U.PORTO

FEUP FACULDADE DE ENGENHARIA
UNIVERSIDADE DO PORTO

AVOID IT

Relatório do Projeto

Laboratório de Computadores

LCOM1718-T4G02

MIEIC 2º Ano

André Esteves
up201606673@fe.up.pt

Bruno Sousa
up201604145@fe.up.pt

Índice

1. Instruções de Utilização

1.1. Menu Inicial

Quando o jogo é iniciado é apresentado ao utilizador o menu inicial (fig.1 - Menu Inicial), sendo utilizado o rato para circular entre os diferentes menus. Quando o rato está dentro do "retângulo" já definido, este é visível e a sprite do rato é alterada, sendo assim percetível a possibilidade de clicar no rato e ir para outro menu, como mostra na imagem abaixo.

Existem neste menu 3 opções: "Start Game", "High Score" e "Quit Game".

Fig. 1 – Menu Inicial

1.2. Menu das Pontuações

Quando selecionado o menu do High Score são apresentadas ao utilizador as 10 melhores pontuações. Para cada pontuação é apresentado: a posiçao [de 1º lugar a 10º lugar]; o nome do jogador; a sua pontuação e a data (dd mmm yy). Ver figura abaixo.

Existe uma opção "Back to Main Menu" para voltar ao menu inicial, caso o utilizador clique nela.

Fig. 2 – Menu das Pontuações

1.3. Gameplay & Start Game

Quando iniciado o jogo o utilizador deve movimentar o seu personagem com as teclas A (movimenta o jogador para a esquerda) e D (para a direita). Para além disso, o utilizador pode usar o rato. Para isso, o utilizador deve clicar no botão esquerdo do mesmo e movimentar o rato. Pode também usar os dois tipos de movimentação ao mesmo tempo. A cada segundo é atualizado no canto inferior direito, o tempo e a respetiva pontuação (fig. 3 e fig. 4). Os obstáculos são gerados de forma aleatória de 5 em 5 segundos, e vão ficando mais rápido a movimentarem-se consoante o tempo de jogo progride. Com o uso do Real Time Clock (RTC) o fundo do jogo é diferente. Quando o jogo é iniciado, dependendo da hora, o jogo tem o tema diurno (entre as 8 e as 20) ou o tema noturno.

