TÉCNICO LISBOA

Projeto de Fundamentos de Programação

Documentação técnica

GRUPO Nº 09

106426	Pedro Rosa pedro.b.rosa@tecnico.ulisboa.pt
105995	João Miguel joao.tomas.miguel@tecnico.ulisboa.pt

Licenciatura em Engenharia Mecânica

Lisboa, junho de 2023

Índice

1. Documentação técnica

1.1. Arquitetura do programa

O programa contempla três grandes módulos: o nível_1, o nível_2 e o nível_3. Em qualquer nível que o utilizador escolha, pode sempre voltar ao menu. Na última implementação – nível_3 – existem diversos menus de forma que o utilizador possa escolher entre as formas possíveis de obter coordenadas de obstáculos e de sujidades. Nestes menus é também dada a opção ao utilizador de "voltar atrás", tal como esquematizado na figura abaixo.

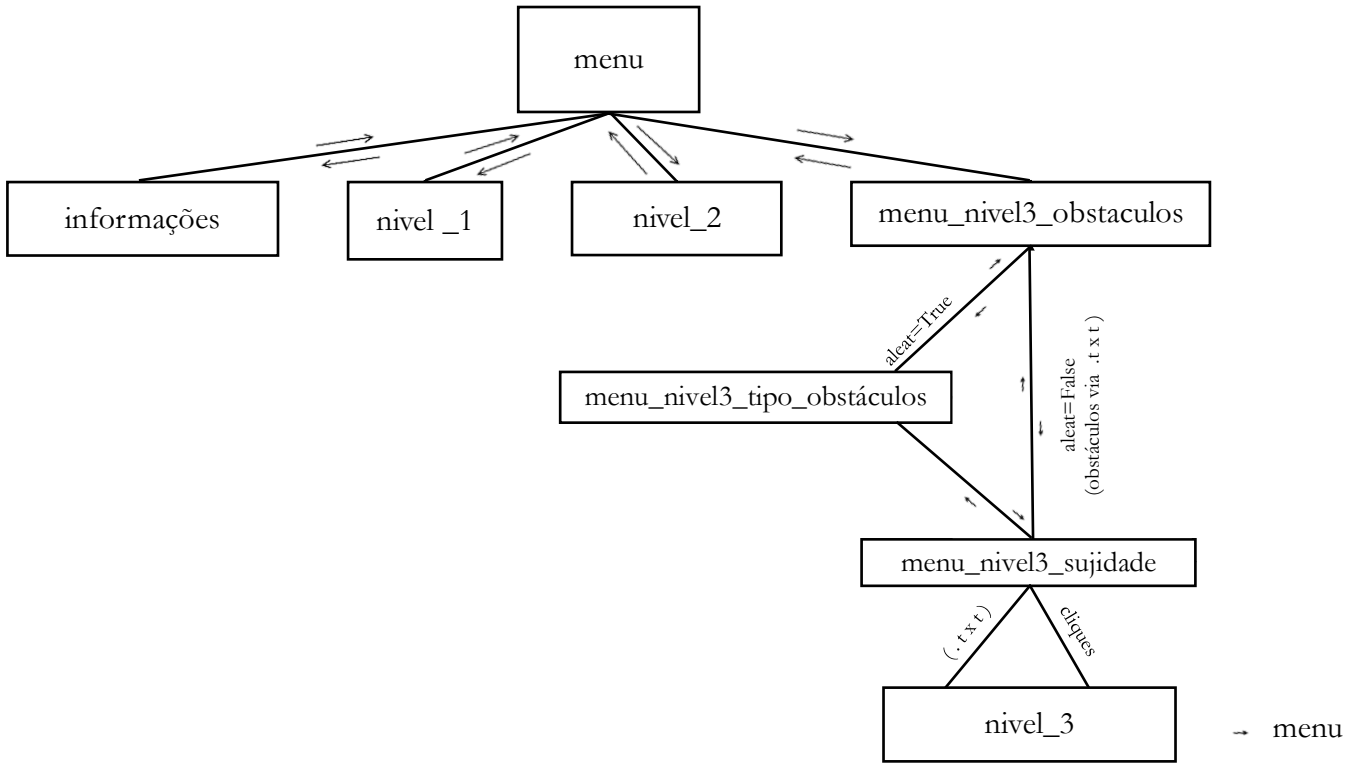


Figura 1: diagrama com os vários módulos, funções e variável *aleat*

1.2. Projeto detalhado

Classes:

- **Waiter** - Esta classe representa um *robot*. O *robot* possui vários atributos, como a sua posição (x, y), cor, estado da bateria e uma representação gráfica usando círculos numa janela gráfica.

  O método "**__init__**" inicializa os atributos do *robot* e desenha a sua representação gráfica na janela fornecida (**win**).

  O método "**troca**" remove a representação gráfica do *robot* da janela.

  O método "**mover**" move o *robot* para uma posição especificada (x, y). Calcula também a distância a percorrer, divide-a em passos e move o *robot* gradualmente para criar uma animação. Por fim, atualiza a posição do *robot*.

  O método "**ir_para_docking_station**" move o *robot* de volta para a estação de carregamento mais próxima (lado esquerdo ou direito) para carregar a sua bateria. Exibe uma mensagem de carregamento, simulando um tempo de carregamento de 2 segundos e, de seguida, atualiza o estado da bateria e oculta a mensagem de carregamento.

