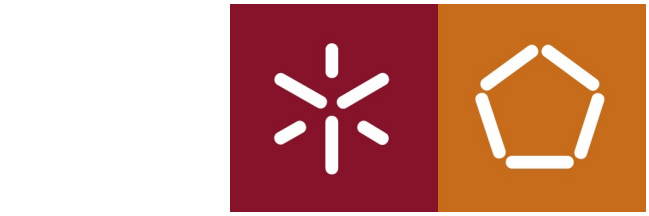
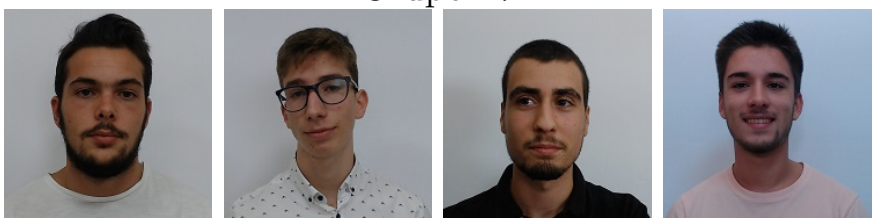


Universidade do Minho
Mestrado Integrado em Engenharia Informática
3ºano - 2º Semestre

Computação Gráfica

Relatório sobre a Fase 3

Grupo 47



a83732 – Gonçalo Rodrigues Pinto
a84197 – João Pedro Araújo Parente
a84829 – José Nuno Martins da Costa
a85059 – Diogo Paulo Lopes de Vasconcelos

4 de Maio de 2020

Conteúdo

Lista de Figuras

1 Introdução

No 2º semestre do 3º ano do Curso de Engenharia Informática da Universidade do Minho, existe uma unidade curricular denominada por Computação Gráfica, que tem como objectivo ajudar os estudantes caracterizar as transformações geométricas e os referencias utilizados na computação gráfica, aplicar transformações para construção de modelos geométricos complexos e posicionamento da câmara, dar a conhecer algoritmos de iluminação local e global tais como Gouraud, Phong, Ray-tracing, Radiosity and Virtual Point Lights, têm também como meta que os estudantes apliquem texturas e definam coordenadas de textura, além de permitir uma análise de soluções do ponto de vista do desempenho recorrendo a profilers, utilizando apropriadamente soluções de eliminação de geometria, recorrendo a partição espacial e por fim aplicação de análise de algoritmos para geração de sombras

O presente trabalho pretendeu desenvolver um mecanismo 3D baseado em mini gráficos de cenas e fornecendo exemplos de uso que mostrem o seu potencial.

2 Contexto

Partindo do trabalho realizado na fase 2, onde foram utilizadas todas as classes previamente criadas na fase 1, no qual se alterou a forma como a aplicação Engine efectua a leitura do documento XML, deixando a parte referente à leitura do documento com extensão .3d e posteriormente representação em formato matricial da mesma forma procedida na fase 1.

De forma a realizar as transformações geométricas requeridas anteriormente, como foi abordado reestruturou-se as estruturas de dados para armazenar a informação que os ficheiros XML possuem.

3 Fase 3 - Curvas, Superfícies Cúbicas e VBOs

Nesta fase, foi nos requerido que a aplicação Generator fosse capaz de criar um novo tipo de modelo baseado nos patches de Bezier. O Generator deve receber como parâmetro o nome de um documento em que os pontos de controlo de Bezier são definidos, bem como é requerido o nível de tessellation (divisões). O documento resultante vai conter uma lista dos triângulos para desenhar a superfície.

Em relação à aplicação Engine, foi nos solicitado estender as transformações geométricas mais precisamente a translação e a rotação. Considerando a translação, será fornecido um conjunto de pontos para definir uma curva Catmull-Rom cúbica, bem como o número de segundos para percorrer toda a curva. O objectivo é realizar animações com base nestas curvas. Os modelos podem ter uma transformação dependente do tempo, ou uma transformação estática como nas fases anteriores. No nó de rotação, o ângulo pode ser substituído pelo tempo, ou seja, o número de segundos para realizar uma rotação completa de 360 graus em torno do eixo especificado.

Foi nos sugerido para medir o tempo utilizar a função glutGet(GLUT_ELAPSED_TIME).

```
...
<translate time=10 >
      <point X=1 Y=0 Z=1 />
      <point X=0.707 Y=0.707 Z=1 />
      <point X=0 Y=1 Z=1 />
      ...
      <point X=-1 Y=0 Z=1 />
</translate>
...
```

Figura 1: Exemplo da translação para esta fase.