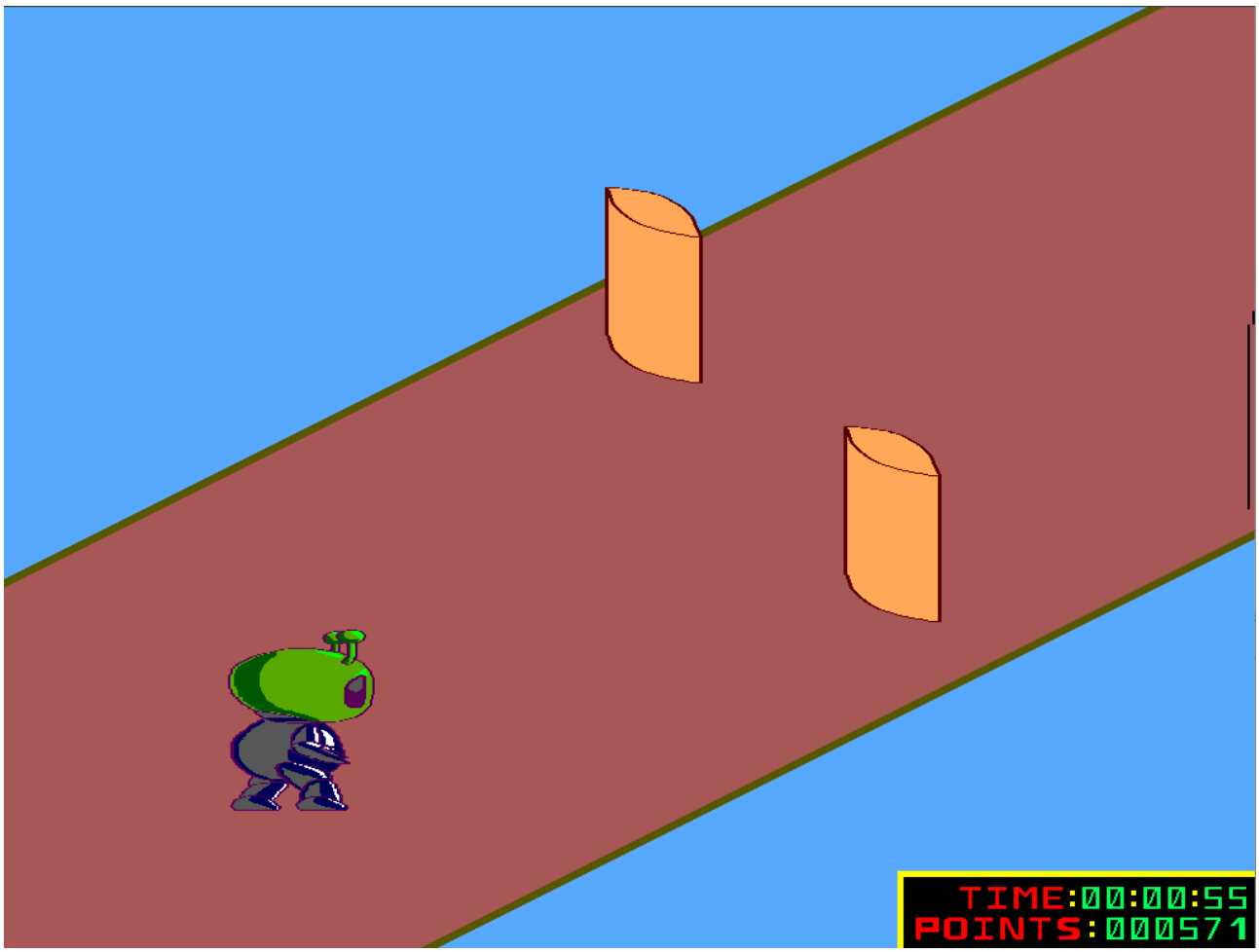


Fig. 3 – Ambiente de jogo (diurno)

