U.PORTO

FEUP FACULDADE DE ENGENHARIA UNIVERSIDADE DO PORTO

Fall Diet

Grupo 10 – Turma 2

Maria Beatriz Russo Lopes dos Santos (up201906888)
Ricardo Filipe da Costa Cabral Ferreira (up201907835)

Índice

Índice de figuras

1. Introdução

O projeto realizado pelo grupo 10, no âmbito da unidade curricular Laboratório de Computadores, consiste num jogo sobre um jogador com excesso de peso. O objetivo é conseguir comer o máximo de comida considerada saudável possível e destruir a que prejudica a sua saúde. Se o jogador falhar em destruir a comida não saudável, irá perder uma vida, tendo no total três. No caso de falhar comer a comida saudável, nada acontece, apenas é perdida uma oportunidade de ganhar pontos.

Ao perder as vidas todas, o jogo terminará e terá a possibilidade de aparecer em primeiro, segundo, ou terceiro lugar na tabela de pontuações, caso as consiga atingir.

2. O Jogo

Figura 1

O jogo é localizado num restaurante. Como tal, ao iniciá-lo irá aparecer a sua entrada, como na figura representativa ao lado. Os botões serão selecionados com o rato. Após carregar no botão “Main Menu” (pendurado na porta), a imagem alterar-se-á para o menu principal.

O menu é constituído por três botões:

- ***Single Player:*** inicia o jogo, apenas com um jogador.
- ***Instructions:*** indica o objetivo do jogo e como o jogar.
- ***Exit:*** irá encerrar o programa.

Figura 2

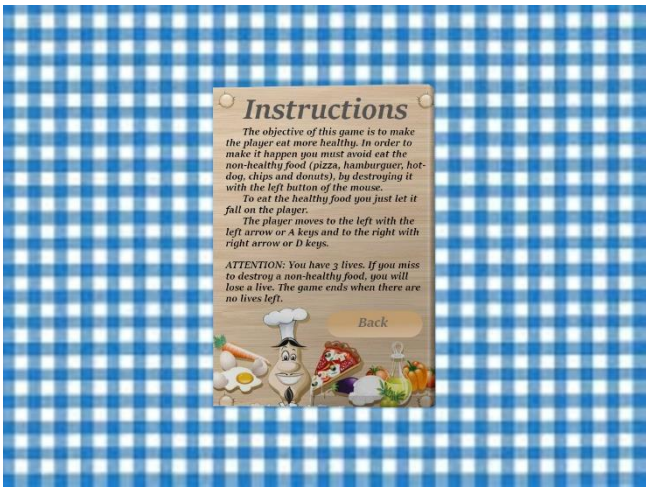

Figura 3

As instruções apenas permitem a sua leitura e existe o botão ***Back*** que retorna ao menu.

Ao selecionar no botão que inicia o jogo, o jogador aparece na cozinha e move se tanto com as setas, como com as teclas A e D, para a esquerda e para a direita, respetivamente. No ecrã é possível visualizar os pontos e as vidas. Após esperar uns segundos, a comida começa a descer com velocidade reduzida, indo gradualmente aumentado até um máximo, dificultando do jogo.

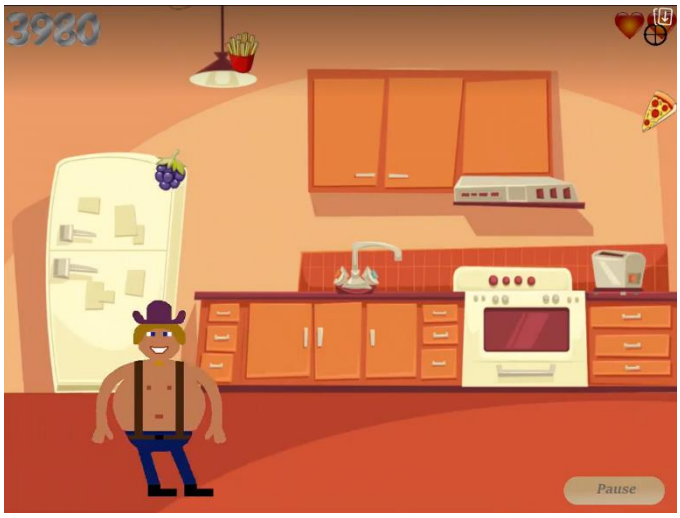

Figura 4

Figura 5

Durante o jogo é possível parar, selecionando com o rato o botão ***Pause***. Neste menu, existem as opções de continuar, rever as instruções e sair do programa. Caso o jogador selecione a opção ***Continue***, o jogo retomará onde parou.

