\\192.168.50.233\redirection\up201404293\Desktop\logótipo com cores oficiais.tif

Professor: Pedro Souto

Monitor: Rodolfo

Turma: 4

Grupo: 6

Trabalho realizado por:

* Carolina Centeio Jorge
* Luís Vilar Barbosa

Índice

[Instruções de utilização 3](#_Toc439679422)

[Estado do projecto 5](#_Toc439679423)

[Timer 6](#_Toc439679424)

[Teclado 7](#_Toc439679425)

[Rato 8](#_Toc439679426)

[Placa gráfica 9](#_Toc439679427)

[RTC 11](#_Toc439679428)

[UART 12](#_Toc439679429)

[Organização/estrutura do código 13](#_Toc439679430)

[Game.c 14](#_Toc439679431)

[Graphics.c 15](#_Toc439679432)

[Graphics\_vbe.c 16](#_Toc439679433)

[Graphics\_video\_gr.c 17](#_Toc439679434)

[KBD.c 18](#_Toc439679435)

[Logger.c 19](#_Toc439679436)

[Main.c 20](#_Toc439679437)

[Menu.c 21](#_Toc439679438)

[Mouse.c 22](#_Toc439679439)

[Player.c 23](#_Toc439679440)

[read\_xpm.c 24](#_Toc439679441)

[RTC.c 25](#_Toc439679442)

[sprite.c 26](#_Toc439679443)

[Timer.c 27](#_Toc439679444)

[Gráfico de chamada de funções 28](#_Toc439679445)

[Detalhes de implementação 29](#_Toc439679446)

[Conclusões 31](#_Toc439679447)

# Instruções de utilização

O objetivo do jogo é conseguir o maior número de jogadas possível (valor no canto superior direito). Para isto, o saldo tem de se manter positivo (valor no centro, em baixo, inicialmente a 30, por predefinição).

Cada jogada consiste no lançamento dos dados pelo menos uma vez. O objetivo de cada jogada é conseguir uma pontuação mais próxima de 21 possível (sendo o ideal conseguir 21), não ultrapassando este valor. Esta pontuação resulta da soma dos lançamentos aleatórios dos dados, dois a dois, de cada vez.

Início:

O jogador deve premir a tecla de seta para cima ou para baixo para escolher entre START (se pretender jogar) ou QUIT (se pretender sair do jogo) e, quando a opção pretendida se mostrar realçada, premir ENTER.

Se escolher START:

Seguidamente, deve escolher o modo de jogo que pretende: maioritariamente utilizando o teclado ou rato, premindo, respetivamente a tecla K ou M, como especificado no ecrã.

Se escolher:

Teclado: o jogador usufrui das teclas com seta para manobrar o copo de dados, até conseguir arrastar para o quadrado verde fluorescente.

Rato: o jogador deve mexer a luva branca até ao copo de dados e, em seguida, clicando no botão esquerdo do rato, trazê-lo até ao quadrado verde.

Quando o copo atinge o quadrado verde fluorescente, são lançados os dois dados que somam à pontuação do jogador (apresentada ao centro do ecrã, em cima). O jogador deve agora decidir (se a pontuação não tiver excedido 21) se pretende continuar a jogada ou desistir, premindo Y ou N, respetivamente.

Quando pressionado N, ou a pontuação exceder 21, o saldo será atualizado:

Atualização do saldo:

Se a pontuação final do jogador na jogada for :

21: soma 10 ao saldo;

> 21: subtrai 5;

< 21: soma a pontuação – 18 (ex: se a pontuação for 19, soma 19-18=1);

Se o saldo continuar positivo: o jogador voltará ao menu inicial, onde pode escolher continuar a jogar, pressionando START para uma nova jogada, ou QUIT para sair do jogo.

Se não continuar positivo: são mostrados dois valores no ecrã, sendo o centrado o número de vezes que conseguiu jogar e o de baixo, o saldo com que terminou e o jogo terminará.

# Estado do projecto

|  |  |  |  |
| --- | --- | --- | --- |
| Dispositivo | Para quê: | Interrupções | Varrimento |
| Timer | Suspender o jogo por algum tempo | Y | N |
|  |  |  |  |
| Teclado | Ler as escolhas do utilizador | Y | Y |
|  | Movimentar o copo durante o jogo | N | Y |
|  |  |  |  |
| Rato | Movimentar o copo com dados durante o jogo | Y | N |
|  |  |  |  |
| Placa gráfica | Mostrar o ambiente de jogo | N | Y |
|  |  |  |  |
| RTC | Contar o tempo de jogo e da jogada | N | Y |
|  |  |  |  |
| UART | –––––––––––––––––––––––––––––––––––– | –––––––––– | ––––––––– |
|  |  |  |  |

## Timer

Relativamente ao “timer”, este está a ser utilizado em modo de interrupção e serve para suspender o funcionamento do jogo por um certo tempo (que pode ser menor que um segundo).

