CT301 – Introdução à Programação

Cursos: AN, EN-AEL, EN-MEC, M – 1º Ano

1º Semestre

Docentes: STEN TSN (ELT) Gaspar Merca

Projeto

Objectivo

Este projecto tem como objectivos:

- Desenvolver as capacidades de programação;
- Desenvolver as capacidades de programar com recurso à utilização de subprogramas (funções);
- Instruir sobre formas de armazenamento de dados e respetiva leitura;
- Dar a conhecer aos alunos as funcionalidades gráficas do MATLAB;
- Desenvolver a capacidade de raciocínio e investigação.

Introdução ao problema

- Atualmente é usual a utilização de sistemas de apoio à decisão baseados em modelos informatizados que permitem uma rápida simulação, reparametrização, observação e arquivo de resultados.

- Uma aplicação que simule uma situação de batalha, idêntica a um jogo de batalha naval, poderá servir como apoio na decisão da quantidade e tipo de armamento necessário para uma missão.

Desenvolvimento

1 – Complete as 20 funções necessárias para o desenvolvimento deste projeto. Os parâmetros, de entrada e saída, têm que obedecer às descrições presentes neste documento. Juntamente com este documento é fornecido em anexo o *script "Projeto.m"* que, com as funções que tem que de desenvolver, deverá apresentar um jogo de aventura em texto ao utilizador. No fim dos slides encontra-se o fluxograma deste *script*. Este jogo, que consiste num jogo de batalha naval, deverá permitir ao utilizador as seguintes opções:

-Optar pela quantidade de navios que quer colocar numa área(0,5x0,5 milhas náuticas);

-Escolher o navio, posição e rumo ou gerar aleatoriamente estes parâmetros;

-Pedir as coordenadas para os mísseis e indicar se algum navio foi atingido;

-Mostrar graficamente o resultado do jogo;

-Permitir guardar e carregar jogos.

Desenvolvimento

2 – Desenvolva as seguintes funções:

2.1 – Função “tipo_jogo”:

- Parâmetros de entrada: ( )
- Parâmetros de saída: [novo]

- Esta função deverá perguntar ao utilizador se ele deseja começar um novo jogo ou carregar um jogo anterior. O parâmetro de saída deverá ser do tipo lógico, verdadeiro quando o utilizador escolhe jogar novo jogo e falso quando deseja carregar um jogo anterior. Caso o utilizador não escolha uma opção válida, deverá ser interrogado novamente.

Desenvolvimento

2.2 – Função “aleatorio”:

- Parâmetros de entrada: ( )
- Parâmetros de saída: [esp]

- Esta função deverá perguntar ao utilizador se ele deseja especificar as embarcações e respetivas posições ou gerá-las aleatoriamente. O parâmetro de saída deverá ser do tipo lógico, verdadeiro quando o utilizador escolhe especificar e falso quando deseja que o jogo seja aleatório. Caso o utilizador não escolha uma opção válida, deverá ser interrogado novamente.

Desenvolvimento

2.3 – Função “pedir”:

- Parâmetros de entrada: ( )
- Parâmetros de saída: [num]

- Esta função deverá pedir ao utilizador o número de embarcações que deseja colocar na área. Caso o valor seja um número válido, o parâmetro de saída deverá ser o número inserido pelo utilizador. Caso não seja um número válido, deve ser pedido um novo número.

Desenvolvimento

2.4 – Função “pedir_navios”:

- Parâmetros de entrada: (num)
- Parâmetros de saída: [posicoes_x,posicoes_y, cores]

- Esta função deverá devolver as matrizes “posicoes_x” e “posicoes_y” com as coordenadas de “num” embarcações. Cada linha destas matrizes deverá ter as coordenadas de uma embarcação. A *string* “cores” deverá ter um caracter que represente a cor de cada embarcação das matrizes. As embarcações devem ser escolhidas pelo utilizador, as suas coordenadas convertidas para milhas náuticas e o utilizador deve escolher a posição e rumo, respetivamente, através das seguintes funções:

- “mostrar_navios“
- "converter“
- "pedir_rumo”

Desenvolvimento

2.5 – Função “navios”:

