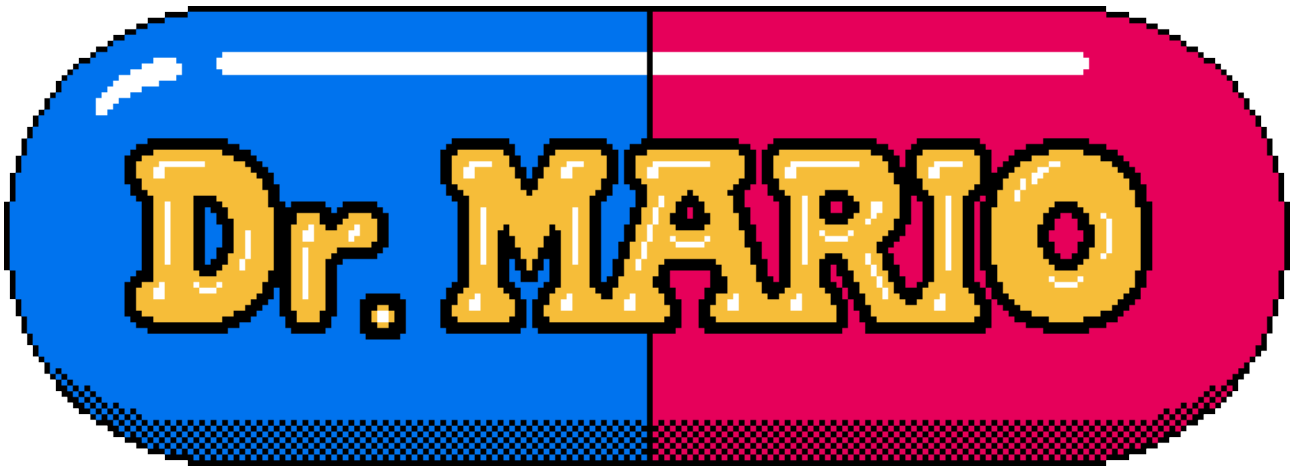


Laboratório de computadores 2019/2020

Turma 2 Grupo 7

João Diogo Martins Romão: up201806779
Romeu Gonçalves: up201806287

1. Instruções de utilização do programa

1.1. Menu Pricipal

Quando o programa é iniciado é apresentado no ecrã o menu pricipal do jogo. O utilizador pode optar por utilizar as setas to teclado e a tecla ENTER ou o rato e o botão do lado esquerdo para selecionar a opção pretendida.

Singleplayer – redireciona o utilizador para o menu de escolha do nível do jogo em modo singleplayer.

Multiplayer – redireciona o utilizador para o menu de escolha do nível do jogo em modo multiplayer.

Serialport – reencaminha o utilizador para um submenu onde permanecerá até a conexão com outro dispositivo seja estabelecida.

Ícone no canto inferior esquerdo – caso o utilizador clique sobre ele com o botão esquerdo do rato o programa terminará.

Ícone no canto inferior direito – caso o utilizador clique sobre ele com o botão esquerdo do rato o programa apresentará o menu de configurações.

1.2. Menus Secundários (escolha de nível)

Uma vez selecionada a opção SINGLEPLAYER ou MULTIPLAYER no menu principal o utilizador será redirecionado para um menu secundário onde poderá escolher o nível no qual pretende jogar.

Neste menu o utilizador poderá ainda receber informações sobre como jogar ao colocar o cursor do rato sobre o ícone que se encontra no canto inferior direito do ecrã, aparecendo no ecrã a seguinte informação:

1.3. Menu de Configurações

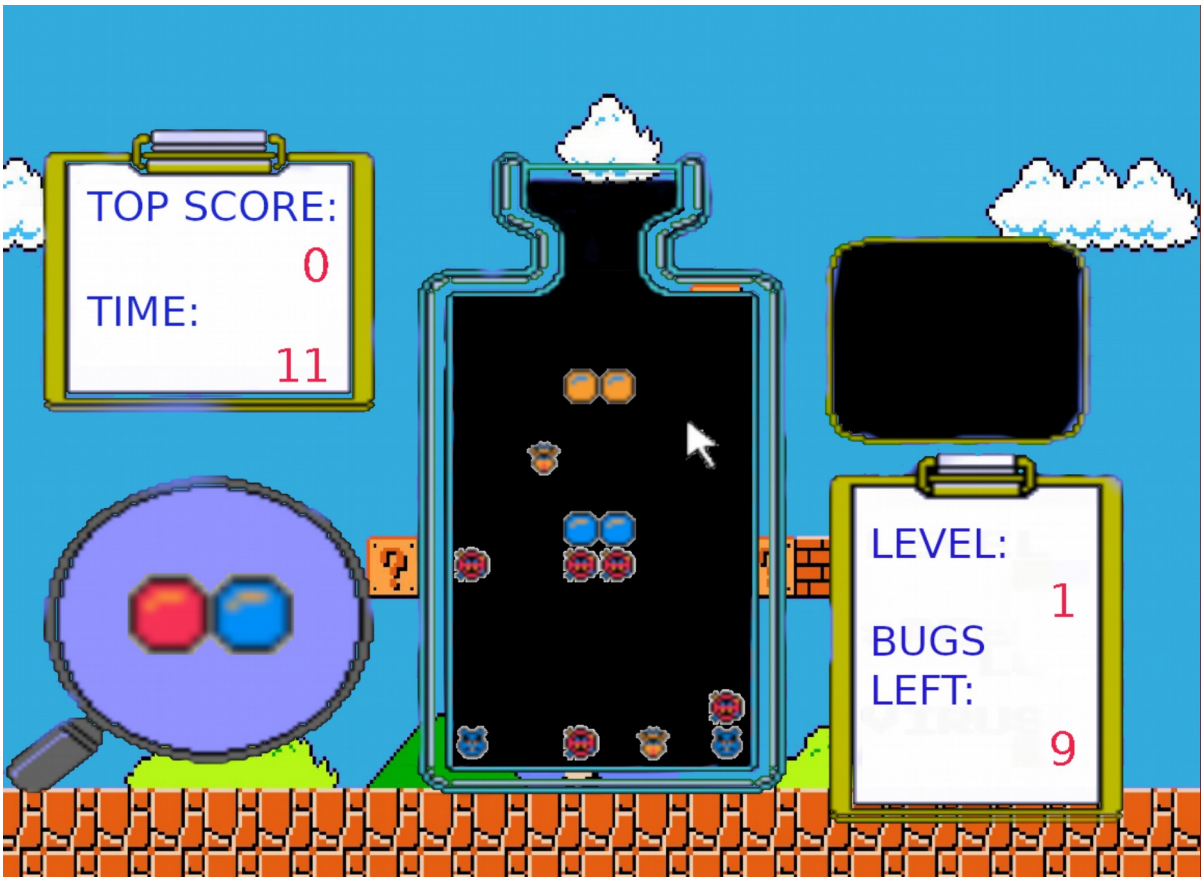
A manipulação deste menu é realizada de forma semelhante ao anterior (com recurso ao rato e ao teclado).

Neste menu o utilizador pode consultar a hora do dia e caso pretenda alterá-la, isto porque o background do jogo é desenhado consuante a hora do dia, possuindo cores mais claras durante o dia e cores mais escuras durante a noite.

1.3.1. Alterção do tempo de jogo

O utilizador poderá numa primeira fase, com recurso ao teclado, alterar a hora do dia. De seguida, após ser premida a tecla ENTER o utilizador poderá selecionar os minutos em questão. O utilizador é de seguida automaticamente direcionado para o menu principal.

1.4. Modo Singleplayer



Neste modo o utilizador tem modo objetivo eliminar todos os “bugs” (peças de forma diferente que mudam de forma de meio em meio segundo) presentes no mapa de jogo. Poderá fazê-lo ao alinhar quatro peças da mesma cor quer na direção horizontal quer vertical. Uma vez havendo quatro peças decor igual alinhadas elas desaparecerão do mapa de jogo. O objetivo é eliminar todos os bugs no menor tempo possível. O utilizador pode saber qual a peça que virá a seguir se observar a lupa que se encontra no canto inferior esquerdo do ecrã. O utilizador perde quando já não conseguir colocar qualquer peça no ecrã.

1.4.1. Movimentação da peça de jogo

A **peça de jogo desloca-se verticalmente** com velociade constante, que vai aumentando ligeiramente com o decorrer do jogo. A peça termina o movimento quando colidir com outra.