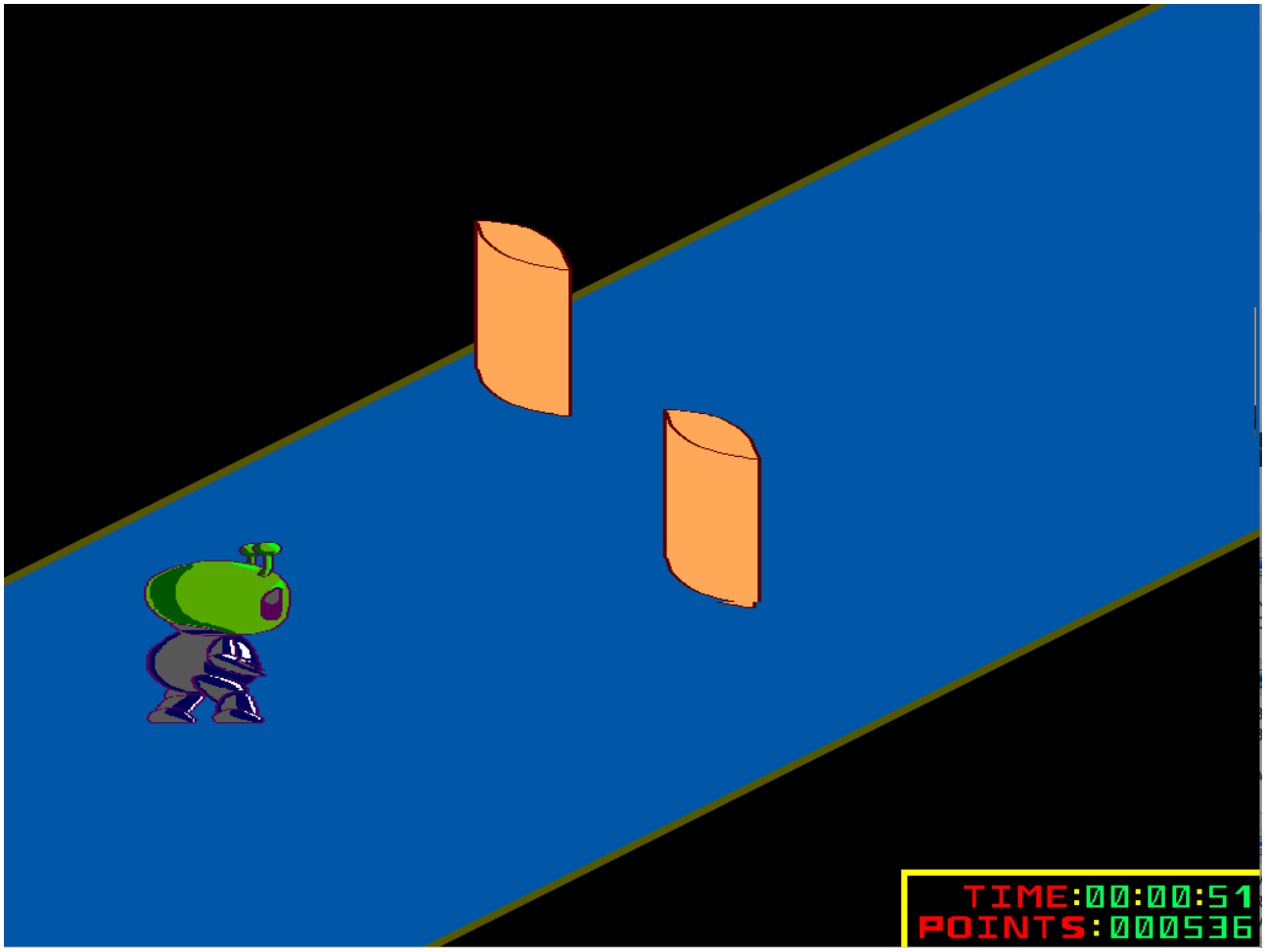


Fig. 4 – Ambiente de jogo (noturno)

1.4. Menu de fim de jogo

Quando o jogador colide com algum obstáculo, perde o jogo, e é-lhe mostrado o menu de fim de jogo (fig. 5 e fig.6). O mesmo se aplica se o utilizador precisar de terminar o jogo antes de perder basta carregar na tecla escape (ESC).

Antes de ser apresentado o menu de fim de jogo é verificada a sua pontuação final.

1.4.1 Pontuação mínima desejável não alcançada

Caso a pontuação não esteja entre as 10 melhores apenas é apresentado o menu com 2 opções, "Back to Main Menu" e "Quit Game" (fig. 5).

Fig. 5 – Menu de fim de jogo (1)

1.4.2 Pontuação mínima desejável alcançada

Caso a pontuação esteja entre no top10 o utilizador pode escrever o seu nome para ser posteriormente apresentado no menu High Score, sendo para isso apensa necessário clicar no retângulo à frente de "Your Name:", escrever o nome[apenas aceites letras] e clicar Enter. Caso não deseje gravar o seu nome ou já o tenha feito basta clicar em Quit Game ou em Back to Main Menu (fig.6).

Fig. 6 – Menu de fim de jogo (2)

Fig. 7 – Menu de fim de jogo (3)

2. Estado do Projeto

Não foi implementada a porta de série.

Periféricos	*Função*	*Interrupção*
Timer	Para gerar objetos de x em x tempo, para mostrar um contador e controlar a framerate	Sim
Teclado	Movimento da personagem e pedido do nome para o High Score	Sim
Rato	Movimentação nos diferentes menus, interação com as diferentes opções e jogabilidade com a personagem	Sim
Placa Gráfica	Exibição dos gráficos	Não
RTC	Para guardar a data dos melhores scores Para alterar o fundo do jogo dependendo da hora	Não

2.1. Timer

O timer é utilizado apenas durante o jogo, para:

- Apresentar um contador no ecrã com o tempo desde que o jogador iniciou o jogo.

- Gerar os obstáculos de x em x tempo.