  O método "**movimento_espiral**" move o *robot* numa trajetória espiral circular, movendo o *robot* gradualmente para criar uma animação. Também verifica a existência de obstáculos de forma a que se o *robot* estiver a descrever a espiral e encontre um obstáculo, para antes de colidir.

  O método "**tirar_a_posição**" devolve a posição atual do *robot* uma lista de coordenadas.

  O método "**distancia_percorrida**" atualiza a distância percorrida e reduz o nível da bateria em conformidade.

  O método "**atualizar_leds**" atualiza a cor do contorno do *robot* (LED) com base no nível da bateria. Se o nível da bateria estiver criticamente baixo, retorna **True** para indicar a necessidade de recarga.

  Variáveis:

  1. x: Variável que representa a coordenada x do *robot*.
  2. y: Variável que representa a coordenada y do *robot*.
  3. win: Variável que representa a janela gráfica onde o *robot* é exibido.
  4. distancia_percorrida: Variável que armazena a distância percorrida pelo *robot*.
  5. robot: Objeto gráfico que representa o *robot*.
  6. color: Variável que armazena a cor do *robot*.
  7. bateria_maxima: Variável que armazena a capacidade máxima da bateria do *robot*.
  8. bateria_atual: Variável que armazena a carga atual da bateria do *robot*.
  9. posiçãox_animação: Variável que armazena a posição x durante a animação de movimento do *robot*.
  10. posiçãoy_animação: Variável que armazena a posição y durante a animação de movimento do *robot*.
  11. doc_stat: Variável que indica a direção da estação de carregamento mais próxima.

12. texto: Objeto gráfico usado para exibir mensagens na interface gráfica.
13. nivel: Variável que representa o nível de espiral do movimento do *robot*.
14. aleatorio: Variável que indica se o movimento do *robot* é aleatório ou não.
15. lista: Lista que armazena as coordenadas dos obstáculos.
16. x1, y1, x2, y2: Variáveis que armazenam as coordenadas dos limites dos obstáculos.
17. i, n: Variável de controle usada em loops.
18. posição_inicial: Lista que armazena as coordenadas iniciais do *robot*.
19. dist: Variável que representa a distância percorrida em cada movimento do *robot*.
20. carregar: Variável que armazena um valor booleano que transmite se o *robot* está a carregar.

- **Docking_Stations** - A classe "**Docking_Stations**" representa as estações de recarga para o *robot*.

No método "**__init__**", a classe inicializa as estações de recarga como retângulos e desenha-as na janela gráfica. São criadas duas estações de recarga, uma no lado esquerdo e outra no lado direito. Além disso, são desenhados dois LEDs (indicadores) de cor vermelha ao lado de cada estação de recarga.

Variáveis:

1. win: Variável que representa a janela onde as estações de carregamento são exibidas.
2. docking_stations: Lista que armazena os objetos de estação de carregamento.
3. doc_stat: Variável usada como uma variável de loop para iterar sobre os objetos de estação de carregamento.
4. led1, led2: Objetos que representam os indicadores de LED para as estações de carregamento.

- **Table** - A classe "**Table**" cria uma mesa com diferentes formatos. O construtor recebe os seguintes argumentos: "**x1, y1, x2, y2**" (coordenadas dos cantos da mesa), "**win**" (janela gráfica) e "**tipo_de_mesa**" (tipo de formato da mesa).

No método "**__init__**", a classe verifica o tipo de mesa especificado. Se for "oval", cria uma mesa oval com dimensões de 12x11. Se for "retangular", cria uma mesa retangular com dimensões de 12x11. Se for "circular", cria uma mesa circular com raio 6.

Para cada tipo de mesa, são definidos os atributos correspondentes, como cor de preenchimento, cor do contorno, largura do contorno e os objetos adicionais como pratos e talheres.

No final, todos os objetos gráficos são desenhados na janela gráfica.

Variáveis:

1. formato: Variável que representa a forma da mesa (oval, retangular ou circular). É um objeto *Oval*, *Rectangle* ou *Circle*, dependendo do tipo de mesa.

2. cor_individual: Variável que armazena a cor do individual, que é diferente para cada formato de mesa.
3. prato1, prato2: Variáveis que representam os pratos da mesa.
4. circulo1, circulo2: Variáveis que representam os círculos decorativos da mesa.
5. talher1, talher2, talher3, talher4: Variáveis que representam os talheres da mesa.
6. copo1, copo2: Variáveis que representam os copos da mesa.
7. individual1, individual2: Variáveis que representam os dois individuais de cada mesa.

- **botao** - A classe "**botao**" constrói botões na janela gráfica. O construtor recebe os seguintes argumentos: "**x1, y1, x2, y2**" (coordenadas dos cantos do retângulo que representa o botão), "**texto**" (texto exibido no botão) e "**win**" (janela gráfica).

  No método "**__init__**", a classe cria um retângulo representando o botão com base nas coordenadas fornecidas. Também é criado um objeto de texto com base nas coordenadas do centro do botão e o texto fornecido. O texto é configurado com a fonte "helvetica", tamanho 16 e estilo "bold italic". Em seguida, é desenhado na janela gráfica.

  A classe também possui um método "**ativar**" que recebe as coordenadas "**x**" e "**y**" de um clique do mouse. Esse método verifica se as coordenadas do clique estão dentro dos limites do retângulo do botão. Se estiverem, o método devolve **True**, indicando que o botão foi clicado.