Para **deslocar a peça horizontalmente** o utilizador pode fazê-lo com recuro às setas do teclado, sendo que a seta da esquerda desloca a peça uma unidade para a esquerda e a seta da direita uma unidade para a direita.

Para **rodar a peça** o utilizador tem que premir a seta para cima do teclado. São apenas permitidas quatro rotações: da direita para baixo, de baixo para a esquerda, da esquerda para cima e de cima para a direita.

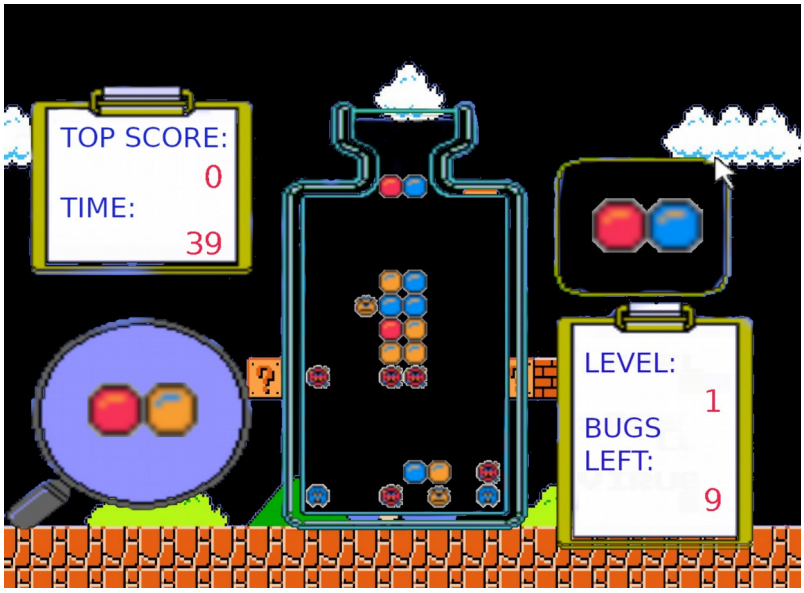
Para **acelerar o movimento de queda** da peça o utilizador terá que realizar um movimento descendente premindo o botão esquerdo do rato, largando-o no final. A peça acelerará mais quanto maior for o movimento realizado com o rato.

1.4.2. Stack

A stack encontra-se inicialmente vazia e é representada pela forma quadrática preta que se encontra na parte superior do lado direito da imagem.

Caso o utilizador não pretenda jogar a peça nesta jogada poderá guardá-la na stack realizando um movimento para o lado direito com o rato, enquanto prime o botão esquerdo do dipositivo. O movimento é concluido quando o utilizador larga o botão.

Caso a stack já possua alguma peça, a peça atual irá substituí-la, tomando a peça que se encontrava na tack a posição de jogo.



(imagem contém uma peça na posição da stack)

1.4.3. Fim de jogo

Como já foi referido anteriormente o utilizador ganha o jogo quando já não houver mais “bugs” no mapa de jogo e perde quando já não for possível inserir qualquer peça no mapa de jogo. Dependendo da forma como o jogo é terminado o utilizado é reencaminhado para diferentes fases do programa (as quais podem ser observadas nas imagens em seguida). Uma vez neste estado, ao premir a tecla ENTER o utilizador regressa ao menu principal.

1.4.4. Atualização do melhor tempo

Uma vez concluido o jogo, caso o tempo obtido seja inferior ao anteriormente registado ou ainda não tenha sido registado qualquer tempo a secção TOP SCORE do nível jogado é atualizada com este novo valor:

1.5. Modo Multiplayer

A utilização do modo multiplayer é semelhante ao singleplayer, contudo não possui movimentos com o rato e também não possui uma stack.

1.5.1. Intruções para o utilizador 1

O utilizador 1 (que joga do lado esquerdo do ecrã) poderá movimentar a peça de jogo utilizando para tal as teclas AWSD.

Para **deslocar a peça horizontalmente** para a esquerda terá que usar a tecla A e para a direita terá que utilizar a tecla D.

Para **rodar a peça** terá que utilizar a tecla W.

