Chess Game

Turma 7 – Grupo 3

João Filipe Oliveira Ramos up202108743

Matilde Isabel da Silva Simões up202108782

Miguel Diogo Andrade Rocha up202108720

Miguel José Pinho Freitas de Carvalho Pedrosa up202108809

Licenciatura em Engenharia Informática e Computação

Laboratório de Computadores

2022/2023

Índice

Introdução

O nosso projeto "Chess Game" é uma adaptação digital do xadrez tradicional, destacado por uma interface interativa que realça as possíveis jogadas ao selecionar uma peça.

Uma vez iniciado, o jogo segue as regras estabelecidas do xadrez. Se a jogada for inválida, o jogo volta ao estado anterior e o jogador terá que tentar outro movimento. Além disso, o jogo é limitado por tempo, adicionando desafio à partida. O jogo termina quando um jogador coloca o rei do adversário em xeque-mate, quando o tempo de um dos jogadores termina ou por empate, por afogamento.

Assim, este jogo proporciona aos jogadores uma experiência de jogo de xadrez tradicional utilizando uma plataforma digital, mantendo a essência e a estratégia do jogo original.

Instruções de utilização

O jogo começa com um menu inicial que contém o nome e dois botões, um que permite continuar para jogar e o outro para sair. Contém, também, o dia e a hora atual em que o jogo foi inicializado.

Figura 1. Menu inicial

Se o jogador clicar com o rato no botão esquerdo ou na tecla Enter, vai encontrar a secção onde é lhe pedido o nome. Tem a possibilidade de apagar as letras, em caso de engano.

Figura 2. Menu para escrever o nome

Ao clicar na tecla Enter, o jogador é direcionado para o jogo propriamente dito e o seu tempo começa imediatamente a decrescer. Com a utilização do mouse, o jogador seleciona uma peça e arrasta-a para o quadrado que pertender. Se o movimento for válido, a peça é colocada no quadrado, caso contrário, a peça volta ao seu quadrado anterior e o movimento é anulado.

