Computação Gráfica

Mestrado Integrado em Engenharia Informática
3ºano-2ºsemestre

José Pinto A81317 Luís Correia A81141
Pedro Barbosa A82068

Trabalho Prático - Fase 3

Resumo

Foi-nos proposto na UC de Computação Gráfica o desenvolvimento de um mini mecanismo 3D baseado num cenário gráfico utilizando as ferramentas utilizadas na cadeira, *C++* e *OpenGL*.

Este relatório é referente à terceira fase onde nos foi proposto o uso de VBO's para gerar os modelos previamente implementados, a construção de objetos a partir de superfícies de *Bezier* e a implementação de transformações geométricas em função do tempo. No final são demonstrados os resultados obtidos através de uma modelação do sistema solar.

Abril 2019

Conteúdo

Lista de Figuras

1 Introdução

Nesta fase do trabalho prático foi-nos proposto o uso de VBO's para desenhar os modelos previamente criados, de modo a tornar o programa mais eficiente uma vez que os VBO's permitem que a informação seja pré carregada na placa gráfica e não na memória central do sistema. O uso de índices ajuda a reduzir a quantidade de informação redundante. Foi-nos também pedido que fosse possível gerar superfícies de *Bezier* a partir de ficheiros no formato *patch*. Por último, foi pedido a implementação de duas transformações geométricas em função do tempo, a translação e a rotação. Na translação foram utilizadas as curvas de *Catmull-Rom*. Uma vez implementadas as funcionalidades, o ficheiro XML que representa o Sistema Solar foi alterado para que os planetas e um cometa(representado pelo *teapot*) orbitassem o Sol e girassem sobre eles próprios.

2 Vertex Buffer Objects

Os algoritmos desenvolvidos na primeira fase geravam os vértices pela ordem em que eles iriam ser desenhados sem ter em consideração a possível repetição destes. De forma a evitar este problema e de forma a tirar melhor proveito dos *Vertex Buffer Objects* do *OpenGL*, os algoritmos foram adaptados de forma a calcularem dois resultados. Todos os vértices únicos de um modelo e uma lista dos índices dos vértices gerados pela ordem que devem ser desenhados. Toda esta informação é escrita num ficheiro de formato 3d. Neste formato, a primeira linha indica o número de índices, a segunda linha contem os índices separados por vírgulas, a terceira linha contem o número de vértices e as restantes linhas contêm cada uma um vértice.

2.1 Vértices

O processo para gerar os vértices é quase idêntico ao da primeira fase com a diferença de que em cada iteração apenas é calculado um vértice, em vez de todos os vértices de um ou mais triângulos. Deste forma não são calculados vértices repetidos. O desafio desta tarefa encontra-se em gerar os vértices por uma ordem que torne fácil determinar os índices adequados. De seguida segue-se um exemplo para o caso de uma esfera com 5 *stacks* e 9 *slices*.

O primeiro vértice a ser gerado é o do topo da esfera. Depois são gerados todos os vértices que separam cada camada. No final é gerado o vértice no fundo da esfera. A ordem resultante é representada pela figura 1.

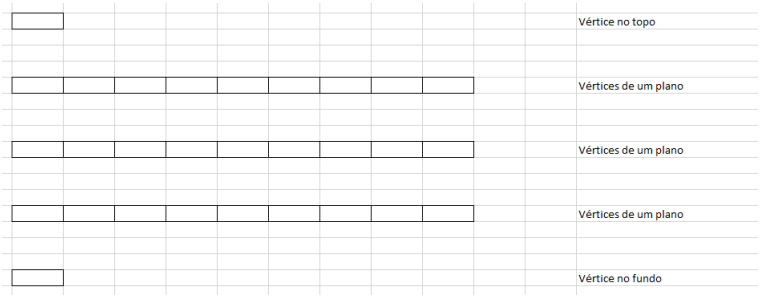

Figura 1: Ordem dos vértices gerados

Desta forma é relativamente simples determinar a lista de índices necessária para desenhar uma esfera.