As funções que fazem uso das potencialidades do “timer” estão declaradas em “Timer.h” e definidas em “Timer.c”:

“timer\_test\_int” (linha 155).

Estas funções são usadas em:

Game.c: “throwDices” (linha 3) que usa “timer\_test\_int”;

Graphics.c: “final\_result” (linha 385) que usa “timer\_test\_int”.

## Teclado

Relativamente ao teclado, este está a ser utilizado tanto em modo de interrupção como em modo de varrimento (na realidade, existe uma função, que é chamada num ciclo, que implementa um ciclo de interrupções de forma a receber todo o scancode da tecla premida, de forma a ser possível receber scancodes com mais do que 8 bits) e serve para ler as teclas premidas pelo utilizador e com base nas teclas pressionadas movimentar as opções de um menu, movimentar os dados e escolher a opção de abortar ou continuar o jogo no final de cada jogada, ou seja, o teclado é usado para controlo do jogo e, em termos de introdução de texto, selecionar opções.

As funções que fazem uso das potencialidades do teclado estão declaradas em “KBD.h” e definidas em “KBD.c”:

“kbd\_subscribe” (linha 3);

“kbd\_unsubscribe” (linha 16);

“kbd\_read” (linha 30);

“kbd\_read\_char” (linha 52).

Estas funções são usadas em:

Menu.c: “initial” (linha 5);

Graphics.c: ”updateDice” (linha 234);

Game.c: “play” que usa “get\_mode” (linha 95).

## Rato

Relativamente ao rato, este está a ser utilizado em modo de interrupção e serve para ajustar a posição de um cursor no ecrã ou de uma forma mais correcta na memória gráfica; faz uso tanto do sensor de deslocamento como dos botões do rato sendo que os botões servem para selecionar um objecto e a posição serve para deslocar o cursor ao longo do ecrã.

As funções que fazem uso das potencialidades do rato estão declaradas em “Mouse.h” e definidas em “Mouse.c”:

“write\_to\_kbc” (linha 24);

“write\_to\_mouse” (linha 44);

“mouse\_subscribe\_int” (linha 68);

“mouse\_unsubscribe\_int” (linha 80);

“en\_mouse” (linha 92);

“dis\_mouse” (linha 105);

“read\_from\_mouse” (linha 118);

“move\_check” (linha 153).

Estas funções são usadas em:

Game.c: “play\_mouse” que usa “move\_check” (linha 28).

## Placa gráfica

O modo de vídeo utilizado na placa gráfica é o modo 0x105 cuja resolução é 1024x768. Este modo suporta 256 cores das quais só 64 são utilizáveis, porque as cores 64 a 255 têm uma tonalidade muito próxima do preto. Cada cor é codificada em 16 bits. Está a ser utilizado double buffering. Há objectos que se movem (sprites animados), mas não é feita detecção de colisões. Não foi utilizada nenhuma fonte de letra específica, todo o texto é feito à base de XPMs. Não foram utilizadas funções para alterar a palette.

As funções que fazem uso das potencialidades da placa gráfica estão declaradas em “Graphics.h”, “Graphics\_vbe.h” e “Graphics\_video\_gr.h” e definidas em “Graphics.c”, Graphics\_vbe.c” e “Graphics\_video\_gr.c”:

Graphics.c:

“draw\_square” (linha 13);

“draw\_xpm” (linha 35);

“fillBackground” (linha 74);

“transfer\_mem” (linha 93);

Graphics\_vbe.c:

“vbe\_get\_mode\_info” (linha 22);

Graphics\_video\_gr.c:

“vg\_init” (linha 39);

“vg\_exit” (linha 96).

Estas funções são usadas em:

Menu.c: “initial” (linha 5), “mcontinue” (linha 43);

Game.c: “throwDices” (linha 3), “play\_mouse” (linha 16), “play\_kbd” (linha 51), “play” (linha 86);

Mouse.c: “move\_check” (linha 153).

## RTC

Relativamente ao “real-time clock”, este está a ser utilizado em modo de varrimento e serve para mostrar a hora actual, contar o tempo de uma jogada e contar o tempo total de jogo de um jogador, sendo que apenas é lida a data e a hora.

As funções que fazem uso das potencialidades do RTC estão declaradas em “RTC.h” e definidas em “RTC.c”:

“read\_reg” (linha 6);

“wait\_valid\_rtc” (linha 90);

“rtc\_date” (linha 125);

“diff\_time\_seconds” (linha 158).

