Projeto LC Grupo T05G06

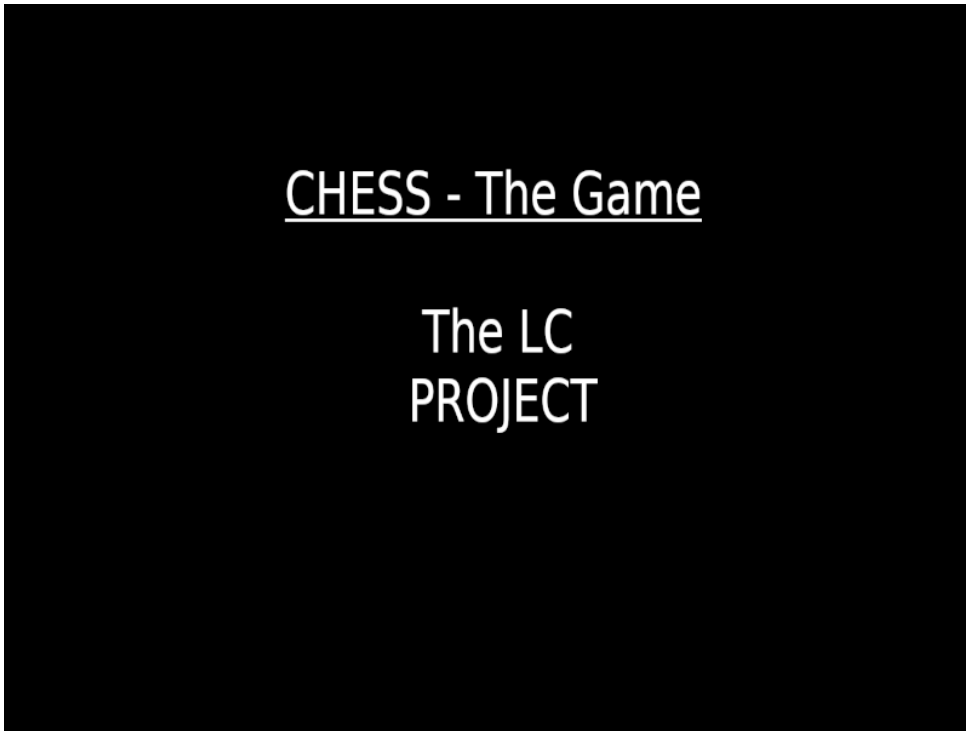
CHESS - The Game

User instructions:

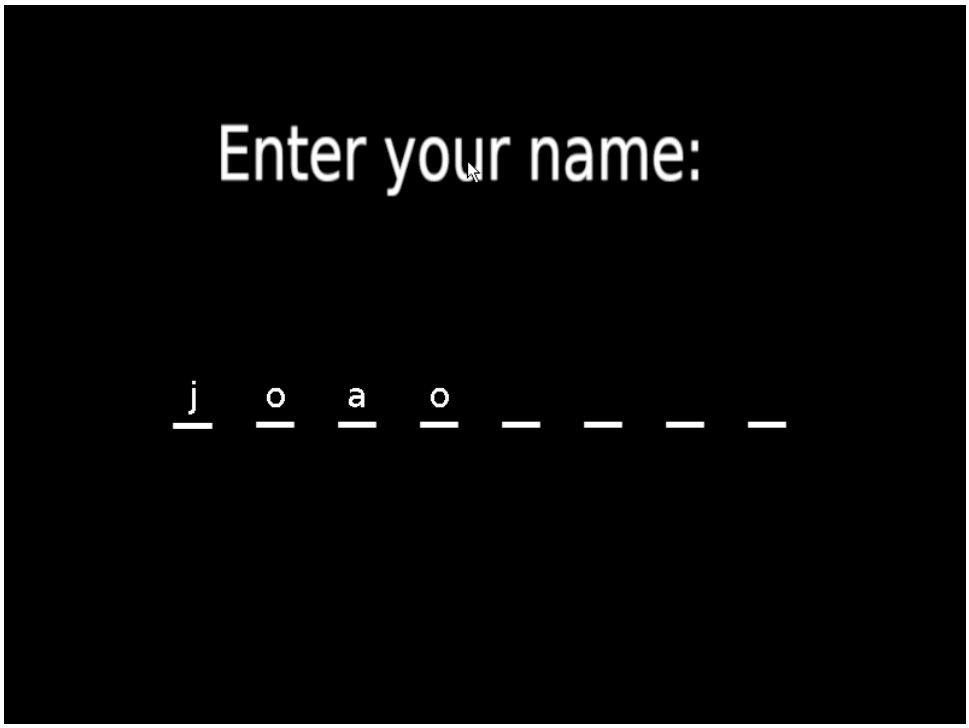
Para o tema do nosso projeto escolhemos desenvolver uma réplica do jogo clássico de xadrez, mas em forma virtual.

