Licenciatura em Ciências da Computação

Computação Gráfica

Curves, Cubic Surfaces and VBOs

Filipe Barbosa	Luís Bigas	Hugo Ferreira
A77252	A76964	A78555

15 de Abril de 2019

Conteúdo

Capítulo 1

Introdução

1.1 Introdução

No âmbito da unidade curricular Computação Gráfica, foi proposta a extensão da segunda fase deste trabalho.

Nesta terceira fase foram expandidos os dois programas **engine.cpp** e **generator.cpp**, com novas funcionalidades de geração e transformação de figuras num espaço tridimensional.

Capítulo 2

Utilização e Compilação do programa

2.1 Compilação

- make eng

  compila a engine.cpp

- make gen

  compilação do generator.cpp

2.2 Utilização

Geração dos modelos para serem utilizados no programa principal.

- ./generator sphere 1 10 10 sphere.3d
- ./generator box 1 1 1 10 box.3d
- ./generator teapot.patch 40 teapot.3d
- ./engine "nome_ficheiro".xml

  utilizando um ficheiro XML apropriado

Capítulo 3

Ficheiro XML

3.1 Estrutura

Para esta fase do trabalho foram adicionadas mais opções de Nodos para a criação de um ficheiro XML, estas permitem realizar a movimentação das figuras pretendidas num espaço 3D, através de movimentos de rotação e translação com curvas Catmull-Rom.

Novos Nodos Possíveis no ficheiro XML:

- <translate time="tempo">
  <point X="x_cord" Y="y_cord" Z="z_cord" >
  .....
  <point X="x_cord"Y="y_cord"Z="z_cord">
  </translate>
- <rotate time="tempo" X="x_rot"Y="y_rot"Z ="z_rot"/>

3.1.1 Exemplo

Exemplo 1:

Uma esfera percorre a curva definida pelos 6 pontos indicados, percorrendo todo o percurso em 50 segundos.

```
<scene>
    <group>
        <translate time="50" >
            <point X="40" Y="0" Z="200" />
            <point X="-40" Y="0" Z="200" />
            <point X="-70" Y="0" Z="0" />
            <point X="-40" Y="0" Z="-200" />
            <point X="40" Y="0" Z="-200" />
            <point X="70" Y="0" Z="0" />
        </translate>
        <models>
            <model file="sphere.3d" />
        </models>
    </group>
</scene>
```

Exemplo 2:

Uma esfera roda em relação ao eixo dos Y's, uma rotação a cada 10 segundos.

```
<scene>
    <group>
        <rotate time="10" axisX="0" axisY="1" axisZ="0" />
        <models>
            <model file="sphere.3d" />
        </models>
    </group>
</scene>
```

Capítulo 4

Engine

O programa engine nesta fase foi expandido para realizar o parsing das novas regras do ficheiro XML e para a sua execução.

4.1 Parsing

Na fase de parsing do programa fizemos algumas alterações.
Foram criadas novas estruturas:

- **modelPoints**

  Guarda os pontos representantes de um modelo um vector de floats. E um parâmetro para cada uma das três componentes da cor (RGB).

- **translateT**

  Guarda os pontos de controlo para o calculo da curva Catmull num multi vector, e o tempo total para percorrer a curva.

- **rotateT**

  Guarda um parâmetro com o tempo para a execução da rotação e um float para cada uma das coordenadas para a direção desta rotação.

4.2 VBO's

Para utilizar vbo's inicializamos um buffer com **GLuint buffers[1]**.
Em vez de imprimir cada um dos vértices das figuras um a um, guardamos os pontos do modelo atual no buffer e então são passados os parâmetros necessários para a função **glDrawArrays()**, o que permite uma melhor performance do programa.

4.3 Curvas Catmull-Rom

Na engine utilizamos a matéria lecionada durante as aulas desta cadeira para a implementação do calculo de posições nestas curvas.
Para o calculo dos pontos numa curva necessitamos de no mínimo 4 pontos de controlo, dados pelo ficheiro XML.