- Para controlar o tempo entre cada atualização do ecrã (quando esta não é necessária ser atualizada por movimento do jogador).

Está implementado no ficheiro timer.c (funcões *timer_subscribe_int(), timer_unsubscribe_int()* e *timer_int_handler()*) e é utilizado na função startmenu() o subscribe e unsubscribe de interrupts e utilizado na função *game()* o interrupt handler.

2.2. Teclado

O teclado é utilizado durante o jogo para mover o personagem do jogador (com as teclas A e D) e sair do jogo; no menu GameOver para o jogador escrever o seu nome.

É utilizado tanto para controlo de jogo, como para input de texto. Utiliza interrupts. Está implementado no ficheiro kbd.c (funcões *kbd_subscribe_int(), kbd_unsubscribe_int()* e *kbd_int_handler()*) e é utilizado na função startmenu() o subscribe e unsubscribe de interrupts e utilizado na função *game()* e na função *menu_handler()* o interrupt handler.

2.3. Rato

O rato é utilizado durante o jogo e em todos os menus. Durante o jogo, serve para movimentar o jogador (clicando no botão e movendo para um dos lados). Dentro dos menus, serve para movimentar a sprite do mouse e clicar nas opções desejadas navegar entre menus.

São utilizados tanto os movimentos do rato como os botões tanto no menu como no jogo. Utiliza interrupts e stream mode.

Está implementado no ficheiro mouse.c (funcões *mouse_subscribe_int, mouse_unsubscribe_int(), mouse_int_handler(), mouse_stream_mode()* e *mouse_disable_data_report()*) e é utilizado na função *startmenu()* o subscribe e unsubscribe de interrupts, para além de stream_mode(função que faz set de stream mode e enable data report) e disable_data_report e utilizado nessa função e também na função *game()* e na função *menu_handler()* o interrupt handler.

2.4. Placa Gráfica

A placa gráfica é utilizada em todo o programa para mostrar tanto menus como o jogo.

É utilizado o modo 0x105 com resolução 1024 x 768 com 256 cores, estando estas indexadas (sem alteração da palete). A vram é atualizada tanto periodicamente (no jogo, com uma frequência fixa) como por evento (nos menu, quando o utilizador move o rato e no jogo quando move o personagem). É utilizado double buffering. São utilizadas as funções VBE 0x1 e 0x2. São utilizadas “sprites” (implementadas em sprite.c) com deteção de colisão.

Está implementada em video_gr.c e em vbe.c (este com as funções vbe). As funções principais são vg_init(inicia a placa grafica num determinado modo), vg_exit(retorna para text mode), vbe_get_mode_info (obtém informaçoes sobre o modo que queremos utilizar), video_draw_xpm (que lê e desenha um xpm), vg_pixel_color(pinta um determinado pixel duma determinada cor) e copy_to_video_mem (copia do segundo buffer para o buffer associado à VRAM o que o programa quer mostrar ("double buffering")). É utilizado em praticamente todo o código.

2.5. RTC

O RTC é utilizado para obter a data para guardar juntamente com os melhores scores e para obter a hora para determinar que tema utilizar.

Não utiliza interrupts, sendo apenas lida a data/hora escrevendo o address location correspondente para o adress register e lendo do data register.

Está implementado em rtc.c (funções *rtc_get_hour()* e *rtc_get_date()*), sendo ambas as funções apenas utilizadas na função *menu_handler()*.

3. Organização e Estrutura de código

3.1. game.c

Neste módulo estão implementadas as funções correspondentes ao jogo em si. Está implementada a função de teste de colisões entre a personagem e os obstáculos e está implementada a função principal do jogo, onde recebe os interrupts do timer/keyboard/mouse para o controlo do jogo. Esta ultima função é responsável pelo movimento do jogador, pela geração aleatória de obstáculos e pela movimentação destes e pelo aumento da dificuldade consoante o tempo aumenta.

Algumas das funções foram implementadas por um dos membros e outro pelo outro membro, não existiu um só membro como responsável.

