TP1-Tangram
Robótica Industrial

Luís Silva, nmec 88888
luisfgbs@ua.pt
Mestrado em Robótica e Sistemas Inteligentes
Universidade de Aveiro

I. Introdução

Este relatório foi redigido no âmbito da UC Robótica Industrial, lecionada pelo professor Vítor Santos e tem o objetivo de documentar o trabalho realizado no decorrer do projeto. O Tangram é composto por 7 peças de 5 tipos diferentes um quadrado, 2 triângulos retângulos isósceles pequenos, 1 triângulo retânguo isósceles médio, 2 triângulos retângulos isósceles grandes e um paralelogramo.

O quebra-cabeças tem como objetivo a criação de diferentes de diferentes figuras usando sempre o conjunto de peças descrito anteriormente, que são construídas ao alterar as posições e rotações das peças. Este trabalho prático pretende que se proceda à animação de movimentos das peças do jogo referentes a 3 figuras do Tangram criando as peças e definindo as transformações necessárias à montagem.

Outubro 2022

II. Execução

A. Criação

1) createPieces(): Esta função recebe um cell_array contendo os nomes das peças a criar e devolve uma struct chamada *config* que para cada peça *piece* contém os seguintes campos:

config.piece.object - contém matriz homogénea referente aos pontos que compõem piece
config.piece.color - a cor com que devem ser representadas as faces do objeto
config.piece.pI - que guardará o resultado da acumulação sucessiva de transformações aplicadas a piece. É inicializada como a matriz identidade
config.piece.handle - handle gráfico de piece

Existem também funções dedicadas a criar cada um dos 5 tipos de peça. Estas funções devolvem a matriz guardada em config.piece.object bem como uma matriz que indica as arestas que compõem cada face que vai ser uasda para desenhar o objeto. Serve como exemplo o excerto de código apresentado na figura 1. Passamos a estas funções *l* que representa o comprimento do lado do quadrado que será usado como referência para a construção de todas as peças bem como futuramente para as transformações e *t* que representa espessura da peça.

```
%criar quadrado
function [P, F]=create_square(l, t)
    P=[0 l l 0 0 l l 0
       0 0 l l 0 0 l l
       0 0 0 0 t t t t
       1 1 1 1 1 1 1 1];

    F=[1 2 3 4
       5 6 7 8
       1 4 8 5
       2 3 7 6
       1 2 6 5
       3 4 8 7];
end
```

Fig. 1. Função create_square

B. setInitialPosition()

Esta função é responsável por fazer a inclinação do plano de montagem das peças consoante os valores de *theta* e *phi* e por colocar as peças na sua posição inicial dentro do mesmo. *theta* e *phi* são argumentos da função e representam, em graus, as rotações do plano de montagem sobre o eixo xx e yy, respetivamente. Depois da rotação do plano as peças assumem aleatoriamente uma de 7 posições definidas numa formação circular, para tal cada peça é rodada localmente sobre o seu eixo zz pelo ângulo apropriado e sofre depois uma translação local segundo o seu eixo xx.

```
T(:,:,:,1)=mrotx(angles_theta);
T(:,:,:,2)=mroty(angles_phi);
T(:,:,:,3)=mrotz(linspace(0, angles(i), s));
T(:,:,:,4)=mtrans(linspace(0,3*l, s), 0, 0);

order=[0, 0, 1, 1];
```

Fig. 2. Transformações aplicadas para atingir posição inicial

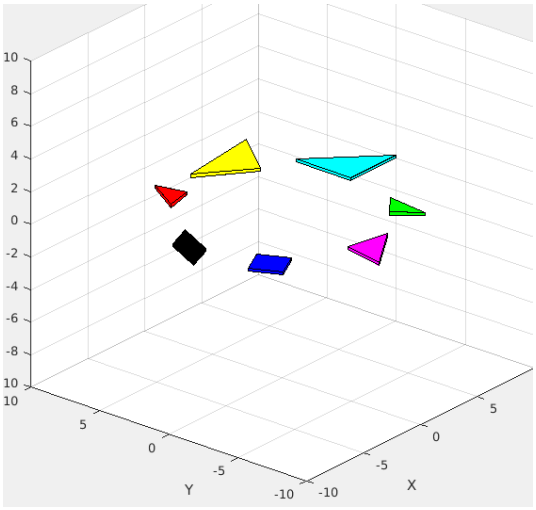

Fig. 3. Posição inicial sobre o plano xOy

C. *createFormations()*

Dentro do código *formation* é sinónimo de figura do tangram e o que *createFormations()* faz é definir as transformações para cada peça, dentro de cada formação para formar a figura correta. No total definem-se 5 tranformações (2 translações e 3 rotações), todas locais, para cada peça em cada formação. Começa-se por desfazer as translações e rotações aplicadas à peça na formação anterior para que esta regresse à origem e depois aplica-se a translação e rotação necessárias para que a peça fique corretamente colocada na sua poisção final. De notar que para a primeira figura a ser montada, as tranformações a serem desfeitas são as que foram aplicadas em *setInitialPosition()* após a inclinação do plano de montagem.

```
formations.house.t1.pos1=-formations.cat.t1.pos2;
formations.house.t1.pos2=[l+l^2+sqrt(2)*l/2, l+sqrt(2)*l/2, 0];
formations.house.t1.rot1=-formations.cat.t1.rot3;
formations.house.t1.rot2=0;
formations.house.t1.rot3=3*pi/4;
```

Fig. 4. Definição das 5 transformações de *t1* para a figura *house* quando esta é precedida pela figura *cat*

```
T(:,:,:,1)=mrotz(linspace(0,formations.(formation_name).(piece).rot1 ,s));
T(:,:,:,2)=mtrans(linspace(0, transf_vector1(1), s), linspace(0, transf_vector1(2), s), linspace(0, transf_vector1(3), s));
T(:,:,:,3)=mrotz(linspace(0,formations.(formation_name).(piece).rot2 ,s));
T(:,:,:,4)=mtrans(linspace(0, transf_vector2(1), s), linspace(0, transf_vector2(2), s), linspace(0, transf_vector2(3), s));
T(:,:,:,5)=mrotz(linspace(0, formations.(formation_name).(piece).rot3,s));

order=[1,1,1,1,1];
```

Fig. 5. Aplicação das 5 transformações definidas

D. *Tolerância*

A implementação concebida deve permitir que facilmente seja criados novos tipos de peças e que se adicionem mais peças às formações, bastando que se indiquem as transformações adequadas. Para alterar a ordem de construção das figuras basta que se altere a referência em cada delas à figura que a precede.

E. *displayCompleted()*

Esta função limita-se a tornar mais explícita a conclusão do processo de montagem de cada uma das figuras alternando as cores de cada uma das peças entre a sua cor original e o cinzento.

F. *resetBoard()*

Quando terminadas as construções de todas as figuras definidas esta função é chamada para devolver cada peça de novo à sua posição na formação circular inicial com o plano de montagem coincidente com xOy.

III. Future Work

Um aspeto a ser melhorado no projeto é substituir a animação sequencial das peças para cada figura por animação simultânea das mesmas.