Cada uma das movimentações ao longo da curva é calculada pela função **catmullMov()** que recebe o tempo total para percorrer o percurso e um vetor com todos os pontos de controlo fornecidos. E sub-sequencialmente utiliza **glTranslatef()** para colocar a matriz na posição correta para o movimento.

Começamos por utilizar as funções fornecidas nas aulas pelo professor, mas devido a problemas de execução do programa optamos por realizar os cálculos de forma direta na função **getCatmullRomPoint()**

Capítulo 5

Generator

O programa generator foi expandido para possibilitar a leitura de um ficheiro de Patches de Bezier, e posteriormente realizar o calculo dos pontos representantes da superfície gerada pelos diferentes patches.

5.1 Patches de Bezier

5.1.1 Parsing

Para processar um ficheiro de Patches, o programa vai lendo o ficheiro dado linha a linha.

Com a primeira linha do ficheiro guardamos o número da patches. Seguidamente guardamos numa matriz as patches e os respetivos índices dos pontos de controlo (16 pontos de controlo por patch).

Seguidamente é lido o quantos pontos disponíveis temos e os próprios pontos de controlo ficam guardados noutra matriz.

5.1.2 Calculo de Pontos

Os pontos calculados dependem do nível de tecelagem pretendido pelo utilizador, dado na execução por programa.

Desta forma temos dois ciclos que percorrem o nível de tecelagem na "vertical"e "horizontal".
Por cada iteração do ciclo interior calculamos um conjunto que 4 pontos que irão representar dois triângulos.

Não entrarei mais em detalhe na resolução desta parte do trabalho pois não conseguimos calcular os pontos corretos de forma a criar forma pretendida, sendo possível que o programa ainda seja substancialmente alterado para a resolução do problema.

Capítulo 6

Sistema Solar

Por fim podemos demonstrar o funcionamento do nosso programa.

Como pedido no enunciado do trabalho apresentamos um sistema solar dinâmico, com um cometa que percorre todo este sistema num género de uma elipse cujos pontos são gerados por uma curva CatMull, neste caso o cometa é apenas uma esfera já que não conseguimos calcular o modelo com os patches de bezier.