  Variáveis:

1. xmin, xmax, ymin, ymax: Variáveis que armazenam as coordenadas mínimas e máximas do botão para fins de detecção de clique.
2. botao: Objeto Rectangle que representa o botão na janela.
3. txt: Objeto Text que representa o texto a ser exibido no botão.

- **Input_Box –** Esta classe foi criada de modo ao utilizador poder inserir o número de objetos de cada tipo que deseja ter no nível 3.

  O método "**__init__**" cria a imagem gráfica das caixas de textos disponibilizadas pela biblioteca graphics. O método **analisar_valores** analisa a string introduzida pelo utilizador e devolve um inteiro. O método **zerar_valores** serve para colocar a entrada de uma caixa de texto a '0', útil quando o utilizador excede o número de obstáculos permitidos, por exemplo.

  Variáveis:

1. x: posição *x* da caixa de texto
2. y: posição *y* da caixa de texto
3. caracteres: número de caracteres visíveis na caixa
4. texto: texto inicial que contempla a caixa

5. win: variável de janela gráfica

Funções:

- **Funções que geram as janelas gráficas onde o utilizador irá interagir:**
  1. **menu()**: Abre o menu principal do jogo. Lê informações do arquivo "menu.txt" para definir as propriedades da janela, como tamanho e nome. De seguida, cria a janela gráfica, exibe o título e os botões do menu.
  2. **nivel1(nivel)**: Abre o nível 1 do jogo. Lê informações do arquivo "nivel1.txt" para definir as propriedades da janela, como tamanho e nome. De seguida, cria a janela gráfica, exibe o título e outras informações relacionadas ao nível. Chama a função sujidade_e_mesa() para exibir sujidade e mesas na janela, cria um objeto Waiter, e chama a função detetor_do_rato1() para capturar cliques do utilizador.
  3. **nivel2(nivel)**: Abre o nível 2 do jogo. Lê informações do arquivo "nivel2.txt" para definir as propriedades da janela, como tamanho e nome. De seguida, cria a janela gráfica, exibe o título e outras informações relacionadas ao nível. Chama a função ler_tipo_de_obstaculo() para ler e exibir informações sobre obstáculos no nível. Cria um objeto Waiter, chama a função detetor_do_rato2() para capturar cliques do usuário e gerar sujidade aleatoriamente.
  4. **menu_nivel3_coordenadas_obstáculos(nivel)**: Abre o menu do nível 3 para definir a disposição esparsa dos objetos. Cria uma janela gráfica, exibe caixas de entrada para definir o número de objetos, e botões para gerar e começar o jogo. Essa função também contém um loop que verifica os cliques do usuário nos botões.
  5. **menu_nivel3_sujidade(nivel, lista_coordenadas, aleatorio)**: Abre o menu do nível 3 para decidir a forma de gerar sujidade. Cria uma janela gráfica, exibe botões relacionados às opções de geração de sujidade e coordenadas, e verifica os cliques do usuário.
  6. **nivel3(win, lista_coordenadas, nivel, aleatorio, coor)**: Abre o nível 3 do jogo. Lê informações do arquivo "nivel3.txt" para definir as propriedades da janela, como tamanho e nome. De seguida, cria a janela gráfica, exibe o título e outras informações relacionadas ao nível. Chama a função botoes_nivel3() para exibir botões relacionados às ações do jogador no nível.
  7. **informaçoes()**: Esta função é responsável por desenhar uma janela gráfica chamada "Informações" e exibir algumas informações sobre o projeto. Ela também cria um botão "Voltar" e aguarda interação do usuário.
  8. **menu_nivel3_obstaculos(nivel)**: Esta função abre o menu para o nível 3 de um jogo. Lê informações de um arquivo, cria uma janela gráfica com base nas dimensões especificadas e exibe um menu para os obstáculos do nível 3.

Módulo Graphics

Este módulo fornece uma interface orientada a objetos simples para criar e manipular janelas gráficas.

Variáveis:

1. janela: Variável que armazena as informações sobre a janela gráfica, como largura e altura.
2. nome_da_janela: Variável que armazena o nome da janela gráfica.
3. win: Variável que representa a instância da janela gráfica criada com o uso da classe "GraphWin".
4. botao_de_terminar, botao_de_começar, bot_1, bot_2, bot_3: Variáveis que representam instâncias da classe "botao", que criam botões em diferentes partes do jogo.
5. docking_stations: Variável que representa uma instância da classe "Docking_Stations", que cria uma estação de carregamento para o objeto "Waiter".
6. robot: Variável que representa uma instância da classe "Waiter", que representa o objeto Waiter na janela gráfica.
7. coor: Lista que armazena as coordenadas da sujidade a serem limpas.
8. clique: Uma variável que armazena as coordenadas de um clique do mouse na janela gráfica.
9. x, y: Variáveis que armazenam as coordenadas x e y extraídas do "clique".
10. linhas: Uma lista que armazena as linhas lidas de um arquivo.
11. janela: Uma lista que armazena as dimensões de uma janela extraídas do arquivo.
12. nome_da_janela: Uma variável de string que armazena o nome de uma janela extraído do arquivo.
13. ficheiro: Um objeto de arquivo usado para ler dados de um arquivo.

