TÉCNICO LISBOA

Projeto de Fundamentos de Programação

Documentação técnica

GRUPO Nº 49

114108	*Pedro Alves* *pedrojorgealves@tecnico.ulisboa.pt*
114120	*José Lima* *jose.pedro.leal.lima@tecnico.ulisboa.pt*

Licenciatura em Engenharia Mecânica

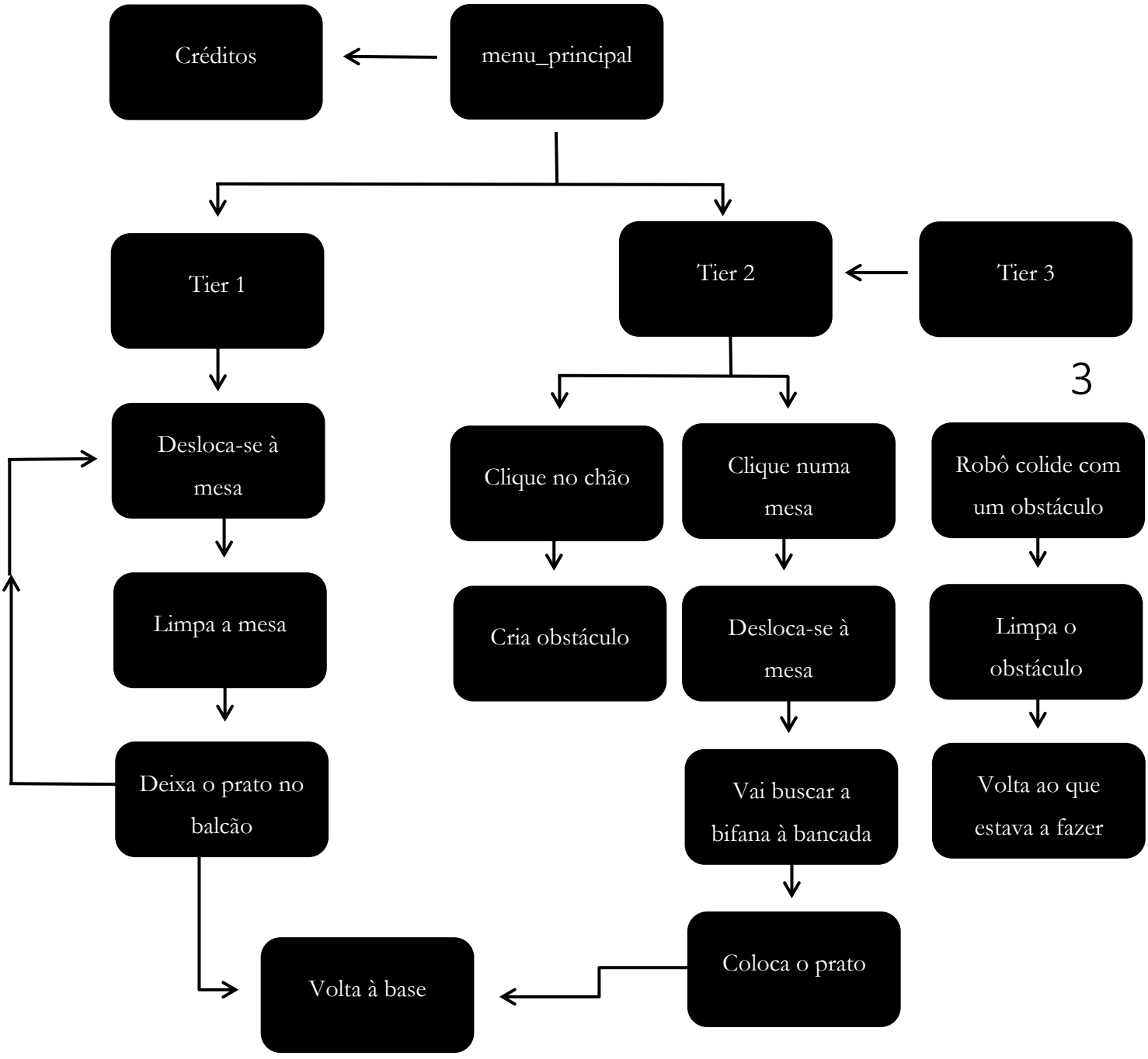
Lisboa, junho de 2025

Índice

1. Documentação técnica

1.1. Arquitetura do programa

Esquema de organização do programa:



O programa é organizado em três ficheiros principais:

- **classes.py:** define as classes que modelam os elementos do sistema (por exemplo, Waiter, Sala, Table, Button). Cada classe encapsula dados e métodos associados aos objetos correspondentes.
- **projeto.py:** contém as funções centrais de lógica, como as rotinas de navegação go_to_table_tier1 e go_to_table_tier2, e outras funções auxiliares de controlo do robô e da simulação.
- **menu_principal.py:** fornece a interface principal com o utilizador. Apresenta um menu (no terminal ou numa janela gráfica) para escolher o ficheiro de mapa e opções de execução. Depois de selecionadas as opções, este módulo chama as funções de projeto.py passando os parâmetros adequados. Inclui uma função menu_principal() protegida por if __name__ == "__main__": menu_principal() para arrancar o programa.

Em execução, o programa lê o ficheiro de mapa via Sala, instancia os objetos (salas, mesas, obstáculos, e o próprio Waiter) e inicia a simulação. O fluxo geral é: o menu lê as escolhas do utilizador → carrega o mapa na classe Sala → cria um Waiter associado à sala → invoca a função de navegação do robô(Tier 1, 2 ou 3).

Principais Classes

- **Waiter (Robô-Empregado):** Representa o robô que circula pelo restaurante. A classe armazena atributos como a posição atual (x, y) ou (linha, coluna), a direção de deslocação, e referências à Sala em que se encontra. Os principais métodos são:
    - __init__: define posição inicial, associa o robô a uma instância de Sala para conhecer o ambiente, e inicializa a representação gráfica.
    - go_to_table_tier1(table): faz o robô deslocar-se até a uma mesa table sem considerar obstáculos (Tier 1). Normalmente faz movimento passo a passo horizontal/vertical.
    - go_to_table_tier2(table): variante do movimento que inclui deteção de obstáculos (Tier 2). Antes de cada passo verifica colisão e tenta contornar obstáculos.
    - Outros métodos auxiliares podem incluir: mover um passo (move_step), atualizar o desenho gráfico do robô na janela, verificar colisão, entre outros.

- **Sala:** Modela o restaurante (ambiente). Armazena as dimensões (largura, altura) da sala e contém coleções dos objetos presentes (por exemplo, lista de Table e lista de obstáculos). Nos atributos da classe podem estar: a janela gráfica (GraphWin), os limites das paredes e o conteúdo lido do ficheiro de mapa. Métodos principais:
    - __init__: lê o ficheiro de mapa (ex.: sala49.txt), define tamanho da sala e instancia objetos de mesas e obstáculos conforme definido no ficheiro.
    - draw(): desenha na janela gráfica todas as mesas, obstáculos e itens fixos (p. ex., o chão, paredes ou portas).
    - Métodos de suporte: por exemplo, para converter coordenadas de grelha em pixels, ou para verificar se uma posição contém obstáculo.
- **Table (Mesa):** Representa cada mesa a ser servida. A classe pode ter atributos como a posição (linha, coluna) e possivelmente um identificador ou estado (por exemplo, se a mesa já foi servida). Os métodos podem incluir:
    - __init__(id, linha, coluna): armazena a localização da mesa.
    - draw(win): desenha a mesa na janela (por exemplo, como um rectângulo ou imagem) usando a biblioteca gráfica.
    - Outros métodos: talvez um método que retorne sua posição, ou altere cor quando servida.
- **Button (Botão Gráfico):** Trata-se de uma classe utilitária para criar botões interativos na janela gráfica (caso a interface use botões clique). Provavelmente tem atributos como forma (Rectangle), posição e rótulo (label). Métodos típicos:
    - __init__(win, center, width, height, label): cria um botão retangular numa janela win centrado em center.
    - is_clicked(p): devolve True se o ponto p (coordenada de clique do rato) está dentro do retângulo do botão.
    - draw(), undraw(): desenhar ou remover o botão da janela.
    - activate()/deactivate(): mudar estado visual do botão.