6.0.1 Ficheiro XML Utilizado

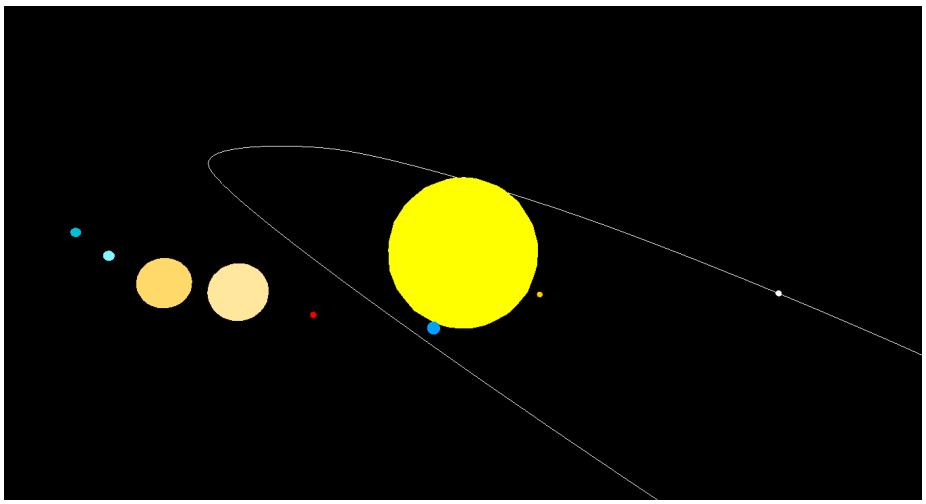
```
<scene>
    <group>
        <!-- SOL -->
        <group>
            <scale X="30" Y="30" Z="30" />
            <rotate time="50" axisX="0" axisY="1" axisZ="0" />
            <models>
                <model file="sphere.3d" R="9833" G="1" B="0" />
            </models>

        </group>

        <!-- MERCURIO -->
        <group>
            <rotate time="100" axisX="0" axisY="1" axisZ="0" />
            <translate X="0" Y="0" Z="35" />
            <rotate time="10" axisX="0" axisY="1" axisZ="0" />
            <models>
                <model file="sphere.3d" R="0.65" G="0.4117" B="0" />
            </models>
        </group>
```

```
<!-- VENUS -->
<group>
    <rotate time="150" axisX="0" axisY="1" axisZ="0" />
    <translate X="0" Y="0" Z="45" />
    <scale X="1.5" Y="1.5" Z="1.5" />
    <rotate time="10" axisX="0" axisY="1" axisZ="0" />
    <models>
        <model file="sphere.3d" R="1" G="0.8" B="0" />
    </models>
</group>

<!-- TERRA -->
<group>
    <rotate time="200" axisX="0" axisY="1" axisZ="0" />
    <translate X="0" Y="0" Z="60" />
    <scale X="2" Y="2" Z="2" />
    <rotate time="20" axisX="0" axisY="1" axisZ="0" />
    <models>
        <model file="sphere.3d" R="0" G="0.6333" B="1" />
    </models>
</group>

<!-- MARTE -->
<group>
    <rotate time="270" axisX="0" axisY="1" axisZ="0" />
    <translate X="0" Y="0" Z="70" />
    <scale X="1.7" Y="1.7" Z="1.7" />
    <rotate time="20" axisX="0" axisY="1" axisZ="0" />
    <models>
        <model file="sphere.3d" R="1" G="0" B="0" />
    </models>
</group>

<!-- JUPITER -->
<group>
    <rotate time="350" axisX="0" axisY="1" axisZ="0" />
    <translate X="0" Y="0" Z="85" />
    <scale X="10" Y="10" Z="10" />
    <rotate time="5" axisX="0" axisY="1" axisZ="0" />
    <models>
        <model file="sphere.3d" R="1" G="0.905" B="0.62" />
    </models>
</group>

<!-- SATURNO -->
```

```
    <group>
        <rotate time="400" axisX="0" axisY="1" axisZ="0" />
        <translate X="0" Y="0" Z="110" />
        <scale X="9" Y="9" Z="9" />
        <rotate time="5" axisX="0" axisY="1" axisZ="0" />
        <models>
            <model file="sphere.3d" R="1" G="0.8525" B="0.41" />
        </models>
    </group>

    <!-- URANO -->
    <group>
        <rotate time="500" axisX="0" axisY="1" axisZ="0" />
        <translate X="0" Y="0" Z="140" />
        <scale X="2" Y="2" Z="2" />
        <rotate time="5" axisX="0" axisY="1" axisZ="0" />
        <models>
            <model file="sphere.3d" R="0.53" G="0.9608" B="1" />
        </models>
    </group>

    <!-- NEPTUNO -->
    <group>
        <rotate time="600" axisX="0" axisY="1" axisZ="0" />
        <translate X="0" Y="0" Z="170" />
        <scale X="2" Y="2" Z="2" />
        <rotate time="5" axisX="0" axisY="1" axisZ="0" />
        <models>
            <model file="sphere.3d" R="0" G="0.7517" B="0.82" />
        </models>
    </group>
</group>


<!-- COMETA -->
<group>
    <translate time="50" >
        <point X="40" Y="0" Z="200" />
        <point X="-40" Y="0" Z="200" />
        <point X="-70" Y="0" Z="0" />
        <point X="-40" Y="0" Z="-200" />
        <point X="40" Y="0" Z="-200" />
        <point X="70" Y="0" Z="0" />
    </translate>
    <models>
        <model file="sphere.3d" />
```

```
        </models>

    </group>

</scene>
```

6.0.2 Resultado de Execução



./engine config.xml

Capítulo 7

Conclusão

Nesta fase do trabalho expandimos o nosso conhecimento quanto ao calculo e funcionamento relativo às curvas CatMull-Rom e Patches de Bezier.

Conseguimos compreender o impacto de performance entre diferentes formas de desenho em OpenGL.

Sentimos grande dificuldade no calculo das curvas. Assim como no calculo dos Patches, que esperamos conseguir resolver mais tarde.