Faculdade de Engenharia da Universidade do Porto

Chess Duels

LCOM 2023 - Projeto Final - Licenciatura em Engenharia Informática e Computação

Realizado por:

- António Azevedo (up202108689@edu.fe.up.pt);
- Duarte Gonçalves (up202108772@edu.fe.up.pt);
- Tomás Martins (up202108776@edu.fe.up.pt);

Índice

1. Introdução

No âmbito da cadeira de LCOM, decidimos criar um jogo de nome **“Chess Duels”**. Este consiste na recriação do jogo Xadrez no qual dois utilizadores irão mover unicamente e uma vez por turno uma peça da respetiva cor, seguindo as regras do Xadrez.

2. Instruções de utilização do programa

2.1 Ecrã inicial

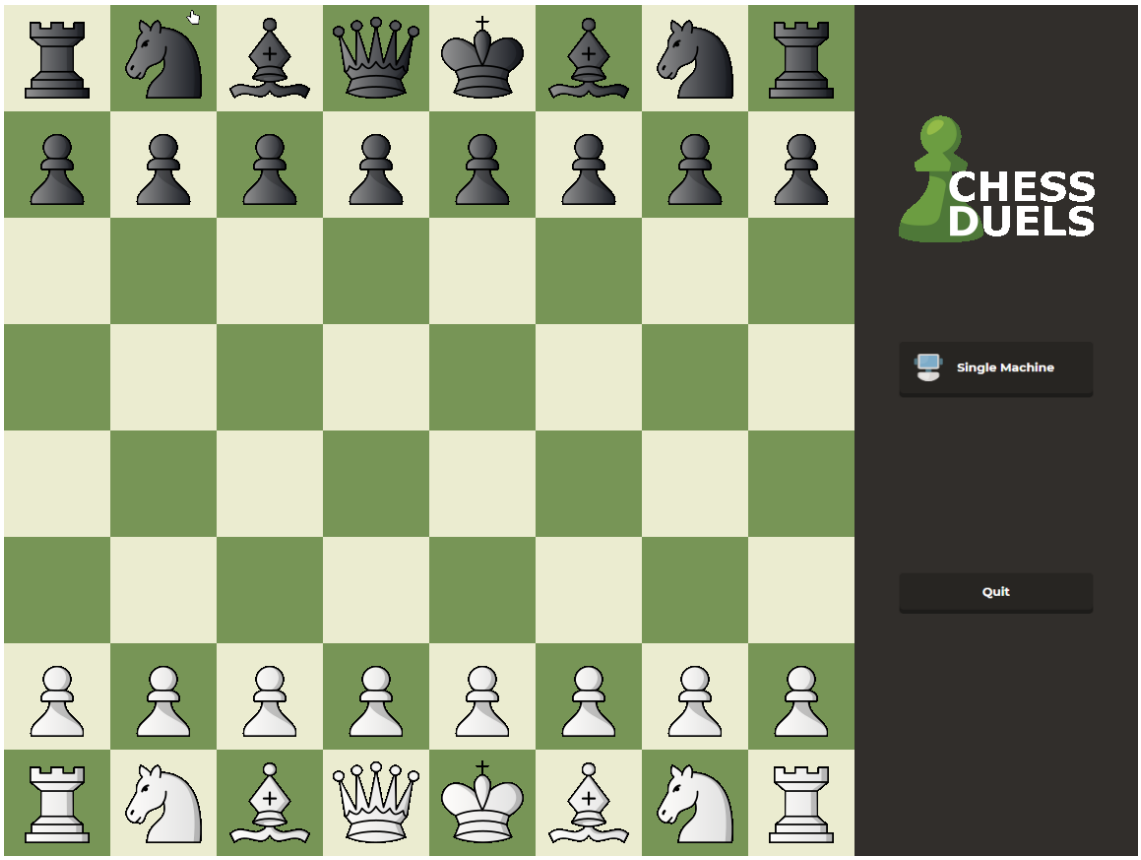

Imagem 1 – ecrã inicial

Ao iniciar o jogo, é apresentado o tabuleiro de xadrez com as peças colocadas no local correto. Além disso, é mostrado um menu vertical que irá permitir aos utilizadores iniciarem o programa. Caso o botão “Single Machine” não seja premido, todos os tipos de interações com as peças estarão bloqueados, sendo possível iniciar o jogo pressionando o botão mencionado anteriormente. O outro botão no menu, tal como está escrito no mesmo, permite ao utilizador encerrar o jogo.