Após terem sido perdidas as três vidas, aparece uma tabela de três classificações com as respetivas datas em que estas foram realizadas. Caso o jogador não atinga nenhuma delas, consegue na mesma visualizar quanto obteve. Neste menu existem as opções de regressar ao menu principal ou de encerrar programa.

Figura 6

Ao encerrar o programa, é mostrada uma mensagem de ***Loading…*** e existe um momento de espera até este encerrar definitivamente. Caso o jogador se encontre no menu principal, esta mensagem aparecerá como na figura 7, caso seja a meio do jogo ou no fim, aparecerá como na figura 8. Em qualquer momento do jogo é possível terminar ou com a tecla ***ESC*** ou com os vários botões ***Exit***.

Figura 8

Figura 7

3. Estado do Projeto

Na tabela que se segue estão presentes os periféricos utilizados no nosso programa, bem como a sua funcionalidade.

Periférico	Utilização	Interrupção
Timer	Atualização a imagem do jogo (*frame rate*)	Sim
Keyboard	Movimento do jogador e saída do jogo	Sim
Mouse	Navegação nos menus e destruidor de comida não saudável	Sim
Video Card	Mostrar o jogo e as suas imagens	Não
Real Time Clock (RTC)	Indicar o dia em que o jogador fez uma certa pontuação e aumentar a velocidade de deslocamento da comida	Sim

3.1 Timer

O periférico *Timer* possui uma função muito importante no jogo. Este atualiza o seu estado, tal como os gráficos e as posições do jogador, rato, comida, etc.

É utilizado através de interrupções, tendo sido usadas as funções ***timer_subscribe_int()***, ***timer_unsubscribe_int()*** e ***timer_int_handler()*** desenvolvidas no Lab2.

Após o tratamento das interrupções, o programa entra numa função chamada ***handler()***, cujos parâmetros são duas *emuns*, uma delas enumera em que parte o jogo se encontra (entrada, menu, single player, pausa…) e a outra enumera que tipo de interrupção se trata. Neste caso, o segundo parâmetro encontrar-se-ia como TIMER. Desta forma, podemos dizer que esta função se trata de uma máquina de estados.

3.2 Keyboard

O teclado é responsável pelo movimento do jogador, ou para encerrar o programa. Para tal, são utilizadas as funções ***kbd_subscribe_int()***, ***kbd_unsubscribe_int()*** e ***kbd_ih()***, desenvolvidas no Lab3. A última função referida tem como objetivo ler o *scancode* gerado pelo teclado. Após a sua leitura, o byte, ou os bytes, são tratados dentro do ciclo *driver receive*.

Tal como no *Timer*, é chamada a função ***handler()***, neste caso como segundo parâmetro KEYBOARD.

3.3 Mouse

O rato funciona como um dos dispositivos mais importantes do jogo, pois permite a sua navegação ao longo do mesmo, sendo utilizado o botão do lado esquerdo para selecionar a opção pretendida ou destruir os alimentos.

Optamos por substituir a imagem do rato por uma colher enquanto o jogo não decorre ou se encontra em pausa e, durante a sua execução este passa a uma mira para acertar na comida, porém, quando o rato é aproximado do botão pausa, este volta a mudar para uma colher e vice-versa.

Foram assim utilizadas as funções ***mouse_subscribe_int()***, ***mouse_unsubscribe_int()*** e ***mouse_ih()***, desenvolvidas no Lab4. Para além destas principais funções, o nosso grupo desenvolveu a sua própria de ***mouse_enable_data_reporting()***.

No ciclo *driver receive* os três pacotes são colocados num *array* e são construídos na função ***packet_make()*** os pacotes do rato para a *struct packet pp*. Após isto, é chamada a função ***handler()*** com o segundo parâmetro MOUSE.