É de realçar que devido à definição da curva Catmull-Rom é sempre exigido um ponto inicial antes do segmento da curva inicial e outro ponto após o último segmento. O número mínimo de pontos é de 4.

```
...
<rotate time=10 axisX=0 axisY=1 axisZ=0 />
...
```

Figura 2: Exemplo da rotação para esta fase.

Nesta fase foi também exigido que os modelos sejam desenhados com VBOs, por oposição ao modo imediato utilizado nas fases anteriores.

A cena de demonstração foi um sistema solar dinâmico, incluindo um cometa com uma trajectória definida através de uma curva Catmull-Rom. O cometa foi construído utilizando patches Bezier, por exemplo, com os pontos de controlo fornecidos para o bule.

3.1 Aplicação Generator

3.1.1 Estruturas utilizadas

As estruturas utilizadas para esta fase foram as seguintes:

- **guarda_pontos**: uma lista para guardar os vários pontos;
- **pointer_pontos**: é um inteiro que diz quantos pontos temos no "guarda pontos";
- **patches**: outro inteiro para guardar o número de patches;
- **Estrutura patch**: uma estrutura que possui como variáveis um array de inteiros designado *indices* para armazenar os índices dos pontos que pertencem ao patch em consideração e um inteiro *pointer_indices* para armazenar o número de indíces que tem o array;
- **patchss**: um array de estruturas *patch*, para guardar todos os patchs que fazem parte de um objeto;
- **final**: um array com uma matriz de pontos para cada patch (esta matriz é uma matriz onde a ordem por que estão os pontos representa os triângulos, para melhor esclarecimento consultar os relatórios das fases anteriores, onde é explicado ao pormenor);

3.1.2 Abordagem

Como foi abordado previamente foi solicitado nesta fase complementar a aplicação Generator criada até ao momento com criação um novo tipo de modelo baseado nos patches de Bezier.

A curva de Bezier é uma curva polinomial expressa como a interpolação linear entre alguns pontos representativos, chamados de pontos de controlo. Pontos esses que se encontram num ficheiro tal como o número de divisões que irá ser processado pela aplicação.

De forma a atingir esse fim após ser passado todos os parâmetros o Generator irá iniciar a leitura do documento, onde guarda o valor da primeira linha que contém o número de patches na variável (*patches*). Posteriormente, percorre todos esses patches, efectuando a leitura de cada linha que contém os índices de cada patch e respectivamente guarda na estrutura *patchss* na posição correcta.

Portanto neste momento, o Generator já tem todos os pontos de controlo na sua memória e a indicação para cada patch os pontos que fazem parte dele, após isto para cada patch vai ser calculado os pontos de Bezier para a tesselation dada e colocados numa matriz, como foi dito anteriormente as posições da matriz já representam os triângulos e tendo assim esta matriz podemos reutilizar o código das fases anteriores, nomeadamente a função (*"escreve ficheiro"*) que pegando numa matriz escreve-a para um ficheiro que depois o engine irá ler.

3.2 Aplicação Engine

Nesta fase o Engine passou a suportar translações e rotações como animações, sendo necessário fornecer o tempo (em segundos) que representa a duração da animação, tendo em conta que se no caso de uma translação são necessários no mínimo 4 pontos para que seja possível calcular a posição actual, através de curvas CatMull-Roll, na animação, se for uma rotação o eixo em torno de que se vai fazer a rotação.

Nota: Nestas rotações novas passa a ser escusado fornecer o ângulo, já que nestas o ângulo máximo será 360 e estará a evoluir constantemente dependendo da duração da animação.

3.2.1 Estruturas utilizadas

As estruturas continuam a ser as da fase anterior, apenas com uma pequena mudança em relação à estrutura *grupo*, que agora vai conter informação necessária para desenharmos os objectos com VBO's pois foi um requisito para esta fase e adicionou-se alguns atributos à estrutura *operacao* para que fosse possível guardar o tempo de uma animação.