Estas funções são usadas em:

Game.c: “play\_mouse” (linha 16), “play\_kbd” (linha 51);

Graphics.c: “draw\_mouse\_game” (linha 168), “draw\_kbd\_game” (linha 186).

## UART

Relativamente à porta série, esta ia ser utilizada em modo de varrimento e servir para transmitir as pontuações e o tempo de uma jogada. Seriam utilizados como parâmetros de comunicação 8 bits por caractere, 1 stop bit e 9600 bits por segundo. A informação seria transmitida no final de cada jogada. No entanto, devido a problemas de configuração dos portos TCP na VirtualBox não foi exequível implementar as funções desenvolvidas no projecto.

# Organização/estrutura do código

Nesta secção vamos explicar para que serve cada ficheiro de código em linguagem C:

A Carolina Jorge foi responsável por quase todas as funções que estivessem relacionadas com o ambiente de jogo (desde menus, gráficos, incluindo o design de todas as imagens usadas, e dinâmica e algoritmo de jogo) e o Luís Barbosa foi responsável por corrigir e adaptar ao projeto todas as funções relacionadas com a interação com o hardware para a Carolina conseguir implementar. No entanto, em alguns módulos do projeto ambos os membros do grupo são responsáveis pelo código, pois só conseguiriam ser concretizados em conjunto. Também se deve referir que um antigo professor da disciplina é o responsável pelo código que se encontra no ficheiro “read\_xpm.c” e ainda o uso do site <http://www.office-converter.com/Convert-to-XPM> que ajudou a converter mais facilmente as imagens para um formato xpm, não evitando a alteração necessária a nível de cabeçalho da imagem, para que a imagem e modo 105 as conseguissem reproduzir.

Em termos de contribuição e participação de cada elemento vamos apenas indicar a sua contribuição e participação para a melhoria e construção de código no projeto, sendo que vai ser ignorada a contribuição e participação nos labs. Pode-se considerar que a participação e contribuição de cada estudante são equivalentes, cada estudante teve uma percentagem de participação equivalente à de contribuição.

A documentação “doxygen” das funções cuja documentação não foi fornecida, ou seja, as nossas funções, foi redigida por Luís Barbosa. Esta documentação possui um pequeno lapso devido a falta de informação: na função driver\_receive() não é indicado o tipo do argumento “ANY” nem se há outros argumentos possíveis para este parâmetro.

## Game.c

O ficheiro “Game.c” possui código para “atirar os dados” e fazer uma jogada, gerindo o seu ambiente gráfico. Existe a possibilidade de jogar usando o rato ou usando o teclado, havendo funções específicas para cada caso.

O responsável por este código é Carolina Centeio.

A participação e contribuição de Luís Barbosa é 10% e a de Carolina Centeio é 90%.

O peso relativo deste módulo no projecto é de 5%.

## Graphics.c

O ficheiro “Graphics.c” possui código para “desenhar um quadrado”, desenhar um XPM, preencher o fundo, iniciar o modo gráfico, transferir a memória de trabalho para a VRAM, terminar o modo gráfico, desenhar os diversos menus iniciais, desenhar pontuações menores que 100, desenhar o ambiente inicial de jogo, atualizar as posições dos dados, desenhar os dados, repor o ambiente inicial de jogo, desenhar o menu de opção de continuação ou finalização do jogo, desenhar o cursor, calcular a nova posição do rato, verificar a posição dos dados e verificar a posição do cursor em relação ao copo com os dados.

O responsável por este código é Carolina Centeio.

A participação e contribuição de Luís Barbosa é 20% e a de Carolina Centeio é 80%.

O peso relativo deste módulo no projecto é de 20%.

## Graphics\_vbe.c

O ficheiro “Graphics\_vbe.c” possui o código que permite obter as características da placa de vídeo.

Os responsáveis por este código são ambos os membros do grupo, pois foi feito no lab5 e ligeiramente melhorado por Luís Barbosa para este projecto.

A participação e contribuição de Luís Barbosa é 50% e a de Carolina Centeio é 50%.

O peso relativo deste módulo no projecto é de 5%.

## Graphics\_video\_gr.c

Este ficheiro possui duas estruturas de dados: a estrutura vbe\_mode\_info\_t (que serve para receber e guardar informação sobre um dado modo da placa gráfica) e a estrutura VBEInfoBlock (que serve para receber e guardar as propriedades da placa gráfica).