3.4 Video Card

A placa gráfica permite-nos visualizar o jogo e o que está a acontecer neste.

Foram utilizadas as funções ***vg_init()*** e ***vg_color_pixel()*** desenvolvidas no Lab5, tendo sido implementado por nós uma versão da função ***vbe_get_mode_info()***. Permitimos o *double buffering* para um melhor desempenho do jogo com a função ***clear_sec_buff()***.

O modo que decidimos utilizar no nosso projeto foi o modo VBE 0x14C, contendo 1152 pixéis de largura e 864 de altura, 16.777.216 cores, 32 bits por pixel (4 bytes) e em modo direto. Optamos por este modo por ter um tamanho grande e para os alimentos terem mais espaço de queda.

As imagens são do tipo *xpm_image_t* e são carregadas todas ao iniciar o programa com o comando *lcom_run proj*. Optamos por fazer assim, pois, apesar de o programa não iniciar logo, estando a carregar todas as imagens, o jogo ficará mais rápido ao trocar de imagens, uma vez que esta já foi transferida para a respetiva *xpm_image_t.* Resolvemos criar um *array* deste tipo referido para imagens do mesmo género, isto é, imagens de fundo encontram se todas localizadas em *xmp_image_t backgrounds[ ]*, tal como botões em *xpm_image_t buttons[ ]* e assim sucessivamente.

A nível de colisões de imagens, desenvolvemos algumas funções que, dependendo da colisão tratada, verificam se os objetos se encontram sobrepostos (no caso do cursor) ou o final de uma imagem encontra-se na mesma posição do início de outra (no caso de as frutas caírem sobre o jogador).

3.5 Real Time Clock (RTC)

O RTC é utilizado no nosso projeto para aumentar a velocidade, tanto da fruta que desce no ecrã, como do deslocamento do jogador na horizontal, e para ler o dia em que a pontuação foi feita. No caso de esta ser uma de entre as três melhores, irá ficar na tabela de pontuação com o respetivo dia até alguém conseguir pontuar mais. Foram usadas as interrupções periódicas.

4. Organização e estrutura de código

4.1 XPMS

O módulo xpms contém toda a biblioteca de imagens utilizadas no projeto. Maioria destas foram retiradas da internet com licença de utilização, estado maior parte inalteradas. Apenas o jogador e os botões foram inteiramente da nossa autoria.

Módulo realizado inteiramente por Maria Beatriz Santos.

4.2 Utils

Este módulo permite apenas a leitura de um valor para uma variável, invocando a função ***sys_inb().*** Foi desenvolvido durante o Lab2.

4.3 i8042

Neste módulo encontram se as constantes simbólicas necessárias para o projeto usadas sobre o teclado e o rato. Foi desenvolvido no Lab3 e algumas constantes foram adicionadas ao longo do Lab4.

Módulo realizado inteiramente por Ricardo Ferreira.

4.4 Timer

O módulo *Timer* contém as funções relativas ao correspondente periférico, como indicado anteriormente na descrição dos periféricos na secção 3.1. Foi desenvolvido no Lab2. E, tal como referido, tem a funcionalidade de atualizar as imagens no ecrã. De referir que foram usados 60 fps.

Módulo realizado por ambos os membros de forma igual.

4.5 Keyboard

Este módulo contém as funções relativas ao periférico keyboard. Todas as suas funções foram indicadas anteriormente na descrição dos periféricos. Foi desenvolvido no Lab3.

Durante a execução do programa, ao premir a tecla ***ESC*** o jogo é encerrado e nenhuns dados, como a pontuação, são guardados, caso este esteja a meio de um jogo. Para além disso, com as teclas ***A*** e a seta para a esquerda o jogador move-se para a esquerda e com as teclas ***D*** e a seta para a direita, o jogador move-se para a direita.

Módulo realizado por ambos os membros de forma igual.

4.6 Mouse

O módulo *mouse* contém as funções relativas ao periférico rato, como indicado anteriormente na descrição dos periféricos e as suas respetivas funções. De destacar a função fornecida pela LCF, ***mouse_enable_data_reporting(),*** cuja versão presente no trabalho foi realizada por nós. Este módulo foi desenvolvido no Lab4.