Principais Funções

- **go_to_table_tier1(waiter, table):** Função (ou método) que movimenta o robô waiter até à mesa table utilizando a lógica do Tier 1. Costuma fazer movimentos passo-a-passo em direção à mesa (por exemplo, primeiro em x, depois em y) sem verificar obstáculos. Cada passo é desenhado na janela.
- **go_to_table_tier2(waiter, table):** Variante que implementa o Tier 2. Além de mover em direção à mesa, em cada passo verifica se há algum obstáculo no caminho (consultando atributos da sala). Se existir obstáculo bloqueando a trajetória, a função tenta ajustar o rumo (p. ex., tentativas de contorno). Assim, o robô avança conscientemente evitando colisões simples.
- **Outras funções auxiliares (em projeto.py):** Podem existir funções como move_waiter(direction), check_collision(), ou funções de processamento dos dados do mapa. Cada função principal deve ter um papel claro: ex., preparar o robô, atualizar o ambiente, gerar logs, etc.

Bibliotecas Utilizadas

O programa recorre a bibliotecas Python para suportar gráficos, cálculos matemáticos e controlo de tempo:

- **graphics:** Utilizada para criar a janela e desenhar os objetos (o robô, mesas, obstáculos, botões). Trata-se da biblioteca *Zelle Graphics* (arquivo graphics.py), concebida para facilitar gráficos em Python. Conforme a documentação, esta biblioteca "é um pacote simples orientado a objetos para gráficos, projetado para tornar muito fácil para programadores iniciantes experimentar com gráficos de computador". É através dela que criamos formas (círculos, retângulos, linhas) e a janela onde a simulação é visualizada.
- **time:** Importada para introduzir pausas e controlos de tempo na animação. A função time.sleep(segundos) suspende a execução do programa durante um dado número de segundos, permitindo que cada movimento do robô seja visível e não instantâneo. Em Python, a biblioteca padrão *time* fornece esta funcionalidade básica de temporização. Usamos sleep() após cada passo do robô para desacelerar a animação.
- **math:** Usada para operações matemáticas necessárias (por exemplo, calcular distâncias ou ângulos, caso se deseje mover em diagonal ou rotacionar). O módulo math inclui funções e constantes matemáticas (como math.sqrt, math.pi, etc.) e

"fornece acesso a funções e constantes matemáticas comuns". No nosso contexto, poderia ser usada para calcular distâncias entre coordenadas ou converter coordenadas de forma precisa.

1.2. Projeto detalhado

1.2.1. Módulo menu_principal

Este ficheiro define e executa o menu principal. O módulo menu_principal tem como principal objetivo gerir a interface gráfica inicial do programa "Zé das Bifanas". Este menu inclui os modos de jogo Tier 1, Tier 2 e Tier 3, a opção para visualizar os proprietários e a possibilidade de sair.

A classe principal deste módulo é menu_principal. Quando instanciada, esta cria uma janela gráfica, através do *GraphWin,* com resolução 1600x900 e coordenadas invertidas para facilitar o posicionamento dos elementos gráficos. O estado interno do menu é controlado através do atributo estado_menu, que pode assumir os valores "*main*" (menu principal) ou "*authors*" (submenu dos proprietários). Logo após a criação da janela, é chamado o método *show_main_menu,* que desenha o menu principal.