- Parâmetros de entrada: (num)
- Parâmetros de saída: [posicoes_x,posicoes_y, cores]

- Esta função deverá devolver as matrizes “posicoes_x” e “posicoes_y” com as coordenadas de “num” embarcações. Cada linha destas matrizes deverá ter as coordenadas de uma embarcação. A *string* “cores” deverá ter um caracter que represente a cor de cada embarcação das matrizes. As embarcações devem ser selecionadas aleatoriamente, as suas coordenadas convertidas para milhas náuticas e atribuídos uma posição e rumo aleatórios, respetivamente, através das seguintes funções:

- “lernavio“
- "converter“
- "rumo”

Desenvolvimento

2.6 – Função “mostrar_navios“:

- Parâmetros de entrada: ()
- Parâmetros de saída: [emb_x, emb_y, cor]

- Esta função deverá devolver os vectores “emb_x”, “emb_y” e a respetiva “cor” de uma embarcação escolhida pelo utilizador. Para que o utilizador escolha uma embarcação, deverão ser listadas as classes existentes no ficheiro em anexo “Embarcações.xlsx” (Inicialmente pode usar o ficheiro “Embarcações.mat”). Conteúdo dos ficheiros embarcações:

Embarcação:	Cor:		Coordenadas					
Classe "Vasco da Gama"	y	x	0	80	115,9	80	0	0
		y	0	0	7,1	14,2	14,2	0
Classe "Bartolomeu Dias"	b	x	0	85	122	85	0	0
		y	0	0	7,2	14,4	14,4	0
NRP "Bérrio"	k	x	0	127	140,6	127	0	0
		y	0	0	9,6	19,2	19,2	0
Classe "Viana do Castelo"	c	x	0	60	83,1	60	0	0
		y	0	0	6,475	12,95	12,95	0
Classe "Tridente"	m	x	0	2,9	65	67,9	65	2,9
		y	3,15	0	0	3,15	6,3	6,3

Desenvolvimento

2.7 – Função “converter“:

- Parâmetros de entrada: (emb_x, emb_y)
- Parâmetros de saída: [emb_x, emb_y]

- Uma vez que as coordenadas das embarcações, presentes nos ficheiros “Embarcações”, se encontram em metros, deverão ser convertidas para milhas náuticas. Os vectores introduzidos como parâmetros de entrada (emb_x, emb_y), após serem convertidos são também os parâmetros de saída da função “converter”.

Desenvolvimento

2.8 – Função “pedir_rumo“:

- Parâmetros de entrada: (emb_x, emb_y)
- Parâmetros de saída: [emb_x, emb_y]

- Uma vez que as coordenadas das embarcações, presentes nos ficheiros “Embarcações”, se encontram todas na posição (0,0) e com um rumo de 0 graus, deverá ser pedido ao utilizador quais as posições (x e y) e o rumo (em graus) que pretende. As posições deverão ser entre 0 e 0,5 para que embarcação seja colocada dentro da área em questão. O rumo deverá encontrar-se entre 0 e 360. Caso o utilizador não escolha valores dentro destes parâmetros, os dados deverão ser pedidos novamente. Aos vectores introduzidos como parâmetros de entrada (emb_x, emb_y), deverão ser alteradas as posições e o rumo para que estes vectores sejam também os parâmetros de saída da função “pedir_rumo”.

Desenvolvimento

2.9 – Função “lernavio“:

- Parâmetros de entrada: ( )
- Parâmetros de saída: [emb_x, emb_y, cor]

- Esta função deverá devolver os vectores “emb_x”, “emb_y” e a respetiva “cor” de uma embarcação escolhida aleatoriamente. À semelhança da função “mostrar_navios” os vectores que contêm as coordenadas das embarcações deverão ser retirados do ficheiro em anexo “Embarcações.xlsx” (Inicialmente pode utilizar o ficheiro “Embarcações.mat”).