Para **acelerar o movimento de queda vertical** terá que utilizar a tecla S.

1.5.2. Intruções para o utilizador 2

O utilizador 1 (que joga do lado esquerdo do ecrã) poderá movimentar a peça de jogo utilizando para tal as setas do teclado

Para **deslocar a peça horizontalmente** para a esquerda terá que usar a seta para o lado esquerdo e para a direita terá que utilizar a seta para o lado direito.

Para **rodar a peça** terá que utilizar a seta para cima.

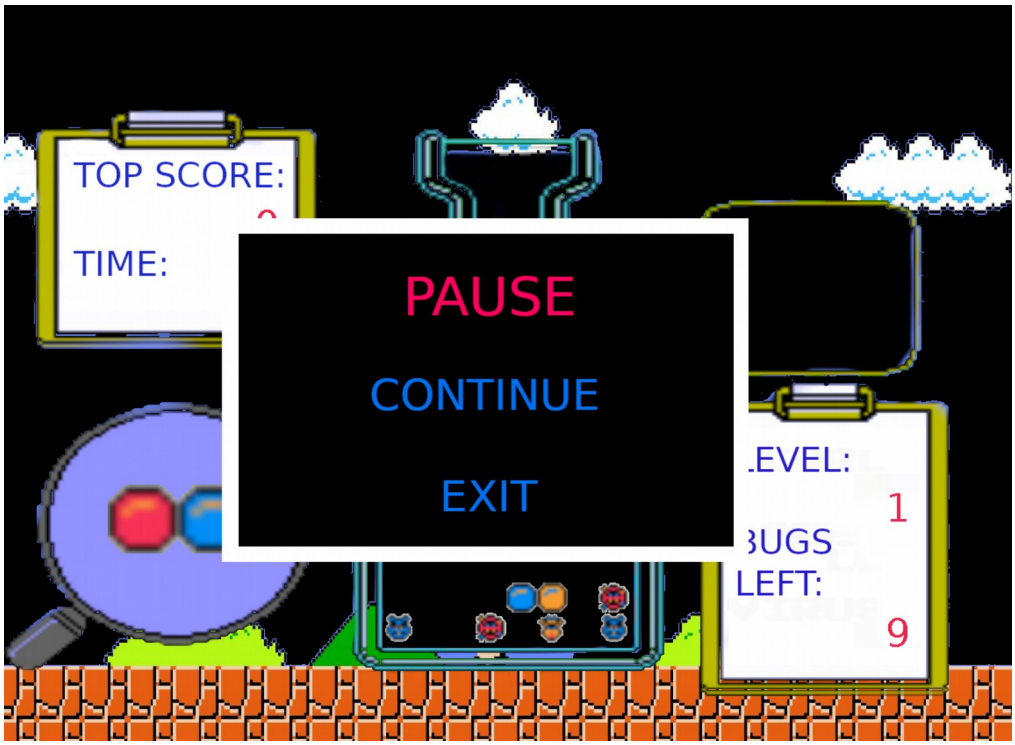
Para **acelerar o movimento de queda vertical** terá que utilizar a seta para baixo.

1.5.3. Fim de jogo

O jogo em modo multiplayer termina quando um dos jogadores já não apresentar “bugs” no seu mapa de jogo ou quando um dos jogadores já não puder colocar qualquer peça no mapa de jogo. Deste modo um dos jogadores perde e outro ganha não sendo possível outro caso.

1.6. Menu pausa

Durante qualque modo de jogo caso o utilizador pressione a tecla ESC será apresentado o menu pausa. O utilizador terá depois a opção de continuar ou sair do jogo.



2. Estado do projeto

2.1. Dispositivos utilizados

Dispositivo	Utilização	Interrupções
Timer	Permite uma atualização contínua do estado do jogo.	Sim
Teclado	Permite a seleção de menus e execução de comandos de jogo.	Sim
Rato	Permite a seleção de menus e execução de comandos de jogo.	Sim
Placa gráfica	Permite a visualização de toda a informação relativa ao estado em que o jogo se encontra	Não
RTC	Permite guardar pontuações que o utilizador obteve no modo singleplayer. Permite também a existência de um modo que adapta a cor do background do jogo à hora do dia.	Sim
Porta série	Tentámos implementar a porta série para não limitármos o modo multiplayer a um só jogador, subscrevendo as suas interrupções, mas não conseguimos estabelecer uma conexão entre duas máquinas.	Sim

2.2. Descrição da funcionalidade de cada dispositivo

2.2.1. Timer

O timer foi utilizado para controlar a execução do programa. Foram subscritas as suas interrupções (timer_subscribe_int() - ficheiro proj.c). Cada vez que se recebe uma notificação de interrupção a varável counter (int) é incrementada, através da função timer_int_handler() (ficheiro proj.c), sendo o programa atualizado sempre que há a notificação de interrupção.