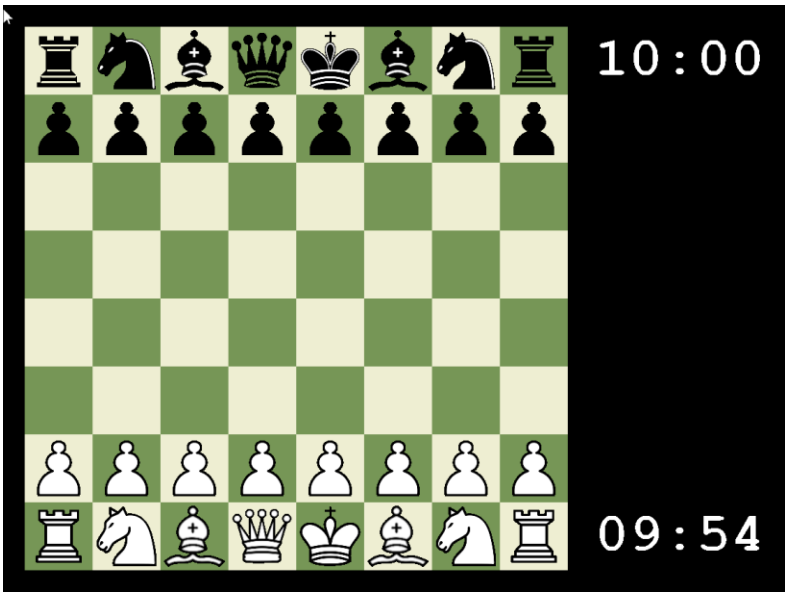


Figura 3. Tabuleiro inicial

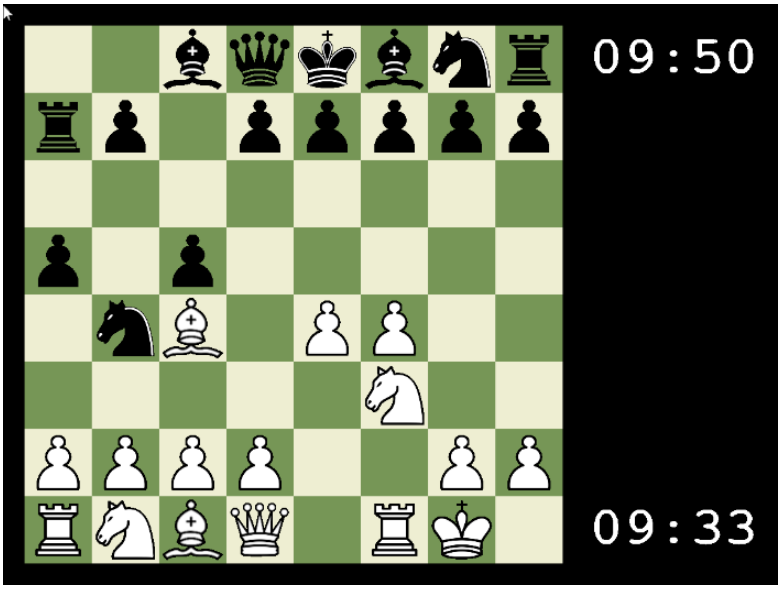


Figura 4. Castle

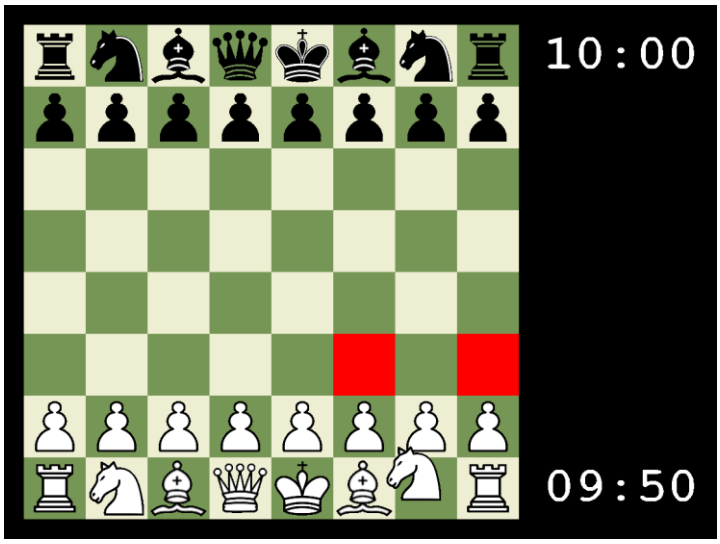
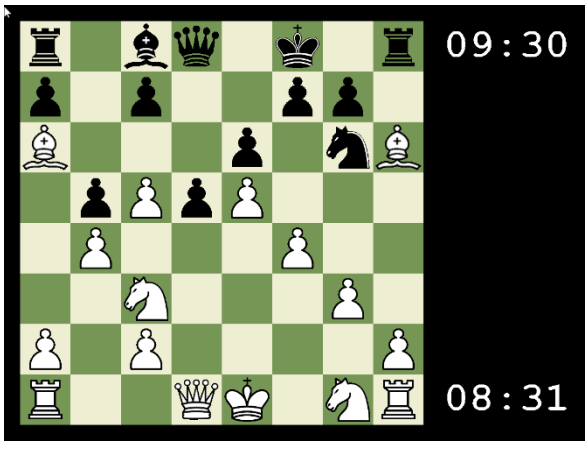


Figura 5. Movimento

Figura 6. Promotion



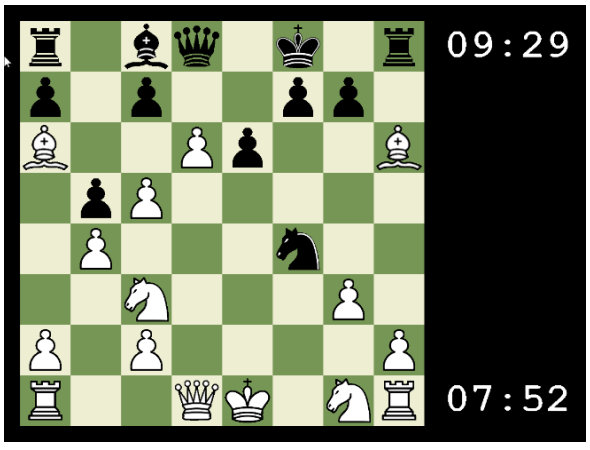


Figura 8 e 9. En passant

No jogo, as peças são movíveis com o rato e largam-se nas posições desejadas.

A partida termina em determinadas situações: quando o tempo do jogador ou do bot se esgota, quando ocorre um check-mate ou um stalemate. Nesse momento, o jogador tem acesso ao game over menu, onde pode visualizar a razão pelo qual o jogo acabou e se ganhou ou perdeu. Além disso, o menu apresenta dois botões que permitem ao jogador escolher entre jogar novamente ou sair do jogo.

Figura 8. Checkmate

Figura 9. Stalemate

Figura 10. Time Out

Dispositivos

Timer

Peso relativo – 15%
Baseado no lab 2 das aulas práticas (implementado por todos)

O timer tem a função de limitar o tempo de cada jogo e gerir a frame-rate da placa gráfica, realizando atualizações no ecrã 60 vezes por segundo. Todas as funcionalidades relacionadas com o timer são implementadas no ficheiro "timer.c".