Desenvolvimento

2.10 – Função “rumo “:

- Parâmetros de entrada: (emb_x, emb_y)
- Parâmetros de saída: [emb_x, emb_y]

- Uma vez que as coordenadas das embarcações, presentes nos ficheiros “Embarcações”, se encontram todas na posição (0,0) e com um rumo de 0 graus, deverão ser geradas aleatoriamente novas posições (para x e y) e rumo (em graus). As posições deverão ser entre 0 e 0,5 para que a embarcação seja colocada dentro da área em questão. O rumo deve ser entre 0 e 360. Aos vectores introduzidos como parâmetros de entrada (emb_x, emb_y), deverão ser alteradas as posições e o rumo para que estes vectores sejam também os parâmetros de saída da função “rumo”.

Desenvolvimento

2.11 – Função “atingir_emb“:

- Parâmetros de entrada:

(posicoes_x,posicoes_y,num)

- Parâmetros de saída: [p_x,p_y,dentro]

- Esta função deverá começar por informar o utilizador que “num” embarcações estão na área e perguntar se deseja tentar atingi-las. Caso o utilizador pretenda atingir as embarcações, deverá solicitar as coordenadas (X e Y) para onde o utilizador deseja enviar mísseis e guardá-las nos vectores “p_x” e “p_y”. Se o míssil atingir uma embarcação, o vector “dentro” deverá ficar com verdadeiro no elemento correspondente às coordenadas (p_x,p_y). Caso contrário, deverá ficar com falso. Esta função deverá conter outras duas funções:

- “perguntar_at“
- "atingir“

Desenvolvimento

2.12– Função “perguntar_at “:

- Parâmetros de entrada: (num)
- Parâmetros de saída: [at]

- Esta função deverá indicar ao utilizador quantas embarcações estão na área (num) e perguntar se o mesmo deseja tentar atingi-las. Caso o utilizador tente atingir as embarcações, deverá definir o parâmetro de saída (at) como verdadeiro. Caso contrário, deve dizer ao utilizador que escolheu não tentar atingir as embarcações e que o gráfico irá ser mostrado e ainda definir “at” como falso.

Desenvolvimento

2.13 – Função “atingir“:

- Parâmetros de entrada: (posicoes_x,posicoes_y)
- Parâmetros de saída: [p_x,p_y,dentro]

- Esta função deverá, enquanto o utilizador pretender, perguntar as coordenadas (X e Y) para onde deseja enviar mísseis e guardá-las nos vectores “p_x” e “p_y”. Se o míssil atingir uma embarcação, o vector “dentro” deverá ficar com verdadeiro no elemento correspondente às coordenadas (p_x e p_y), caso contrário deverá ficar com falso. Deve também informar se as embarcações são atingidas ou não.

Desenvolvimento

2.14 – Função “listar“:

- Parâmetros de entrada: ( )
- Parâmetros de saída: [ ]

- Esta função serve para listar os jogos guardados anteriormente. Os jogos devem estar guardados numa variável do tipo estrutura e dentro de um ficheiro “.mat”, ambos com o nome “jogos”. Em anexo é fornecido um exemplo (jogos.mat). O formato da lista deve ter o seguinte aspecto:

1: 14:25 05-11-2015
Navios: 2
Mísseis lançados: 0
Navios atingidos: 0

2: 23:00 05-11-2015
Navios: 2
Mísseis lançados: 2
Navios atingidos: 1

3: 00:22 06-11-2015
Navios: 2
Mísseis lançados: 3
Navios atingidos: 2

Desenvolvimento

2.15 – Função “carregar“:

- Parâmetros de entrada: ( )

- Parâmetros de saída: [posicoes_x,posicoes_y,p_x,p_y,cores,dentro]

- Esta função deve solicitar um número ao utilizador correspondente a um jogo listado pela função “listar”. Após verificar que é um número válido, deve carregar toda informação desse jogo para os parâmetros de saída (posicoes_x,posicoes_y,p_x,p_y,cores,dentro).

Desenvolvimento

2.16 – Função “contar_emb“:

- Parâmetros de entrada: (num)
- Parâmetros de saída: [posicoes_x]

- Esta função deve contar o número de embarcações que o jogo carregado contém. Para isso tem como parâmetro de entrada a matriz “posicoes_x”. A quantidade de embarcações deve ser verificada e colocada no parâmetro de saída “num”.