2.2 Índices

Novamente, os algoritmos são semelhantes aos das fases anteriores. Em vez de em cada iteração serem calculados os vértices para desenhar um triângulo, são calculados os índices que representam esses vértices. Voltando ao caso da esfera, a figura 2 representa exemplos de associações de vértices e respetivos índices que permitem desenhar os triângulos da superfície da esfera.

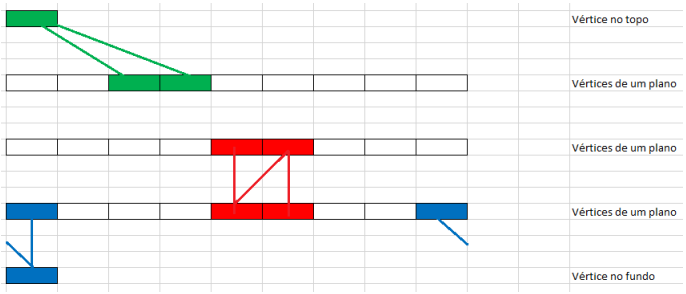

Figura 2: Associações de vértices que dão origem aos triângulos pretendidos

Por exemplo, para obter um dos triângulos da camada superior da esfera basta seleccionar sempre o índice 0 (vértice superior), um índice no intervalo 1 a $N + 1$, sendo N o número de *slices*, e o próximo índice (considerando que o índice a seguir ao $N + 1$ é o 1). Ao iterar para todos os índices no intervalo referido então é possível obter todos os triângulos da camada superior da esfera. Para a camada inferior da esfera o processo é o mesmo. Nas restantes camadas o processo é semelhante. Escolhe-se um índice I de um dos vértices do topo da camada. Depois seleciona-se os índices $I+1$, $I+N$ e $I+N+1$. (não esquecendo a exceção do último ponto de cada intervalo). Com os índices dos 4 vértices é possível obter 2 triângulos que compõem uma secção da superfície da esfera.

2.3 Engine e Buffers

Ao processar o ficheiro 3d são criados dois *arrays*, um para os índices e outro para os vértices. A ordem de ambos é conservada de forma a que os índices indiquem a posição correta no *array* de vértices e para que sejam desenhados os triângulos corretos. Na primeira chamada da função *renderScene* são gerados e preenchidos os *buffers* de vértices e índices de cada grupo. Quando a *renderScene* é novamente invocada não é necessário gerar e preencher novamente os *buffers*, bastando apenas aplicar as transformações geométricas necessárias, ativar os *buffers* corretos e desenhar os triângulos.

3 Bezier Patch

No contexto do projeto, uma superfície de *Bezier* é definida por 16 pontos de controlo. Para obter a informação sobre estes pontos é necessário efetuar o processamento do ficheiro em formato *patch*. Para cada *patch* é criado um *array* com os seus índices. É ainda criado um *array* com todos os pontos de controlo presentes no ficheiro, conservando a ordem destes.

Para calcular os vários pontos da superfície de *Bezier* foi usada a seguinte fórmula:

$$M = \begin{bmatrix} -1 & 3 & -3 & 1 \\ 3 & -6 & 3 & 0 \\ -3 & 3 & 0 & 0 \\ 1 & 0 & 0 & 0 \end{bmatrix}$$

$$B(u,v) = \begin{bmatrix} u^3 & u^2 & u & 1 \end{bmatrix} * M * \begin{bmatrix} P_{00} & P_{01} & P_{02} & P_{03} \\ P_{10} & P_{11} & P_{12} & P_{13} \\ P_{20} & P_{21} & P_{22} & P_{23} \\ P_{30} & P_{31} & P_{32} & P_{33} \end{bmatrix} * M^T * \begin{bmatrix} v^3 \\ v^2 \\ v \\ 1 \end{bmatrix}$$

Os parâmetros u e v variam entre 0 e 1. Para calcular as três coordenadas de um ponto é necessário utilizar a fórmula 3 vezes, uma para cada coordenada, sendo a terceira matriz composta pelas coordenadas respetivas dos pontos de controlo.