Módulo realizado por ambos os membros de forma igual.

4.7 RTC

Este módulo, como referido no ponto 3.5. permite o aumento da velocidade de queda dos alimentos e a leitura da data em que o jogo está a ser decorrido. Caso a pontuação feita esteja entre as três melhores, a data ficará gravada na tabela de pontuação e no ficheiro onde a vai buscar para imprimir na tabela.

Foram assim utilizadas as principais funções ***rtc_subscribe_int()***, ***rtc_unsubscribe_int()*** e ***rtc_ih()***. Esta última permite a leitura da data, uma vez que chama três funções, cada uma retornando o dia, mês e ano.

Para além disso, foi desenvolvida uma função que permite habilitar ou desabilitar (existe um parâmetro que se for *true* a função funcionará como *enable,* caso contrário funcionará como disable) as interrupções para ler a data.

Módulo realizado por ambos os membros de forma igual.

4.8 Video

Este módulo contém todas as funções que interagem diretamente com a parte gráfica, tendo sido importado do código das aulas práticas do Lab5. Foi indicado na secção 3.4, na explicação do periférico vídeo card, as principais funções que incluía. Para além das referidas, existem também a função ***refresh()*** que atualiza a memória virtual *video_mem.* Para além disso, para pintar os pixéis é chamada a função ***vg_color_pixel()*** que pinta pixel a pixel. De destacar a função implementada por nós, também fornecida pela LCF, ***vbe_get_mode_info().***

Módulo realizado por ambos os membros de forma igual.

4.9 Sprite

O módulo *sprite* contém duas *structs*, uma chamada *Sprite* e outra chamada *Food*. A primeira referida é constituída pelas características da imagem, como a posição inicial, altura, largura, velocidade. Todas as imagens são *Sprite*, com à exceção da comida. A segunda *struct* engloba a *Sprite*, mas constitui mais uma característica, se esta é classificada como saudável ou não.

Para além do referido, o módulo é constituído por funções que criam e destoem *sprites* ou *foods*, tanto como as desenha no ecrã, chamando a função ***vg_color_pixel()*** do módulo *video.*

Por fim, neste módulo está presente a função que permite o movimento das imagens sem apagar as debaixo, atualizando o ecrã ao desenhá-las de novo.

Módulo realizado maioritariamente por Maria Beatriz Santos.

4.10 Mainmenu

O módulo consiste em funções que mostram os diferentes tipos de menus e o cursor no ecrã. Cada função indica as coordenadas dos botões nos menus e desenha-os.

Módulo realizado maioritariamente por Ricardo Ferreira.

4.11 Initiate

Este módulo consiste em funções necessárias quando se inicia o programa, tal como as subscrições dos periférico, funções de *enable*, ***vg_init()***, o *load* de todos os xpms e a criação de sprites, e quando se vai encerrar, sendo estas as funções de *unsubscribe*, de *disable,* ***vg_exit()*** e a destruição das Sprites.

Módulo realizado maioritariamente por Maria Beatriz Santos.

4.12 Highscores

O módulo *highscores* permite a impressão da pontuação do jogo ao longo do mesmo. Para além disso, a função ***deal_with_point()*** vai retirar do ficheiro *HighScores.txt* as pontuações previamente obtidas no jogo. Caso a pontuação feita pelo jogador seja superior às existentes no ficheiro, a mais pequena é eliminada e a atual entra. Por fim, a terceira e última função deste módulo imprime a tabela de pontuações e as datas.

Módulo realizado maioritariamente por Ricardo Ferreira.

4.13 Game

Este módulo é o módulo principal do projeto, pois é lá que o jogo é desenvolvido. Foi criada uma função (***handler()***) que recebe uma máquina de estados que indica o que fazer em cada parte do jogo e quando mudar para a próxima. Estes estados consistem na ENTRACE (entrada do restaurante), MENU (onde se encontram as opções), SINGLEPLAYER (estado em que o jogo decorre), INSTRUCTIONS (é mostrado as instruções do jogo), PAUSE (para quando o jogador pressiona sobre o botão de pausa), GAMEOVER (quando o jogador fica sem vidas) e, por fim, EXIT (encerra o jogo).