O estado da variável counter vai permitindo a execução de várias ações ao longo do programa como por exemplo o movimento de queda de uma peça (update_array_horizontal / update array vertical – ficheiro singleplayer.c).

2.2.2. Teclado

O teclado foi utilizado quer para a navegação de menus (utilização das setas do teclado juntamente com a tecla ENTER), quer para a execução de comandos de jogo. Destaca-se para tal a função key_handler() do ficheiro display.c que interpreta a informação resultante das notificações de interrupção do dispositivo consoante o estado em que o programa se encontra. Depois desse reconhecimento são chamadas funções secundárias. Destas últimas para a seleção de menus destacam-se key_main_menu() (seleção de opções dentro do menu principal), key_level_menu_s() (seleção de nível para o modo singleplayer), key_level_menu_m() (seleçaõ de nível para o modo multiplayer). Quanto à execução de comandos de jogo destacam-se key_playing(), que permite o movimento horizontal, pill_rotation(), permite a rotação de uma peça e accelerate_mov(), permite a aceleração do movimento de queda durante o modo multiplayer. Quanto à inserção de texto destaca-se ask_for_hours() do ficheiro settings.c que pede ao utilizador um número inteiro.

2.2.3. Rato

Tal como o teclado também opera na seleção de menus e na execução de comandos de jogo.

Destaca-se a função mouse_handler(), que sucessivamente incrmenta as coordenadas da posição do rato e intrepreta os cliques dos botões do lado esquerdo e direito consoante o estado do programa. Para a seleção de menus foi utilizado o botão esquerdo.

Utilizadas durante o modo singleplayer as funções mouse_accelerate() e mouse_send_to_stack() reconhecem gestos realizados com o rato intrepretando-os. A primeira reconhece um movimento descendente enquanto se clica no botão esquerdo, permitindo a aceleração do movimento de queda de uma peça. A segunda permite o reconhecimento de um movimento para o lado direito enquanto o botão direito é premido e permite que a peça de jogo seja enviada para a stack.

2.2.4. Placa gráfica

A placa gráfica foi inicializada com o modo 0x14c, o qual possui uma resolução de 1152x864, um modo de cor direto e consegue representar 16 777 216 cores. Tirámos vantagem da técnica de “double buffering”, sendo que numa fase inicial toda a informação é enviada para um “buffer secundário” sendo depois copiada para a VRAM com o auxílio da função memcpy(). O programa apresenta funções de movimento para a peça que está em jogo, que se vai movendo ao longo do tempo (update_array_horizontal / update array vertical – ficheiro singleplayer.c). As colisões entre as peças não foram geradas com o auxílio da placa gráfica, mas sim com um array bidimensional de carateres. Caso hajam dois carateres seguidos na direção vertical do array estamos perante uma colisão.

2.2.5. RTC

O RTC foi utilizado a fim de obter a hora do dia podendo assim criar um modo de noite de faz com que a cor azul do background seja mais clara ou mais escura. Para tal utilizámos as interrupções periódicas do dispositivo.

Também foram utilizados os seus registos para guardar informação sobre as melhores pontuações

3. Estrutura e organização de código

3.1. Módulo Timer

Contém o código desenvolvido no lab2, tendo sido utilizadas as funções timer_subscribe_int(); timer_unsubscribe_int() e time_int_handler().

Tem como principal função permitir e controlar a atualização do jogo.