O método *show_main_menu* define visualmente os cinco botões principais: "TIER1", "TIER2", "TIER3", "SAIR" e "PROPRIETÁRIOS". Cada botão é criado através da função *Button.create_button,* que devolve um objeto gráfico (retângulo e texto), sendo depois desenhado na janela e armazenado numa lista para facilitar a sua gestão posterior. As imagens associadas aos botões, como os ícones dos tiers ou o botão de sair, são também desenhadas com objetos *Image,* melhorando o aspeto da interface.

Para garantir uma navegação, o método *clear_main_menu* permite remover da janela todos os elementos gráficos associados ao menu principal, incluindo botões e imagens. Isto é útil, por exemplo, quando o utilizador clica em "PROPRIETÁRIOS", momento em que o menu principal é limpo e substituído pelo submenu de autores.

O método *show_authors_menu* é responsável por exibir este submenu dos proprietários. Também aqui é desenhado um fundo específico e adicionados botões e imagens. O botão "VOLTAR" permite ao utilizador regressar ao menu principal. Quando isso acontece, o método *clear_authors_menu* é chamado para apagar o menu e repor o menu principal.

É através da função *run* que o programa corre. Este executa um ciclo infinito que aguarda cliques do utilizador através do método *win.getMouse()*. No menu principal, clicar em "TIER1", "TIER2" ou "TIER3" encerra a janela e importa o módulo correspondente de projeto, iniciando o modo de jogo selecionado.

1.2.2. Módulo projeto

Neste módulo existem enumeras classes que são a espinha dorsal de todo o programa apresentado.

A classe “Waiter” trata da representação, movimento, bateria e obstáculos apresentados em todos os tiers. Está definido no início através de funções do *graphics*. A primeira função *consumir_bateria* trata da atualização de uma variável “quantidade” que se refere à quantidade de bateria que o robô possui. A segunda função, *Carregar_bateria*, como o nome indica, define a zona de carregamento e, aquando da presença do robô nela, carrega o robô num loop que carrega o robô em 10 pontos percentuais a cada quinto de segundo.

A função *cor_bateria* altera as cores da bateria e do mostrador aquando da mudança de bateria. *mover_tier1* executa o movimento do robô no tier 1 através do cálculo da distância do ponto querido e da posição atual do robô, os movimentos são executados sempre em 100 etapas, independentemente da distância, logo demorarão sempre o mesmo tempo. Na função *move_tier2* é adicionada ao que já existe na anterior uma deteção de clique que executa a função *obstáculo,* função essa que cria um obstáculo na sala caso o clique não tenha sido efetuado dentro de certas áreas ou em cima das mesas ou das divisórias. Esta função também define os limites do obstáculo pois sendo só uma imagem teria apenas um ponto. Os limites são utilizados pela função *colisao* que deteta se existe um choque entre o robô e os obstáculos. Na função *move_tier2* caso haja *colisao* ele remove o obstáculo. O *move_tier3* é muito semelhante.

A *table_check* deteta se o clique efetuado foi em cima de uma mesa. Em par com esta existe a *go_to_table_tier1* que executa o movimento para as mesas de uma forma muito simples, movendo se entre pontos e removendo a comida.

A função *go_to_table_tier2* executa os movimentos do robô quando chamado para uma certa mesa numa certa coluna, este vai à mesa, vai à cozinha, entrega a comida e volta à sua estação para carregar a bateria. A função *go_to_table_tier3* funciona pois à priori foram criadas ordens possíveis para a entrega e ela executa as na menor distância possível.

A função do *mostrador* desenha um mostrador e atualiza-o ao longo da mudança de valores percentuais da bateria.

