\\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](#_Toc439518031)

[Estado do projecto 4](#_Toc439518032)

[Timer 5](#_Toc439518033)

[Teclado 6](#_Toc439518034)

[Rato 7](#_Toc439518035)

[Placa gráfica 8](#_Toc439518036)

[RTC 9](#_Toc439518037)

[UART 10](#_Toc439518038)

[Organização/estrutura do código 11](#_Toc439518039)

[Game 12](#_Toc439518040)

[Graphics 13](#_Toc439518041)

[Graphics\_vbe 14](#_Toc439518042)

[Graphics\_video\_gr 15](#_Toc439518043)

[KBD 16](#_Toc439518044)

[Logger 17](#_Toc439518045)

[Main 18](#_Toc439518046)

[Menu 19](#_Toc439518047)

[Mouse 20](#_Toc439518048)

[Player 21](#_Toc439518049)

[Read\_xpm 22](#_Toc439518050)

[RTC 23](#_Toc439518051)

[Sprite 24](#_Toc439518052)

[Timer 25](#_Toc439518053)

[UART 26](#_Toc439518054)

[Gráfico de chamada de funções 27](#_Toc439518055)

[Detalhes de implementação 28](#_Toc439518056)

[Conclusões 29](#_Toc439518057)

# Instruções de utilização

# 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”

(indicar em que ficheiros e linhas são chamadas funções do “timer”)

## 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 “Keyboard.h” e definidas em “Keyboard.c”

(indicar em que ficheiros e linhas são chamadas funções do teclado)

## 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”

(indicar em que ficheiros e linhas são chamadas funções do rato)

## 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”

(indicar em que ficheiros e linhas são chamadas funções da placa gráfica)

## RTC

Relativamente ao “real-time clock”, este está a ser utilizado em modo de varrimento e serve para contar o tempo de uma jogada e contar o tempo total do jogo, 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”

(indicar em que ficheiros e linhas são chamadas funções do RTC)

## UART

Relativamente à porta série, esta está a ser utilizada em modo de varrimento e serve para transmitir as pontuações e o tempo de uma jogada. Estão a ser utilizados como parâmetros de comunicação 8 bits por caractere, 1 stop bit e 9600 bits por segundo. A informação é transmitida no final de cada jogada.

As funções que fazem uso das potencialidades da porta série estão declaradas em “UART.h” e definidas em “UART.c”

(indicar em que ficheiros e linhas são chamadas funções do UART)

# Organização/estrutura do código

Nesta secção vamos explicar para que serve cada ficheiro de código em linguagem C. A Carolina Centeio foi responsável por quase todas as funções que estivessem relacionadas com o ambiente de jogo e o Luís Barbosa foi responsável por corrigir e adaptar ao projecto todas as funções relacionadas com a interacção com o hardware, no entanto, em alguns módulos do projecto ambos os membros do grupo são responsáveis pelo código, pois foi feito 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”. 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 projecto, 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.

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 e verificar a posição dos 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) e ler um scancode completo por interrupção. Também existe uma porção de código assembly associada a este ficheiro que serve para ler um caractere do buffer.

O responsável por este código é Luís Barbosa, sendo que o corrigiu e adaptou ao projecto.

A participação e contribuição de Luís Barbosa é 100% e a de Carolina Centeio é 0%, considerando a participação e contribuição específica para o projecto.

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

## Logger.c

(se for para colocar)

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.

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

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.

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

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 e retornar um apontador para um array de chars que contém a data. 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 e criar o sprite “Dados”.

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%.

## UART

## Gráfico 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 projecto 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.

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 projecto 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 projecto.)

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 projecto bloquear.

falar do UART e RTC feito

Código baseado em eventos feito

Máquinas de estados feito

# 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.

(Contribuição de cada um para o relatório)