3.2.2 VBOs

Como referido anteriormente, nesta fase é necessário desenhar os objectos com VBO's, como tal decidimos optar por utilizar VBO's com índices, pois é uma forma de não termos informação de pontos repetida. É de notar que VBO's são afectados pelas transformações geométricas e portanto não os temos de alterar cada vez que haja uma transformação, sendo assim possível só alterar os VBO's na sua criação. Ao aplicar uma transformação geométrica a um VBO ela é aplicada a todo o VBO, não existe maneira de só aplicar a parte do VBO portanto convêm juntar todos os objectos num único VBO que sofre todos as transformações, ou seja, todos os objectos que estão ao mesmo nível num grupo, por isso mesmo em cada grupo terá dois VBO's (pontos e índices) que representarão todos os objectos desse grupo nesse nível (não vai incluir os objectos dos grupos dentro deste grupo).

3.2.3 Abordagem

Os cálculos referentes às translações foram simplesmente aplicação das formulas de CatMull-Roll, e como na rotações foi necessário termos uma evolução de tempo e calcular basicamente uma velocidade da animação e a partir disso calcular a posição actual da animação.

Devido ao facto de termos feito a representação dos objectos em forma de matriz auxiliou-nos bastante, pois a conversão para um vector para ser usado nos VBO's é bastante fácil e o cálculo do vector dos índices também pois o código para calcular os triângulos dessa matriz já estava todo realizado devido a ter sido desenvolvido nas fases anteriores.

4 Cena de demonstração

A cena de demonstração foi um sistema solar dinâmico, incluindo um cometa com uma trajectória definida através de uma curva Catmull-Rom. O cometa foi construído utilizando patches Bezier, por exemplo, com os pontos de controlo fornecidos para o bule de chá.

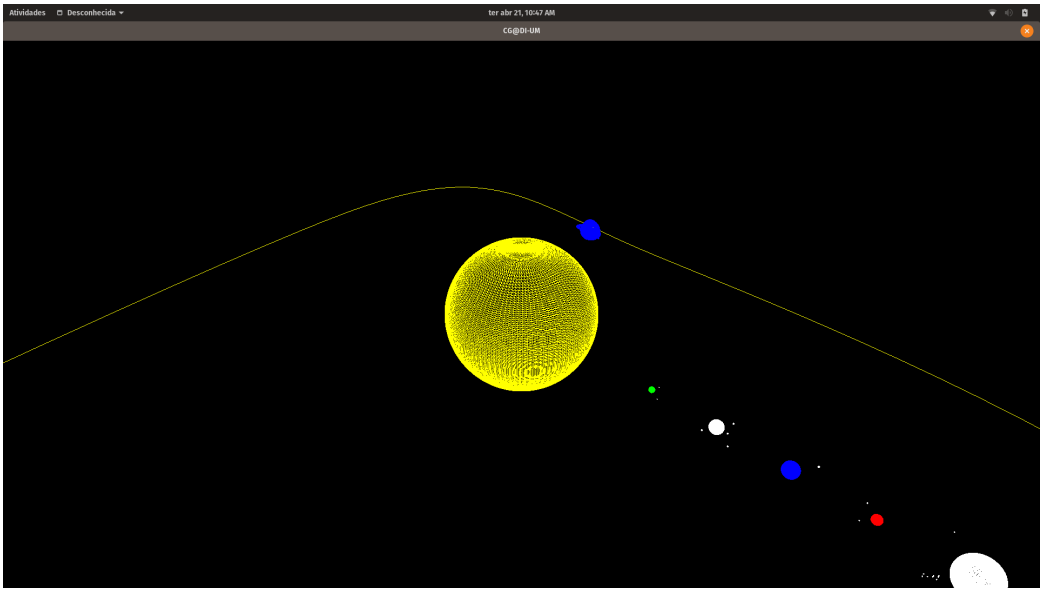


Figura 3: Cena de demonstração.

Para incluir o cometa com uma trajectória adicionou-se ao ficheiro XML desenvolvido na fase anterior que desenhava um sistema solar um grupo com a indicação da cor e também uma translação com um determinado tempo. sdddd

5 Conclusão

O presente relatório descreveu, de forma sucinta, a resolução desta fase mediante os requisitos apresentados.

Consideramos que os principais objectivos foram cumpridos.

Sentimos que a realização deste projecto consolidou os nossos conhecimentos em ferramentas associadas à computação gráfica como o OpenGL e o GLUT, permitiu adquirir conhecimentos da linguagem C++, essencial para o desenvolvimento desta fase e conseguimos obter uma noção dos algoritmos que estão associados à criação de algumas primitivas gráficas.

Em suma, esperamos que o trabalho realizado até ao momento seja essencial e fulcral para as fases seguintes do projecto.