2.2 Ecrã de jogo

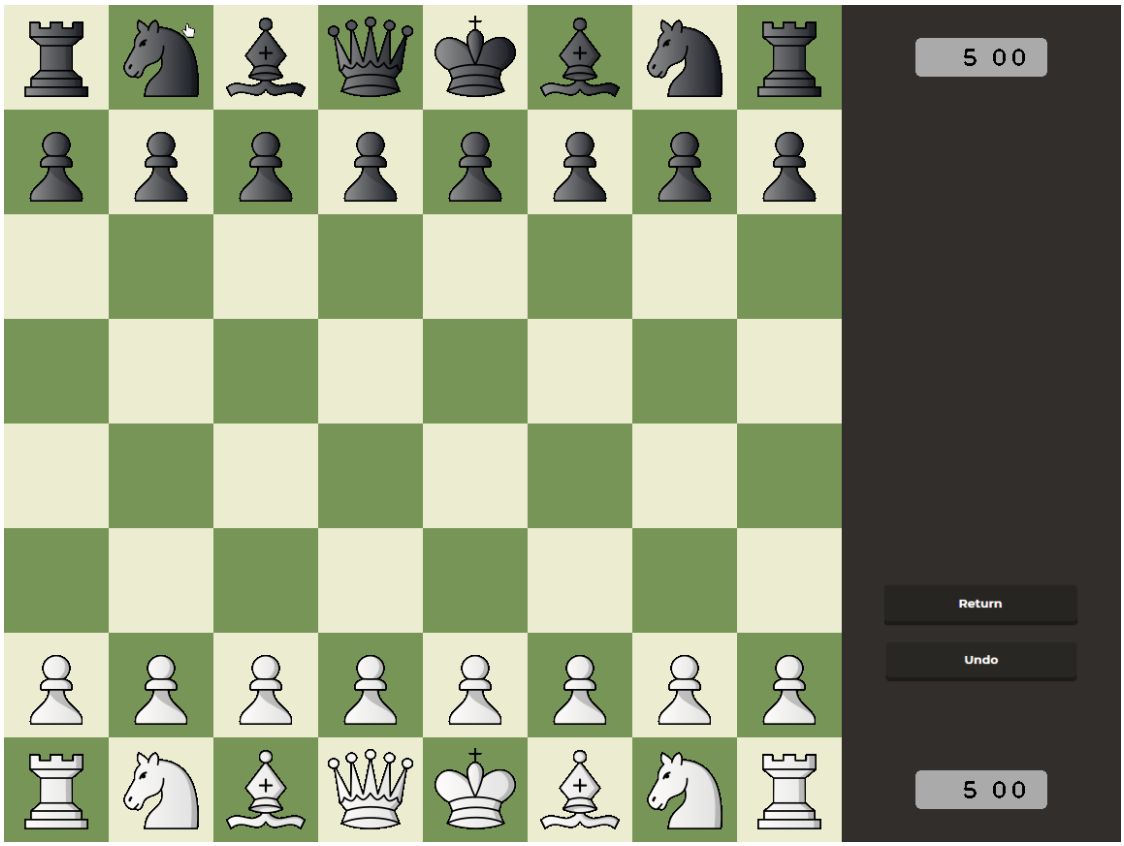

Imagem 2 - ecrã de jogo

Após o botão que inicia o jogo ter sido premido, é exibido um diferente menu vertical que apresenta dois temporizadores e dois botões. Estes temporizadores indicam o tempo restante que cada jogador tem para acabar o jogo. Sempre que um dos jogadores termina a sua jogada o seu respetivo temporizador para de contar e o do adversário começa. Se eventualmente um dos temporizadores chegar a 0, o utilizador respetivo perde o jogo. O botão *return*, quando clicado, muda o ecrã para o anterior e o botão *undo* faz com que o jogo volte atrás uma jogada, permitindo ao jogador refazer o seu movimento.

