Robótica Industrial
Aula prática nº 2
Transformações geométricas em 3D
Composição de transformações
Pré- e pós-multiplicação

Vítor Santos

Universidade de Aveiro

3 Out 2022

Exercício 1

- Criar funções de transformação geométrica no espaço (matrizes de 4×4) para as 4 operações seguintes onde a se mede em radianos:
  - `M=trans(x,y,z)`
  - `M=rotx(a)`
  - `M=roty(a)`
  - `M=rotz(a)`
- Criar a mesma representação do triângulo $A1$ do exercício 1 da aula anterior, mas no espaço a $3D$ (usar a função `fill3` para visualizar)
- Fazer a animação do triângulo em torno do eixo dos xx. (Animar duas voltas completas)

Matrizes de transformação homogéneas

$$trans(x,y,z) = \begin{bmatrix} 1 & 0 & 0 & x \\ 0 & 1 & 0 & y \\ 0 & 0 & 1 & z \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad rotx(\alpha) = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & C\alpha & -S\alpha & 0 \\ 0 & S\alpha & C\alpha & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$roty(\alpha) = \begin{bmatrix} C\alpha & 0 & S\alpha & 0 \\ 0 & 1 & 0 & 0 \\ -S\alpha & 0 & C\alpha & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad rotz(\alpha) = \begin{bmatrix} C\alpha & -S\alpha & 0 & 0 \\ S\alpha & C\alpha & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Exercício 2

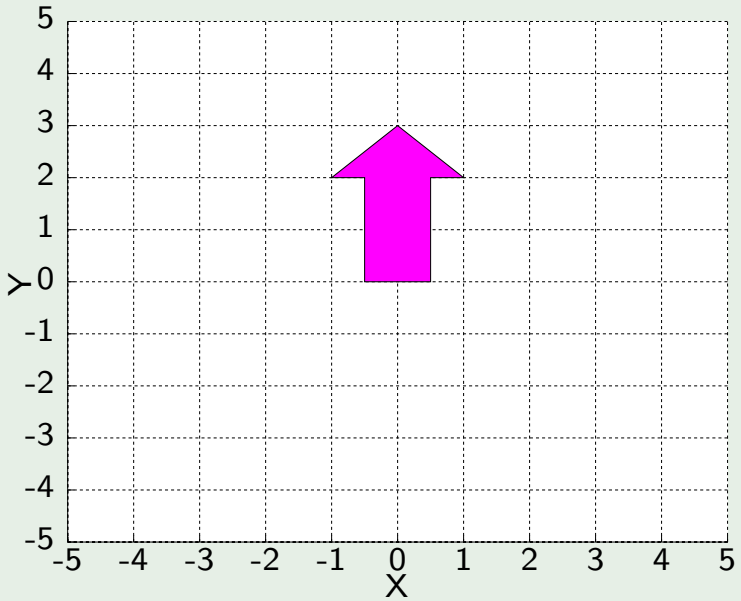
Criar o objeto P indicado abaixo (a seta) no plano XY

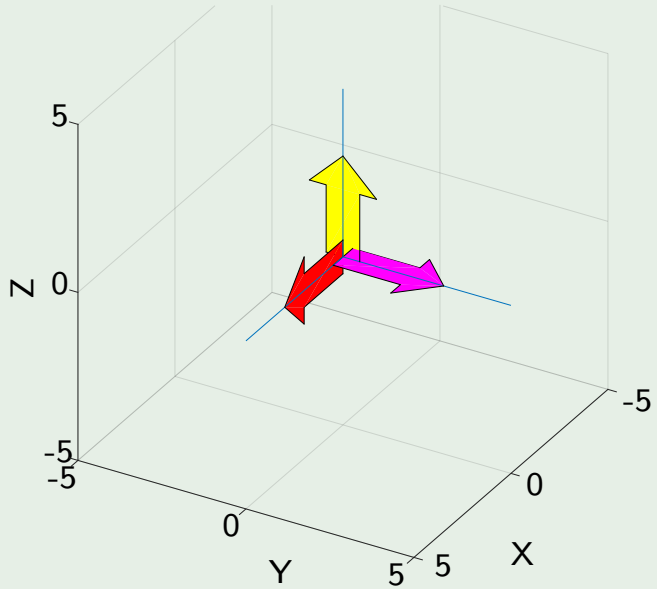
- Sugere-se as seguintes coordenadas:

```
P=[0.5 0.5 1  0  -1  -0.5  -0.5
   0   2   2  3   2    2    0
   0   0   0  0   0    0    0
   1   1   1  1   1    1    1   ];
```

- Representá-lo em 3D usando o comando `fill3()` do MATLAB.
- Mediante as transformações geométricas `rotx()`, `roty()`, etc., criar e representar réplicas da seta P nos outros dois planos (XZ e YZ).