Keyboard

Peso relativo – 5%
Baseado no lab 3 das aulas práticas (implementado por todos)

O keyboard é usado para permitir a inserção do nome do jogador, que será usado para apresentar o resultado no final do jogo. Além disso, o teclado também é utilizado para navegação pelas diversas secções. Por exemplo, é utilizado para sair do estado de solicitação do nome e entrar no jogo propriamente dito. Da mesma forma, o teclado é utilizado para retornar ao menu principal em qualquer estado do jogo, onde se tem a opção de sair completamente do jogo ("exit").Todas as funcionalidades relacionadas com o teclado são implementadas no ficheiro "keyboard.c".

Mouse

Peso relativo – 40%
Baseado no lab 4 das aulas práticas (implementado por todos)

O mouse desempenha um papel fundamental no menu principal, onde são apresentadas as opções de "Jogar" para entrar no jogo e "Sair" para encerrá-lo. Além disso, o mouse é utilizado em todas as funcionalidades e movimentos do jogo, incluindo o menu de "Game Over". Permite ao jogador interagir com elementos na tela, como clicar em botões, arrastar as peças e executar os movimentos dentro do jogo. Todas as funcionalidades relacionadas com o mouse são implementadas no ficheiro "mouse.c".

Real Time Clock (RTC)

Peso relativo – 10%
Implementado por todos

O RTC é utilizado para dispor a data e hora atual de cada partida no menu inicial. Todas as funcionalidades relacionadas ao RTC são implementadas no ficheiro "rtc.c".

Video Graphics

Peso relativo – 30%
Baseado no lab 5 das aulas práticas (implementado por todos)

A placa gráfica é responsável por criar uma interface gráfica para exibir as diferentes secções que foram criadas no projeto do jogo, com as quais o utilizador irá interagir. Todo o projeto foi desenvolvido no modo de vídeo, onde os caracteres e elementos visuais são exibidos usando XPMs que foram gerados pelo software GIMP, a partir de imagens retiradas da internet e criadas no próprio software. No projeto, o modo de vídeo utilizado é o 0x14C, que, entre os modos disponíveis, é o modo com maior resolução dentro dos modos de cor direta, permitindo criar uma interface maior e mais detalhada. Todas as funcionalidades relacionadas com a placa gráfica são implementadas no ficheiro "gpu.c".

Double Buffering

Para renderização do jogo no ecrã, estamos a usar a técnica de double buffering. Ediretamente na *vram* poderia causar dessincronisações de frames caso este processo fosse interrompido por um interrupt do timer. Tendo isto em conta, implementámos a utilização da placa de vídeo com double buffering, pelo que cada frame é preparado num buffer idêntico à placa de vídeo, e apenas quando este está completo, é que é copiado na sua totalidade para a placa de vídeo, evitando assim problemas de dessicronização na renderização do jogo.

Animatied Sprites

Utilizámos animated sprites para o rato e para mover peças no teclado. Através de imagens prédfinidas, sprites, conseguiamos recriar os objetos que pretendemos na interface gráfica. A cada frame, este objetos movíveis no ecrã, são apagados das posiçoões anteriores e renderizados nas sua novas posições, criando uma ideia de movimento. De forma a apagar o estes objetos sem perder a imagem de fundo, optámos por criar um terceiro buffer chamado *screenshot*. Quando entrando numa nova página ou estado de jogo, o fundo é desenhado no

buffer e depois copiado para o screenshot. Assim, quando temos que apagar um objeto do buffer, podemos ir ao screenshot buscar a informação de fundo que ficou copiada e guardada, sem assim termos a necessidade de rescrever o buffer inteiramente e melhorando a performance do programa.

Project Status

Device	What for	Interrupts
Timer	controlar frame rate/ tempo de jogo	yes
Keyboard	Navegação no menu, inserir o nome do jogador	yes
Mouse	Interações com o menu e movimentos das peças no jogo	yes
Video Card	Sprites e as várias secções do jogo	yes
RTC	Disposição da hora e data atual	yes

Organização e estrutura do código

main.c

No ficheiro *main.c*, corre o game loop. Antes deste é feita toda a inicialização do programa, isto inclui: carregar os sprites do jogo para memória, definir a frequência do timer, ativar o *mouse data reporting*, subscrever os interrupts do *timer*, *keyboard*, *mouse* e *rtc*, inicializar o modo de video e mapear a memória *vram* em *user space* e incializar o estado do jogo. Durante o main loop, para cada interrupt dos dispositivos é chamada a função correspondente, dentro da máquina de estados. Após sair do main loop, é libertada toda a memória ainda alocada, são desativadas todas as subcrições dos interrupts dos dispositivos e volta-se a colocar o video em modo de texto.

state.c

No ficheiro *state.c*, está definida a máquina de estados. Existem 5 estados: MENU, WRITE_NAME, GAME, GAME_OVER e EXIT. Neste existem funções para cada uma das interrupções dos dispositivos. Dentro de cada uma destas funções, são chamadas as funções correspondentes de acordo com o estado, para inputs do teclado ou rato, ou para atualizar o ecrã, sendo esta atualização feita uma vez por tick do timer, ou seja, 60 vezes por segundo.

