Simulação de movimentos compostos em 3D

Departamento de Engenharia Mecânica
RI - Universidade de Aveiro - Vitor Santos

Filipe André Seabra Gonçalves - 98083

Outubro de 2023

1 Introduction

Este relatório tem por objetivo explicar a abordagem e o código desenvolvido para obter uma aplicação em Matlab que mostre, de forma animada, a simulação do movimento de objetos compostos em 3D, dada a sua trajetória, bem como outras métricas.

Para a construção do objeto composto, foram usadas 3 pirâmides quadrangular, com o mesmo tamanho, mas com rotações diferentes ao longo do eixo Z de modo a que fiquem numa organização de trevo de 3 folhas, como visto na figura 1.

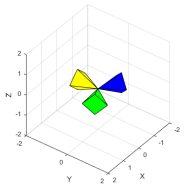

Imagem 1: Posição inicial do objeto composto

A trajétoria e todas as outras métricas são dadas ao ler o ficheiro **traj.csv**, e se não existir na pasta quando o projeto for executado, então o projeto irá produzir as transformações necessárias para uma trajétória hardcoded.

2 Funções e Classes criadas

Para representar cada pirâmide foi criado uma classe **Piramide** que recebe como argumento de entrada o index (de 1 a 3) com qual pirâmide é suposto criar.

É importante notar que cada objeto tem um **handler** gráfico e uma matriz de transformações, que são atualizadas sempre que é feita uma nova transformação geométrica. Deste modo, podemos criar transformações geométricas específicas para cada objeto que compõe o objeto composto.

Também foi criada uma função **PointsFromArc** que, dados os pontos por onde o objeto composto deve seguir, calcula as listas para as translações. Existem três listas, cada uma para cada coordenada cartesiana (x, y, z), que depois serão usadas na matriz de translação do objeto.

Além das funções referidas anteriormente, foram também usadas as funções auxiliares **mrotx**, **mroty** e **mrotz** para obter as matrizes de rotação pelos eixos de **X**, **Y** e **Z**, **mtrans** e **trans** para obter as matrizes de translação e **seixos3** para obter o referencial local do objeto composto.

3 Procedimento

3.1 Orientação

Para a orientação do objeto em relação à trajetória, primeiramente foi calculado, usando a função do matlab **cart2sph**, os ângulos das coordenadas esféricas da trajetória. Depois foram feitas quatro rotações, animadas numa só, usando a propriedade associativa das transformações geométricas, em que as duas primeiras são de colocar o objeto composto na posição normal global, e as últimas duas para colocar o objeto na nova orientação. As rotações são feitas usando o complementar do ângulo **theta** em relação ao eixo **Z** local e o ângulo **phi** em relação ao eixo **Y** local.

3.2 Translação

Para cada trajetória, apenas é preciso calcular, para cada passo, qual a quantidade de **X**, **Y** e **Z** que o objeto composto tem de "andar" em torno do referencial global. No entanto, existem vezes em que para além do objeto ter de fazer a translação sobre a trajetória, também tem de rodar em torno de si mesmo, **twist**, **n** vezes. Para tal, ao invés de fazermos uma animação por tranformaçãpo geométrica, fazemos animação por passo. Desta maneira, não podemos simplesmente criar uma translação normal pela trajetória, e em vez disso, verificamos a diferença entre dois passos consecutivos, e fazemos a translação de acordo com esse desvio, pelo referencial local do objeto composto. Assim, a rotação, ao invés de ser uma rotação normal, é o desvio entre passos consecutivos da rotação.

Contudo, a adição do **twist** não é a princial dificuldade, porque para além do possível **twist**, o objeto composto também pode seguir uma trajetória circular, ao longo de um arco, que começa no ponto inicial da trajetória e acaba no ponto final da trajetória, passando por um ponto auxiliar a **h** de distância. Este arco faz com o eixo da trajeória um ângulo de **beta**.

Qualquer um dos valores de **n**, **h** e **beta** são dados pelo ficheiro **traj.csv** ou hardcoded no projeto.

Para calcular os pontos do arco, primeiramente encontramos o ponto médio da trajetória e calculamos a posição do ponto auxiliar, **P4**, que faz um ângulo reto com a trajetória, onde **phi = 0**, em coordenadas esféricas.

Depois, calculamos o centro da circunferência que passa pelo ponto inicial, **P1**, pelo ponto final **P2** e por **P4**, usando a função **arc3** da toolbox "3D arc passing through 3 points" do **File Exchange**. Com o raio, conseguimos calcular os pontos pertencentes ao arco que começa em **P1** e termina em **P2**, passando por **P4**, usando as fórmulas de translação de coordenadas esféricas para coordenadas cartesianas.

Finalmente, para calcular os pontos, com a rotação de **beta**, podemos usar a função **axang2rotm**, que dada a trajetória e o ângulo, faz a rotação automática dos pontos.

Podemos ver gráficamente os passos dados para o cálculo do arco com rotação **beta** pela figura 3.2.

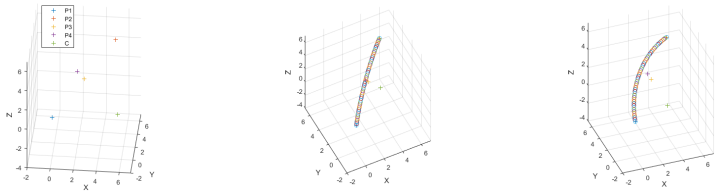

Imagem 2: Passos desde o cálculo de P4 até ao cálculo dos pontos do arco com rotação beta

3.3 Movimento de objeto Singular

Para o movimento relativo a cada uma das pirâmides do objeto composto, foram ciradas três translações, uma para cada pirâmide, com valores diferentes de **X**, **Y**, e **Z**. Cada pirâmide irá afastar-se do eixo da trajetória, e no ponto médio desta, ou no ápice da trajetória em arco, irá estar no ponto mais afastado da trajetória, fazendo assim, no final do seu movimento o retorno até se juntarem os objetos num só. Podemos ver o afasto das pirâmides na figura 3.3.

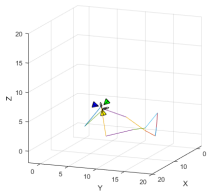

Imagem 3: Passos desde o cálculo de P4 até ao cálculo dos pontos do arco com rotação beta

Para conseguir este efeito, foi usada a função **LinspaceVect**, criada nas aulas, que junta dois vetores, cada metade do seu movimento, desde 0 a 0.5 e de 0.5 a 0 de coordenadas.

O movimento de cada pirâmide ocorre quando não existe o movimento de **twist**.

4 Conclusão

Concluindo, foram cumpridos todos os requisitos do projeto até ao nível 4, sendo possível observar a animação do objeto composto orientado pela trajetória pretendida, de acordo com todos os parâmetros referidos.

Por fim, segue-se o vídeo demonstrativo: `https://youtu.be/I1Ng-n3CN8I`.

Qualquer problema que tenha com a submissão de código, tem o link para o github aqui: `https://github.com/FlipGoncalves/3D_ObjectSimulation`.