O ficheiro “Graphics\_video\_gr.c” possui o código que permite iniciar o modo de vídeo, inicializar algumas variáveis relativas à placa gráfica e desligar o modo de vídeo.

Os responsáveis por este código são ambos os membros do grupo, pois foi feito no lab5.

A participação e contribuição de Luís Barbosa é 50% e a de Carolina Centeio é 50%.

O peso relativo deste módulo no projecto é de 10%.

## KBD.c

O ficheiro “KBD.c” possui o código que permite subscrever e des-subscrever as interrupções do KBC, ler por varrimento um scancode (que estará completo se só for composto por 8 bits senão ainda falta a outra metade), ler um scancode completo por interrupção e escolher qual o modo do jogo (rato ou teclado). Também existe uma porção de código assembly associada a este ficheiro que serve para ler um caractere do buffer, no entanto a função que ia fazer uso desta porção de código não foi utilizada, sendo que o código funciona, mas não é executado.

O responsável por este código é Luís Barbosa, sendo que o corrigiu e adaptou ao projecto (a função “get\_mode” foi construída pela Carolina Centeio).

A participação e contribuição de Luís Barbosa é 95% e a de Carolina Centeio é 5%, considerando a participação e contribuição específica para o projecto.

O peso relativo deste módulo no projecto é de 15%.

## Logger.c

O ficheiro “Logger.c” possui código para inicializar o sistema de log e fazer o log de diferentes variáveis: strings, inteiros decimais e inteiros hexadecimais.

O responsável por este código é Carolina Centeio.

A participação e contribuição de Luís Barbosa é 0% e a de Carolina Centeio é 100%.

O peso relativo deste módulo no projecto é de 2%.

## Main.c

O ficheiro “Main.c” contém a função que serve para iniciar o programa, permitir operações de I/O ao programador, iniciar a semente da função de geração de números aleatórios, chamar as funções que iniciam o modo gráfico, chamar a função que inicia o ambiente de jogo e terminar o modo gráfico.

Os responsáveis por este código são ambos os membros do grupo.

A participação e contribuição de Luís Barbosa é 50% e a de Carolina Centeio é 50%.

O peso relativo deste módulo no projecto é de 3%.

## Menu.c

O ficheiro “Menu.c” contém uma função que mostra o menu inicial de forma a ser possível iniciar o jogo ou terminar o programa e uma função que é chamada no final de cada jogada perguntando se deseja continuar ou sair do jogo.

O responsável por este código é Carolina Centeio.

A participação e contribuição de Luís Barbosa é 15% e a de Carolina Centeio é 85%.

O peso relativo deste módulo no projecto é de 5%.

## Mouse.c

O ficheiro “Mouse.c” contém as funções necessárias para interagir com o rato, mais concretamente o KBC e receber a informação produzida pelo rato.

O responsável por este código é Luís Barbosa, sendo que o corrigiu e adaptou ao projecto (a função “move\_check” foi construída pela Carolina Centeio com base em outras funções já corrigidas do rato).

A participação e contribuição de Luís Barbosa é 70% e a de Carolina Centeio é 30%.

O peso relativo deste módulo no projecto é de 8%.

## Player.c

“Player” é uma estrutura de dados que guarda informação sobre um jogador, nomeadamente “Saldo”, “Nome”, “Pontuação”, “Activo” (se o jogador está neste momento a jogar ou não) e “Tempo de jogo” (do jogador).

O ficheiro “Player.c” contém as funções necessárias para criar um jogador, colocá-lo inactivo, adicionar pontuações, verificar se é possível fazer uma aposta e apagar o jogador da memória.

Os responsáveis por este código são ambos os membros do grupo.

A participação e contribuição de Luís Barbosa é 50% e a de Carolina Centeio é 50%.

O peso relativo deste módulo no projecto é de 10%.

## read\_xpm.c

O ficheiro “read\_xpm.c” contém a função necessária para transformar um XPM num mapa de caracteres e saber a sua altura e largura.

O responsável por este código deverá ser João Cardoso, antigo professor de LCOM, no entanto, este código foi ligeiramente alterado por Carolina Centeio.

A participação e contribuição de Luís Barbosa é 0% e a de Carolina Centeio é 3%.

O peso relativo deste módulo no projecto é de 3%.

## RTC.c

O ficheiro “RTC.c” contém as funções necessárias para ler os registos do RTC, esperar por um momento em que haja a certeza que a informação retornada pelo RTC é válida, retornar um apontador para um array de chars que contém a data e calcular a diferença entre duas datas. Também existe uma porção de código assembly associada a este ficheiro que serve para activar e desactivar interrupções.

O responsável por este código é Luís Barbosa.

