TÉCNICO LISBOA

Projeto de Fundamentos de Programação

Documentação técnica

GRUPO Nº 66

103847	Guilherme Fernandes guilhermegrancho@tecnico.ulisboa.pt
103332	Gonçalo Drago goncalo.drago@tecnico.ulisboa.pt

Licenciatura em Engenharia Mecânica

Licenciatura em Engenharia e Arquitetura Naval

Lisboa, junho de 2022

Índice

2

1. Documentação técnica

1.1. Arquitetura do programa

A arquitetura do programa ficou separada num ficheiro anexado, "1.1._Arquitetura do programa.pdf".

1.2. Projeto detalhado

Classes principais:

Tem-se: as classes dadas pelo módulo graffics, a classe Harve que tem uma série de métodos para trabalhar com o robô, a classe mapMemmory que tem métodos para guardar informação dos mapas de objetivos e de obstáculos para trabalhar com estes mapas. E por último a classe Tree que tem métodos para desenhar e trabalhar com os obstáculos.

Funções principais:

Tem-se a função mainMenu é a função encarregue do menu primário, esta função chama a função buttonsForTheMainMenu, que é responsável de colocar os botões do primeiro menu gráfico. A partir desta função podemos encerrar o programa no botão exit, podemos chamar a função help que abre uma janela informativa. Dentro desta função tem-se uma função chamada mainMenu2 que por sua vez chama a função buttonsForTheMainMenu2 que é responsável pelos botões do segundo menu gráfico. Se clicar no botão return chama a função mainMenu e volta ao menu gráfico primário. Também pode-se chamar da primeira à terceira implementação chamando as funções respetivas firstImplementacion, secondImplementacion e thirdImplementacion. Para aceder à quarta implementação será chamada a função mainMenu3. Esta função carrega o terceiro menu gráfico e chama buttonsForTheMainMenu3, esta função se for clicando no botão return chama a função mainMenu2 e volta para o menu gráfico secundário. O terceiro menu gráfico permite-nos escolher entre as duas versões da quarta implementação chamando fourthImplementacion1 e fourthImplementacion2.

4

A função firstImplementacion chama a função backGround que cria a janela gráfica da primeira implementação. Depois atribui uma variável a classe mapMemory, cujas suas funcionalidades já foram ditas anteriormente, chama a função obstáculos que cria e desenha os obstáculos (utilizando os métodos da classe Tree). Atribui uma variável à classe Harve. Chama a função bodyForTheBot que atribui a parte gráfica ao robô, chama a função objectives que guarda a informação sobre a posição dos objetivos no mapa e desenha-os graficamente, chama a função robot que utiliza a inicialmente a função goalFinder para detetar se já está no centro do objetivo e calcular o ângulo a que o robô precisa de se mexer para ir ter ao objetivo. A função run que está dentro da função robot e não dentro da função goalFinder faz o robô andar utilizando a função freeToPass para evitar obstáculos. A função robot também utiliza a função baseFinder para calcular o ângulo que o robô tem de se mexer para voltar à base recorrendo mais uma vez à função run. Tanto a função goalFinder como a baseFinder utilizam a função angleFinder para calcular o ângulo da trajetória. Após todos estes passos a função frstImplementacion é encerrada e chama-se outra vez a função mainMenu2 para voltarmos ao menu gráfico secundário.

Ao ser chamada a função secondImplementacion o processo mantem-se igual ao da função firstImplementacion com a exceção de que a função obstacules utiliza métodos diferentes da classe Tree para diversificar o tipo de obstáculos e em vez da função robot é chamada a função

botForTheSecondImplementacion que utiliza o mesmo processo que a função robot à exceção de que chama as funções setBotBattery e batteryForTheSecondImplementacion para inicializar a bateria e garantir o correto funcionamento da bateria do robô, respetivamente.

Ao chamar a função thirdImplementacion esta por sua vez chama a função informationForTheThirdImplementacion. Esta função chama a função readFileObstacules utiliza uma sequência de subfunções para retirar toda a informação importante do ficheiro "Ambiente.txt". Seguindo a sequência do algoritmo da função é desenhado e criado a janela gráfica atribuindo uma variável à classe mapMemory e é chamada a função obstaculesForTheThirdImplementacion que utiliza a informação do ficheiro "Ambiente.txt" para criar os obstáculos. De seguida é chamada a função objectives e robot que tem as mesmas funcionalidades que foram descritas na função firstImplementacion. Quando acabado o algoritmo da thirdImplementacion é chamada a função mainMenu2 para voltar ao segundo menu gráfico.

Ao chamar a função fourthImplementacion1, que é a função da quarta implementação, esta segue um algoritmo de chamada de funções igual ao de firstImplementacion com exceção de que os objetivos são definidos pela função objectivesForTheFirstImplementacion1 que utiliza a função readFileObjectives para retirar a informação do ficheiro "Limpeza.txt" que nos indica a localização dos objetivos. Para alem disto fourthImplementacion1 ao chamar a função obstacules cria uma maior diversidade e quantidade de obstáculos comparativamente com a função firstImplementacion. Por fim, a função fourthImplementacion2 segue o mesmo algoritmo da função firstImplementacion à exceção de que quando chama a função obstacules cria uma maior quantidade e diversidade de obstáculos.