Para além disso, contrariamente ao que acontecia no ecrã anterior, neste momento os utilizadores já são capazes de interagir com as peças. Para isso podem tanto utilizar o rato como o teclado. Caso se deseje utilizar o rato basta mover o rato e clicar na peça desejada e escolher uma "casa" vazia. Se esse movimento for permitido, dependendo do tipo de peça que se deseja mover, a peça clicada será movida, caso contrário essa será desselecionada e nenhuma peça se moverá até o jogador realizar um movimento válido. Do mesmo modo, para selecionar uma peça com o teclado basta ao jogador escrever as coordenadas da peça e após isso as coordenadas da "casa" para que deseja movimentar a peça.

Selecionar uma coluna (A-H) seguido de uma linha (1-8), ex: A2 -> A4, A7 -> A6…

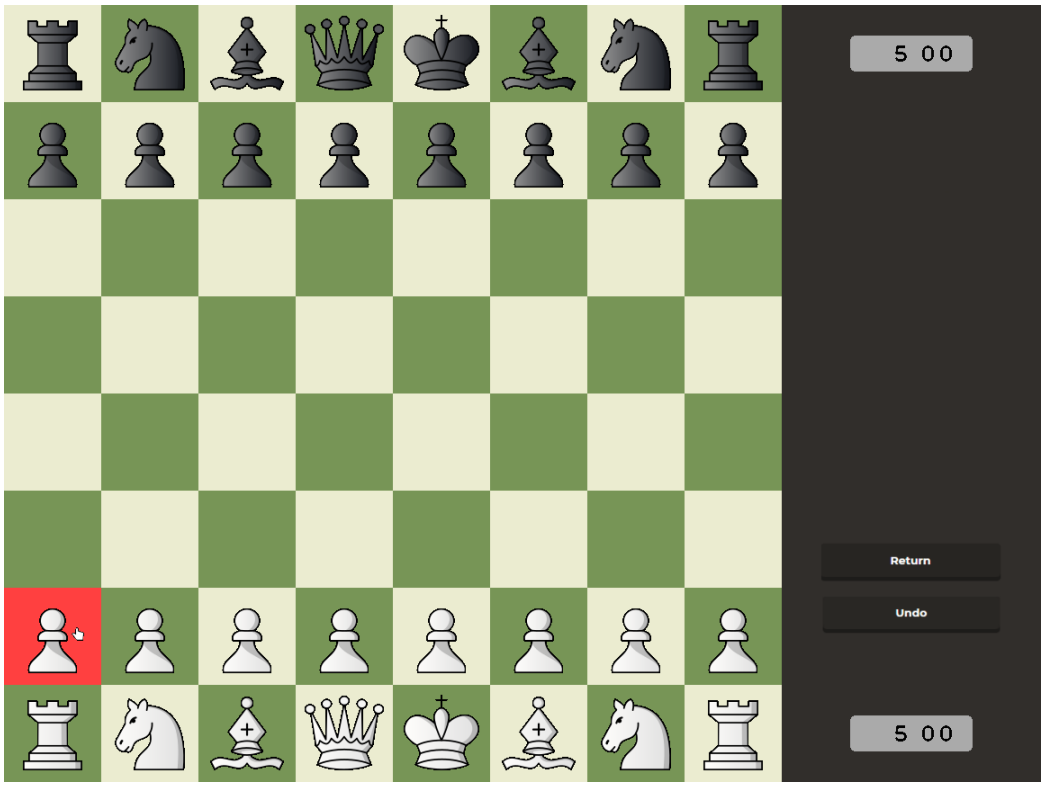

Imagem 3 - peça selecionada

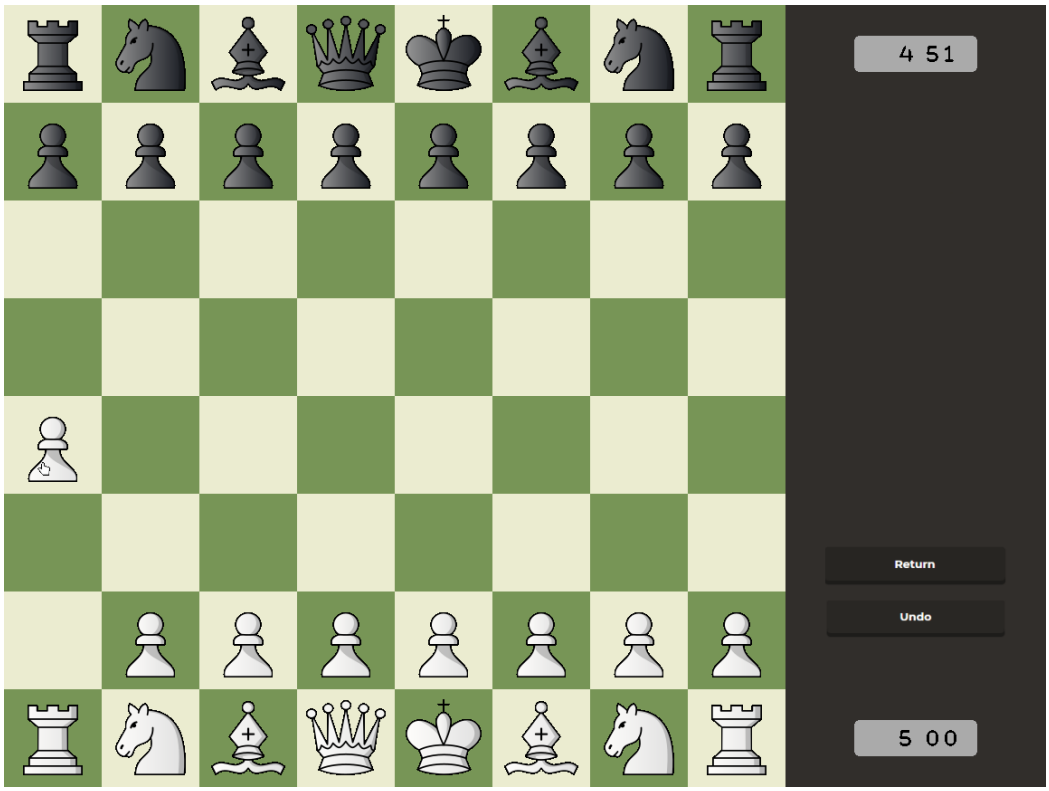