- **Funções que vão buscar dados aos arquivos txt:**
  1. **lista_de_coordenadas_obstaculos(nivel, aleatorio)**: Uma função que retorna uma lista com as coordenadas dos obstáculos, dependendo do nível e se os obstáculos são gerados aleatoriamente.
  2. **nivel_3_acrescento(aleatorio)**: Uma função que retorna uma lista com as coordenadas dos obstáculos do nível 3, sendo geradas aleatoriamente ou não.
  3. **ir_buscar_coordenadas_sujidade(win, inicio, nivel)**: Uma função que cria uma lista com as coordenadas de sujidades, dependendo do nível e da posição inicial.

Arquivos:

1. "nivel1.txt": Um arquivo que contem as coordenadas dos obstáculos para o nível 1.
2. "nivel2.txt": Um arquivo que contem as coordenadas dos obstáculos para o nível 2.
3. "dados.txt": Um arquivo que contem as coordenadas dos obstáculos para o nível 3 (se geradas aleatoriamente).

4. "Sala.txt": Um arquivo que contem as coordenadas dos obstáculos para o nível 3 (se não geradas aleatoriamente).
5. "Limpeza.txt": Um arquivo contendo informações de limpeza.

Variáveis:

1. nivel: Uma variável que indica o nível do jogo.
2. aleatorio: Uma variável que indica se os obstáculos são gerados aleatoriamente ou não.
3. ficheiro: Uma variável que representa um arquivo.
4. linhas: Uma lista que armazena as linhas lidas de um arquivo.
5. lista_de_coordenadas_obs: Uma lista que armazena as coordenadas dos obstáculos.
6. n: Uma variável utilizada em loops para percorrer as linhas ou os valores em uma linha.
7. linha: Uma lista que armazena os valores separados de uma linha.
8. i: Uma variável utilizada em loops para percorrer os valores numa linha.
9. coordenadas_de_sujidade: Uma lista que armazena as coordenadas de sujidades.
10. win: Um objeto que representa a janela do jogo.
11. inicio: Uma variável que indica a posição inicial para o loop.

- **Funções que desenham objetos na janela gráfica:**
  1. **ler_tipo_de_obstaculo(nivel,win)**: Esta função lê o tipo de obstáculos de um arquivo, com base no nível a ser executado, e chama a função "desenhar_obstaculos" para desenhá-los na janela "win". Ela recebe como parâmetros o nível (nivel) e a janela onde serão desenhados os obstáculos (win).
  2. **desenhar_obstaculos(linha,nivel,win,i)**: Essa função recebe uma linha que contém as informações sobre um obstáculo e desenha o obstáculo correspondente na janela win. Os parâmetros são a linha de informações sobre o obstáculo (linha), o nível a ser executado (nivel), a janela (win) e um índice (i) para ter acesso às coordenadas do obstáculo na lista de coordenadas.
  3. **vaso(x1,y1,x2,y2,win), piano(x1,y1,x2,y2,win), cadeira(x1,y1,x2,y2,win)**: Essas funções são responsáveis por desenhar os obstáculos específicos na janela "win", com base nas coordenadas fornecidas (x1, y1, x2, y2). Cada função desenha um tipo de obstáculo diferente.
  4. **sujidade(suj_x, suj_y, cor1, cor2, win)**: Esta função desenha uma sujidade na janela "win", com base nas coordenadas (suj_x, suj_y) e nas cores fornecidas (cor1 e cor2). A sujidade é representada por várias formas (círculos e ovais).
  5. **sujidade_e_mesa(win, nivel, inicio=7)**: Esta função separa as coordenadas dos pontos de sujidade e as coloca no jogo, juntamente com a mesa. Ela recebe como parâmetros a janela "win", o nível em execução "nivel" e um valor opcional "inicio", que indica a linha inicial das coordenadas de sujidade. A função

seleciona aleatoriamente um tipo de mesa e obtém as coordenadas da mesa e da sujidade a partir de outras funções.

6. **mensagem_de_erro(win)**: Esta função exibe uma mensagem de erro na janela "win". Desenha um retângulo branco na janela e exibe um texto que informa o utilizador sobre o erro nas coordenadas. Após exibir a mensagem, a função aguarda um clique do rato e fecha a janela, chamando a função "**menu()**".
7. **fim_do_nivel(win)**: Esta função exibe uma mensagem de fim de nível na janela win. Desenha um retângulo branco na janela e exibe um texto informando que a limpeza foi concluída. A função aguarda um clique do rato num botão "sair" para fechar a janela e chamar a função "**menu**()".
8. **txt(texto, x, y, tamanho, estilo, win)**: Esta função facilita a escrita de texto personalizado numa janela gráfica. Ela cria um objeto **Text** com o texto especificado e posiciona-o na coordenada (x, y) da janela. Além disso, permite personalizar o tamanho e o estilo do texto.
9. **desenho1(coordenadas_de_sujidade, win, nivel, aleatorio):** Esta função desenha a sujidade numa janela gráfica. A função percorre as coordenadas da sujidade, verifica se estão dentro da janela e não estão dentro dos limites dos obstáculos, e desenha a sujidade na janela usando a função "**sujidade**". Em caso de a sujidade não estar num sitio em que pode estar, a função "**mensagem_de_erro**" é chamada.
10. **led(doc_stat, cor, win):** Esta função desenha os LEDs das *docking stations* numa janela gráfica. A função cria um retângulo com as coordenadas adequadas dependendo da posição da *docking station* e define a cor de preenchimento desse retângulo. Em seguida, o retângulo é desenhado na janela gráfica.
11. **limpeza_da_sujidade(x, y, robot, win, coor, n, nivel, aleatorio):** Esta função é responsável por simular a limpeza de uma sujidade específica na janela gráfica.