O jogo é bastante intuitivo na sua forma de jogar.

O programa começa com uma imagem xpm (welcome screen) onde é identificado o jogo.



Em seguida o jogador tem de escolher o seu nome até 8 caracteres alfabéticos utilizando o teclado.

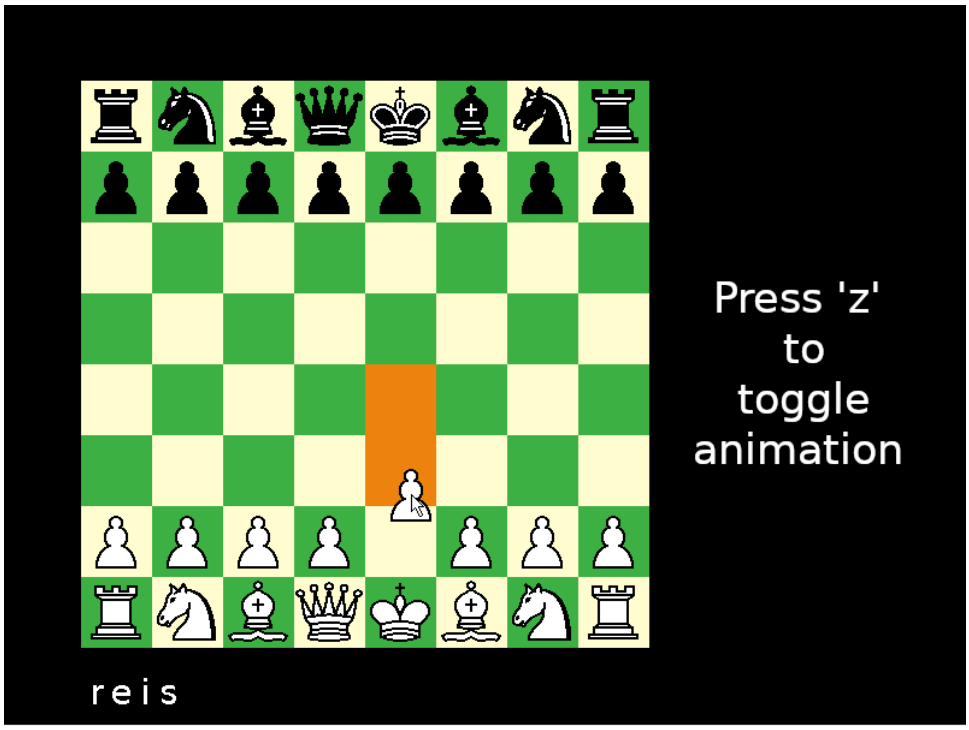


Enquanto um dos jogadores escolhe o nome no outro ecrã é apresentado um ecrã loading.