Peso no projeto: 5%
João Romão: 50%
Romeu Gonçalves: 50%

3.2. Módulo Keyboard

Contém o código desenvolvido no lab3

Peso no projeto: 5 %
João Romão: 50%
Romeu Gonçalves: 50%

3.3. Módulo Mouse

Contém o código desenvolvido no lab4

Peso no projeto: 5%
João Romão: 50%
Romeu Gonçalves: 50%

3.4. Módulo Graphics

Contém o código desencolvido no lab5, tendo este sido adaptado ao modo de vídeo utilizado neste projeto. A função set_vbe_mode(), a qual não tinha sido corretamente implementada no lab5 foi agora corrigida. A função vg_color_pixel() foi adaptada para um modo gráfico com mais do que um byte por pixel e para a técnica de “double buffering” que foi utilizada ao longo do projeto.

Peso no projeto: 5%
João Romão: 100%

3.5. Módulo RTC

Contém código utilizado para subscrição e manipulamento das interrupções do RTC.

handle_update_int() - atualiza os valores das horas, minutos e segundos guardados em variáveis quando é recebida uma notificação de interrupção.

update_record() - recebe um valor candidato a novo recorde, verifica se esse valor é de facto o novo recorde e atualiza esse valor.

set_hour() / set_minutes() - atualiza os valores das horas e minutos fornecidos pelo utilizador no menu de configurações.

get_hour() / get_minutes() / get_seconds() - obtém os valores atuais para horas minutos e segindos.

draw_background_color() - Desenha como background um tom de azul (que representa o céu) que é mais escuro ou claro consuante a hora do dia.

Peso no projeto: 5%
João Romão: 100%

3.6. Módulo Serial port

Não foi correntamente implementado, uma vez que não conseguimos estabelecer uma conexão entre duas máquinas.

3.7. Módulo Proj

Módulo que contém o loop de driver_receive e a função main(). É de destacar que é neste módulo que estão definidas as enumerações que detetam o estado em que o programa se encontra e vão sendo atulizadas com o decorrer do tempo.

- enum display menu = MAIN_MENU; → deteta o estado em que o menu principal se encontra sendo mostrado de acordo no ecrã.
- enum display_single single_menu = SINGLE_MENU; → deteta o estado do menu de seleção de níveis no modo singleplayer.
- enum display_multi multi_menu = MULTI_MENU; →deteta o estado do menu de seleção de níveis no modo multiplayer.
- enum state_of_program state = IN_MAIN_MENU; → deteta o estado em que o jogo se encontra (Menus, Singleplayer, Multiplayer, Settings, Pausa…)

Peso no projeto: 5%
João Romão: 100%

3.8. Módulo Settings

Módulo em que permite ao utilizador a visualização e alteração da hora de jogo.

ask_for_hours() - pede ao utilizador um novo número para as horas e minutos. Caso o número seja superior a 23 ou 59 respetivamente, o programa assume esses valores.

draw_clock() - apresenta o relógio ao utilizador, com os valores fornecidos pelo RTC.

Peso no projeto: 8%
João Romão: 100%

3.9. Módulo Display

Este módulo é responsável por mostar no ecrã informação relativa à seleção de menus e interpreta os inputs do utilizador quer do rato (através da função mouse_handler()) quer do teclado (key_handler()). (A funcionalidade destas funções já foi descrita anteriormente).

Este módulo possui também a função **mouse_cursor()** que desenha o cursor do rato acompanhado background de onde o jogo se encontra. Uma vez que o cursor do rato se encontra presente em quase todo o trabalho esta função revelou-se bastante importante, sendo mesmo responsável por chamar as funções **draw_game_board()** e **throw_pill()** do módulo Singleplayer e que permitem o denrolar do jogo e serão descritas mais à frente.

Outra função de elevada importância é a **load_images()** responsável por carregar todos os XPMs no início do programa.