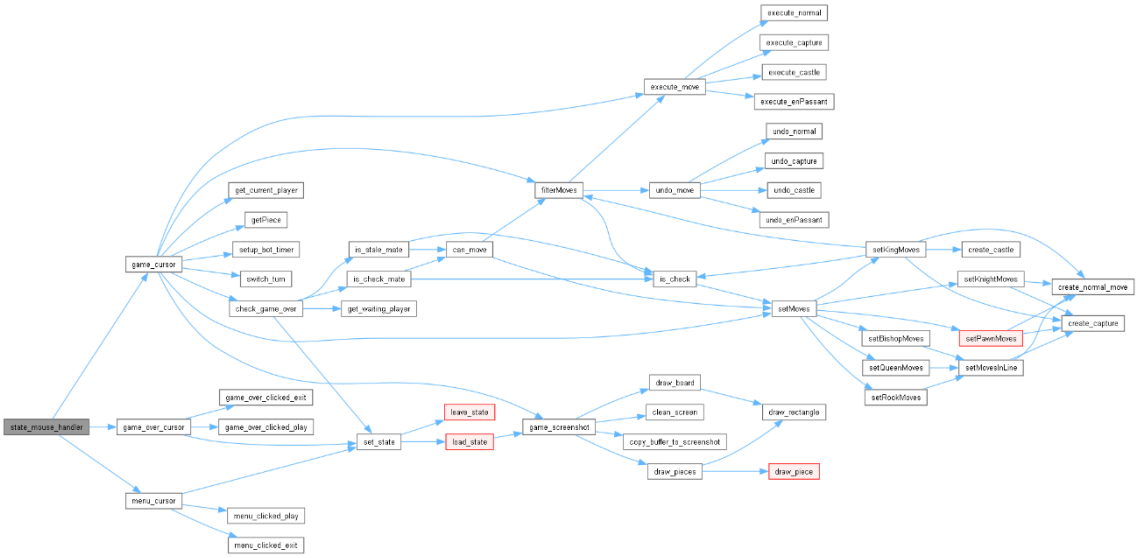


Figura 11. State Mouse Handling

Figura 12. State Keyboard Handling

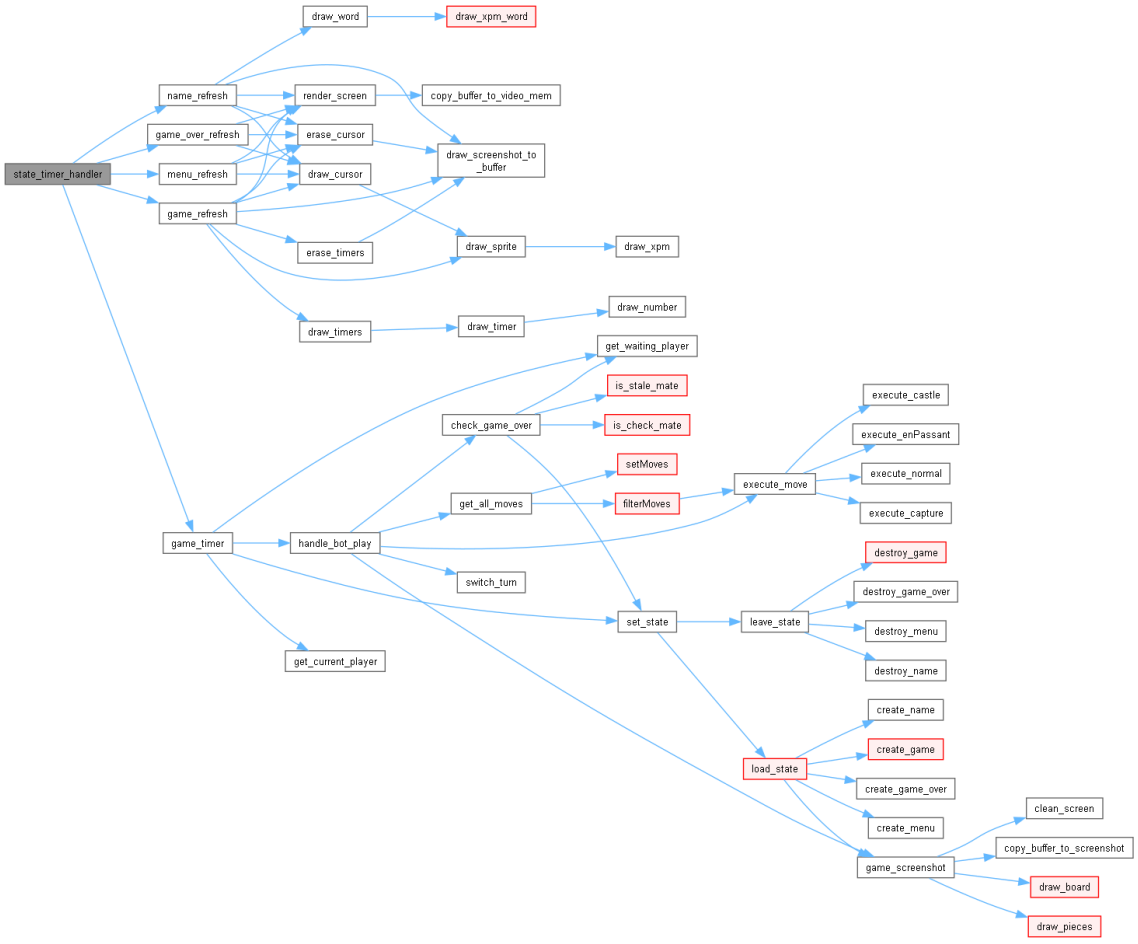


Figura 23. State Timer Handling

menu.c

No ficheiro *menu.c*, está definido o modelo do Menu. A estrutura de dados *Menu* contém um *Title*, que será exibido no ecrã, dois *Button*'s, um sendo o *playButton* e outro o *exitButton*, e um *Sprite* para o fundo. Tanto o título como os botões possuem

coordenadas, largura e altura, de modo a facilitar o processo de renderização dos mesmos e do tratamento de inputs do rato.

menu_controller.c

No ficheiro *menu_controller.c*, são tratados os inputs do Menu. Quando no estado Menu, a cada interrupt do keyboard, ao acabar de ler um *scan code*, irá chamar a função *menu_kbd()*, que irá receber o scan_code e tratá-lo. No caso deste corresponder ao da *break code* da tecla *ENTER* este irá alterar o estado para WRITE_NAME. Se corresponder ao *break code* da tecla *ESC*, este irá alterar o estado para EXIT, o que terminará o programa. Para cada interrupt do rato, se a posição do rato coincidir com a área de um dos botões, definidos no modelo do Menu, o estado alterar-se-á. Se o clique for no botão de *play*, passará ao estado *WRITE_NAME*, mas se o clique fôr no *exit*, passará ao estado de EXIT e terminará o programa.

menu_viewer.c

No ficheiro *menu_viewer.c* está definido o viewer do Menu. Este possui duas funções principais, *menu_screenshot()* e *menu_refresh()*. Ao entrar no estado de Menu, a função *menu_screenshot ()* desenha no buffer a sua imagem de fundo, titulo e botões e guarda uma cópia do buffer chamada de *screenshot*. A cada tick do *timer*, quando neste estado, é chamada a função *menu_refresh()*, que atualiza a hora e data, e a posição do rato no ecrã.

name_input.c

No ficheiro *name_input.c*, está definido o modelo do NameInput. A estrutura de dados *NameInput* contém coordenadas para onde o input do utilizador será exibido, uma string onde o input do utilizador será guardado e atualizado, assim como um limite de caracteres, e um sprite que serve como fonte para o input que será exibido.