Classes:

É utilizada a classe "**Table**"

Variáveis:

1. nivel: Variável que indica o nível que está em funcionamento. É um número inteiro.
2. win: Variável que representa a janela onde os obstáculos serão desenhados. É uma instância da classe "**GraphWin**".
3. ficheiro: Variável que representa o arquivo de onde são lidos os dados dos obstáculos. É um objeto de arquivo.
4. linha: Variável que armazena a linha atual do arquivo sendo lido. É uma string.
5. linhas: Lista que armazena todas as linhas do arquivo lido. Cada elemento da lista é uma string.
6. i: Variável de controle usada nos loops. É um número inteiro.
7. lista: Lista que armazena as coordenadas dos obstáculos. São números inteiros ou reais.
8. impureza1, impureza2, impureza3, impureza4, impureza5, impureza6: Variáveis que representam as formas que compõem a sujidade. Elas podem ser objetos gráficos, como ovais e círculos.

9. cor1, cor2: Variáveis que representam as cores utilizadas na sujidade. São strings que representam as cores em formato RGB ou nomes de cores padrão.
10. mesa: Variável que representa a mesa. É um objeto da classe Table.
11. x1, y1, x2, y2: Variáveis que armazenam as coordenadas da mesa. São floats que representam as posições x e y.
12. coordenadas_de_sujidade: Variável que armazena as coordenadas dos pontos de sujidade. É uma estrutura de dados que pode ser uma lista ou outro tipo adequado para armazenar as coordenadas.
13. tamanho: O tamanho do texto.
14. estilo: O estilo do texto/Fonte.
15. texto: O texto que será escrito.
16. doc_stat: Variável que indica a posição da *docking station* ("esquerda" ou "direita").
17. cor: Variável que indica a cor do LED.
18. led: Objeto retângulo que representa o LED.

- **Funções para desviar de obstáculos:**
  1. **desviar_obstaculo(x, y, robot, nivel, aleatorio)**: Esta função simula o caminho que o *robot* irá percorrer para verificar se ele passa por cima de algum obstáculo. A função calcula a distância até o destino, divide-a em passos e percorre cada passo, verificando se houve a necessidade de desviar do obstáculo usando a função "verificar_a_posição_e_desviar_da_mesa".
  2. **verificar_a_posição_e_desviar_da_mesa(robot, posição_x, posição_y, nivel, mudou_direção, aleatorio)**: Esta função verifica se o *robot* passa por cima de algum obstáculo. A função obtém a lista de coordenadas dos obstáculos usando a função "lista_de_coordenadas_obstaculos" e percorre essa lista, verificando se a posição atual está dentro das coordenadas de algum obstáculo. Se estiver, chama a função "muda_a_direção1" ou "muda_a_direção2" dependendo do nível, para desviar do obstáculo.
  3. **muda_a_direção1(x1, x2, y1, y2, posição_x, posição_y, robot)**: Esta função é chamada quando estamos no primeiro nível e o *robot* precisa de se desviar de um obstáculo. Com base nas coordenadas do obstáculo e na posição atual do *robot*, a função determina a nova posição para a qual o robô deve se mover para se desviar do obstáculo.
  4. **muda_a_direção2(x1, x2, y1, y2, posição_x, posição_y, robot)**: Esta função é chamada quando estamos no segundo ou no terceiro nível e o *robot* precisa de se desviar de um obstáculo. Tal como na função anterior, com base nas coordenadas do obstáculo e na posição atual do robô, a função determina a nova posição para a qual o robô deve se mover para desviar do obstáculo.

     Variáveis:

1. x e y: Representam as coordenadas de destino para as quais o *robot* está a mover-se. São números inteiros ou reais.
2. robot: É um objeto que representa o *robot*. É uma instância da classe "Waiter" que possui métodos para controlar o movimento do *robot*.
3. nivel: Indica o nível do programa. É um número inteiro que determina o comportamento do *robot* e a escolha dos obstáculos.
4. aleatorio: Indica se a geração dos obstáculos é aleatória ou não. É um valor booleano (verdadeiro ou falso).
5. x_inicial e y_inicial: Armazenam as coordenadas iniciais do *robot* antes de começar um novo movimento. São números inteiros ou reais.
6. dx e dy: Representam as diferenças nas coordenadas x e y entre a posição atual do *robot* e as coordenadas de destino. São números inteiros ou reais.
7. distância: Armazena a distância entre a posição atual do *robot* e as coordenadas de destino.
8. passos: Representa o número de passos necessários para percorrer a distância até o destino. É um número inteiro.
9. posição_x e posição_y: Representam as coordenadas atuais do *robot* enquanto ele simula o seu movimento. São números inteiros ou reais.
10. mudou_direção: É uma variável booleana que indica se houve uma mudança de direção durante o desvio de obstáculos.
11. lista: É uma lista que armazena as coordenadas dos obstáculos.

- **Funções para detetar cliques do rato e funcionar de acordo:**

  Menu:

  1. **botoes(win):** Esta função permite que o usuário escolha o nível por meio de botões na janela "win". Ela cria os botões na posição especificada e aguarda o clique do mouse do usuário. O botão clicado determina a ação a ser realizada. Se o botão "Nível 1" for clicado, a janela é fechada, e a função "**nivel1(nivel)**" é chamada com "nivel" definido como 1. O mesmo ocorre para os botões "Nível 2" e "Nível 3". Se o botão "Informações" for clicado, a janela é fechada, e a função "**informacoes()**" é chamada. Se o botão "Terminar" for clicado, a janela é fechada.