Imagem 4 - peça movida para a "casa" desejada

2.3 *Dark mode* e *Light mode*

Apesar de ser uma alteração meramente visual, os menus anteriores podem apresentar uma variação de cores. O tabuleiro que anteriormente possuía cores mais claras (*light mode*) passa a possuir cores mais escuras (*dark mode*). Esta variação de cores ocorre dependendo da hora em que se iniciou o jogo.

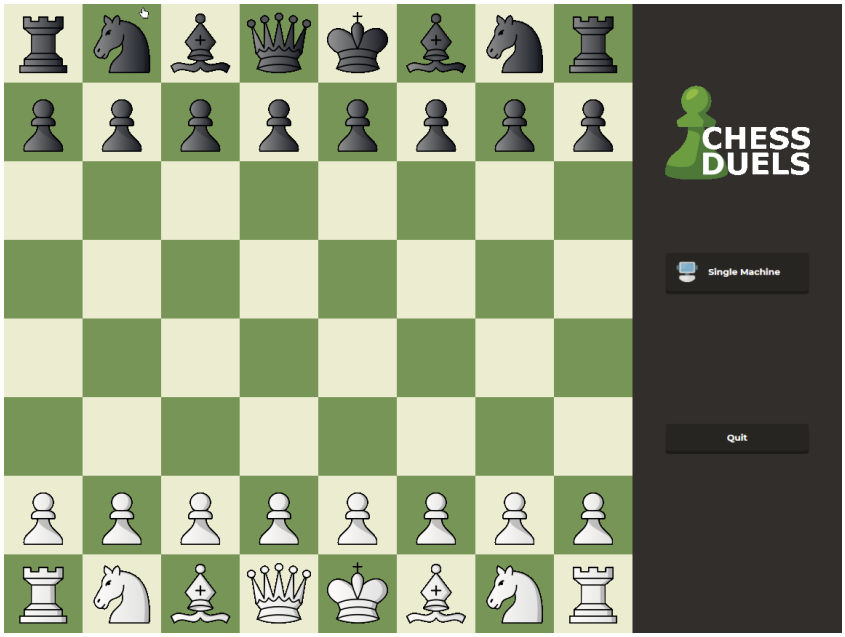

Imagem 5 - Light Mode

Imagem 6 - Dark Mode

3. Estado do projeto

3.1 Funcionalidades dos dispositivos

Dispositivo	Funcionalidades	Interrupts?
Timer	Controlo do temporizador de cada jogador, tal como o controlo do frame rate.	Sim
Keyboard	Interação com os menus e movimentação de peças.	Sim
Mouse	Navegação nos menus e movimentação de peças	Sim
Graphics Card	Display de todos os assets gráficos no ecrã.	Não
Real Time Clock	Alterar entre *light mode* e *dark mode.*	Sim