Para além da enumeração dos vários estados, esta função recebe um *emun* chamado *type_of_interrupt* que permite, em cada etapa do jogo, indicar ao programa que tipo de interrupção ocorreu, como por exemplo, quando o jogo se encontrar em MENU, o rato irá receber movimento pela parte do utilizador, entrando na interrupção MOUSE que lhe atualizará a posição e se algum evento por parte deste foi ocorrido.

Esta é a principal função, tanto do projeto, como deste módulo. Contudo, existem outras que nos auxiliam o desenvolver do projeto, como a escolha da comida que cai ao longo do programa. Esta é posta num *array* do tipo *xpm_image_t,* como referido anteriormente. Na função referida, o programa irá buscar de forma aleatória com a funcionalidade ***rand()*** um alimento desse *array,* permitindo, assim, a queda de alimentos aleatórios no decorrer do jogo. Para além disso, é neste módulo que as colisões são tratadas, como a colisão do cursor com um botão (fazendo com que este se destaque, mudando de cor) ou a colisão dos alimentos com o jogador.

Módulo realizado por ambos membros de forma igual.

4.14 Proj

Este último módulo é a base de todos os outros. Tem apenas uma função e é nesta que o ciclo *driver receive* é chamado, bem como as funções de inicialização e encerramento do programa. Nas páginas seguintes é possível visualizar o gráfico gerado por esta função, inicialmente repartido em duas partes e por fim o geral, uma vez que, pelo tamanho do mesmo, a visibilidade do completo não é a melhor.

Módulo realizado por ambos membros de forma igual.

4.15 Gráfico

Como é possível visualizar no gráfico abaixo, a função ***proj_main_loop()***, onde se situa o ciclo *driver receive*, é a que liga todas as outras.

Neste primeiro gráfico encontram-se as funções de subscrição de interrupções, de enable, tal como as que fazem o inverso das referidas e a criação e destruição de sprites.

