Trabalho CG - 2013/14

LEI & LCC DI - UM

Pretende-se com este projecto que os alunos consolidem alguns dos conhecimentos leccionados na disciplina de CG. O objectivo final deste projecto é construir um mini motor 3D, e criar um exemplo de utilização que permita explorar algumas das potencialidades do motor criado.

Este trabalho deverá ser realizado em 4 fases, correspondendo a 4 entregas (todas a realizar via blackboard).

Fase 1 - Primitivas gráficas simples

Nesta fase dar-se-á início ao desenvolvimento do motor 3D por forma a que este desenhe modelos 3D armazenados em ficheiros individuais, num formato a definir pelo grupo.

Para criar estes ficheiros será necessária desenvolver uma aplicação (independente do motor) que receba como parâmetro o tipo de primitiva gráfica e os dados necessários à sua definição. Esta aplicação deverá criar um ficheiro onde irá armazenar os triângulos necessários ao desenho da primitiva.

Para a primeira fase pretende-se que esta aplicação seja capaz de gerar áreas planas, paralelepípedos, esferas e cones, no mínimo com os parâmetros das funções equivalentes de funções do GLUT.

Por exemplo, considerando que a aplicação tem por nome g*erador* e que se pretende criar uma esfera de raio 1, com 10 fatias e 10 camadas e armazenar a lista de triângulos correspondente no ficheiro “esfera.3d”, o seguinte comando poderá ser um exemplo de invocação desta aplicação:

```
gerador esfera 1 10 10 esfera.3d
```

O motor receberá como parâmetro um ficheiro XML onde serão definidos os dados da cena a desenhar. Para a primeira fase, este ficheiro de projecto conterá somente a indicação dos modelos a incluir na cena. Cada modelo deverá estar armazenado num ficheiro individual como uma sequência de triângulos, podendo conter mais informação se o grupo achar conveniente para a sua leitura. Após a leitura o motor deverá desenhar os modelos indicados.

Para fazer a leitura de ficheiros XML existem várias alternativas disponíveis como por exemplo o tinyXML (http://www.grinninglizard.com/tinyxml/).

Exemplo de um ficheiro XML para este efeito:

```
<cena>
     <modelo ficheiro=”esfera.3d” />
     <modelo ficheiro=”plano.3d” />
</cena>
```

Nota: Os ficheiros esfera.3d e plano.3d devem ter sido criados previamente pela aplicação mencionada acima.

Para a entrega e posterior apresentação, para além do código fonte e executáveis da aplicação que gera primitivas e do motor 3D, os grupos deverão entregar ficheiros com modelos diversos, e ficheiros XML que permitam ilustrar as funcionalidades especificadas.

Fase 2 – Transformações Geométricas

Nesta segunda fase pretende-se incluir transformações geométricas na definição da cena por forma a criar um modelo hierarquico.

Para tal considera-se que a cena é definida como uma árvore, em que cada nodo contém um conjunto de transformações geométricas (translação, rotação e escala) e um conjunto de modelos associado. Cada nodo poderá ter também associado um conjunto de nodos "filhos".

Exemplo de um ficheiro XML com um nodo:

```
<cena>
	<grupo>
		<translação X=5 Y=0 Z=2 />
		<rotação angulo=45 eixoX=0 eixoY=1 eixoZ=0 />
		<modelos>
			<modelo ficheiro="esfera.3d" />
		</modelos>
	</grupo>
</cena>
```

Exemplo de um ficheiro XML com um nodo "pai" e um nodo "filho":

```
<cena>
	<grupo>
		<translação X=1 />
		<modelos>
			<modelo ficheiro="esfera.3d" />
		</modelos>
		<grupo>
			<translação Y=1 />
			<modelos>
				<modelo ficheiro="cone.3d" />
			</modelos>
		</grupo>
	</grupo>
</cena>
```

Neste segundo exemplo, o grupo do cone, o nodo "filho" irá sofrer cumulativamente as transformações geométricas do nodo "pai" e do próprio nodo.

Nota: atenção à ordem das transformações geométricas.

