Universidade do Minho

CG - Fase 3 do Trabalho Prático

Grupo nº39

Hugo Filipe de Sá Rocha
(A96463)

Gabriel Alexandre Monteiro da Silva
(A97363)

José Diogo Lopes Faria
(A95255)

5 de maio de 2023

Conteúdo

Capítulo 1

Contextualização

1.1 Alterações no Generator

Nesta terceira fase do trabalho prático, na aplicação **generator**, adicionamos um novo modelo que consiste em desenhar um teapot através da análise de um ficheiro **patch** que contém os pontos de controlo de **Bezier** que nos permitem desenhar a cena.

1.2 Alterações no Engine

Quanto ao **Engine**, foi atualizado de forma a conseguir efetuar rotações em função de um parâmetro temporal e ainda a renderizar curvas de **Catmull-Rom** e conseguir colocar modelos a percorrer as mesmas, eventualmente, de forma alinhada (não necessariamente). Além disso, o desenho dos modelos está ser efetuado com os respetivos vértices armazenados em **VBOS**.

Capítulo 2

Concepção da solução

2.1 Teapot (generator)

Para a análise do ficheiro patch, criámos uma estrutura **map** que, para cada **patch**, associa um **vetor** de pontos que corresponde aos respetivos pontos de controlo de **Bezier**. Após ter essa informação, percorremos cada patch, e para cada coordenada X,Y e Z dos pontos de controlo, criámos uma matriz 4 por 4 que será multiplicada à esquerda e à direita pela matriz de Bezier obtendo ainda como resultado uma matriz 4 por 4. Esta última, será multiplicada à esquerda por um vetor de valores exponenciais **U** e à direita por um outro vetor de valores exponenciais **V**, obtendo assim uma coordenada de um ponto da curva. Os parâmetros u e v, que definem os vetores exponenciais, percorrem valores entre **0 e 1**.