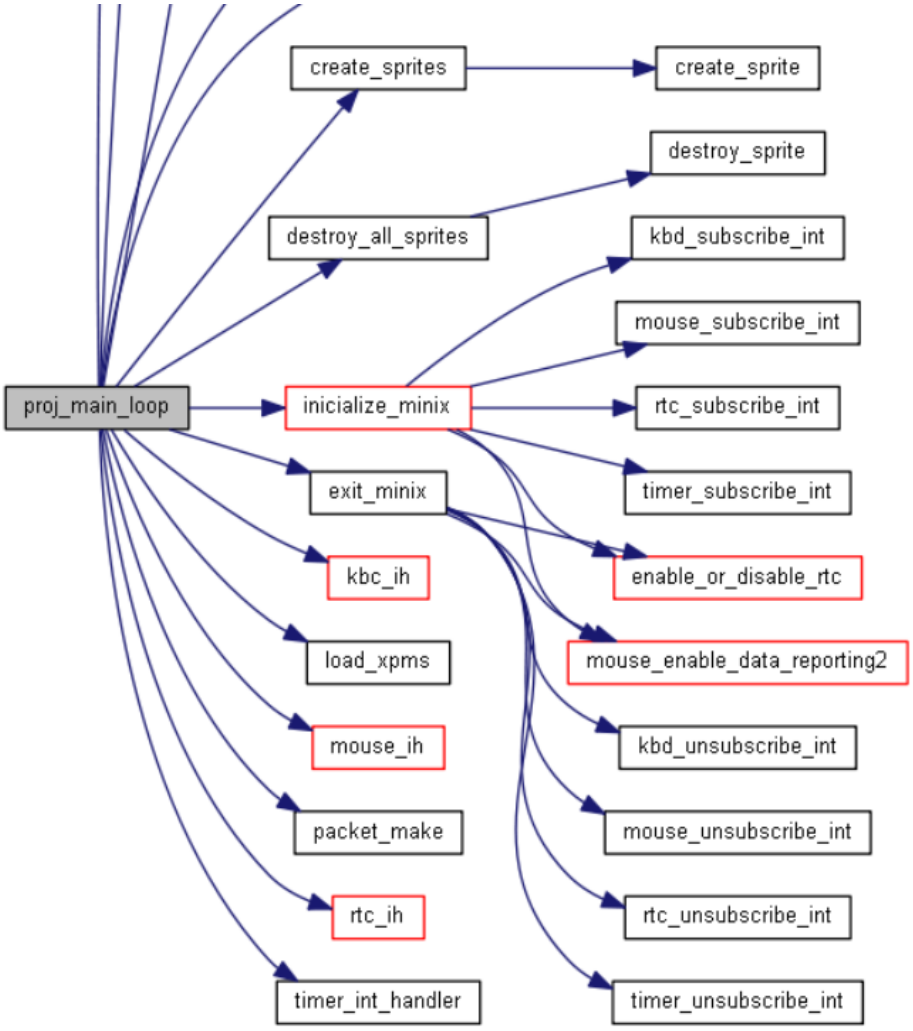

Figura 9

Na página seguinte encontra-se a restante parte do gráfico. De notar que a função ***handler()*** invoca todas as outras, sendo assim possível o funcionamento do programa.

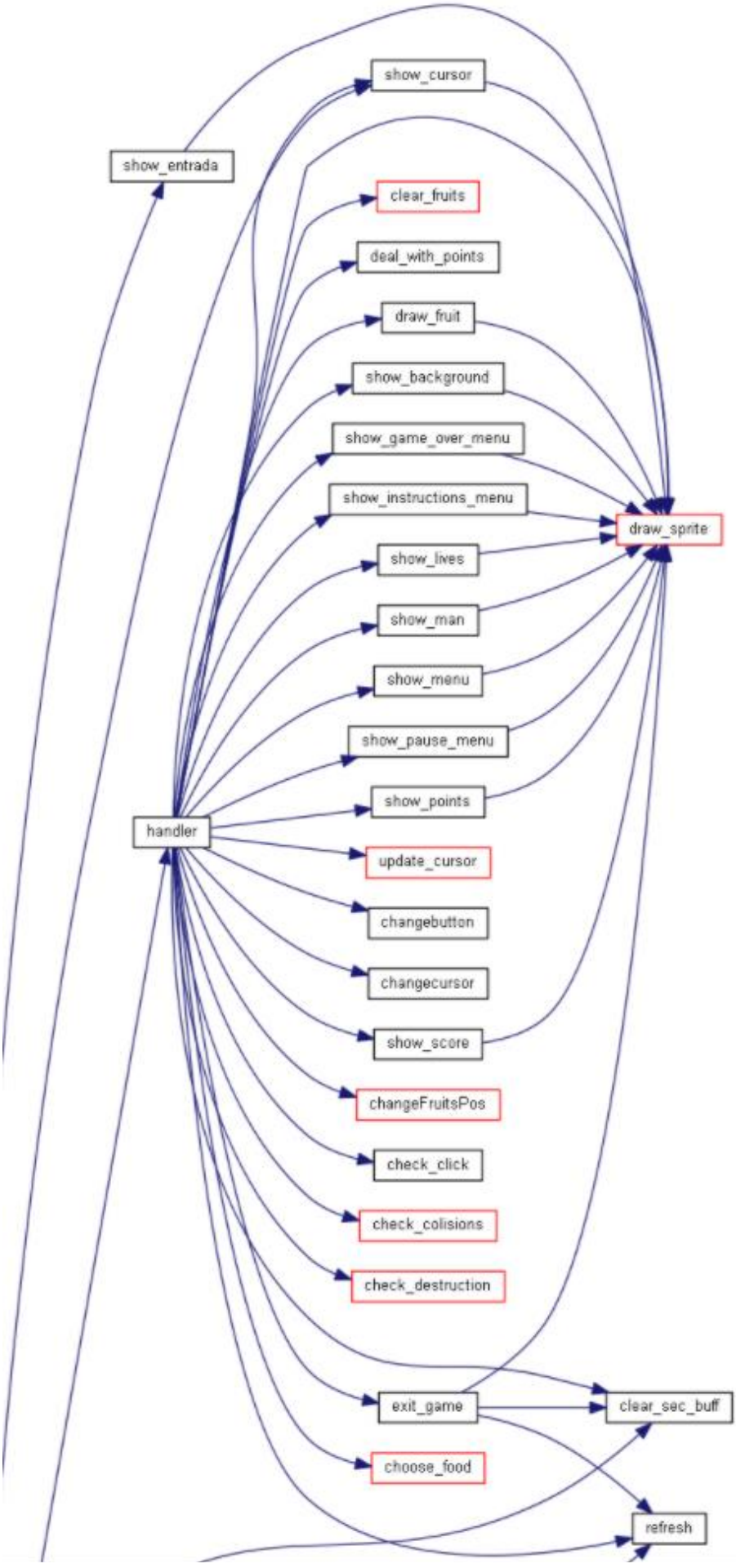
show_cursor
show_entrada
clear_fruits
deal_with_points
draw_fruit
show_background
show_game_over_menu
show_instructions_menu
draw_sprite
show_lives
show_man
show_menu
show_pause_menu
show_points
handler
update_cursor
changebutton
changecursor
show_score
changeFruitsPos
check_click
check_colisions
check_destruction
exit_game
clear_sec_buff
choose_food
refresh