3.2 Timer

O **timer**, tal como descrito anteriormente, tem como funcionalidades controlar o temporizador de cada jogador e controlar o frame rate do jogo. Iniciando-se o jogo, a cada interrupção do timer, a função timer_handler() é chamada. Nessa, são utilizadas outras funções que: atualizam o temporizador do turno de cada jogador e verificam se já chegou a zero (clock_update() e clock_timeout()); incrementam o contador do timer em 1 (timer_int_handler()); verificam se naquele momento algum dos jogadores se encontra em *checkmate,* saindo do jogo em caso verdadeiro (is_checkmate()); gerem a atualização dos *assests* gráficos no ecrã. As funções de configuração do timer encontram-se no ficheiro **timer.c** e as chamadas às outras funções mencionadas encontram-se no ficheiro **dispatcher.c**.

3.3 Keyboard

O **keyboard** foi utilizado para permitir ao utilizador, a navegação pelos menus existentes e a movimentação das peças durante o jogo. Caso o utilizador queira navegar pelos menus, basta clicar nas teclas ***arrow_up*** ou ***arrow_down*** para selecionar a opção de cima ou de baixo, respetivamente e pressionar a tecla ***enter*** para escolher a opção. Para além disso, caso o jogador esteja no ecrã de jogo, se este desejar escolher a opção *undo*, basta clicar na tecla "**z**" ao invés de selecionar a opção com as ***arrow_keys*** do teclado. Para concluir a descrição das ações do teclado na navegação entre ecrãs, é também possível encerrar o jogo, caso o utilizador o deseje,

pressionando a tecla "**esc**". Estas ações são realizadas através da chamada à função menu_handle_keyboard(), que é acionada após a ocorrência de uma interrupção do **keyboard**.

Quanto à seleção das peças a mover, basta ao utilizador inserir as coordenadas desejadas, pressionando no teclado uma tecla com uma letra de A a H seguida de uma tecla com um número de 1 a 8. Consequentemente aparecerá um fundo vermelho na peça selecionada e neste momento o utilizador irá repetir a mesma ação no teclado que realizou anteriormente de forma a escolher as coordenadas do local onde a peça se irá movimentar, caso seja possível. Se o jogador desejar selecionar outra peça, este terá de inserir umas novas coordenadas ou terá de pressionar a tecla ***backspace*** e realizar o movimento a seguir (só se ainda não tiver realizado o movimento). A interação com as peças é possível através da função keyboard_handle_input() que é chamada após a ocorrência de uma interrupção do **keyboard**. Ambas as chamadas das funções mencionadas anteriormente são feitas no ficheiro **dispatcher.c** na função keyboard_handler(), e a configuração do dispositivo é feita nos ficheiros **keyboard/kbc.c** e **keyboard.c**.

3.4 Mouse

De certo modo, o **mouse** é capaz de realizar as mesmas funcionalidades descritas no keyboard, mas nos menus é necessário colocar o cursor sobre a opção desejada e de seguida pressionar o botão esquerdo do rato (menu_handle_mouse()), e na seleção e movimentação das peças, o utilizador precisa de colocar o cursor sobre a peça desejada e clicar no botão esquerdo do rato, realizando de seguida a mesma ação, mas escolhendo a "casa" para a qual, se possível, irá colocar a peça (mouse_select_piece() e mouse_move_piece()). As estas funções são executadas sempre que a função mouse_handler() é chamada sendo que esta ocorre aquando de uma interrupção do mouse. A configuração do dispositivo encontra-se nos ficheiros **mouse/kbc.c** e **mouse.c** e a chamada à função que define os acontecimentos após uma interrupção encontra-se no ficheiro **dispatcher.c**.

