiecode.com/js_webgl/terrain/index.html))

a) O vertex shader deve ser aplicado ao plano e alterar as coordenadas de cada vértice de forma a que a sua coordenada Y (altura) varie segundo a informação contida na imagem correspondente ao mapa de alturas.

b) O fragment shader deve aplicar a textura de cor sobre o terreno.

**NOTA:** as dimensões das texturas usadas devem ser potências de 2

## Requisitos da cena

Deve ser criada uma cena que utilize as funcionalidades referidas acima, nomeadamente:

- \* Instanciação e animação do veículo animado de acordo com as indicações acima (preferencialmente voando e/ou pousando/levantando).
- \* Utilização superfícies 2D e 3D.
- \* Instanciação de uma área de terreno usando a primitiva "terrain" proposta (numa parte do terreno usado, por exemplo uma região montanhosa envolvendo a cena, ou uma zona específica da cena com um determinado relevo).

**Nota Importante:** Apesar de não ser obrigatório, sugere-se a utilização/extensão do parser LSX realizado no TP1 para suportar as novas funcionalidades solicitadas neste enunciado.

## 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:** A avaliação será feita em aula prática, numa apresentação de cada grupo ao docente respetivo.

**Avaliação Individual:** Na prova de avaliação individual, serão pedidas várias funcionalidades adicionais, a implementar sobre o código original desenvolvido em trabalho de grupo.

**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 enuncionadas, serão considerados os seguintes critérios para efeitos de **Avaliação do Trabalho de Grupo**:

Estruturação e Documentação do código	3 valores
Animação de veículo	5 valores
Utilização de Superfícies 2D e 3D	5 valores
Instanciação de terreno com Shaders	5 valores
Aspeto geral e criatividade	2 valores

O enunciado incorpora, em cada alínea, a sua classificação máxima, correspondendo esta a um ótimo desenvolvimento, de acordo com os critérios seguintes, e que cumpra com todas as funcionalidades enunciadas. Sem perda da criatividade desejada num trabalho deste tipo, não serão contabilizados, para efeitos de avaliação, quaisquer desenvolvimentos além dos que são pedidos.

### Planeamento do Trabalho:

- \* Semana 1 (início em 19/10/2015): Animação
- \* Semana 2 (início em 26/10/2015): Superfícies de grau 1, 2 e 3
- \* Semana 3 (início em 02/11/2015): Semana de interrupção
- \* Semana 4 (início em 09/11/2015): shaders
- \* Semana 5 (início em 16/11/2015): entrega dos trabalhos dia 16, avaliação dos trabalhos de grupo durante as aulas práticas
- \* **Avaliação prática individual:** 18/11/2015

**Entrega:**

- \* Data limite de entrega do trabalho completo: 16/11/205
- \* Por via eletrónica/moodle (instruções a divulgar).

# Sugestão de extensão à Linguagem LSX

A linguagem LSX encontra-se definida no questionário do trabalho prático 1. Nesta secção são apresentadas as extensões ao formato LSX de modo a poder comportar as funcionalidades descritas neste enunciado.

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 mensagens de erro ou avisos, identificando eventuais erros encontrados ou situações anómalas ou indesejáveis.

Na descrição abaixo, os símbolos utilizados têm o seguinte significado:

ii: valor inteiro  
ff: valor em vírgula-flutuante  
ss: string  
ee: carácter "x" ou "y" ou "z", especificando um eixo  
tt: valor Booleano na forma "true" ou "false"

Segue-se uma listagem representativa da sintaxe pretendida, relativa às extensões à linguagem LSX. As tags / atributos acrescentados encontram-se escritos a vermelho. A cinzento encontram-se elementos definidos na versão original da linguagem LSX, usados para melhor contextualizar as alterações.

```
<SCENE>
...
<!-- informacao de animacao -->
<animations>
<!-- pode não existir qualquer nó "animation" se a cena não tiver
animações -->
<!-- span é o tempo, em segundos, que a animação deve demorar *
<!-- nesta versão do formato LSX, type pode ter o valor "linear" ou
"circular" -->
    <animation id="ss" span="ff" type="linear">
        <controlpoint xx="ff" yy="ff" zz="ff" />
        ...
    </animation>
    <animation id="ss" span="ff" type="circular" center="ff ff ff"
radius="ff" startang="ff" rotang="ff" />
</animations>
```

```

<LEAVES>
    <!-- Nova primitiva: plano, gerado por NURBS -->
    <!-- ex: <plane parts="5" /> um plano de dimensões 1 x 1
    unidades -->
    <!-- assente em XZ, centrado na origem e com a face visível
    apontando para +Y -->
    <!-- com divisão em cinco partes por eixo -->
    <LEAF id="ss" type="plane" parts="ii" />

    <!-- Nova primitiva: patch, gerada por NURBS -->
    <!-- parâmetros: -->
    <!-- order: ordem, pode ser 1,2,3-->
    <!-- partsU: divisão em partes no domínio U a ser usada para o
    cálculo da superfície-->
    <!-- partsV: divisão em partes no domínio V a ser usada para o
    cálculo da superfície -->
    <!-- o número de pontos de controlo dentro da primitiva patch
    é (ordem+1)^2 -->
    <LEAF id="ss" type="patch" order="ii" partsU="ii" partsV="ii"
>
    <controlpoint x="ff" y="ff" z="ff" />
    ...
</LEAF>

    <!--Nova primitiva: corresponde a um veículo voador. Inclui
    pelo menos uma superfície não-plana gerada utilizando evaluators -->
    <LEAF id="ss" type="vehicle"/>

    <!-- Nova primitiva: terreno baseado em shaders -->
    <!-- parâmetros: -->
    <!-- ficheiro de textura jpg ou png com a textura que deve ser
    visualizada sobre o terreno (dimensões devem ser potências de 2) -->
    <!-- ficheiro jpg ou png com o mapa de alturas que deve ser
    usado para formar o terreno (dimensões devem ser potências de 2) -->
    <LEAF id="ss" type="terrain" texture="ss" heightmap="ss"/>
    ...
</LEAVES>

```

```
<NODES>

    <!-- tem de existir, pelo menos, um bloco "node" -->

    <NODE id="ss">
        ...
        <!-- referência ao bloco de animação por cada "node". nó
opcional -->
        <animationref id="ss" />
    </NODE>
</NODES>
</SCENE>
```