Laboratório de Aplicações com Interface Gráfica

(Aulas Práticas)

MIEIC – 2017/2018

Trabalho Prático 2 – Aperfeiçoamento de Técnicas de Utilização de WebGL

O objetivo deste trabalho é introduzir novas técnicas gráficas, como animação e 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. O parser de LSX, realizado no TP1, deve ser utilizado e ampliado de forma a suportar as novas funcionalidades solicitadas neste enunciado.

1 Funcionalidades Pretendidas

1.1 Animação

Uma animação neste contexto corresponde a uma transformação geométrica representável em função do tempo. Assim, em intervalos de tempo aproximadamente regulares, a função devolve a transformação geométrica a aplicar a um objeto. Nota: assume-se que a "frente" do objeto aponta na direção positiva do eixo dos ZZ.

Nesta fase do trabalho, dever-se-á implementar um conjunto de classes `Animation`, `LinearAnimation`, `CircularAnimation`, `BezierAnimation`, `ComboAnimation`, de acordo com as sub-secções seguintes.

1.1.1 Classe Animation

Implementar a classe Animation como classe base (abstrata) para aplicar animações a um objeto.

1.1.2 Classe LinearAnimation

a) Criar a classe LinearAnimation, derivada de Animation, para trajetórias lineares, que permita definir uma animação caracterizada por um vetor de Pontos de Controlo e pela velocidade de movimentação (esta última medida em unidades 3D por segundo)

Exemplo:
Pontos de Controlo = {(0,0,0), (1,0,0), (1,1,0)}
Velocidade = 10 unidades/segundo

b) 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, o movimento é tipo "helicóptero").

1.1.3 Classe CircularAnimation

a) Criar a classe CircularAnimation, derivada de Animation, para trajetórias circulares em planos paralelos a ZX (rotação em torno de um eixo vertical), que permita definir uma animação caracterizada pelo centro e raio de circunferência, ângulo inicial (medido em relação à direção positiva do eixo XX), ângulo de rotação (este último a adicionar ao ângulo inicial), ambos em graus, velocidade linear (esta última correspondendo a um comprimento de arco, medido em unidades 3D, por segundo).

Exemplo:
Centro = (10, 10, 10)
Raio = 5
Ângulo Inicial = 40º
Ângulo de rotação = 20º
Velocidade = 10 unidades/segundo

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

1.1.4 Classe BezierAnimation

Criar a classe BezierAnimation para trajetórias no espaço 3D. A animação é caraterizada pelas coordenadas dos quatro pontos de controlo da curva de Bézier correspondente, assim como velocidade linear da mesma. O objeto animado deve manter uma orientação tangencial à trajetória no plano ZX, ou seja, rodando em torno do eixo vertical Y.

NOTA: o comprimento da curva pode ser aproximado pelos dois *convex hull* que resultam da aplicação do primeiro nível do algoritmo de Casteljau.

Exemplo:
P1, P2, P3, P4= {(0,0,0), (1,0,0), (1,1,0), (0,1,0)}
Velocidade = 10 unidades/segundo

1.1.5 Classe ComboAnimation

Implementar uma classe ComboAnimation que permita realizar uma animação constituída por uma sequência de instâncias das classes anteriores, segundo a extensão proposta abaixo para a linguagem LSX.

Um objeto animado desta forma deverá manter a sua horizontalidade (ou seja, manter-se paralelo ao plano XZ), apenas devendo alterar a sua orientação por rotação em torno de um eixo vertical; a orientação do objeto no plano ZX deve ser coerente com a direção e sentido do seu movimento (como p.ex. um helicóptero).

NOTA: uma ComboAnimation não pode conter outra ComboAnimation.

1.1.6 Aplicação em Cenas LSX

Criar uma cena em LSX que inclua objetos animados segundo as classes de animação definidas anteriormente.

No mínimo, deverá incluir duas animações:

a) Uma animação composta por, pelo menos, dois segmentos de reta (classe LinearAnimation), intermediados por um segmento circular (CircularAnimation).

b) Uma animação composta por, pelo menos, duas curvas de Bézier (classe BezierAnimation) e um ou mais troços com base em segmentos de reta ou circulares.

1.2 Modificação dinâmica de aparência com *shaders*

Para assinalar a seleção de um determinado objeto numa cena, pretende utilizar-se a técnica de *shaders* de forma a alterar a forma e cor do objeto dinamicamente e de forma periódica (ver imagem abaixo). Em particular, deverá ser implementado:

- Um *vertex shader* que altere a dimensão de um objeto, de forma semelhante aos exemplos fornecidos (movendo os vértices na direção das suas normais), mas variando o fator de escala em função do tempo.
- Um *fragment shader* que altere a cor do objeto entre o seu aspeto normal e uma cor saturada - cor de seleção.

(vídeo: https://drive.google.com/file/d/1JCyTflbdBNjsxO7POIV2K0pzJFUPjllJ/view?usp=sharing)

Sugere-se para este efeito que declarem uma variável do tipo *uniform float* (e.g. *timeFactor*) em ambos os *shaders*, que seja usada no cálculo dos *offsets* dos vértices e das cores, respetivamente.

Essa variável deve ser atualizada na função *update* da cena, através da função *setUniformsValues* do *shader* criado.