name_controller.c

No ficheiro name_controller.c, está definido o controlador do NameInput. Quando no estado WRITE_NAME, a cada interrupt do keyboard, ao acabar de ler um *scan code*, irá chamar a função *name_kbd()*, que irá receber o scan_code e tratá-lo. No caso deste corresponder ao da *break code* da tecla *ENTER* este irá alterar o estado para GAME e inicirará a partida. Se corresponder ao *break code* da tecla *ESC*, este irá alterar o estado para MENU, e voltará para o menu do jogo.

name_viewer.c

No ficheiro *name_viewer.c* está definido o viewer do NameInput. Este possui duas funções principais, *name_screenshot*() e *menu_refresh()*. Ao entrar no estado de WRITE_NAME, a função *name_screenshot()* desenha no buffer a sua imagem de fundo e titulo, e guarda uma cópia do buffer chamada de *screenshot*. A cada tick do *timer*, quando neste estado, é chamada a função *name_refresh()*, que atualiza o nome de acordo com o input fornecido, e a posição do rato no ecrã.

pieces.c

O ficheiro pieces.c, define a estrutura de dados *Piece* e dispõe de método que servem de contrutores para as mesmas. *Piece* possui uma *position* que possui coordenadas x e y, uma *color* que pode ser branca ou preta, um *type*, podendo ser este um peão, torre, bispo, cavalo, rei ou rainha, um *status*, viva ou capturada, um *boolean has_moved* que indica se a peça já foi movida, um sprite, correspondente ao tipo da peça, e um array de movimentos que servirá para guardar os movimentos disponíveis da peça quando selecionada.

player.c

O ficheiro player.c define a estrutura de dados Player. Esta reprenta um jogador no jogo, e possuir de atributos *name*, o nome do jogador, uma *color* correspondente no contexto do jogo, um *counter* para o seu tempo dentro da partida, um *type*, sendo este AI ou USER, um array de 16 peças onde serão guardados apontadores para todas as peças do jogador.