Desenvolvimento

2.17 – Função “concatenar“:

- Parâmetros de entrada:

(p_x,p_y,dentro,p_x2,p_y2,dentro2)

- Parâmetros de saída: [p_x,p_y,dentro]

- Esta função deve concatenar no mesmo vector os mísseis que já se encontravam gravados anteriormente (p_x,p_y,dentro) com os mísseis que foram lançados no jogo atual (p_x2,p_y2,dentro2). No fim, todos os mísseis devem estar nos parâmetros de saída (p_x,p_y,dentro).

Desenvolvimento

2.18 – Função “grafico”:

- Parâmetros de entrada:

(posicoes_x,posicoes_y,p_x,p_y,cores,dentro)

- Parâmetros de saída: [ ]

- Esta função deve representar graficamente todas as embarcações, numa área de 0,5x0,5 milhas náuticas, com as respetivas cores. Deve também mostrar os mísseis lançados: a verde os que atingiram alguma embarcação e a vermelhos os restantes. Coloque título e etiquetas nos dois eixos. No fim destes slides encontra-se um exemplo de como deve ser o gráfico mostrado.

Desenvolvimento

2.19 – Função “guardar”:

- Parâmetros de entrada:

(posicoes_x,posicoes_y,p_x,p_y,cores,dentro)

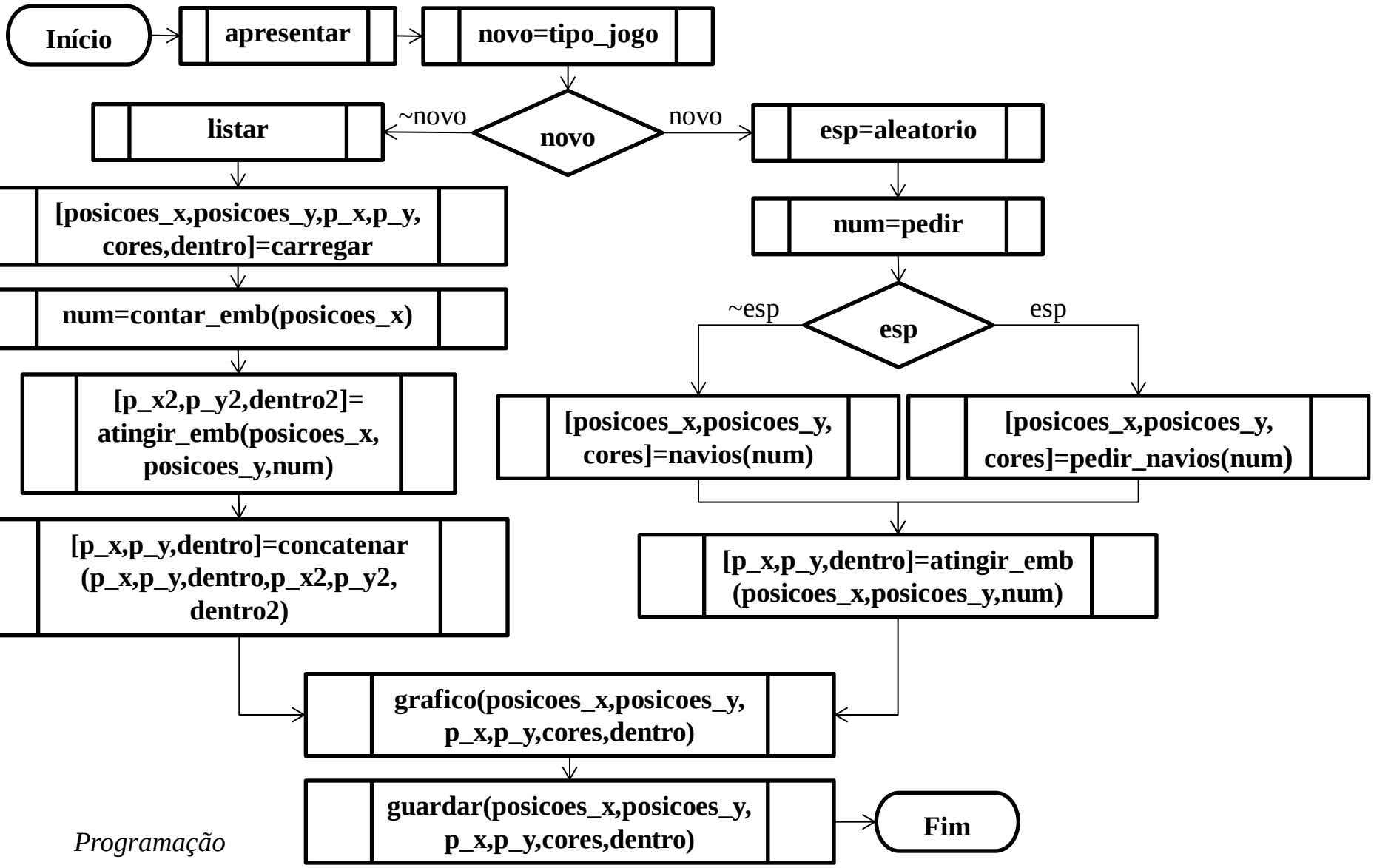
- Parâmetros de saída: [ ]