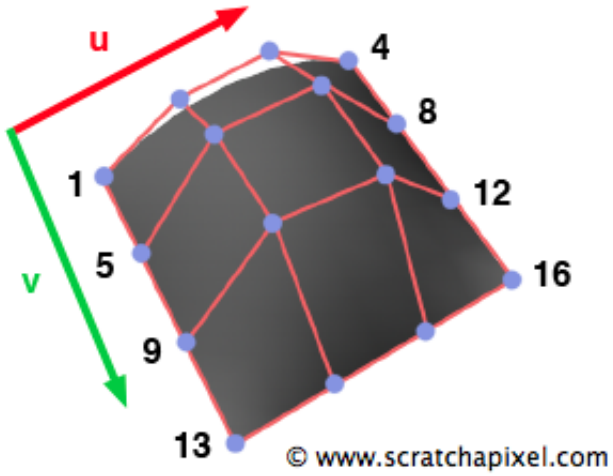

Figura 3: Pontos de controlo de uma superfície *Bezier*

De forma simplificada o significado dos parâmetros u e v é o seguinte: o valor de u implica uma deslocação nas curvas definidas pelos pontos $1-2-3-4$, $5-6-7-8$, $9-10-11-12$ e $13-14-15-16$. Esta deslocação dá origem a 4 pontos que podem ser usados para auxiliar na definição de uma outra curva na direção do eixo do v. Uma deslocação nesta curva de acordo com o parâmetro v dá origem a um ponto da superfície de *Bezier*. Ao aplicar a fórmula para diferentes valores de u e v obtém-se vários pontos pertences à superfície pretendida.

Para obter o nível de tesselação pretendido (t) é necessário iterar os valores de u e v entre 0 a 1 em intervalos de $1/t$. Em termos de implementação isto corresponde a dois ciclos aninhados, sendo que cada um é responsável pelo aumento de cada parâmetro e cada iteração dá origem a um ponto. Os pontos são gerados pelo ordem representada na figura 4.

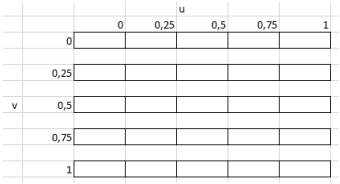

Figura 4: Ordem em que são gerados os vértices

Para cada ponto gerado com os parâmetros u_i e v_j, sendo u_i e v_j diferentes de 1, são selecionados também os pontos gerados pelos parâmetros (u_i, v_j+1/t), (u_i+1/t ,j_v) e (u_i+1/t ,j_v+1/t). Estes 4 pontos permitem gerar 2 triângulos que compõem parte da superfície pretendida.

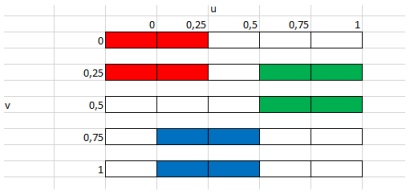

Figura 5: Exemplos de associações entre vértices

4 Extensão da translação e rotação

4.1 Translação

Nesta fase do projeto é necessário a implementação de uma funcionalidade que permita uma translação dependente do tempo. A translação é controlada por um conjunto de pontos de controlo que definem uma curva *Catmull-Rom* cúbica e por um valor em segundos que indica o tempo que demora a percorrer a curva.

A qualquer momento durante a execução é necessário ser possível determinar em que segmento da curva é que o modelo deve estar. O resto da divisão do tempo desde o início de execução pela duração total da curva corresponde ao tempo desde o início da volta atual. Na situação em que sejam necessários 10 segundos para percorrer uma curva e já passaram 25 segundos desde o início de execução, o resto da divisão de 25 com 10 é 5 ou seja, já passaram 5 segundos desde o início da volta atual. Os cálculos a efetuar necessitam de um parâmetro t entre 0 e 1. Esse valor é obtido ao dividir o resto pela duração total da curva.