board.c

O ficheiro board.c define a estrutura de dados Board,. Este dispõe dum array de 64 quadrados, podendo estes ter um apontador apra uma peça ou *NULL*. O índice duma peça no tabuleiro, corresponde à sua posição também (índice = y * 8 + x). Board dispõe de métodos que servem como construtores e que incializam todas as suas peças nas respetivas posições e associam cada uma ao jogador correspondente.

move.c

O ficheiro move.c define a estrutura de dados Move correspondente a uma jogada. Este dispõe de uma posição de origem, *origin*, e uma de destino, *destination*, um apontdaor para a peça que será movida, *piece*, um segundo apontador opcional para uma peça que possa ser capturada ou uma torre que possar ser usada para um castle, *secondary_piece*, e atributos que guardam o estado anterior das peças, préviamente às jogadas, como *had_moved* e *old_type*.

game.c

O ficheiro game.c define o modelo do Game. Este dispõe da estrutura de dados *Game*, que possui um *Board*, dois *Player*'s e um *Turn*, que indica a cor jogador de quem é a vez de jogar. No ficheiro estão definidos inúmeros métodos para gestão do modelo

do Game, como funçoes para verficação do estado de jogo: is_check(), has_moves(), is_check_mate() e is_stale _mate(). Assim como também existem funções para gestão de jogadas no jogo: undo_move(), execute_move(), filter_moves(), set_moves() e switch_turn().

game_controller.c

O ficheiro game_controller.c define o controlador do jogo.

Quando no estado Game, a cada interrupt do keyboard, ao acabar de ler um *scan code*, irá chamar a função *game_kbd()*, que irá receber o scan_code e tratá-lo. No caso deste corresponder ao da *break code* da tecla *ESC*, este irá alterar o estado para MENU, o que terminará a partida e passará para o menu do jogo.

Para cada clique no botao esquerdo do rato, se a posição do rato coincidir com uma peça do jogador que tiver a vez de jogar, essa é selecionada e as suas jogadas disponíveis são determinadas e guardadas. A peça manter-se-á selecionada se o jogador mantiver o botão esquerdo premido. Ao largar o botão, caso o rato esteja na posição de uma jogada válida, esta será executada, caso contrário, a peça voltará ao estado anterior. Se o jogador clicar no botão direito do rato, enquanto tem uma peça selecionada, esta será deselecionada.

Para cada interrupção do *timer*, o contador do tempo do jogador que tiver a vez de jogar será decrementado. Caso o turno seja de um Bot/AI, sempre que o seu turno é iniciado, este gera um tempo aleatório de jogada que, e a cada interrupção do timer, será verificado se este tempo já passou. Caso tenha, a jogada do bot será executada e o jogo prosseguirá.

game_viewer.c

O ficheiro game_viewer.c define o modelo do Game. Ao entrar no modo Game, é chamada a função game_screenshot(), que desenha o tabuleiro, todas as peças vivas e não selecionadas e jogadas da peça selecionada, caso seja o caso. A cada interrupção do timer é chamada a função game_refresh() que atualiza os timers dos jogadores, assim como a posição do rato e, caso haja uma peça selecionada, desenha esta na posição do rato também.

game_over.c

O ficheiro game_over.c define o model do GameOver. Este dispõe duma estrutura de dados *GameOver* que contém dois *Button's*, um de *Exit* para sair para o menu do jogo e um de *Play* para jogar outra vez. Para além disto dispõe duas strings, *result* e *winner*, sendo que *result* guarda os desfecho da partida, i.e. "Checkmate", "Stalemate", etc., e *winner* guarda a mensagem que anuncia o vencedor, i.e. "Bot wins", "Draw", etc.

game_over_controller.c

O ficheiro game_over_controller.c define o controlador do GameOver. Quando no estado Menu, a cada interrupt do keyboard, ao acabar de ler um *scan code*, irá chamar a função *game_over_kbd()*, que irá receber o scan_code e tratá-lo. No caso deste corresponder ao da *break code* da tecla *ENTER* este irá alterar o estado para GAME, e começará uma nova partida. Se corresponder ao *break code* da tecla *ESC*, este irá alterar o estado para MENU, o que passará para o menu. Para cada interrupt do rato, será chamada a função game_over_cursor(), em que se a posição do rato coincidir com a área de um dos botões, definidos no modelo do Menu, o estado alterar-se-á. Se o clique for no botão de *play*, passará ao estado *GAME,* e começará uma nova partida mas se o clique fôr no *exit*, passará ao estado de MENU e voltará para o menu do jogo.