- Esta função deve questionar se o utilizador pretende guardar o jogo. Caso a resposta seja afirmativa, deve armazenar todos os dados anteriores numa estrutura. Os campos dessa estrutura devem ter o mesmo nome dos parâmetros de entrada. Deve também ser acrescentado um campo (Data) com a hora e data com que o jogo foi gravado (no formato HH:MM dd-mm-yyyy);

Desenvolvimento

3 – Desenvolva um script de apresentação, com o nome “apresentar”, para ser corrido no início do jogo. Este script deve apresentar os autores do projeto, a data de início do projeto e o número de dias que passaram desde que o projeto foi iniciado.

Fluxograma


Início
apresentar
novo=tipo_jogo
novo
~novo
listar
novo
esp=aleatorio
[posicoes_x,posicoes_y,p_x,p_y,
cores,dentro]=carregar
num=pedir
num=contar_emb(posicoes_x)
~esp
esp
esp
[p_x2,p_y2,dentro2]=
atingir_emb(posicoes_x,
posicoes_y,num)
[posicoes_x,posicoes_y,
cores]=navios(num)
[posicoes_x,posicoes_y,
cores]=pedir_navios(num)
[p_x,p_y,dentro]=concatenar
(p_x,p_y,dentro,p_x2,p_y2,
dentro2)
[p_x,p_y,dentro]=atingir_emb
(posicoes_x,posicoes_y,num)
grafico(posicoes_x,posicoes_y,
p_x,p_y,cores,dentro)
guardar(posicoes_x,posicoes_y,
p_x,p_y,cores,dentro)
Fim