Como a curva é cúbica, na eventualidade de a curva ser definida por mais do que 4 pontos de controlo é necessário determinar que pontos usar. Para tal utilizamos o método utilizado nas aulas práticas(MANTER ISTO?). Primeiro multiplica-se o valor obtido na etapa anterior pelo número de pontos de controlo. O resultado permite tanto identificar o segmento atual como o novo parâmetro t correspondente ao segmento. Por exemplo, se o resultado for 2,5 então o segmento é aquele entre o segundo e o terceiro ponto de controlo e metade desse segmento já foi percorrido. Os pontos de controlo a utilizar são os 2 que definem o segmento atual, o ponto anterior e o ponto a seguir ao segmento.

Uma vez escolhidos os pontos de controlo e um valor para o parâmetro t, já devidamente enquadrado no segmento em questão, vai ser possível calcular as coordenadas do ponto que queremos.

$$Ponto = \begin{bmatrix} t^3 & t^2 & t & 1 \end{bmatrix} * \begin{bmatrix} -0.5 & 1.5 & -1.5 & 0.5 \\ 1 & -2.5 & 2 & -0.5 \\ -0.5 & 0 & 0.5 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix} * \begin{bmatrix} P_0 \\ P_1 \\ P_2 \\ P_3 \end{bmatrix}$$

4.2 Rotação

À semelhança da translação, a rotação também vai precisar de calcular o tempo de maneira a descobrir em que posição se encontra na rotação (resto da divisão inteira entre o tempo desde o inicio do programa e o tempo de rotação). Uma vez calculado este valor, vai-se multiplicar o mesmo por 360 para obter um ângulo em graus que diz respeito à etapa da rotação atual.

5 Modelo do Sistema Solar

De maneira a representar o trabalho desenvolvido, o modelo do sistema solar desenvolvido para a fase anterior foi atualizado para encorporar as extensões feitas às transformações geométricas. Neste novo modelo, os planetas seguem uma trajetória definida por uma curva *Catmull-Rom* cúbica que representa a órbita do planeta. Para além disso, a cada planeta é aplicada uma rotação dependente o tempo sobre ele próprio. É ainda modelado um cometa a partir de superfícies de *Bezier*. Na figura seguinte, é possível verificar todos os planetas e algumas luas, assim como as respetivas órbitas em torno do Sol, ou no caso da Lua, em torno da Terra.