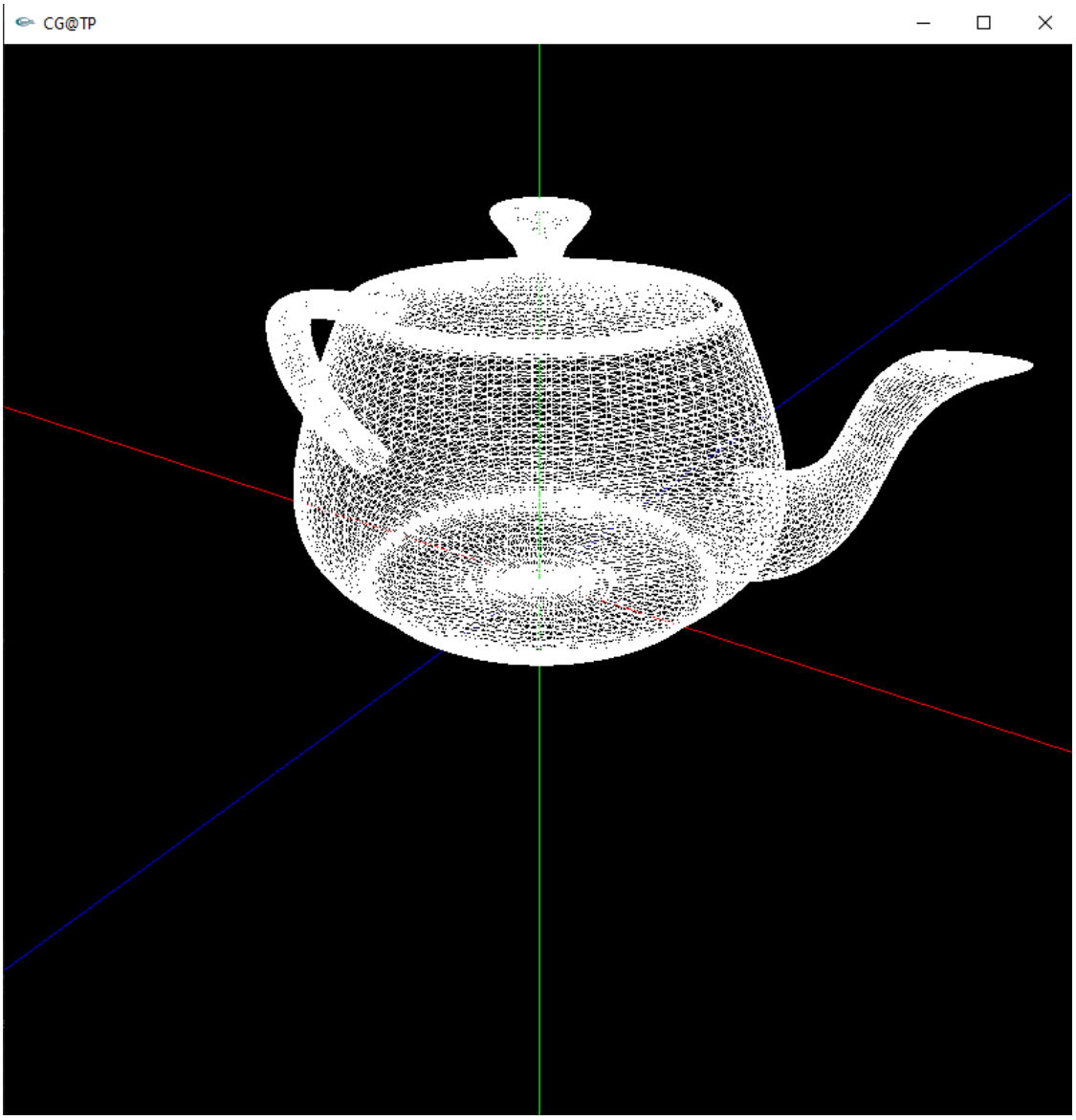

Figura 2.1: Teapot

2.2 Engine

Nesta aplicação foi necessário adaptar a função de **parsing** do ficheiro XML de forma a ler corretamente os dados necessários para as novas transformações. Na struct **Transformação** adicionamos um novo campo, um vetor de pontos, que guarda os pontos de controlo das curvas de **Catmull-Rom**. Além disso, acrescentámos outros três atributos booleanos **align**, **align_y** e **align_z** que nos dirão se o objeto deve percorrer a curva de forma alinhada, ou não. Por fim, acrescentámos também o atributo **time** que, no caso de uma rotação em função do tempo, nos dirá o tempo necessário (em milissegundos) para o objeto rodar 360^{o}. No caso das curvas, diz-nos o tempo necessário (em milissegundos) para um modelo percorrer toda a curva.

2.2.1 Rotação em função do tempo

Para efetuar esta transformação, precisamos do **ângulo de rotação** a cada instante que foi obtido através de uma regra de três simples, fazendo uso da função **glutGet(GLUT_ELAPSED_TIME)** e do campo **time** da nossa estrutura. O eixo de rotação é-nos dado diretamente do ficheiro XML.

2.2.2 Curvas de Catmull-Rom

Para a elaboração das curvas, definos uma função de **renderização** da curva que serve para desenhar a curva no ecrã em função de um **vetor** de **pontos de controlo**. Para saber quais os pontos que fazem parte da curva, definimos a função **getGlobalCatmullRomPoint** que dado um parâmetro **gt**, que varia entre **0 e 1** para toda a curva, diz-nos qual as coordenadas do respetivo ponto na curva e ainda a sua derivada. Para obter as coordenadas do ponto na curva, multiplicou-se a **matriz de Catmull-Rom** por um vetor de coordenadas (X,Y ou Z), e, de seguida, multiplicou-se o resultado disto à esquerda por um vetor de valores exponenciais. Para obter a **derivada** nesse ponto, o processo é semelhante, a única alteração é na ultima multiplicação, onde se **deriva o vetor exponencial**. O global **gt** para cada curva é incrementado em função do tempo de forma a garantir que o modelo percorra toda a curva no tempo pedido.

Para o **alinhamento** de um modelo na curva, considerou-se por predefinição, que o **eixo do X** corresponde à **derivada** (normalizada) em cada ponto da curva. Para descobrir o **eixo do Z**, efetuou-se o **produto direto do eixo do X pelo eixo do Y**, voltando-se a normalizar o resultado final. Começou-se por assumir que o eixo do Y corresponde ao vetor **(0,1,0)** para a primeira iteração. Por fim, o **eixo do Y** corresponde ao **produto direto**

do eixo do X pelo eixo do Z, não sendo aqui necessário normalizar visto o eixo do X e do Z já estarem normalizados. Por fim, constrói-se uma **matriz de rotação** com base nestes três eixos.

2.2.3 Desenhos dos modelos através de VBOS

Para desenhar os modelos através de **VBOS**, associou-se um **buffer** a cada modelo e visto termos, para cada modelo, um vetor do tipo **Ponto**, bastou convertê-lo para um vetor do tipo **float** e preencher os respetivos buffers para posteriormente desenhá-los.