5

Ficheiros:

"Ambiente.txt" é utilizado na terceira implementação para definir o tamanho da janela gráfica e as caraterísticas dos obstáculos. "Limpeza.txt" serve para definir os objetivos da primeira versão da quarta implementação. "bodyBotT.py" que inicializa as características gráficas do robô assim como para a sua bateria e também contem a função run para que o robô ande sem chocar contra os obstáculos. "grafics.py" que contem o módulo que nos permite ciar toda a interface gráfica. "Harve.py" que contem a classe Harve que nos permite trabalhar com o robô. "implementacions.py" este ficheiro contém todas as funções necessárias para ler os ficheiros "Ambiente.txt" e "Limpeza.txt" para ler estes dois ficheiros para criar os objetivos e os obstáculos de todas as implementações como firstImplementacion, secondImplementacion, thirdImplementacion, fourthImplementacion1 e fourthImplementacion2. Também tem funções que regulamentam o funcionamento da bateria do robô. "Main.py" é o ficheiro em que chama a função mainMenu apenas para começar o programa. "mapMemory.py" que tem as funções que faz o robô contornar os obstáculos e calcula o caminho q o robô deve percorrer para completar os objetivos, para além disso temos a classe mapMemory que nos permite guardar a informação sobre o mapa e trabalhar com ela. "MenusFile.py" que tem as funções que nos permitem trabalhar com todos os menus gráficos existentes e as suas ramificações correspondentes. "Tree.py" que tem a função que garante que os obstáculos criados não fiquei uns sobre os outros e tem a classe Tree que desenha graficamente todos os obstáculos para todas as implementações.

Objetos:

Temos as variáveis win que representam as janelas gráficas. A variável P é atribuída através do método getMouse do modulo graphics. Esta variável representa o ponto clicado na interface gráfica pelo utilizador. A variável xL que é uma lista com as coordenadas x dos centros geométricos dos obstáculos. A variável yL que é uma lista com as coordenadas y dos centros geométricos dos obstáculos. A variável r1 que é o raio dos obstáculos. O objeto bot que é atribuído à classe harve. O objeto body que representa o corpo gráfico do robô. A variável listOfObjectives que é uma lista com as coordenadas do centro dos objetivos e os seus raios. A varável angle que é uma variável do tipo float que define o ângulo ao qual o robô se movimenta. A variável steps que é um integer que conta o número de passos do robô. O objeto battery que representa a informat de texto gráfico da bateria do robô. A variável information que é uma lista com informação sobre o ficheiro "Ambiente.txt". A variável coordenates que contém uma lista da informação importante do ficheiro "Limpeza.txt". A variável x que é um float responsável pelo dimensionamento da janela na terceira implementação. A variável valid que é um boolean que define se o robô está a chocar com um obstáculo ou não. A variável validate que indica se existem obstáculos uns em cima dos outros.

Instâncias:

Da classe Tree: self.radius que dá-nos o raio do obstáculo, self.type que disntingue os obstáculos bush, grass e stone, self.format que disntingue os obstáculos com um formato de um círculo dos com um formato de um retângulo, self.win que nos dá a janela em que os obstáculos são iniciados e self.center que nos dá o centro geométrico dos obstáculos.

6

Da classe harve: self.xpos que recupera a coordenada x do robô, self.ypos que recupera a coordenada y do robô e self.angle indica-nos o ângulo para que o robô está apontado.

Da classe mapMemory: self.wall que guarda as informações sobre os obstáculos no mapa e self.objectives que guarda as informações sobre os obstáculos no mapa.

1.3. Módulo Tree

Tipo: Classe/função

Função: A função chama-se validate que valida que um obstáculo não está em cima de outro e se um objetivo não é criado em cima de um obstáculo.

1.3.1. Entradas (aplicável para funções)

X1 é a coordenada x do objetivo ou do obstáculo que se está a tentar criar, Y1 é a coordenada y do objetivo ou do obstáculo que se está a tentar criar, xL é uma lista com as coordenadas x dos centros geométricos dos obstáculos, yL é uma lista com as coordenadas y dos centros geométricos dos obstáculos e r1 é o raio dos obstáculos.

1.3.2. Saídas (aplicável para funções)

O validate indica se existem obstáculos uns em cima dos outros.

1.3.3. Parâmetros (aplicável para classes)

- self.Tree (variável que apenas será usada nos métodos seguintes);
- self.radius (raio do obstáculo);
- self.N (numero do obstáculo);
- self.type (tipo do obstáculo usado na terceira implementação bush/stone/grass);
- self.format (formato do obstáculo usado na terceira implementação Circle/Rectangle);
- self.win (janela a qual os obstáculos pertencem);
- self.center (ponto importante para o desenho geométrico dos obstáculos);
- self.center2 (caso seja um retângulo o segundo ponto que define essa figura geométrica tendo em conta o graphics);
- self.virtualRadius (raio da circunferência que circunscreve os obstáculos retangulares);
- self.objective (variável que apenas será usada nos métodos seguintes).

7

1.3.4. Métodos (aplicável para classes)

- draw1(self, mapM), (método que desenha os obstáculos da primeira implementação);
- draw2(self, mapM), (método que desenha os obstáculos da segunda implementação);
- draw3(self, mapM), (método que desenha os obstáculos da terceira implementação).

1.4. Módulo Harve

Tipo: Classe

Função: Não é aplicável neste módulo.

1.4.1. Entradas (aplicável para funções)

Não é aplicável neste módulo.

1.4.2. Saídas (aplicável para funções)

Não é aplicável neste módulo.

1.4.3. Parâmetros (aplicável para classes)

- self.xpos, (coordenada x do robô);
- self.ypos, (coordenada y do robô);
- self.angle, (ângulo de direção e do sentido do robô).

1.4.4. Métodos (aplicável para classes)

- getAngle(self), (método que retorna o ângulo para o qual o robô se encontra direcionado);
- update(self, X, Y), (método que atualiza a posição do robô);
- returnPosition(self), (método que retorna a posição do robô).

8