O seu valor pode ser calculado com base no tempo do sistema, e usando p.ex. uma função trigonométrica para obter um valor normalizado (a variar entre -1 e 1, ou entre 0 e 1).

Desta forma, este fator apenas é calculado uma vez em cada *update*, e é usado para todos os vértices/fragmentos que usarem o *shader.*

(Nota: podem e devem declarar outras variáveis que sejam úteis para parametrizar o *shader*, como por exemplo a cor de seleção).

1.2.1 Aplicação em cenas LSX

Deve estender o parser LSX de forma a que os nós possam ter um atributo opcional “selectable” do tipo booleano, que indica que um dado nó possa ser selecionado (ver .
No arranque do programa, deve ser gerada uma “list box” na interface, com a lista dos identificadores dos nós “selectable”, de forma a que o utilizador possa ativar **um** desses nós. O

nó selecionado deve ser desenhado com o shader de seleção. (encontram um exemplo de “listbox” na demonstração de shaders).

Nota: podem existir nós “selectable” na descendência de um dado nó “selectable”.

Devem apresentar uma cena que contenha três ou mais nós selecionáveis.

2 Notas sobre a avaliação do trabalho

2.1 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.

2.2 Avaliação do Trabalho de Grupo

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

2.3 Avaliação Individual

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

2.4 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%

2.5 Critérios de Avaliação do Trabalho de Grupo

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

Estruturação e Documentação do código	3 valores
Animação Linear	2 valores
Animação Circular	2 valores
Animação Bezier	3 valores
Animação Combo	3 valores
Vertex Shader	2.5 valores
Fragment Shader	2.5 valores
Interface, 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.

Entrega:

- Data limite de entrega do trabalho completo: 27/11/2017 - 08:00 (manhã)
- Por via eletrónica/moodle (instruções a divulgar).

Prova de avaliação individual:

- 29/11/2017

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: caracter "x" ou "y" ou "z", especificando um eixo
tt: valor Booleano na forma "true" ou "false"
```

Segue-se uma listagem representativa da sintaxe pretendida, no que respeita 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>
...
<!-- novo bloco ANIMATIONS, a colocar entre
     O bloco MATERIALS e o bloco NODES             -->

<ANIMATIONS>
    <!-- podem existir zero ou mais nós "animation" -->
    <!-- speed: velocidade LINEAR da animação -->
    <!-- type: pode ter valor "linear", "circular", "bezier" ou “combo”;
               na animação circular os angulos são em graus  -->

    <ANIMATION id="ss" speed="ff" type="linear">
        <controlpoint xx="ff" yy="ff" zz="ff /> <!-- repete conforme  -->
        <controlpoint xx="ff" yy="ff" zz="ff /> <!-- seja necessário  -->
                  ...
    </ANIMATION>

    <ANIMATION id="ss" speed="ff" type="circular"
             centerx="ff" centery="ff" centerz="ff"
             radius="ff" startang="ff" rotang="ff"
    />

    <ANIMATION id="ss" speed="ff" type="bezier">
        <controlpoint xx="ff" yy="ff" zz="ff />     <!-- P1 -->
        <controlpoint xx="ff" yy="ff" zz="ff />     <!-- P2 -->
```

```
        <controlpoint xx="ff" yy="ff" zz="ff" />     <!-- P3 -->
        <controlpoint xx="ff" yy="ff" zz="ff" />     <!-- P4 -->
    </ANIMATION>

    <ANIMATION id="ss" type="combo">    <!-- pode ser repetida para -->
                                        <!-- várias animações -->

        <!-- a próxima instrução deve existir pelo menos uma vez
             com id a referir-se a uma animação definida acima  -->
        <!-- as animacoes devem ser executadas sequencialmente
             pela ordem com que sao declaradas                    -->

       <SPANREF id="ss" />
             ...

    </ANIMATION>

</ANIMATIONS>

...

<NODES>

    <ROOT id="ss" />

    <NODE id="ss" selectable=”tt” > <!-- selectable opcional; falso por omissão-->
        <MATERIAL id="ss" />
        <TEXTURE id="ss" />

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

        <!-- Bloco opcional -->
        <ANIMATIONREFS>
              <!-- podem ser declaradas uma ou mais animacoes -->
              <!-- de qualquer tipo ou classe de entre as -->
              <!-- declaradas no bloco de animacoes; -->
              <!-- as animacoes devem ser executadas sequencialmente -->
              <!-- pela ordem com que sao declaradas.     -->
              <!-- NOTA: as transformacoes geometricas da -->
              <!-- animacao seguem as mesmas regras de ordenacao -->
              <!-- definidas no trabalho anterior para as -->
              <!-- transformacoes geometricas; dado que as -->
              <!-- animacoes sao declaradas apos as transformacoes -->
              <!-- geometricas do mesmo objeto, isso significa -->
              <!-- que as animacoes sao afetadas pelas transformacoes -->
              <!-- geometricas do objeto.  -->

              <ANIMATIONREF id="ss" />

        </ANIMATIONREFS>
```

```
        <DESCENDANTS>
            ...
        </DESCENDANTS>
    </NODE>

</NODES>

</SCENE>
```