Seta em XY, e setas nos três planos





Exercício 3

Colocar os 3 objetos em movimento simultâneo

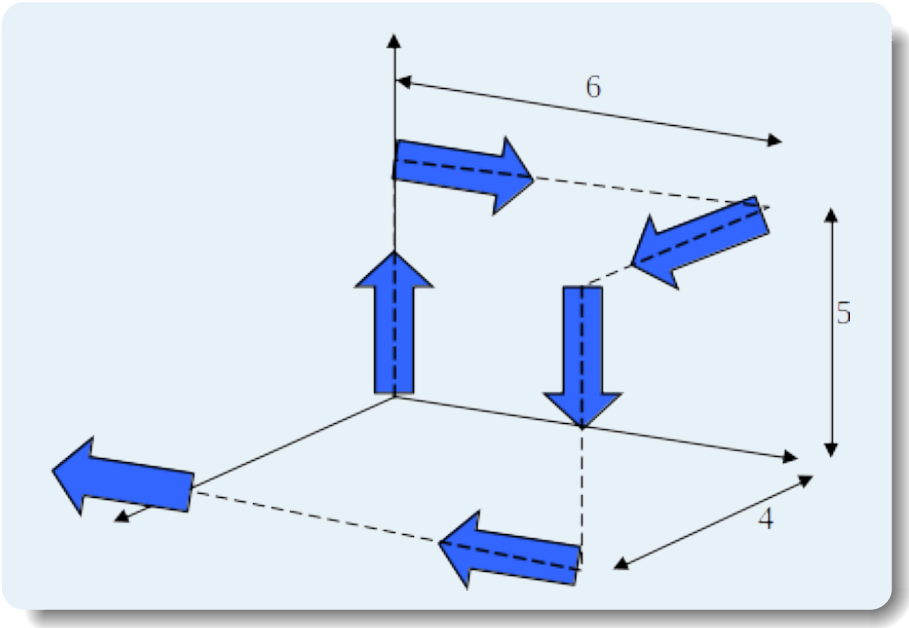
- Animar os três objetos (setas) de tal forma que rodem em torno do eixo onde têm a base apoiada:
  - Fazê-los dar 10 voltas completas girando à mesma velocidade.
  - Usar as rotações apropriadas para cada caso (`rotx`, `roty`, `rotz`).
  - Para um movimento fluído, ajustar os parâmetros do *graphic handle*:
    - alterando os campos da estrutura de dados (`h.XData=...,`)
    - ou usando a função `set` (`set(h,'XData',...)`)
- Ajustar o código para que girem a velocidades diferentes:
  - Por exemplo: 1x, 2x e 3x;
  - NB. Neste caso só um deles dará 10 voltas porque devem parar todos ao mesmo tempo!