2.3 Sistema Solar

Relativamente ao sistema solar e a sua dinâmica, definimos uma curva de **Catmull-Rom** para cada planeta de forma a definir a sua órbita. Cada uma dessas curvas é definida por **oito pontos de controlo** e possuem uma forma aproximadamente circular. Para descobrir esses pontos de controlo, os tempos de órbita em torno do Sol e os tempos de rotação sobre si mesmo, para cada planeta e de forma **proporcionalmente realista**, utilizámos uma pequena script em **Python**. Dois dos pontos de controlo encontram-se no eixo do **X** pelo que a coordenada é a distância do planeta ao sol positiva ou negativa. Dois outros pontos encontram-se no eixo do **Z** e as coordenadas são as mesmas. Os restantes quatro pontos possuem coordenadas polares, nomeadamente a distância do planeta ao sol multiplicada pelo **sin**$\pi/4$ ou pelo **cos**$\pi/4$ (a menos do sinal), em teoria. Visto estes valores serem iguais, utilizou-se apenas o valor do seno.

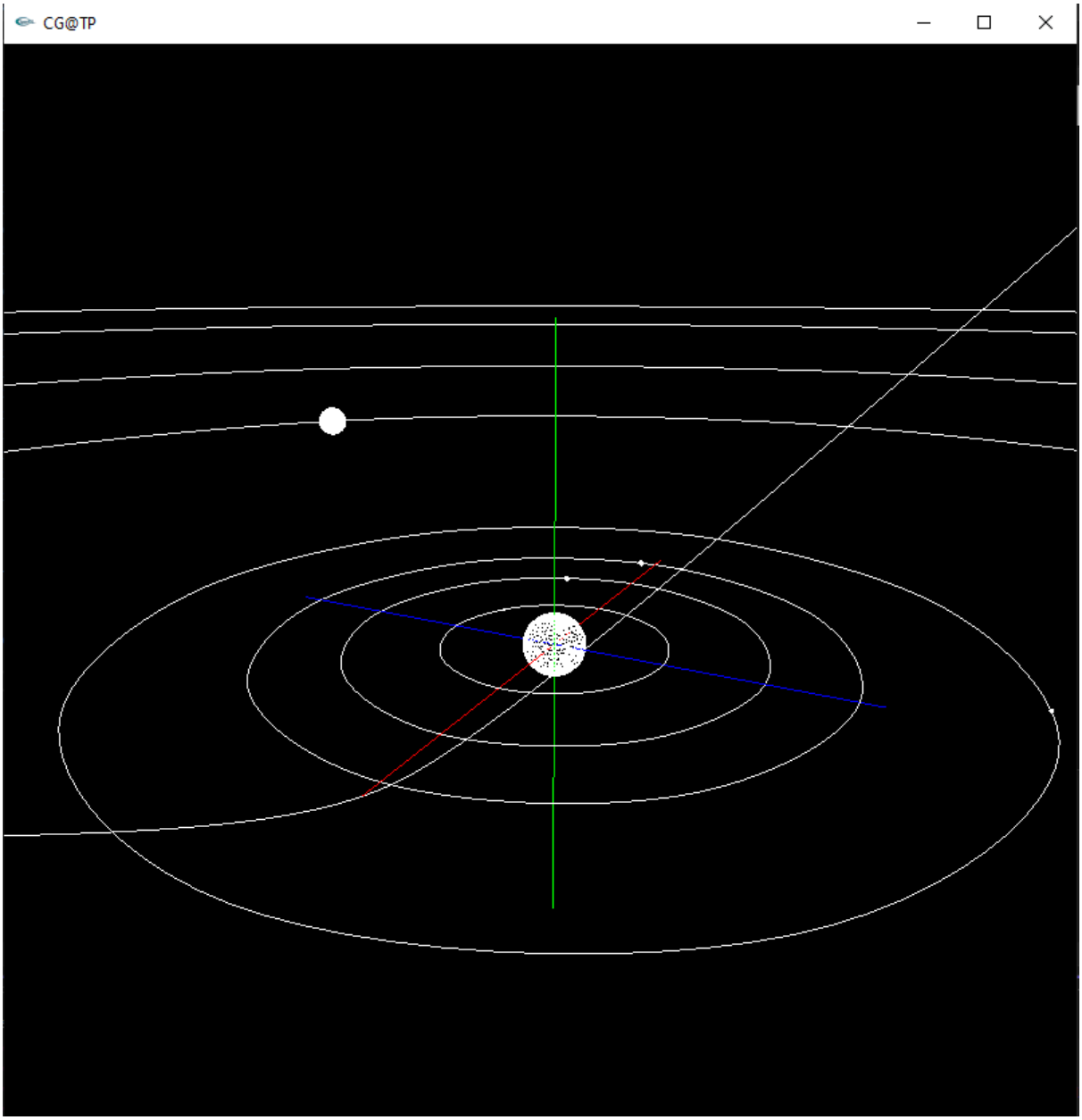

Figura 2.2: Sistema Solar

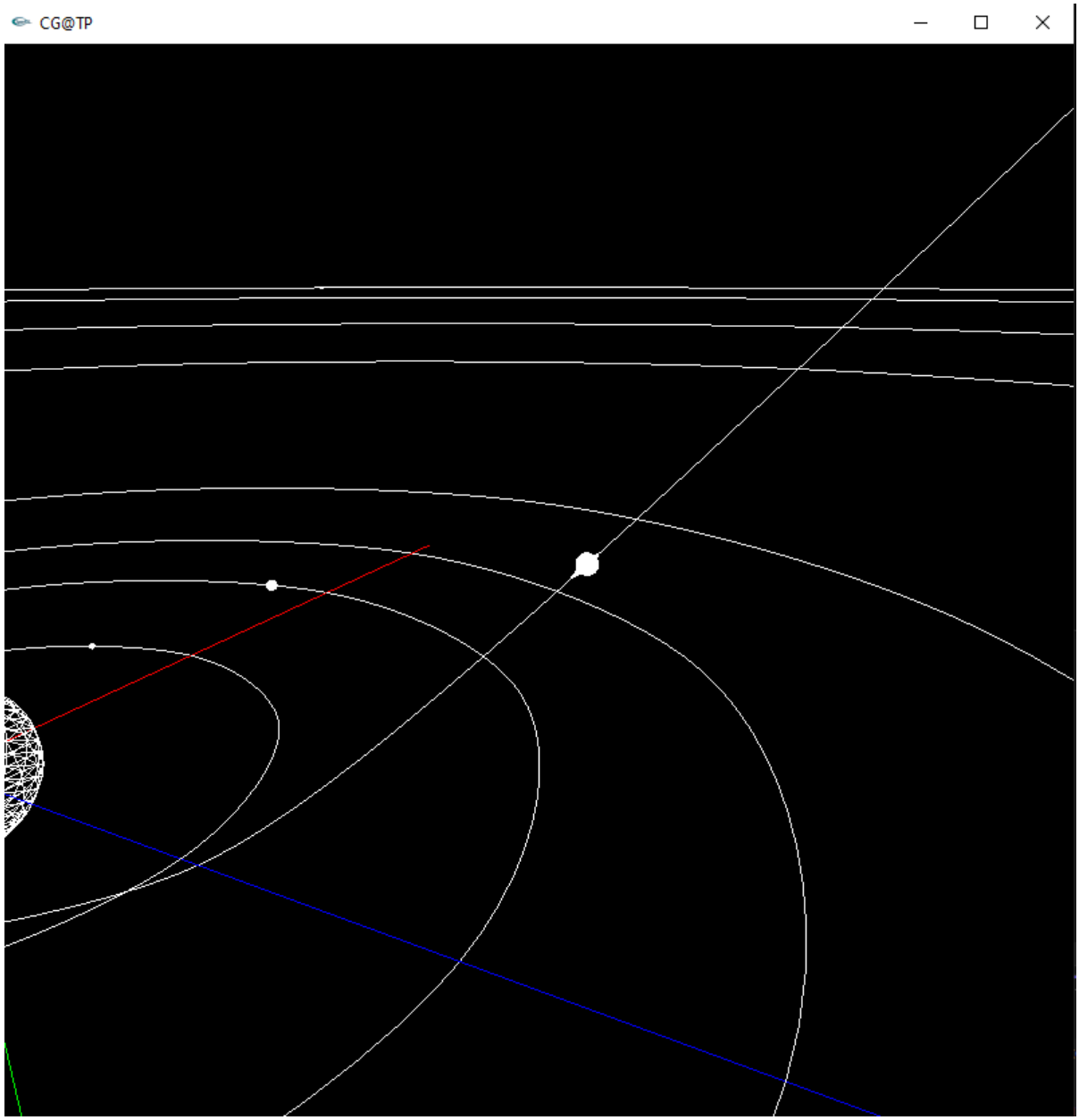

Figura 2.3: Cometa

2.4 Extras

Nesta terceira fase do trabalho implementámos um extra que consiste em adicionar **mais opções de alinhamento** dos modelos ao longo das curvas de **Catmull-Rom**. Além da predefinição, onde a derivada em cada ponto corresponde ao eixo do X, permite-se no ficheiro XML adicionar um atributo **aligny** ou **alignz** onde **a derivada passa a corresponder ao eixo do Y e ao eixo do Z respetivamente**. No caso de o atributo ser o **align** então o eixo do X fica associado à derivada. O processo de cálculo da matriz de rotação é semelhante, trocando-se apenas a definição de cada eixo.