Peso no projeto: 20%
João Romão: 100%

3.10 Módulo Multiplayer

Este módulo é responsável pelo jogo em modo multiplayer. Algumas funções utilizadas encontram-se definidas neste módulo as restantes são reutilizadas do módulo singleplayer.

display_initial_countdown_multi() - responsável por fazer a contagem regressiva que precede o início do jogo. É também responsável por marcar o início do jogo colocando a peça de jogo na posição inicial do mapa de jogo do nível em questão.

throw_pill1() e **throw_pill2() –** são responsáveis por colocar uma peça de jogo na posição inicial quando o movimento de queda da peça anterior chegou ao fim (houve deteção de colisão). São também reponsáveis por mostrar qual a peça que entrará em jogo quando a atual concluir o seu movimento de queda.

check_ending_conditions_1() e **check_ending_condtions_2()** - verificam se jogo chegou ao fim para algum dos jogadores, acabando o jogo de imediato para ambos os jogadores, com posterior indicação de qual dos jogadores ganhou o jogo.

draw_numerical_info_multi() - responsável por representar toda a informação numérica no modo multiplayer.

reset_multi_player() - coloca o jogo de volta ao estado inicial, reiniciando os mapas de jogo e as retantes variáveis de verificação alteradas com o decorrer do jogo do modo multiplayer.

Peso no projeto: 15%
João Romão: 100%

3.11.Módulo Singleplayer

Este módulo é responsável pelo jogo em modo singleplayer, sendo o principal módulo do programa.

display_initial_countdown() - responsável por fazer a contagem regressiva que precede o início do jogo. É também responsável por marcar o início do jogo colocando a peça de jogo na posição inicial do mapa de jogo do nível em questão.

throw_pill() – é responsável por colocar uma peça de jogo na posição inicial quando o movimento de queda da peça anterior chegou ao fim (houve deteção de colisão).
É também reponsável por mostrar qual a peça que entrará em jogo quando a atual concluir o seu movimento de queda.

pill_movement() - resposável por mover a peça de jogo verticalmente ao longo do tempo, atualizando o mapa de jogo.

collision_detector() - deteta se houve colisão no movimento de queda da peça de jogo (há colisão quando dois carateres que representam peças se encontram em posições adjacentes no array bidimensional que representa o mapa de jogo).

check_vertical() e **check_horizontal() -** verifica se há quatro peças alinhadas vertical e horizontalmente no array bidimensional que representa o mapa de jogo, caso haja essas peças são apagadas.

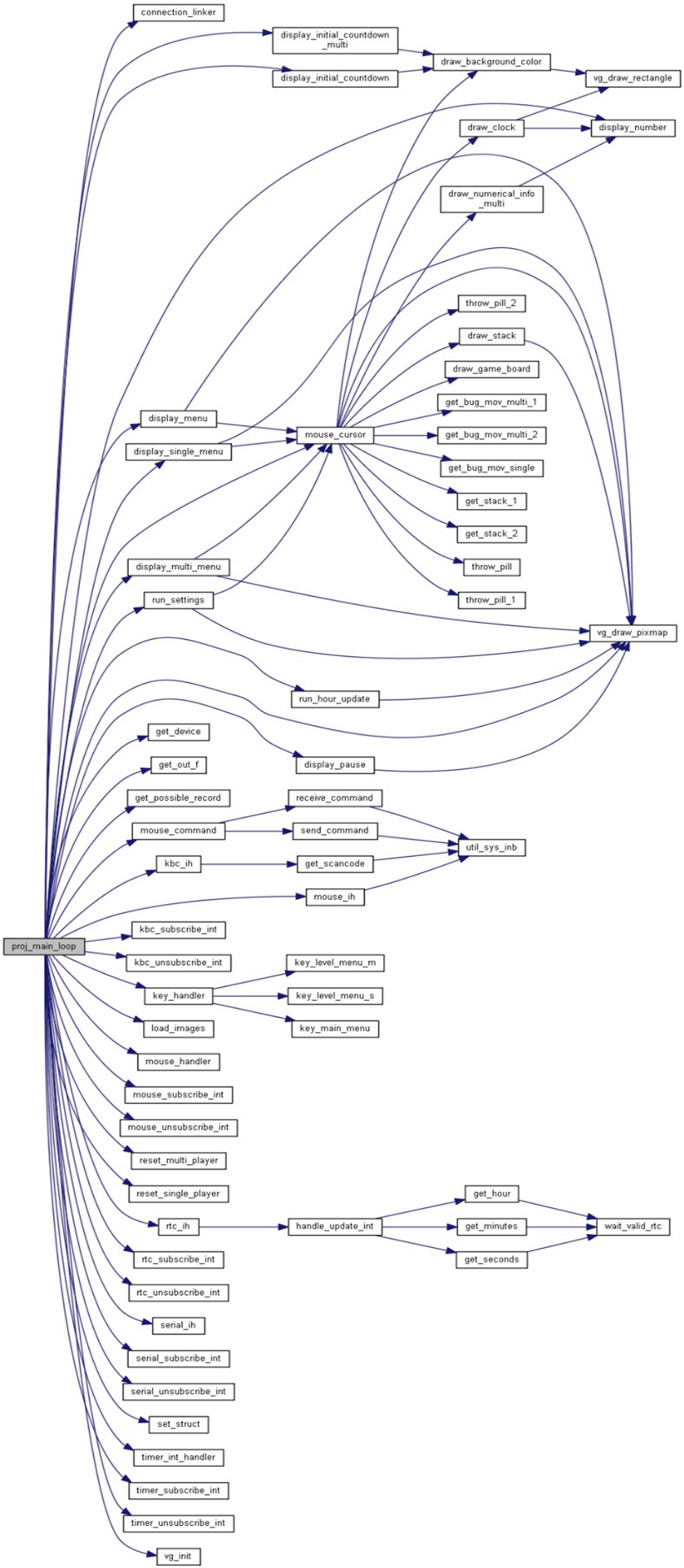
timed_update_lost_pills() - verifica no mapa de jogo, se existe alguma peça isolada. Caso exista o program cria um efeito de gravidade no qual a peça vai caindo até se encontrar junto de outras peças.