game_over_viewer.c

No ficheiro *game_over_viewer.c* está definido o viewer do GameOver. Este possui duas funções principais, *game_over_screenshot()* e *game_over_refresh()*. Ao entrar no estado de Menu, a função *game_over_screenshot ()* desenha no buffer a sua imagem de fundo, titulo e botões e guarda uma cópia do buffer chamada de *screenshot*. A cada tick do *timer*, quando neste estado, é chamada a função *game_over_refresh()*, que atualiza a posição do rato no ecrã.

rtc.c

No ficheiro rtc.c, estão implementadas todas as funções que lidam com o Real-Time Clock. Inicialmente, verifica-se se o RTC está em modo binário, sendo esta informação guardada numa variável global. Se o RTC estiver neste modo, é feita a atualização da hora atual pela função rtc_update_current_time, caso contrário, é necessário convertê-lo para modo binário. Esta função lê e atualiza os componentes do tempo atual (segundos, minutos, horas, dia, mês, ano). Por fim, é implementada a função que lida com os interrupts do RTC, atualizando o tempo atual quando um interrupt ocorre.

gpu.c

O ficheiro gpu.c manipula operações de gráficos numa unidade de processamento gráfico. Contém funções para configuração da resolução e modo de cor, mapeamento de memória para a GPU, desenho de retângulos e padrões (utilizados para gerar o tabuleiro), e manipulação de imagens no formato XPM. Além disso, implementa a função para movimentar um sprite na tela. O ficheiro trabalha também tanto com cores indexadas quanto com cores diretamente especificadas.

sp.c

O ficheiro sp.c define várias funções relacionadas com a manipulação e controlo do "serial port". Estas funções incluem a inicialização do "serial port" (serial_port_initialize), a "subscription" e "unsubscription" de interrupções (serial_port_subscribe_int e serial_port_unsubscribe_int), a leitura e envio de bytes (serial_port_read_byte e serial_port_send_byte), e a manipulação de interrupções (serial_port_interrupt_handler). Por fim, uma queue é usada para armazenar bytes recebidos através do serial port.

data_queue.c

O ficheiro data_queue.c contém a implementação de uma queue projetada para processar dados no contexto da serial port. A implementação desta data_queue inclui funções de inicialização e destruição, funções para dar "push" e "pop" para a queue. Além disso, estão definidas funções adicionais como data_queue_current_size para obter o tamanho atual da estrutura de dados, data_queue_peek para observar o próximo elemento a ser exibido, data_queue_contains para verificar se um determinado elemento está na queue, data_queue_front e data_queue_back para aceder ao primeiro e último elemento da fila, respectivamente, sem os remover. Por fim, estão também implementadas as funções data_queue_is_full e data_queue_is_empty para verificar o estado da fila.

mouse.c

No ficheiro mouse.c, está implementada a interação com o mouse. A função mouse_subscribe_int é usada para subscrever interrupts do mouse, enquanto a mouse_unsubscribe_int é usada para cancelar essa subscrição. A função mouse_ih é o handler de interrupção do mouse, tratando os dados do mouse conforme eles são lidos. As funções enable_data_reporting e disable_data_reporting controlam a transmissão de dados do mouse. As funções get_response e wait_for_input_buffer permitem a leitura de respostas do mouse e a espera pela disponibilidade do buffer de entrada, respectivamente. A função write_to_mouse é utilizada para escrever um comando para o mouse, enquanto write_command permite enviar um comando com argumentos para o mouse.

keyboard.c

No ficheiro keyboard.c, estão definidas as funções relacionadas com o teclado. A função keyboard_subscribe_int é utilizada para subscrever interrupts do teclado, guardando o número de bit da linha de interrupção do IRQ numa variável. De forma inversa, a função keyboard_unsubscribe_int é utilizada para cancelar a subscrição dos interrupts do teclado. A função kbc_ih serve como um handler de interrupção para o teclado. Esta função lê o estado do teclado a partir do buffer de status, verificando se

há algum erro e lendo o scan code do teclado. O scan code é armazenado num array scan_code. A variável last_byte_read indica se o último byte foi lido.

sprite.c

No ficheiro sprite.c, estão definidos vários sprites utilizados no jogo. A função load_sprites é responsável por alocar memória para cada sprite e carregar as respetivas imagens através da função load_sprite. Esta função recebe um mapa xpm correspondente a cada sprite e carrega a imagem para a estrutura de dados do sprite. A função load_sprite utiliza a função load_xpm para carregar o mapa xpm correspondente a um sprite específico. A imagem resultante é armazenada na estrutura de dados do sprite, juntamente com o endereço onde a imagem está armazenada na memória. A função destroy_sprites é responsável por libertar a memória alocada para cada sprite, prevenindo possíveis vazamentos de memória.