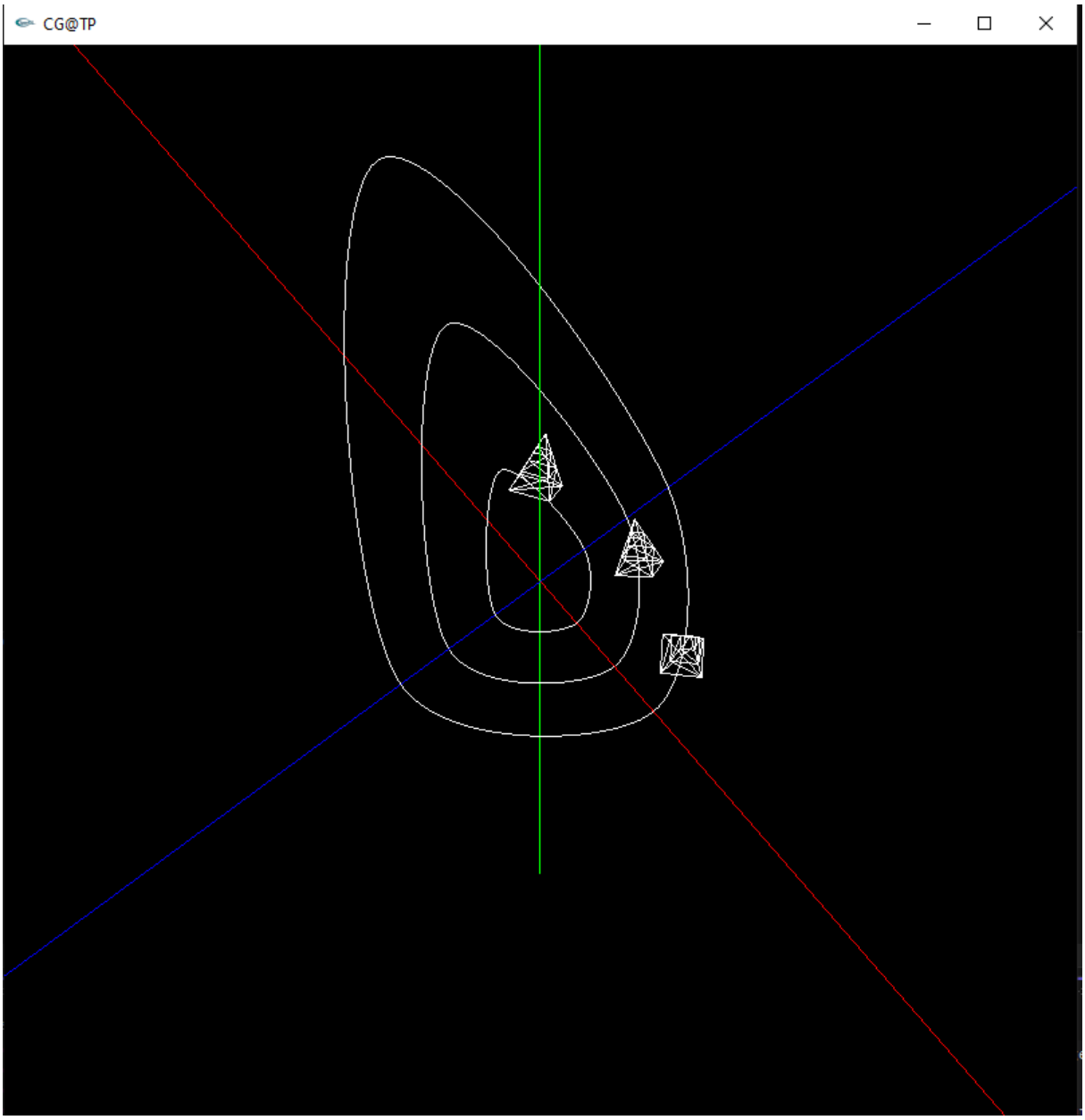

Figura 2.4: Cones e Curvas (exterior: not align, intermédio: aligny, interior: alignz

Capítulo 3

Conclusão

Ao longo do desenvolvimento desta fase do projeto a principal dificuldade esteve **interpretação** correta das curvas de **Bezier** e de **Catmull-Rom**. No entanto, achámos que realizámos o trabalho com a qualidade pretendida pela equipa docente.