draw_numerical_info() - responsável por representar toda a informação numérica no modo singleplayer.

reset_single_player() - coloca o jogo de volta ao estado inicial, reiniciando os mapas de jogo e as retantes variáveis de verificação alteradas com o decorrer do jogo do modo singleplayer.

(as funções mouse_accelerate, mouse_send_to_stack, key_playing e key_rotation que pertencem a este módulo e revelam elevada importância, não foram aqui explicadas uma vez que já o foram anteriormente).

3.12. Function call graph



4. Detalhes da implementação

Dois aspetos fundamentais para a implementação do projeto foi definir como poderíamos fazer a transição entre menus e como poderíamos representar o mapa de jogo no ecrã.

Relativamente à transição de menus optámos por várias variáveis do tipo enumeração que representariam os estados e os subestados em que o programa se encontra. A informação é então disponibilizada ao utilizador com base no estado dessas enumerações.

Relativamente aos mapas de jogos achámos que a solução mais eficaz seria representá-los através de arraus bidimensionais de caratéres. Cada peça possui um caratér identificativo e os espaços em branco são representados por caratéres ‘0’. Esta aproxmação ao problema acabou por se revelar bastante vantajosa, pois permitiu tratar de problemas como colisões e rotações apenas alterando os caratéres presentes no mapa de jogo.

A maior dificuldade encontrada no projeto foi de facto a implementação da porta série. Através da subscrição das interrupções do dispositivo ainda conseguimos fazer com que uma máquina conseguisse enviar informação a outra, mas não as duas em simultâneo. Penso que para uma boa implementação do dispositivo teria sido fundamental uma abordação mais profunda a nível prático.

Por outro lado a implementação do RTC tornou-se bastante intuitiva com base nos conhecimentos adquiridos com os outros dispositivos.

5. Conclusões

LCOM revelou-se uma cadeira bastante trabalhosa e que acaba por aparecer no nosso percuso académico um pouco de paraquedas. Sendo uma cadeira de segundo penso que aborda temas de complexidade relativamente elevada para uma audiência onde grande parte está apenas no segundo ano de contacto com o mundo da programação.

Por esta razão a princípio encontrámo-nos um pouco incapazes, especialmente na primeira aula prática onde praticamente não foi desenvolvido nenhum código. Depois de perceber o mecanismo base da cadeirra de interação com os periféricos, a cadeira tornou-se bastante mais acessível. Deste modo penso que LCOM beneficiaria e muito de uma introdução mais aprofundada.

Outro ponto negativo na cadeira foram os lab handouts e os slides que são algo complexos e não têm a melhor legibilidade, algo que reforça uma leitura atenta e repetitiva, aumentando ainda mais o tempo dedicado à cadeira.

Contudo, acho que LCOM abriu horizontes relativamente à forma de como o computador interage com os periféricos, conhecimento que até à data era praticamente nulo.