Figura 10

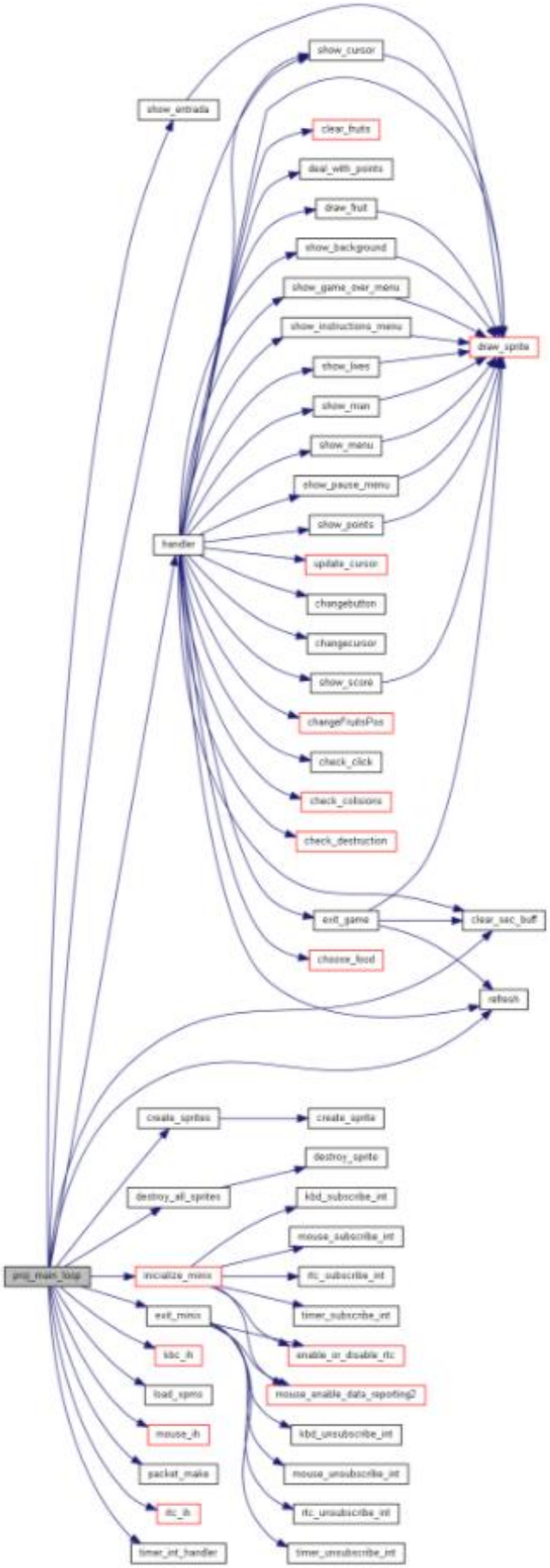
show_cursor
show_entrada
clear_fruits
deal_with_points
draw_fruit
show_background
show_game_over_menu
show_instructions_menu
draw_sprite
show_lives
show_man
show_menu
show_pause_menu
show_points
handler
update_cursor
changebutton
changecursor
show_score
changeFruitsPos
check_click
check_colisions
check_destruction
exit_game
clear_sec_buff
choose_food
refresh
create_sprites
create_sprite
destroy_sprite
destroy_all_sprites
kbd_subscribe_int
mouse_subscribe_int
proj_main_loop
incialize_minix
rtc_subscribe_int
timer_subscribe_int
exit_minix
kbc_ih
enable_or_disable_rtc
mouse_enable_data_reporting2
load_xpms
kbd_unsubscribe_int
mouse_ih
mouse_unsubscribe_int
packet_make
rtc_unsubscribe_int
rtc_ih
timer_int_handler
timer_unsubscribe_int