A classe "Button" através das funções *is_click_in_button* e *create_button* deteta se o clique foi efetuado dentro das fronteiras do botão e cria o mesmo respetivamente. A classe "Saida" define por cima do botão e utilizando a classe "Button" o botão de saída que leva o utilizador ao menu principal. A classe "Table" define as mesas e tem a função *det_table* que permite a deteção do clique em cada mesa. Todas as classes "Divisao", "Cozinha", "Estação" apenas definem estas "coisas" (para não lhes chamar de objetos). As coordenadas são retiradas do salaxx.txt. A classe "Sala" define os limites da sala e cria a janela, desenha todas as classes referidas através do ficheiro .txt podendo ser alteradas as suas posições no mesmo. A função *run* é o que, de facto, executa a sala, existem de pois duas funções *devolver_mesas* e *devolver_Div* que existem para a função *obstaculo* se "lembrar" dos limites.

1.2.3. Módulo projeto

O módulo projeto é muito simples, tem três funções: *tier1*, *tier2* e *tier3* que executam a sala, mais tarde o robô e em seguida as funções dos obstáculos e cliques, este é como que um passo intermédio entre o módulo "classes" e o "menu_principal".

1.3. projeto

De acordo com as funcionalidades do programa, existem entradas de dados de ficheiros (como sala49.txt) e cliques no ecrã. As saídas manifestam-se como alterações visuais na janela do programa, através do desenho de objetos como mesas, divisões, cozinha, estação de carregamento e o robô, bem como a sua interação com estes elementos.

Função:

Este módulo define todas as classes e funções necessárias para a simulação da sala e da operação do robô, incluindo a renderização gráfica dos elementos, movimentação do robô, deteção de cliques, verificação de colisões e lógica de entrega/recolha.

1.3.1. Entradas (aplicável para funções)

- Ficheiro sala49.txt (lido por Sala.desenhar)
- Clique do utilizador no ecrã (window.checkMouse() e getMouse())
- Parâmetros das classes:
    - Waiter: window, center_point, body_radius, battery_level, posicoes_mesa, posicoes_Div
    - Table: win, x1, y1, x2, y2, color, ident
    - Divisao, Cozinha: win, x1, y1, x2, y2, color
    - Estacao: win, x1, y1, radius, color
    - Saida: win, x1, y1, x2, y2
- Coordenadas de destino (destino_x, destino_y) nas funções de movimento (mover_tier1, mover_tier2, mover_tier3)
- Pontos clicados no ecrã para deteção de mesa ou obstáculos

- Listas de mesas e divisões para verificação de colisão e cliques

1.3.2. Saídas (aplicável para funções)

- Alterações visuais na janela:
    - Desenho de objetos (mesas, cozinha, robô, obstáculos)
    - Atualização da cor da bateria e mostrador
    - Movimento animado do robô
    - Entrega de comida e remoção de obstáculos
- Impressões de debug no terminal:
    - Nível da bateria
    - Detecção de cliques e colisões
- Retornos booleanos:
    - table_check, go_to_table_tier2, etc.
- Imagens desenhadas ou removidas:
    - bifana.png, jola.png, sair2.png

1.3.3. Parâmetros (aplicável para classes)

- Waiter: window, center_point, body_radius, battery_level, posicoes_mesa, posicoes_Div
- Table: win, x1, y1, x2, y2, color, ident
- Divisao: win, x1, y1, x2, y2, color
- Cozinha: win, x1, y1, x2, y2, color

- Estacao: win, x1, y1, radius, color
- Saida: win, x1, y1, x2, y2
- Sala: (sem parâmetros diretos, instancia GraphWin internamente)

1.3.4. Métodos (aplicável para classes)

- Waiter:
  - __init__
  - consumir_bateria
  - carregar_bateria
  - mostrador
  - atualizar_mostrador
  - cor_bateria
  - mover_tier1, mover_tier2, mover_tier3
  - table_check
  - go_to_table_tier1, go_to_table_tier2, go_to_table_tier3
  - colisao
  - obstaculo
- Table:
  - __init__
  - det_table
- Divisao, Cozinha, Estacao:

- __init__

- Saida:
  - __init__
- Sala:
  - __init__
  - desenhar
  - run
  - devolver_mesas
  - devolver_Div
- Button:
  - is_click_in_button
  - create_button