Para a entrega e posterior apresentação, para além do código fonte e executáveis da nova versão do motor 3D, os grupos deverão entregar ficheiros com modelos diversos, e ficheiros XML que permitam ilustrar as funcionalidades especificadas, nomeadamente um modelo do sistema solar estático.

Fase 3 – Curvas, Superfícies e VAOs

Pretende-se nesta fase estender a aplicação geradora de primitivas por forma a que seja possível definir criar listas de triângulos correspondentes a superfícies de Bézier. A aplicação receberá como parâmetro um ficheiro onde se encontram definidos os vértices relativos aos pontos de controle e o grau de tesselação pretendido. O resultado será, tal como para as outras primitivas, um ficheiro com a lista de triângulos.

No âmbito do motor 3D, pretende-se a inclusão das curvas como elementos que definem uma translação, e a noção de tempo. Mais concretamente, em vez de especificar uma translação concreta como nos exemplos da fase 2, o parâmetro da translação será o conjunto dos pontos de controle de uma curva e o tempo (por exemplo em segundos) necessário para percorrer toda a curva. A translação concreta será calculada por frame em função da percentagem de tempo decorrido. Esta animação será cíclica, ou seja, quando a curva terminar regressa-se ao início da mesma. Esta funcionalidade implica que o motor 3D disponha de um relógio, ou contador de tempo (por exemplo: `glutGet(GLUT_ELAPSED_TIME)`).

```
...
<translação tempo=10 >
     <ponto X=1 Y=0 Z=1 />
     ...
     <ponto X=-1 Y=0 Z=1 />
</translação>
...
```

Nota: É sempre necessário um ponto extra no início da definição da curva e um ponto extra no final. O número mínimo de pontos é 4.

Pretende-se ainda que a rotação possa ser associada ao relógio, ou seja em vez de se especificar o ângulo de rotação especifica-se o tempo, em segundos, necessário para uma rotação de 360 graus.

```
...
<rotação tempo=10 eixoX=0 eixoY=1 eixoZ=0 />
...
```

Nesta fase é também suposto os grupos adaptarem o motor para utilizar VAOs em vez do modo imediato.

Para a entrega e posterior apresentação, para além do código fonte e executáveis, os grupos deverão entregar ficheiros com modelos diversos, e ficheiros XML que permitam ilustrar as funcionalidades especificadas, em particular um modelo do sistema solar animado (sol, oito planetas, a nossa lua, e um cometa em forma de bule de chá construído à custa de patches de Bézier.

Fase 4 – Normais e Coordenadas de Textura

Nesta fase pretende-se que a aplicação geradora de primitivas gere não só as coordenadas dos vértices mas também as normais e coordenadas de textura para cada vértice.

Na componente do motor será necessário activar a funcionalidade de iluminação e texturização, assim como permitir a especificação de cores e texturas no ficheiro XML.

Por exemplo:

```
      <modelo ficheiro="esfera.3d" textura="terra.jpg"/>
      <modelo ficheiro="sol.3d" diffR=0.8 diffG=0.8 diffB=0.15 />
```

Também é necessário definir as luzes que farão parte da cena no ficheiro XML que descreve a cena. Por exemplo:

```
<cena>
      <luzes>
            <luz tipo=POINT posX=0 posY=10 posZ=0 />
      </luzes>
      <grupo>
            <translação X=5 Y=0 Z=2 />
            <rotação angulo=45 eixoX=0 eixoY=1 eixoZ=0 />
            <modelos>
                  <modelo ficheiro="esfera.3d" />
            </modelos>
      </grupo>
</cena>
```

Datas de entrega

- Fase 1: 21 de Março
- Fase 2: 11 de Abril
- Fase 3: 2 de Maio
- Fase 4: 30 de Maio

Os prazos terminam à meia-noite dos respectivos dias. Os trabalhos podem ser submetidos com atraso, sendo que por cada dia de atraso será atribuída uma penalização de 10% sobre a fase respectiva. Para efeitos de penalização, o fim de semana conta como um único dia.

Na última fase deve ser submetido um relatório final do projecto.

Todas as submissões devem ser realizadas através do blackboard.

Cotação por cada fase: 15%

Relatório: 20%

Extra: 20% (exemplo: movimentação da câmara, cenas criadas, extensões ao formato do XML, view frustum culling, etc...)