Function call Graph

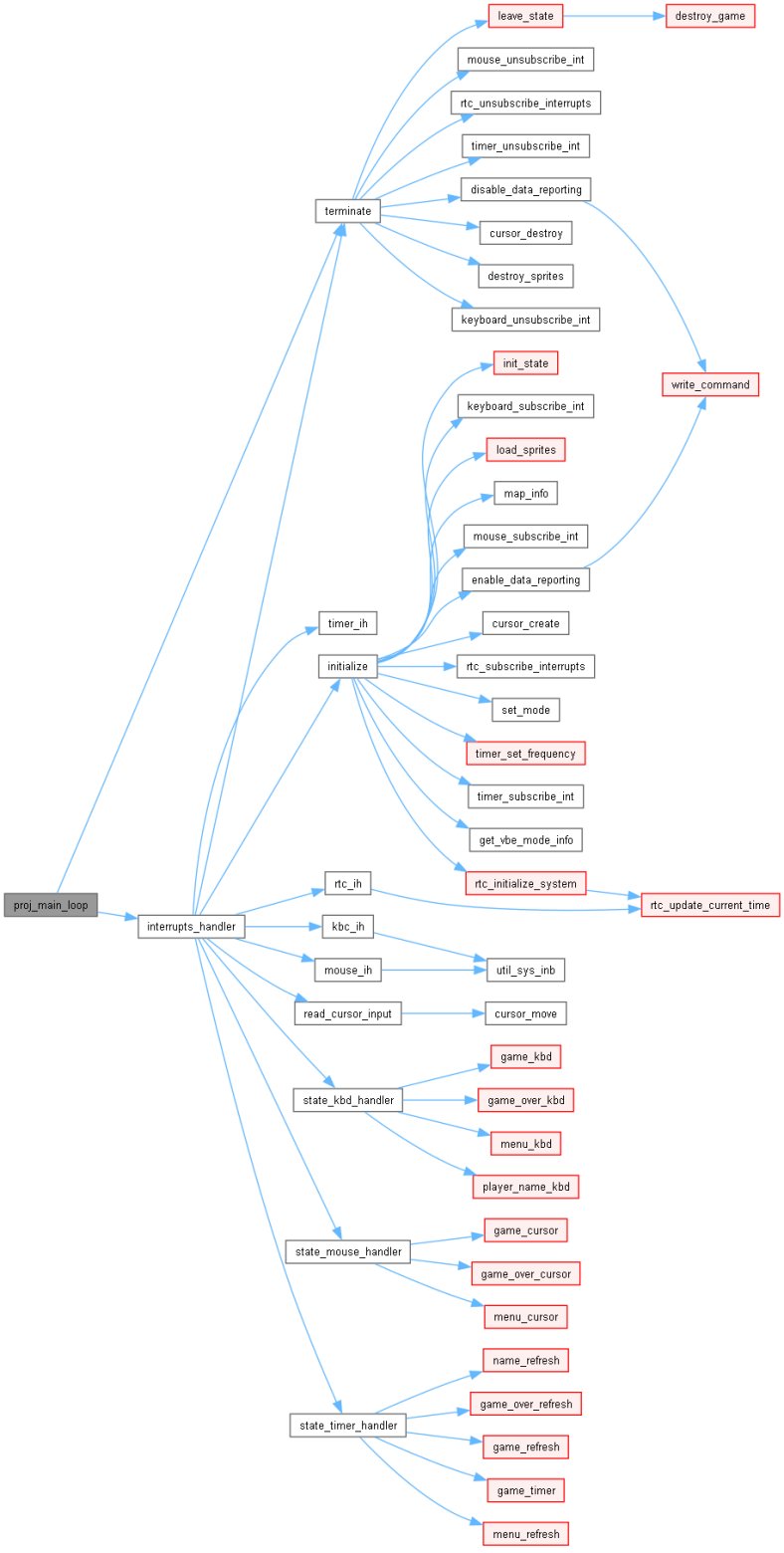


Figura 34. Main Loop Call Graph

Conclusões

Neste projeto, assumimos o desafio de ir além da implementação dos dispositivos obrigatórios, o que nos exigiu uma profunda compreensão do mouse, do RTC e do Serial Port. Obtivemos, assim, uma visão completa e abrangente das funcionalidades e dos requisitos de todos os dispositivos discutidos ao longo das aulas teóricas.

Contudo, não conseguimos integrar todas as funcionalidades inicialmente propostas, devido à complexidade inerente do Serial Port e ao tempo limitado para conclusão do projeto. Porém, efetuámos a definição e a implementação das suas funções.

No nosso jogo, o mouse emergiu como o dispositivo principal. Usado extensivamente, desde a seleção no menu à movimentação das peças no tabuleiro, aprofundamos a nossa compreensão sobre o seu funcionamento e a sua importância num contexto de jogo.

A realização deste projeto foi possível graças a um trabalho de equipa dedicado, com tarefas bem distribuídas, permitindo-nos cumprir os restantes objetivos dentro do prazo.