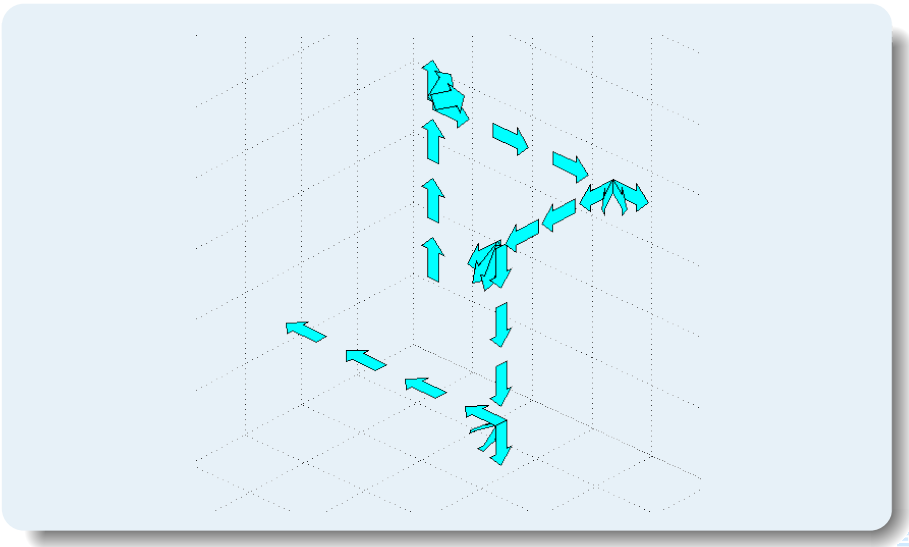
Exercício 4

Objetivo geral

- Simular o movimento do objeto P ao longo do trajeto ilustrado em 10 etapas (5 translações alternadas com 5 rotações de 90°)
- Usar transformações geométricas em **pós-multiplicação**, conforme as instruções nas páginas seguintes usando a função `Animate()` que será desenvolvida para o efeito.

A figura ao lado ilustra algumas posições intermédias.





Exercício 4a

Função `Animate`

- Criar a função `Animate` para simular o movimento de objetos
  - `pFinal=Animate(h, P, pInicial, D, N)`
    - **h** – *handle* gráfico do objeto a animar (movimentar);
    - **P** – matriz de pontos do objeto (no formato homogéneo);
    - **pInicial** – Matriz de transformação da posição inicial do objeto;
    - **D** – vetor $[x, y, z, \phi, \theta, \psi]$ com os **incrementos** totais em posição e orientação vistos do referencial **local** do objeto; este vetor servirá para criar a matriz de transformação que será usada em **pós-multiplicação**.
    - **N** – número de passos da animação da etapa ("duração" do movimento);
    - **pFinal** – matriz com a posição do objeto no final do movimento.

Esta função deverá invocar as funções desenvolvidas anteriormente para transformar o vetor $D = [x, y, z, \phi, \theta, \psi]$ em matriz de transformação geométrica: $M(D) = trans(x, y, z)rotx(\phi)roty(\theta)rotz(\psi)$. Todavia, para criar o efeito de animação, os incrementos totais em D têm de ser fracionados (N partes), e recomenda-se usar o comando `linspace()` do MATLAB.

Exercício 4b

Utilização da função `Animate`

- Para animar um dado troço n, o cálculo das diversas posições da animação faz-se a partir da posição final anterior (T_{n-1}), conforme deixada após a respetiva animação; o procedimento será incremental para ir calculando a posição até ao fim do troço.
- A nova posição (T_n) após o fim do movimento n é o produto da posição anterior com a transformação de movimento do vetor D:
  - $T_n = T_{n-1} M(D)$, sendo $T_0 = eye(4) = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$

A pós-multiplicação

Observe-se que a opção recomendada é de considerar que o movimento é descrito no referencial do objeto e, portanto, as operações incrementais de movimento são obtidas por **pós-multiplicação** $T_n = T_{n-1} M(D)$, e não por pré-multiplicação que seria $T_n = M(D) T_{n-1}$

Exercício 4c

Invocação da função `Animate()` a partir do programa principal

O caminho completo de um movimento será um conjunto de vetores $D_i = [x_i, y_i, z_i, \phi_i, \theta_i, \psi_i]$ que representam as sucessivas transformações relativas, e a função `Animate()` será chamada tantas vezes quanto o número de vetores D_i que constituem o caminho.

Exemplo de caminho com 4 movimentos

Translação de 4 unidades ao longo de z, seguida de rotação em torno de x de -90°, nova translação de 5 unidades ao longo de z e rotação de 90° em torno de y.

i	x_i	y_i	z_i	ϕ_i	θ_i	ψ_i
1	0	0	4	0	0	0
2	0	0	0	$-\pi/2$	0	0
3	0	0	5	0	0	0
4	0	0	0	0	$\pi/2$	0

Valor de retorno de `Animate()`

Para poder ser chamada recursivamente, a função `Animate()` deve retornar a matriz de transformação onde "deixou" o objeto para que a seguinte invocação possa partir de lá.