3.5 Graphics Card

Para este projeto decidimos utilizar a placa de vídeo no modo **0x14C**, com uma resolução de **1152x864** e com um **modo de cor direto**, sendo possível representar $\mathbf{2^{32} = 4294967296}$ cores diferentes. Para inicializar a placa de vídeo com as configurações desejadas utilizamos a função vg_start() que é chamada na função de inicialização do jogo, game_init(). Do mesmo modo, ao encerrar o jogo, a função que tem como papel o termino do programa chama a função vg_exit() que altera o modo de vídeo para o ***text_mode***. Ambas as funções que são chamadas na inicialização e no termino do programa encontram-se no ficheiro **game.c** e as funções que procedem à configuração do dispositivo encontram-se no ficheiro **vídeo.c**.

Para conseguirmos dar display de todos os *assets* gráficos necessários ao jogo e para manter

o movimento do cursor mais fluido e sem "ruido" visual, decidimos implementar ***double buffering***. O buffer auxiliar que foi criado, irá receber e guardar os componentes gráficos que queremos mostrar. Através da função draw_sprite() que consequentemente chama a função vg_draw_pixel(), os *assets* que se deseja dar display são "desenhados" nesse *buffer*. Após isso, é chamada a função vg_copy_buffer() que permite copiar o conteúdo do *buffer* secundário para o *buffer* principal, consequentemente removendo todo o conteúdo do *buffer* auxiliar e dando *display* do conteúdo copiado para o ecrã.

Todos os elementos visuais que queríamos implementar foram convertidos para o formato XPM tendo sido utilizados como *sprites*, depois da chamada à função Sprite *create_sprite(), e consequentemente desenhados no ecrã como referido no parágrafo anterior.

3.6 Real Time Clock

O RTC foi utilizado com o propósito de alterar dinamicamente o visual do jogo. Assim, é se extraído, o dia, o mês, o ano e a hora sempre que uma interrupção seja gerada (rtc_handler() e rtc_ih()), isto é, sempre que o relógio é atualizado. Caso a hora extraída seja igual a 00:00, as interrupções continuarão a ser recebidas até que eventualmente seja recebida uma hora diferente, pelo que consequentemente as seguintes interrupções passarão a ser ignoradas.

Assim, a primeira hora válida que é extraída no RTC irá ditar os tipos de cores que irão ser exibidos no tabuleiro de jogo, como foi mostrado anteriormente no ponto 2.3. De uma forma mais simplificada, decidimos implementar uma espécie de ***dark & light mode*** em que caso a hora atual esteja entre as 20h e as 8h as cores do tabuleiro serão de tons mais escuros e consequentemente nas outras horas as cores do tabuleiro serão de tons mais claros. As funções de configuração deste dispositivo encontram-se no ficheiro **rtc.c** e, no ficheiro **chess.c**, as cores do tabuleiro é alterada na função draw_board().

4. Organização e estrutura do código

4.1 Timer – 5%

Este módulo contém o código desenvolvido no Lab2 que possui diversas funções relacionadas com o timer, tal como, a sua configuração e controlo das respetivas interrupções.

Contribuição:

- António Azevedo;
- Duarte Gonçalves;
- Tomás Martins;

4.2 Keyboard – 5%

Este módulo contém o código desenvolvido no Lab3 que possui diversas funções relacionadas com o teclado, tal como, a sua configuração e controlo das respetivas interrupções.

Contribuição:

- António Azevedo;
- Duarte Gonçalves;
- Tomás Martins;

4.3 Mouse – 10%

Este módulo contém o código desenvolvido no Lab4 que possui diversas funções relacionadas com o rato, tal como, a sua configuração e controlo das respetivas interrupções.

Contribuição:

- António Azevedo;
- Duarte Gonçalves;
- Tomás Martins;

4.4 Video Graphics – 15%

Este módulo contém o código desenvolvido no Lab5 que possui diversas funções relacionadas com a placa de vídeo, tal como, a sua configuração e funções capazes de dar *display* dos sprites.

Contribuição:

- António Azevedo;
- Duarte Gonçalves;
- Tomás Martins;

4.5 RTC – 3%

Este módulo contém diversas funções relacionadas com o real time clock, tal como, a sua configuração e funções capazes de extrair a data e a hora.

Contribuição:

- António Azevedo;
- Duarte Gonçalves;
- Tomás Martins;

4.6 Chess – 25%

Este módulo contém todas as funções que dizem respeito ao funcionamento e lógica do jogo tal como o controlo do que é suposto aparecer no tabuleiro de jogo.