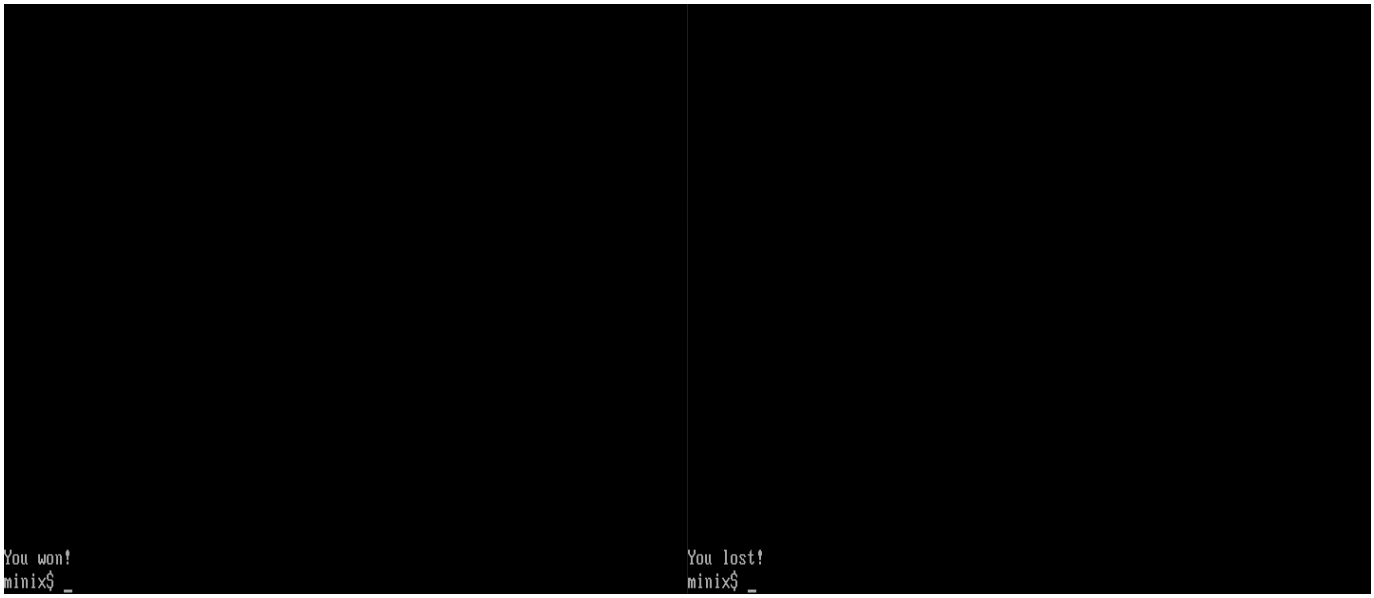
Após isso é mostrado um ecrã onde o jogador que se conectou primeiro pode escolher, usando o rato, a cor das peças que ele deseja (Player2 – Black Pieces) (Player1 – White Pieces).

Enquanto o primeiro jogador não escolhe o jogador que quer usar, no outro ecrã é apresentado um ecrã de loading .

Quando ambos os jogadores escolherem o nome, o tabuleiro é mostrado em ambas os ecrãs. O Player 1 (white pieces) pode então iniciar o jogo.



When a player's king is captured a message "You won!" or "You lost!" is shown.



For this game we did not implement the castling mechanic for both piece colors, we also didn't implement checkmate detection algorithms and lastly we didn't implement the "En passant" rule for pawns.

Project status:

Device	What for	Interrupt/polling
Timer	Controlo dos frame rates para as animações e refresh rates de screen displays	Interrupt
KBD	Input do nome do utilizador	Interrupt
Mouse	Mover peças e selecionar player1 ou player2	Interrupt
Video card	Menus da aplicação e tabuleiro de xadrez, juntamente com as peças	NA
Serial port	Multiplayer	Both

Timer:

Utilizamos o timer para controlar a frame rate para as animações das peças, e refresh rate, para diversos screen displays. Quando um jogador pousa uma peça esta descreve um movimento da sua posição inicial até à posição final.

Usamos este periférico no ficheiro main.c na função name_choice_ih(), playing_ih(), waiting_ih(), choosing_ih(), waiting_choosing_ih().

Keyboard:

Utilizamos o keyboard para receber input do utilizador aquando a inserção do seu nome. É também utilizado como controlo da aplicação pois o utilizador pode clicar na letra z para desativar as animações e vice-versa.

É usado no ficheiro main.c nas funções name_choice_ih() e playing_ih()

Graphics Card:

Utilizamos a placa gráfica para desenhar padrões de cor como, por exemplo, o tabuleiro e também ainda para desenhar imagens contidas em memória não volátil tais como os xpm's das peças.

- VideoMode = 0x115 (800x600, direct color mode)
- Utilizamos um sistema de quadruple buffering: um normal que mapeia a memória da vram, um para corrigir flickering, outro para o tabuleiro e um último para as peças.
- Implementamos também animações nos movimentos das peças em que a peça se move para o seu lugar de destino após ser arrastada para lá. Esta feature pode ser disabled em play-time.
- A principal fonte utilizada nas imagens é Sans-Serif.

Mouse:

Utilizamos o mouse para possibilitar o jogador de escolher se quer usar as peças brancas (Player 1) ou pretas (Player 2), onde vemos se a posição do rato corresponde ao Player 1 ou Player 2, e se o jogador carrega no botão esquerdo nesse lugar. Para além disso, o mouse permite ao jogador arrastar peças e largá-las onde pretende no tabuleiro. Para isso vemos se a posição do rato corresponde a alguma peça e se o jogador pressiona no botão esquerdo. Caso isto aconteça vamos atualizando a posição da peça para a posição do rato até o jogador soltar o butão esquerdo.

É usado no ficheiro main.c nas funções name_choice_ih(), playing_ih(), waiting_ih(), choosing_ih(), waiting_choosing_ih().