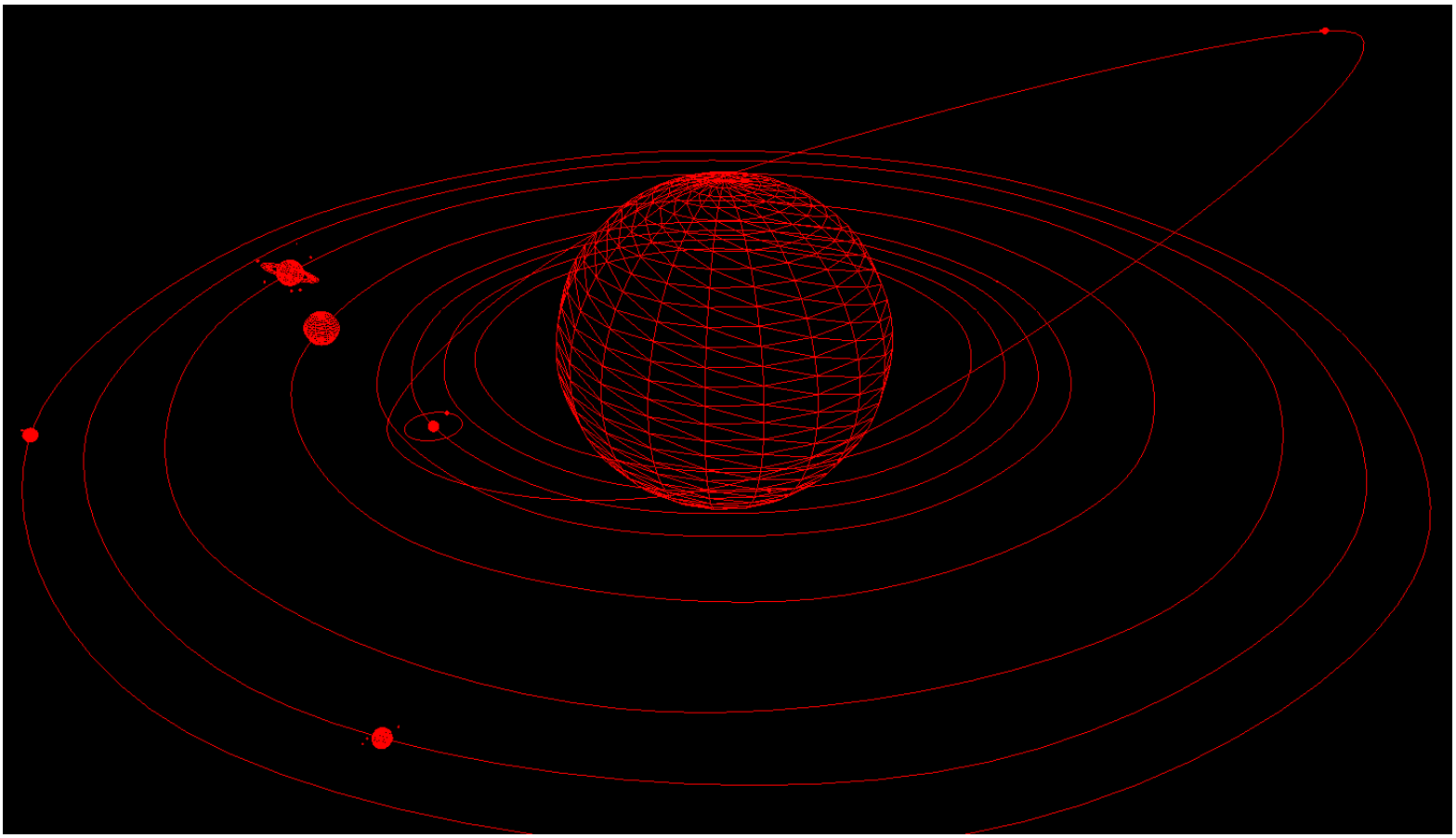

Figura 6: Modelo do Sistema Solar

De forma a representar a a órbita em torno do Sol, a rotação e a órbita da Lua foi usada a seguinte configuração.

```
<group>
    Terra
    <translate time="30" >
        <point X="100" Y="0" Z="0" />
        <point X="70.71067" Y="0" Z="-70.71067" />
        <point X="0" Y="0" Z="-100" />
        <point X="-70.71067" Y="0" Z="-70.71067" />
        <point X="-100" Y="0" Z="0" />
        <point X="-70.71067" Y="0" Z="70.71067" />
        <point X="0" Y="0" Z="100"/>
        <point X="70.71067" Y="0" Z="70.71067"/>
    </translate>
    <rotate time="10" axisX="0" axisY="1" axisZ="0" />
    <scale X="0.3" Y="0.3" Z="0.3" />
    <models>
        <model file="../../ModelsSolarSystem/esfera.3d" />
    </models>
    <group>
        Lua
        <translate time="10" >
            <point X="27" Y="0" Z="0" />
            <point X="19.09188" Y="0" Z="-19.09188" />
            <point X="0" Y="0" Z="-27" />
            <point X="-19.09188" Y="0" Z="-19.09188" />
            <point X="-27" Y="0" Z="0" />
            <point X="-19.09188" Y="0" Z="19.09188" />
            <point X="0" Y="0" Z="27"/>
            <point X="19.09188" Y="0" Z="19.09188"/>
        </translate>
        <scale X="0.37" Y="0.37" Z="0.37" />
        <models>
            <model file="../../ModelsSolarSystem/esfera.3d" />
        </models>
    </group>
</group>
```

Figura 7: Configuração XML da Terra

6 Conclusão

Esta fase contribuiu para solidificar alguns conhecimentos adquiridos nas aulas teóricas. As vantagens do uso de *Vertex Buffer Objects* tornaram-se evidentes. No início do desenvolvimento desta fase a tarefa de implementar superfícies *Bezier* e curvas *Catmull-Rom* parecia intimadora. Ao programar e a testar os resultados obtidos tornou-se mais claro o funcionamento destes métodos. Por último, a modelação do sistema solar serviu para testar e validar a implementação das funcionalidades propostas.