Neste módulo foram definidas *data structures* como:

- **Type**: o tipo de peça - ***empty***, ***king***, ***queen***, ***bishop***, ***knight***, ***rook*** e ***pawn***;
- **Color**: a cor da peça – ***undefined***, ***white*** e ***black***;
- **Piece**: definição dos atributos de uma peça – ***type*** e ***color*** – definidos anteriormente;
- **Position**: local no tabuleiro onde as peças estão – ***row*** e ***col***;

Contribuição:

- António Azevedo;
- Duarte Gonçalves;
- Tomás Martins;

4.7 Dispatcher – 25%

Este módulo contém todas as funções que gerem o que acontece aquando de uma interrupção. Dependendo de que dispositivo gerou a interrupção, uma função de nome *device*_handler() ("device" é substituído pelo respetivo dispositivo que gerou o interrupt) é chamada e nessa encontra-se definido o que acontecerá ao programa, sendo este módulo basicamente o núcleo do bom funcionamento do jogo.

Contribuição:

- António Azevedo;
- Duarte Gonçalves;
- Tomás Martins;

4.8 Utils – 1%

Este módulo contém funções que foram úteis na realização de outros módulos.

Contribuição:

- António Azevedo;
- Duarte Gonçalves;
- Tomás Martins;

4.9 Clock – 1%

Este módulo contém funções relacionadas ao funcionamento do temporizador de jogo.

Neste módulo foram definidas *data structures* como:

- **Time**: ***minutes*** e ***seconds***;
- **Clock**: o tempo que marca e se está ativo – ***time*** e ***running***;

Contribuição:

- António Azevedo;
- Duarte Gonçalves;
- Tomás Martins;

4.10 Game – 1%

Este módulo contém as funções que inicializam, controlam o decorrer e encerram o jogo.

Contribuição:

- António Azevedo;
- Duarte Gonçalves;
- Tomás Martins;

4.11 Menu – 1%

Este módulo contém as funções que controlam como se navega pelos menus e como estes são desenhados.

Contribuição:

- António Azevedo;
- Duarte Gonçalves;
- Tomás Martins;

4.12 Sprite – 7%

Este módulo contém as funções necessárias para converter os xpm em *Sprites* e desenhá-las

no ecrã, para fácil implementação das funções que interagem com elas, como por exemplo o rato a clicar num botão.

Neste módulo foram definidas *data structures* como:

- **Sprite**: struct que contem a informação sobre o xpm carregado - coordenadas ***x*** e ***y***, a ***height*** e ***width*** carregada e um array das cores de cada pixel (***colors***).

Contribuição:

- António Azevedo;
- Duarte Gonçalves;
- Tomás Martins;

4.13 Proj – 1%

Este módulo contém o game loop.

Contribuição:

- António Azevedo;
- Duarte Gonçalves;
- Tomás Martins;