Serial-Port:

Utilizamos o serial-port para a comunicação entre jogadores. Nomeadamente, o jogador que acaba primeiro de escrever o seu nome, após escolher entre as peças pretas e brancas, comunica ao outro jogador o tipo de peças com que vai jogar. Após isto, durante o jogo, este é usado para comunicar ao oponente qual a jogada realizada e que é a sua vez de jogar.

A frequência de troca de dados corresponde a 4 bytes por jogada (um que indica a inicialização da mensagem da jogada, um segundo para a posição antiga da jogada, um terceiro para a posição nova, e um último para indicar que é a vez do oponente) durante o jogo.

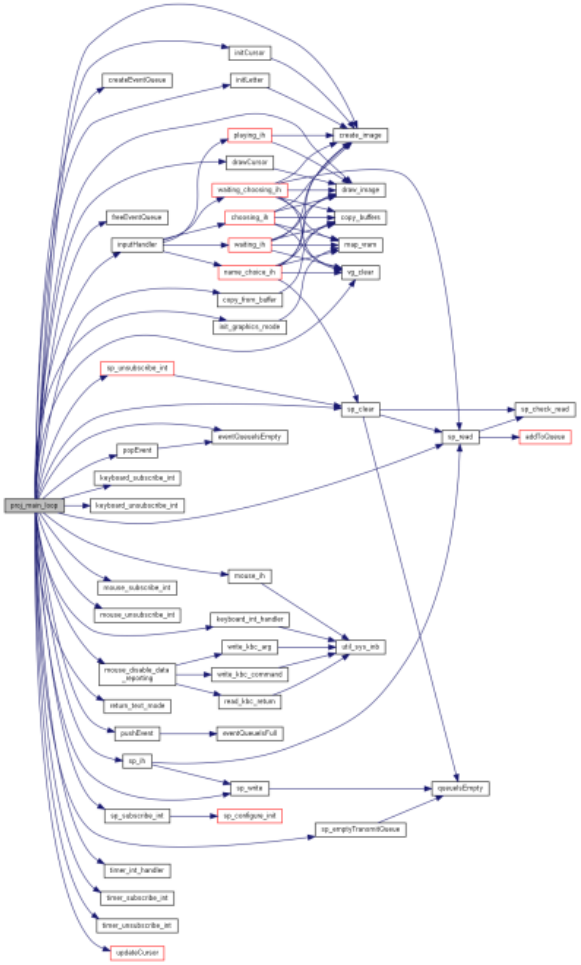
No início são enviados no total 3 bytes. O jogador que acaba em último de escolher o nome manda um byte (VM_CONNECTED) para o que acabou em primeiro lugar, que por sua vez manda um byte (VM_CONNECTED_RECEIVED), a indicar que recebeu este byte.

Quando recebe o VM_CONNECTED, o primeiro jogador tem acesso a um ecrâ para escolher se quer ser o jogador 1 (peças brancas) ou o jogador 2(peças pretas), enquanto o outro jogador fica à espera desta escolha. Após esta selação, é enviado um byte ao oponente a indicar as peças com que ficou, e passa a ter acesso ao tabuleiro. Após receber este byte, o segundo jogador também tem o mesmo acesso. O jogador que tiver as peças brancas (jogador 1) começa a jogar, sendo que enquanto um jogador está a jogar o oponente não consegue pegar em peças.

É utilizado polling para escrever no serial port, interrupções para ler do mesmo e polling para ler as mensagens da fila de mensagens recebidas, em conjunto com FIFO para garantir que não são perdidos dados.

É usado no ficheiro main.c nas funções name_choice_ih(), playing_ih(), waiting_ih(), choosing_ih(), waiting_choosing_ih() e no ficheiro cursor.c na função updateCursor() (para enviar a jogada para o oponente)

Code Organization/Structure:



- **Videocard.c**: Contém tudo acerca da placa gráfica e define todas as funções que desenham algo no ecrã. Também inicializa a placa gráfica no modo direct color 0x115. (15%)
- **Mouse.c:** Contém tudo que gira à volta do rato, sendo a sua principal função o mouse_ih(), que lê os packets gerados pelo rato. (10%)
- **Keyboard.c:** Contém tudo que gira à volta do teclado sendo a sua principal função o keyboard_int_handler() que trata de ler os make-codes e break-codes gerados pelo teclado. (5%)