  Nível 1:

  1. **detetor_do_rato1(win, robot, nivel, x=None, y=None)**: Essa função deteta se o usuário clica na barra de tarefas ou na sala do restaurante. Se x ou y forem **None**, significa que o usuário ainda não clicou na barra de tarefas anteriormente. A função entra num loop infinito que aguarda o clique do mouse do usuário. Se o clique ocorrer acima da linha y=86, a função "**botoes_da_barra1(win, x, y,**

robot, nivel)" é chamada, passando as coordenadas do clique, o objeto robot e o nível. Caso contrário, a função "**ir_limpar(x, y, win, robot, nivel)**" é chamada para realizar a ação de limpeza na posição em que o usuário clicou na sala do restaurante, e a função "**led('esquerda', 'red', win)**" é chamada para acender o LED esquerdo em vermelho.

2. **botoes_da_barra1(win, x, y, robot, nivel)**: Essa função exibe botões na barra de tarefas para permitir que o usuário saia dos níveis e volte para o menu, ou volte para a sala do restaurante. A função entra num loop infinito aguardando o clique do rato do usuário. Se o clique ocorrer dentro das coordenadas especificadas para o botão "Sair", a janela é fechada e o menu é exibido. Se o clique ocorrer abaixo da linha y=86, a função "**ir_limpar(x, y, win, robot, nivel)**" é chamada para realizar a ação de limpeza na posição em que o usuário clicou na sala do restaurante. Caso contrário, a função "**detetor_do_rato1(win, robot, nível)**" é chamada para continuar a deteção de cliques.

Variáveis:

1. x: Representa a coordenada x do último clique do mouse na sala do restaurante ou na barra de tarefas. Seu valor padrão é **None**.
2. y: Representa a coordenada y do último clique do mouse na sala do restaurante ou na barra de tarefas. Seu valor padrão é **None**.
3. win: Representa a janela gráfica em que os botões estão a ser exibidos.
4. robot: Representa o objeto do robô utilizado no programa.
5. nivel: Representa o nível atual do jogo.

Nível 2 e Nível 3:

1. **botoes_nivel3_obstaculos(win)**: Esta função exibe os botões na janela gráfica para que o usuário possa escolher como deseja gerar os obstáculos no nível 3. Os botões permitem gerar os obstáculos através do bloco de notas ou de forma aleatória. Há também um botão para voltar ao menu principal.
2. **botoes_nivel3_sujidade(win, lista_coordenadas, aleatorio)**: Esta função exibe os botões na janela gráfica para que o usuário possa escolher como deseja gerar a sujidade no nível 3. Os botões permitem gerar a sujidade através de um arquivo ou através de cliques na área dos restaurante. Há também um botão para voltar ao menu principal.
3. **botoes_nivel3(win, lista_coordenadas, nivel, aleatorio, coor)**: Esta função exibe os botões na janela gráfica para o nível 3. Os botões permitem iniciar o nível, sair do jogo e interagir com a sujidade e obstáculos na sala do restaurante. Dependendo da escolha do utilizador, os cliques para a sujidade podem ser ou não ser utilizados.
4. **detetor_do_rato2(win, nivel, coor, robot, aleatorio, x=None, y=None)**: Esta função é responsável por detetar o clique do rato na sala do restaurante ou na barra de tarefas durante o nível.

Ela determina se o clique ocorreu em algum botão da barra de tarefas ou em algum ponto da sala do restaurante para interagir com a sujidade ou obstáculos.

5. **botoes_da_barra2(win, x, y, nivel, coor, robot, aleatorio)**: Esta função exibe os botões da barra de tarefas durante o nível 3. Os botões permitem sair do jogo, voltar para o menu principal, começar o nível.
6. **sujidade2(x, y, win, nivel, coor, robot, aleatorio)**: Função responsável por permitir que o usuário escolha onde colocar a sujidade na sala do restaurante por meio de cliques na janela gráfica. Após cada clique, a sujidade é desenhada na posição selecionada, e as coordenadas são adicionadas à lista "coor". A função continua a aguardar cliques até que o usuário clique na barra de tarefas.
7. **carregou_perto_da_mesa(x, y, nivel, aleatorio)**: função que verifica se o clique do rato ocorreu próximo a um obstáculo na sala do restaurante. Ela percorre a lista de coordenadas dos obstáculos, verifica se o ponto (x, y) está dentro da área de uma mesa e, em caso positivo, ajusta as coordenadas x ou y para garantir que o ponto fique fora da mesa. Isso é útil para evitar que a sujidade seja colocada exatamente em cima de uma mesa. Retorna uma lista contendo as novas coordenadas (x, y) ajustadas.
8. **não_carregou_em_cima_de_obstaculo(x, y, nivel, aleatorio):** Esta função verifica se o clique do rato ocorreu em cima de algum obstáculo na sala do restaurante. Ela percorre a lista de coordenadas dos obstáculos e verifica se o ponto (x, y) está dentro da área de algum obstáculo ou da docking station. Se o ponto estiver dentro de algum obstáculo, a função retorna **False**, indicando que o ponto foi carregado em cima de um obstáculo. Caso contrário, retorna **True**.
9. **carrega_perto_da_mesa(x, y, robot, nivel)**: Esta função é responsável por movimentar o *robot* em direção à mesa mais próxima, contornando obstáculos no caminho.