5. Function call Graph

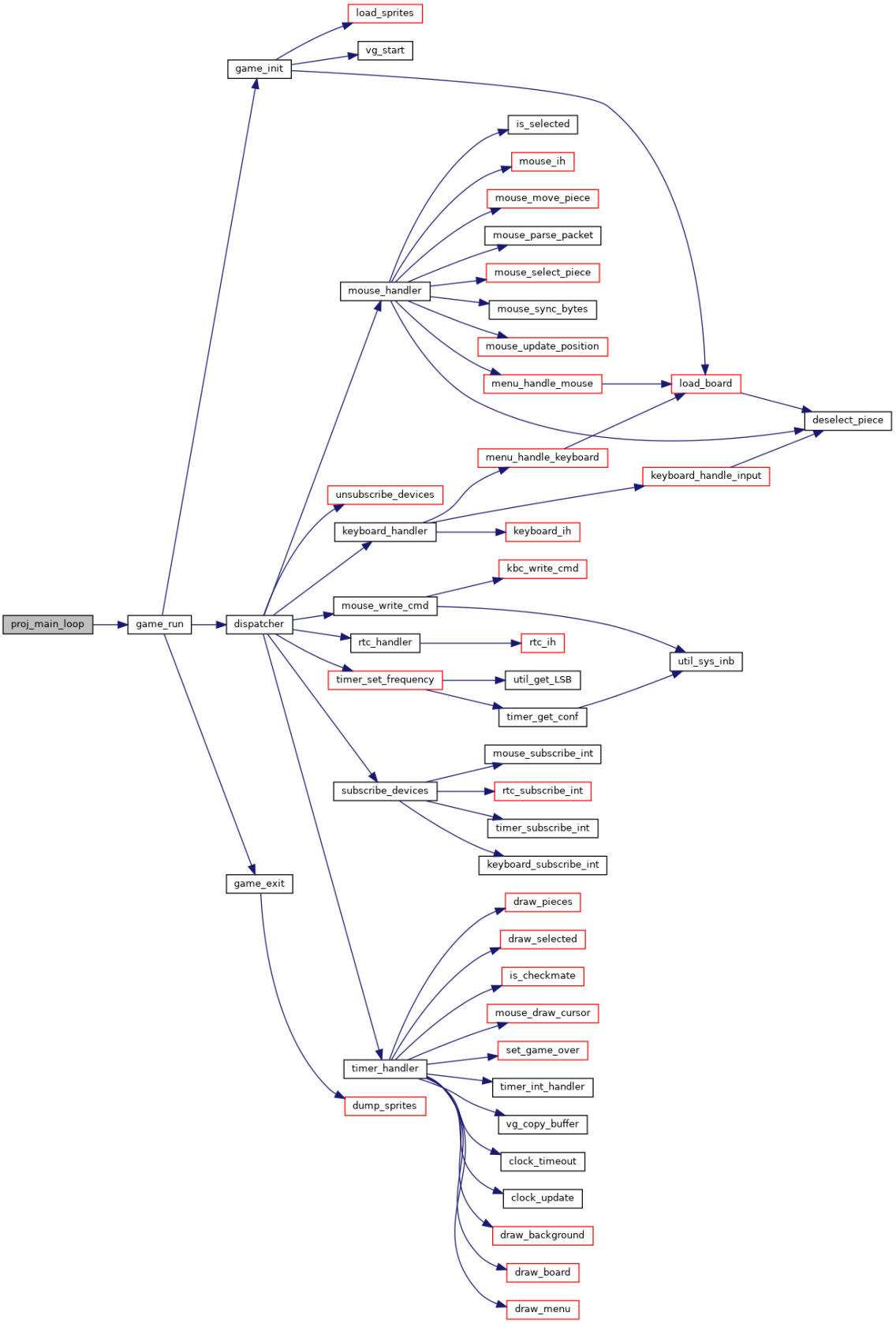

Imagem 7- function call graph

6. Detalhes de Implementação

Para este projeto, decidirmos traduzir os xpm numa struct Sprite, com atributos X e Y das coordenadas de cada sprite, bem como a largura e altura de cada uma.

Isto foi útil especialmente para a função do rato mouse_over_button(), usada para clicar nos botões do menu. Com a implementação de sprites, o desenvolvimento da lógica do rato, clicar em botões, foi trivial, apenas calculando se a posição do rato estaria dentro da "hitbox" retangular da sprite do botão.

7. Conclusões

Este projeto e o trabalho realizado nas aulas ao longo do semestre alargaram o nosso conhecimento quanto às possibilidades que temos ao trabalhar em *low-level* e deu-nos a oportunidade de trabalhar diretamente com o hardware e perceber como este funciona.

Devido a constrangimentos de tempo e complexidade de implementação, infelizmente não conseguimos implementar a UART que tínhamos planeado no início, e se tivéssemos mais tempo era definitivamente o que iríamos adicionar primeiro ao jogo. Também não desenvolvemos todas as mecânicas do jogo de xadrez, como o *roque* e *en passant*, que também adicionaríamos no futuro. Tínhamos pensado inicialmente em implementar o RTC para guardar um histórico de jogos em que obtínhamos informações como, a hora e a data em que o jogo se realizou para depois serem revistos, mas ultimamente decidimos implementar um *dark/light mode*.

A nossa maior conquista neste projeto foi conseguirmos trabalhar em sincronia com todos os dispositivos do aparelho e termos devolvido uma experiência de utilização fluída ao jogador.

Houve alguns obstáculos durante o desenvolvimento deste projeto, nomeadamente durante a tentativa de implementação das funcionalidades que não conseguimos introduzir, mas apesar de tudo estamos orgulhosos com o resultado.