- **Piece.c:** Este ficheiro trata da peça como um objeto. Guarda o tipo de peça, o seu estado, a sua posição e a sua imagem. A sua função mais

importante de sublinhar é a update_piece() que se encarrega de mover uma peça para ela poder ser animada, pois nesta função ela move-se um pixel de cada vez. (5%)

```
typedef enum PieceType{
    b_pawn,
    b_knight,
    b_bishop,
    b_rook,
    b_queen,
    b_king,
    w_pawn,
    w_knight,
    w_bishop,
    w_rook,
    w_queen,
    w_king,
} PieceType;


typedef struct Piece
{
    uint16_t absolute_x;    // screen x
    uint16_t absolute_y;    // screen y
    uint16_t x;             // board x
    uint16_t y;             // board y
    uint16_t target_x;      // target x
    uint16_t target_y;      // target y
    xpm_image_t image;
    PieceType type;
    bool is_moving;
    bool pressed;
    bool pawn_moved;
} Piece;
```

- **Queue.c:** Contém uma struct relativa à transmissão/receção de informação da serial port.(5%)

```
typedef struct {

  uint8_t front, rear, size;
  unsigned capacity;
  uint8_t * array;

} Queue;
```

- **Board.c:** Este ficheiro trata do tabuleiro como um objeto. Guarda um buffer com os píxeis do desenho do tabuleiro, e outro com os das peças. Além disso, guarda uma matriz em que para cada quadrado do tabuleiro existe um apontador para a peça nessa posição (NULL caso não exista nenhuma). Tem inúmeras funções fulcrais para o projeto. Entre elas estão drawBoardPieces() e o updateBoard().(10%)

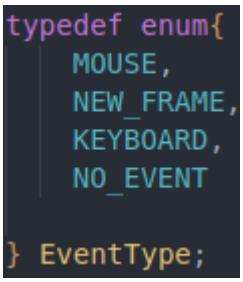
```
typedef struct
{
    char * mem_board;
    char * mem_pieces;
    Piece* board[64];
}Board;
```

- **Cursor.c:** Trata o cursor do rato como um objeto, e tem como principal função guardar as posições do mesmo que são atualizadas no main em resposta a eventos gerados pelo rato, a sua imagem, o seu estado e, no caso de estar a pressionar uma peça, um apontador para a peça pressionada. (15%)

```
typedef struct{
    uint16_t x;
    uint16_t y;
    uint8_t initial_row;
    uint8_t initial_col;
    bool cursorMovingPiece;
    bool pressed;
    xpm_image_t image;
    Piece* pressed_piece;
}Cursor;
```

- **SerialPort.c:** Trata da comunicação entre jogadores, guardando as informações a enviar numa fila (queue.c) até ser possível enviar as mesmas e as recebidas noutra (15%)
- **Main.c:** Recebe interrupções dos dispositivos que produzem eventos e responde a estes, mudando o estado do jogo da forma mais adequada.(15%)

- **Event_Queue.c:** Trata de uma lista de eventos do jogo, de forma a organizar mais o código e poder pôr as mudanças de estado no main.(5%)



- **proj_main_loop():** recebe interrupções, chama os interrupt handlers que geram eventos e chama uma função que consuante o evento e o estado atualiza adequadamente o estado do jogo.

	Afonso Baldo	João Reis	Pedro Gomes	Rui Pires
Main.c	X	X	X	X
Videocard.c	X			X
Mouse.c		X	X	
Keyboard.c		X		X
Piece.c	X		X	X
Queue.c	X		X	
Board.c	X	X	X	X
Cursor.c		X	X	
SerialPort.c	X		X	
Event_Queue.c	X		X	

Detalhes de implementação

Nós usamos na nossa implementação principalmente os conteúdos que foram lecionados nas aulas teóricas, embora alguns deles requereram uma maior aprofundação tais como implementação de state machines para resolver problemas, o Serial Port visto que não resolvemos exercícios nas aulas práticas sobre este conceito e também código orientado a eventos.

Um dos conceitos que não foi lecionado nas aulas, mas foi usado neste projeto foi por exemplo a colisão de objetos. Por exemplo o cursor do rato com a imagem do player1 ou player2 desenhada no ecrã.

Conclusões

Encontramos alguma dificuldade no Serial Port, o que não nos permitiu utilizar interrupts e tivemos de usar polling.

Gostaríamos de implementar a lógica do jogo, por exemplo detenção de checkmate.

Aprendemos a utilizar o Serial Port , state machines e event queue que não tinham ficado aprofundados nos labs.

Appendix

Para dar make e necessário ir para o diretório src (cd proj/src). Temos tambem de ter as duas vm’s ligadas quando se acaba de escolher o nome.