Variáveis:

1. win: Representa a janela gráfica em que os botões estão sendo exibidos.
2. nivel: Representa o nível atual do programa.
3. coor: Lista de coordenadas da sujidade do nível.
4. robot: Representa o objeto do *robot* utilizado no programa.
5. aleatorio: Variável booleana que indica se a geração dos obstáculos é aleatória.
6. x: Representa a coordenada x do último clique do mouse na sala do restaurante ou na barra de tarefas. O seu valor padrão é None.
7. y: Representa a coordenada y do último clique do mouse na sala do restaurante ou na barra de tarefas. O seu valor padrão é None.
8. lista_coordenadas: Lista de coordenadas dos obstáculos do nível.

- **Funções para a limpeza da sala no nível 1:**
  1. **ir_limpar(x, y, win, robot, nivel)**: Esta função é responsável por enviar o *robot* para a posição em que o usuário clicou na janela gráfica. Ela acende um LED correspondente à docking station esquerda, lê as coordenadas de sujidade de um arquivo, verifica se o robô limpou todas as sujidades e executa a função "**verificar_se_limpou**" para lidar com a limpeza das sujidades, o desvio de obstáculos e o retorno à *docking station*.
  2. **verificar_se_limpou(x, y, robot, win, lista, coor, nivel)**: Esta função contém um loop que verifica se o *robot* limpou todas as sujidades. Verifica a posição do *robot* e chama outras funções para desviar de obstáculos. O loop continua até que todas as sujidades tenham sido limpas. A função também aguarda o clique do usuário na janela gráfica para interagir com o programa.
  3. **deteta_e_limpa(limpo, x, y, coor, robot, win, nivel)**: Esta função é responsável por detetar e limpar as sujidades. Ela itera sobre a lista de coordenadas de sujidade e verifica se o *robot* está na da área de uma sujidade. Se o *robot* estiver dentro da área de uma sujidade que ainda não foi limpa, a sujidade é marcada como limpa e a função "**limpeza_da_sujidade**" é chamada para realizar a limpeza. Após a limpeza, a função redefine a posição do **robot** e continua a verificar outras sujidades.

  Variáveis:

  1. x: A coordenada x atual do *robot*. Ou a coordenada x do clique do usuário.
  2. y: A coordenada y atual do *robot*. Ou a coordenada y do clique do usuário.
  3. robot: O objeto da classe "Waiter" que representa o *robot*.
  4. win: A janela gráfica onde o *robot* se está a mover.
  5. lista: Uma lista que contem os números de identificação das sujidades.
  6. coor: Uma lista que contem as coordenadas das sujidades.
  7. nivel: O nível em que o *robot* está a correr.
  8. limpo: Uma lista que contem os números de identificação das sujidades já limpas.

- **Funções para a limpeza da sala no nível 2 e 3:**
  1. **começar(robot, nivel, win, coor, aleatorio)**: Esta função é a função principal que inicia o movimento do *robot*. Ela chama outras funções, como "**ir_voltar**", **"detetor_do_rato2**", "**mover**", "**ir_para_docking_station**" e "**limpeza_da_sujidade**", para realizar diferentes operações.
  2. **ir_voltar(x, ir_ou_voltar, robot, nivel, coor, win, aleatorio)**: Esta função controla o movimento do *robot* para a frente ou para trás. A função usa um loop para mover o *robot* de forma iterativa até atingir uma determinada posição. Ela chama a função "**andar_por_ai**" para mover o *robot* em direções específicas.
  3. **andar_por_ai(robot, x, y, nivel, coor, win, ir_ou_voltar, lista, volta, aleatorio)**: Essa função controla o movimento do *robot* em direções específicas. A função verifica a direção do movimento (ir ou voltar) e

chama outras funções, como "**detetar_obstaculos_e_sujidade**" e "**desviar_obstaculo**", para evitar obstáculos e realizar a limpeza.

Variáveis:

1. robot: Objeto do robot usado para controlar suas ações e movimentos.
2. nivel: Variável que representa o nível do robot.
3. win: Variável que representa a janela do robot.
4. coor: Lista de coordenadas.
5. aleatorio: Variável que indica se o movimento do robô é aleatório ou não.
6. x: Variável que representa a coordenada x do robot.
7. y: Variável que representa a coordenada y do robot.
8. lista: Lista de coordenadas de obstáculos.

- **Funções (módulo) para disposição esparsa dos objetos:**

1. **menu_nivel3_obstaculos(nivel)**: Esta função cria a interface gráfica do menu do nível 3 relativo à escolha da aquisição de coordenadas dos obstáculos: coordenadas aleatórias ou importadas de um documento .txt. Contempla uma função com botões já referida acima. A única variável é *nivel*, que apenas indica e transporta ao longo de funções o nível que o programa está a correr.

2. **gerar_numero_aleat(input_mesas, input_cadeiras, input_vasos,input_pianos,win):** Esta função é corrida quando o utilizador carrega no botão "Gerar". Ela gere o número de tipos de obstáculos aleatoriamente, com a restrição de que a soma destes é sempre igual a 5. Mostra esse resultado nas input boxes disponíveis na janela gráfica.