Figura 11

5. Detalhes de implementação

Dos tópicos abordados em aula, apenas a UART não foi implementada no nosso código.

Achamos relevante mencionar as colisões da personagem com os alimentos como algo que tivemos de descobrir por nós próprios. O programa que usámos para editar as imagens, o Gimp, apenas nos permitia guardá-las de forma retangular. Como a cabeça do jogador é mais alta do que os ombros, ao apenas comparar a altura inicial deste com a altura final do alimento, caso este caísse mais para o lado dos ombros, ocorria uma colisão mesmo não tendo tocado no jogador. Para resolver este problema tivemos de inspecionar a imagem no Gimp e verificar em que pixel começavam os ombros e a cabeça e fazer condições que nos permitissem que a fruta realmente tocasse no jogador.

Para além disso, o tratamento de ficheiros é algo abordado pelo nosso programa. Não nos foram dadas indicações ao longo da cadeira de como as fazer, por isso foi algo que tivemos de pesquisar e aprender antes de poder implementar no nosso código.

Gostávamos também de realçar a função ***handler()*** que é a maquina de estados do jogo, cuja ideia foi abordada nas aulas sobre o Lab4, que facilitou muito o processamento deste e a organização do código. Desta forma, por exemplo, para parar o jogo apenas invocamos o estado PAUSE e, após o jogador desejar retomar, ao selecionar a opção *continue*, a máquina regressa ao estado SINGLEPLAYER com todas as posições das

sprites, pontuação e vidas, iguais às anteriores. Este apenas foi um mero exemplo de uma das funcionalidades em que a máquina nos facilitou no projeto.

O RTC foi algo apenas visto nas aulas teóricas, não nos tendo sido dada a oportunidade de podermos aprofundar mais nas práticas, nem de fazermos testes, como nos Labs desenvolvidos ao longo da cadeira. Por isso, consideramos que foi algo trabalhoso e que necessitou de algum tempo da nossa parte para conseguirmos perceber e implementar bem.

De acrescentar que a impressão da imagem quando as coordenadas são negativas também foi algo que deu algum trabalho e tentativas até o conseguirmos realizar. Isto acontece quando a comida esta no início da sua queda.

Por fim, achamos relevante acrescentar o aumento da velocidade da comida ao longo do tempo, até atingir um máximo.

6. Conclusão

Concordamos que foi um projeto trabalhoso tanto a nível dos Labs como apenas na concentração do projeto, talvez das cadeiras mais complicadas e que consumia mais tempo. Contudo foi um desafio interessante e conseguiríamos acrescentar mais implementações, como o UART, se possuíssemos mais tempo.

Uns aspetos que achamos que conseguiriam ser melhorados, para um melhor aproveitamento da cadeira, são os guiões dos Labs. São, de facto, um pouco confusos e poderiam ser melhor explícitos se feitos de outra forma, principalmente inicialmente, pois estamos a trabalhar com algo novo e o desafio é lançado de uma forma que choca os alunos, em vez se ser gradual.

Para além disso, não possuirmos a nota nem do primeiro teste nem do segundo quando a época de exames já começou é um pouco desmotivante, pois ficamos na incerteza de como está a correr a avaliação de cada um tendo estas sido realizadas há já alguns meses, para além de que é pedido aos alunos que cumpram com os prazos e merecem obter o devido reconhecimento.

Por fim, gostaríamos de agradecer ao nosso professor das aulas práticas, o professor Nuno Paulino, pela ajuda ao longo do semestre e por se encontrar sempre disponível para nos ajudar e tirar dúvidas.