Exemplo

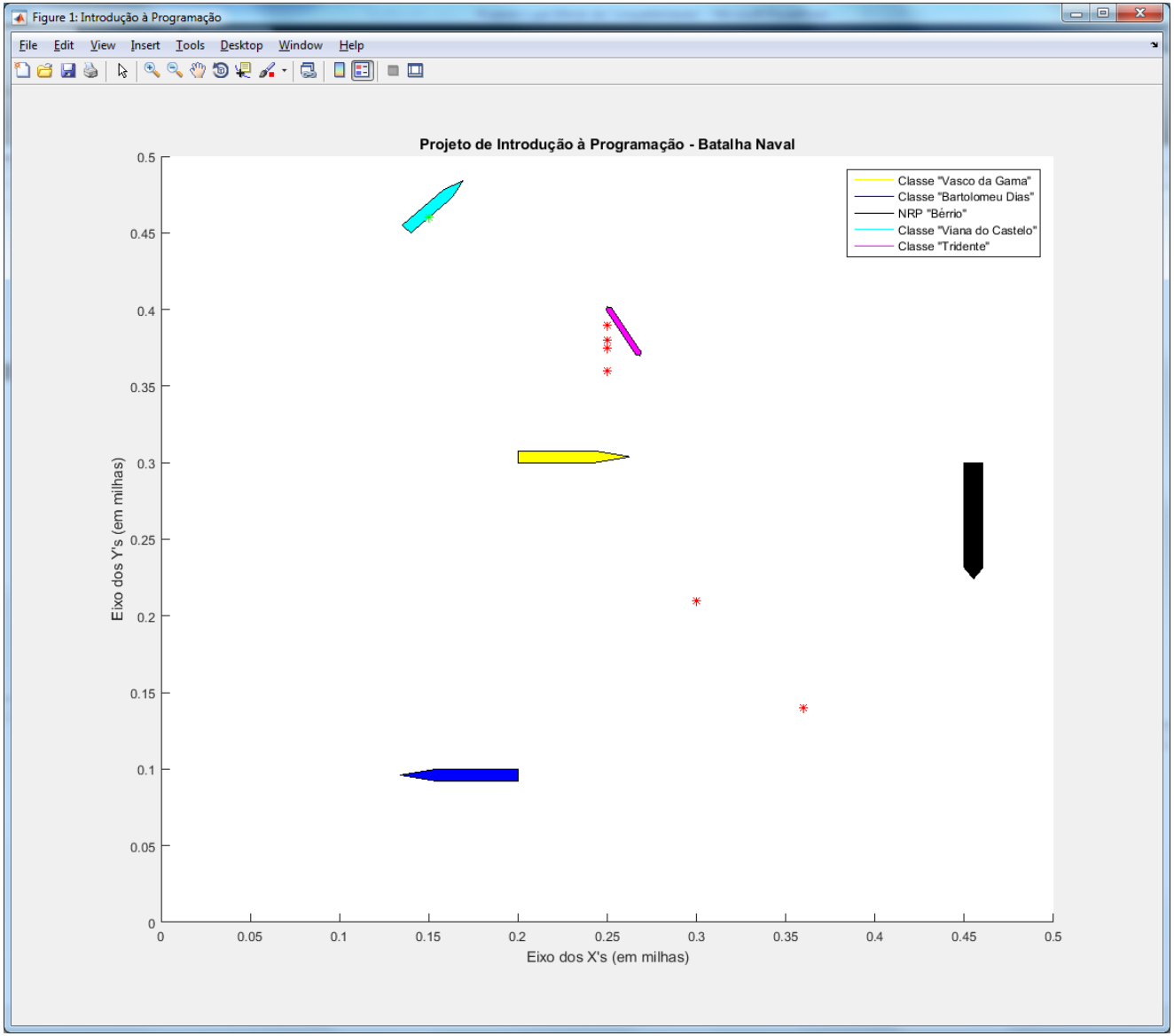


Figure 1: Introdução à Programação
File Edit View Insert Tools Desktop Window Help
Projeto de Introdução à Programação - Batalha Naval
Classe "Vasco da Gama"
Classe "Bartolomeu Dias"
NRP "Bérrio"
Classe "Viana do Castelo"
Classe "Tridente"
Eixo dos Y's (em milhas)
Eixo dos X's (em milhas)
0
0.05
0.1
0.15
0.2
0.25
0.3
0.35
0.4
0.45
0.5

Funções Necessárias

- disp
- input
- fprintf
- datevec
- now
- etime
- round
- num2str
- strcmp
- strcmpi
- zeros
- save
- load
- xlsread
- isempty
- size
- cosd
- sind
- randi
- mod
- true
- false
- datestr
- sum
- inpolygon
- figure
- close
- gcf
- hold
- axis
- plot
- xlabel
- ylabel
- title

Prazo de Entrega (1ª fase)

- Data de entrega:
  - Turma A: 17 de Dezembro
  - Turma B: 14 de Dezembro

- Ficheiros a entregar:
  - apresentar.m
  - tipo_jogo.m
  - aleatorio.m
  - pedir.m
  - listar.m
  - carregar.m
  - contar_emb.m
  - navios.m
  - lernavio.m (de Embarcações.m)
  - converter.m
  - rumo.m

- Respectivos fluxogramas e pseudocódigo

Prazo de Entrega (2ª fase)

- Data de entrega:
  - Turma A: 28 de Janeiro
  - Turma B: 25 de Janeiro

- Ficheiros a entregar:
  - lernavio.m (de Embarcações.xlsx)
  - pedir_navios.m
  - mostrar_navios.m (de Embarcações.xlsx)
  - pedir_rumo.m
  - atingir_emb.m
  - perguntar_at.m
  - atingir.m
  - concatenar.m
  - grafico.m
  - guardar.m

- Respectivos fluxogramas e pseudocódigo