A participação e contribuição de Luís Barbosa é 100% e a de Carolina Centeio é 0%.

O peso relativo deste módulo no projecto é de 5%.

## sprite.c

“Sprite” é uma estrutura de dados que guarda informação sobre um XPM, nomeadamente posição-base no ecrã, largura, altura, velocidade de deslocamento e um apontador para um conjunto de caracteres.

O ficheiro “sprite.c” contém as funções necessárias para transformar um XPM num mapa de caracteres e saber a sua altura e largura, ou seja, criar um sprite, destruir um sprite, simular o movimento do sprite, desenhar um cursor, verificar colisões, desenhar um sprite, apagar do ecrã um sprite, criar o sprite “Copo com dados” e criar o sprite “Rato”.

O responsável por este código é Carolina Centeio.

A participação e contribuição de Luís Barbosa é 0% e a de Carolina Centeio é 100%.

O peso relativo deste módulo no projecto é de 10%.

## Timer.c

O ficheiro “Timer.c” possui código para subscrever e des-subscrever interrupções e fazer uma contagem de tempo em que mais nenhum código pertencente à aplicação é executado.

O responsável por este código é Luís Barbosa (devo dizer que neste código só efectuei correcções).

A participação e contribuição de Luís Barbosa é 100% e a de Carolina Centeio é 0%.

O peso relativo deste módulo no projecto é de 5%.

## C:\Users\User\Desktop\Capturar.JPGGráfico de chamada de funções

# Detalhes de implementação

Para implementar este jogo foram utilizados os conhecimentos já existentes e obtidos ao longo desta unidade curricular. Também foi necessário pesquisar diversas outras fontes para implementar alguns labs, mas principalmente o UART. Ao longo do desenvolvimento do projeto procurou-se estruturar o código por camadas sendo que se calhar até se criaram mais camadas do que as necessárias. Também nos baseamos em programação orientada a objectos ao criarmos a estrutura de dados “Jogador”. Ao nível da placa gráfica trabalhamos num frame em memória auxiliar e quando este estiver pronto é enviado para a VRAM. Foi utilizado código assembly na implementação do RTC e também no teclado, sendo que o assembly usado no teclado foi corrigido em relação ao feito para o lab3 que não funcionava.

Apesar de termos ponderado realizar este trabalho num modo diferente de 105 e com um formato diferente de xpm, achámos que faria mais sentido um trabalho final onde aplicássemos o que foi desenvolvido e aprendido nos labs. Já que referimos os labs, informámos que um traço estranho que aparece em cima dos Sprites é um problema que já acontecia no lab5 e não foi detectada a sua origem.

O RTC foi implementado apenas em modo de varrimento com o intuito de simplesmente ler a data e com base nessa informação contar tempos de execução.

Ao nível do UART, a implementação do seu funcionamento em modo de varrimento foi desenvolvida com aparente sucesso pelo Luís Barbosa, no entanto, devido a dificuldades de testar em todos as execuções a porta série não foi possível fazer o modo de interrupções e não houve tempo para implementar no projeto em si. (Os portos TCP usados entre as VMs tanto estavam a uso como não estavam e por essa razão nem sempre era possível fazer a comunicação entre máquinas virtuais, não sendo seguro utilizar no projeto.)

Usámos código baseado em eventos e em estados durante a execução do jogo, nomeadamente, ao verificar se ocorreu o evento/estado referente ao copo com os dados estar no centro do ecrã.

O rato foi melhorado em relação ao que foi feito no lab4, no entanto, há uma possível falha que não foi exequível corrigir: no final do ciclo de interrupções é recolhido do buffer do rato mais um pacote de forma a limpar o buffer, mas, se usar o rato enquanto as interrupções são des-subscritas pode suceder de mesmo assim o projeto bloquear.

Apresentam-se máquinas de estado ao longo do trabalho, sempre que aparece um menu.

É de referir que este projecto poderá não funcionar correctamente em todos os PCs, pois o mesmo código foi executado no PC de ambos os membros do grupo e num PC o jogo funcionava bem e no outro o jogo encravava sem razão aparente.

# Conclusões

Fornecer a estrutura de dados relativa às propriedades da placa gráfica para o LAB5.

Creio que seria uma ajuda interessante e, para alguns alunos, importante se fosse feita uma abordagem no início do semestre sobre boas práticas de programação a ser seguidas pelos alunos. Isso poderia ajudar muitos alunos a fazer de uma forma mais assertiva os labs, sem receio de estarem a utilizar práticas erradas ou que compliquem a programação.

A contribuição da Carolina Centeio para o relatório foi de 35 % e a de Luís Barbosa foi de 65%.