Variáveis:

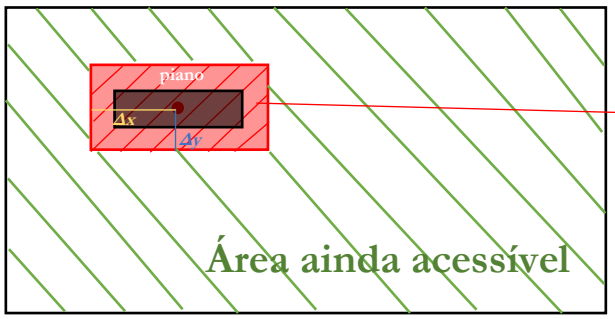
- **input_mesas, input_cadeiras, input_vasos,input_pianos:** representam o valor que está em cada input box que irá ser mudado para valores aleatórios quando é pressionado o botão "Gerar".
- Win – variável da janela gráfica.

3. **avaliar_numero_objetos(nr_mesas ,nr_cadeiras ,nr_vasos ,nr_pianos ,input_mesas , input_cadeiras , input_vasos, input_pianos ,win)**: Esta função avalia se a soma do número de tipos de obstáculos escolhidos pelo utilizador é igual a 5 (quando é escolhida a opção de gerar obstáculos aleatoriamente, na ação de escolher a quantidade de obstáculos). Se não for, o programa corre a função de mensagem_de_erro já especificada acima.

Variáveis:

- nr_mesas ,nr_cadeiras ,nr_vasos ,nr_pianos – Inputs pelo utilizador do número de obstáculos convertidos em inteiros. A sua soma tem que ser igual a 5!
- input_mesas , input_cadeiras , input_vasos, input_pianos – Inputs introduzidos pelo utilizador do número de obstáculos. São colocados todos a 0 caso a soma das variáveis acima seja diferente de 5.
- Win – variável da janela gráfica.

4. **coord_aleatorias**(lista_obstaculos,win): Esta função é a principal da distribuição esparsa dos objetos e gera, aleatoriamente, as coordenadas dos obstáculos. Em primeiro lugar, a função gera uma lista *coordenadas_totais* com todas as coordenadas (x,y) possíveis para colocar um obstáculo. O nosso algoritmo baseia-se no seguinte: começamos por escolher SEMPRE as coordenadas dos objetos maiores e de seguida os menores. Sempre que é gerada uma coordenada (escolhida aleatoriamente da lista *coordenadas_totais*) é retirada dessa mesma lista a periferia do coordenadas do obstáculo, de modo a que as coordenadas em redor do obstáculo gerado já não possam ser escolhidas para um novo obstáculo.



Sendo os pianos (obstáculos maiores) os primeiros obstáculos cujas coordenadas vão ser geradas, a sua periferia não vai ser ocupada pelos restantes 4 obstáculos, sejam eles de qualquer tipo. Depois de se escolherem as coordenadas dos pianos escolhem-se as das mesas, cadeiras e vasos, respetivamente.

NOTA: É sempre gerada a coordenada central do obstáculo

Variáveis:

- **lista_obstaculos**: uma lista do tipo [nr_pianos , nr_mesas, nr_cadeiras, nr_vasos,] que a função recebe de modo a saber que os tipos de objetos a serem gerados.
- **coordenadas_obstaculos**: No fim de todas as coordenadas terem sido atribuídas aos cinco obstáculos, esta função retorna a lista **coordenadas_obstaculos** do tipo [[coordenadas_mesas], [coordenadas_cadeiras], [coordenadas_pianos], [coordenadas_vasos]] (obviamente, sempre com comprimento 4).
- **Win**: variável da janela gráfica.

5. **periferia_obstáculo_x(i,win)**: esta função armazena a periferia de coordenadas a serem eliminadas de acordo com cada tipo de obstáculo. A necessidade da criação desta função advém de o formato e tamanho de um piano ser completamente diferente do formato de um vaso.
   Retorna um ***Δx*** (representado acima) que define a (em termos de coordenada *x*) a região que tem que ser eliminada, de modo a não ocorrer interseção.

Variáveis:

- **i:** tipo de objeto – inteiro cujos únicos valores são 0 (piano), 1 (mesa), 2 (cadeira) e 3 (vaso).
- **Win**: variável da janela gráfica.

6. **periferia_obstáculo_y(i,win)**: Esta função faz exatamente o mesmo (e apresenta as mesmas variáveis) da função de cima, com a exceção de que agora retorna a variação em *y* que tem que ser eliminada – retorna a um ***Δy*** (representado acima).

7. **desenhar_objetos_aleatorios(coordenadas_obstaculos,win)**: Esta função é responsável por desenhar os objetos gerados aleatoriamente. A sua variável **coordenadas_obstaculos** é uma lista que tem sempre o comprimento de 4 do tipo [[coordenadas_mesas], [coordenadas_cadeiras], [coordenadas_pianos], [coordenadas_vasos]]. Mais uma vez, a variável win é a correspondente à janela gráfica.

8. **guardar_dados(coord_mesa, coord_cadeira, coord_vaso, coord_piano):** Esta função guarda, no formato habitual, as coordenadas dos cinco obstáculos. A razão da criação desta função é a uniformidade do código, visto que em todos os níveis optámos por importar de arquivos as coordenadas dos obstáculos.

Variáveis:

- **coord_mesa, coord_cadeira, coord_vaso, coord_piano:** Estas variáveis representam, como o nome indica, as coordenadas d ecada tipo de obstáculo.

1.3. Módulo Nome do Módulo

Tendo consultado o regente da cadeira, foi-nos consentido colocar a descrição dos diferentes módulos/" blocos" do programa no ponto **1.2**.. Desta forma, fica mais claro a explicação do código.