A principal estrutura de dados utilizada neste módulo é a sprite, tanto para o personagem do jogador como para obstáculos.

Peso do módulo no projeto: 25%

3.2. high_score.c

Lê, gere e guarda os 10 melhores scores dum ficheiro txt. Os scores são guardados num array dum estrutura de dados [High_Score] ordenados por score (decrescente).

A principal estrutura de dados utilizada neste módulo é a High_Score que contém um unsigned long com o score, uma string de 20 caracteres com o nome do jogador e uma string de 20 caracteres com a data do high score.

Membro do grupo responsável: Bruno Sousa

Peso do módulo no projeto: 4%

3.3. kbd.c

Neste módulo estão implementadas as funções correspondentes ao teclado, tais como as funções de subscribe, unsusbscribe e handler das interrupções do teclado. Funções efetuadas pelos dois elementos aquando elaborado o laboratório 3.
Peso do módulo no projeto: 5%

3.4. linked_list.c

Implementação de uma estrutura de dados, uma double linked list. Contém um apontador para o inicio e fim da lista e cada elemento contém um apontador para o próximo elemento e o elemento anterior. Foram criadas as seguintes funções sobre ela: create_list, delete_list, push_back, pop_front. Desenvolvida para permitir facilmente retirar os obstáculos fora do ecrã, para libertar a sua memoria e assim permitir que o jogo seja teoricamente infinito.
Membro do grupo responsável: Bruno Sousa
Peso do módulo no projeto: 4%

3.5. menu.c

Neste módulo está implementado a máquina de estados, a qual varia com o estado atual (actual_State) e evento atual (actual_Event) o que faz com o jogo circule entre os diferentes menus, desenhando o que for necessário dependendo do estado e evento atuais. A função inicial *start_menu()* que tem um ciclo do while que recebe os interrupts do rato (o packet) e chama as funçoes que alteram a state_machine para fazerem as alterações necessários.
Funções efetuadas pelos dois elementos.
Peso do módulo no projeto: 15%

3.6. mouse.c

Neste módulo estão implementadas as funções correspondentes ao rato, tais como o subscribe, unsubscribe e handler das interrupções do rato. Para além disso possui também as funções para mudar o rato para stream mode e dar enable/disable de data reporting.

O seu respetivo header file possui algumas macros.

Funções efetuadas pelos dois elementos aquando elaborado o laboratório 4.

Peso do módulo no projeto: 7%

3.7. mouse_sprite.c

Parecida com sprite.c, mas inclui uma funcionalidade extra, que permite trocar entre xpm de ratos diferentes dependendo do local em que está a ser apresentada.

Membro do grupo responsável: André Esteves

Peso do módulo no projeto: 3%

3.8. proj.c

Este módulo apenas possui a função main no qual é chamada as funções *sef_startup()* e *start_menu()* , função inicial da state_machine.

Funções efetuadas pelos dois elementos.

Peso do módulo no projeto: 2%

3.9. rtc.c

Implementação de funções para obter a data e hora usando o RTC.

O seu respetivo header file possui algumas macros.

Funções efetuadas pelos dois elementos.

Peso do módulo no projeto: 4%

3.10. sprite.c

Estrutura de dados sprite, implementada para guardar informações do personagem do jogador e dos obstáculos. Responsável por guardar as suas xpms, velocidades, posições e o que estava na posição onde estão antes de se moverem para lá. Estão implementadas neste modulo funções que permite mostrar as sprites, move-las (guardando e mostrando depois o que lá estava antes).

Membro do grupo responsável: Bruno Sousa

Peso do módulo no projeto: 4%

3.11. timer.c

Neste módulo estão implementadas as funções correspondentes ao timer, tal como a *timer_subscribe_int()* , *timer_unsusbscribe_int()* e *timer_int_hanlder()*. O seu respetivo header file possui algumas macros.

Funções efetuadas pelos dois elementos aquando elaborado o laboratório 2.

Peso do módulo no projeto: 4%

3.12. vbe.c

Implementação da função VBE get mode info.

Funções efetuadas pelos dois elementos aquando elaborado o laboratório 5, da placa gráfica.

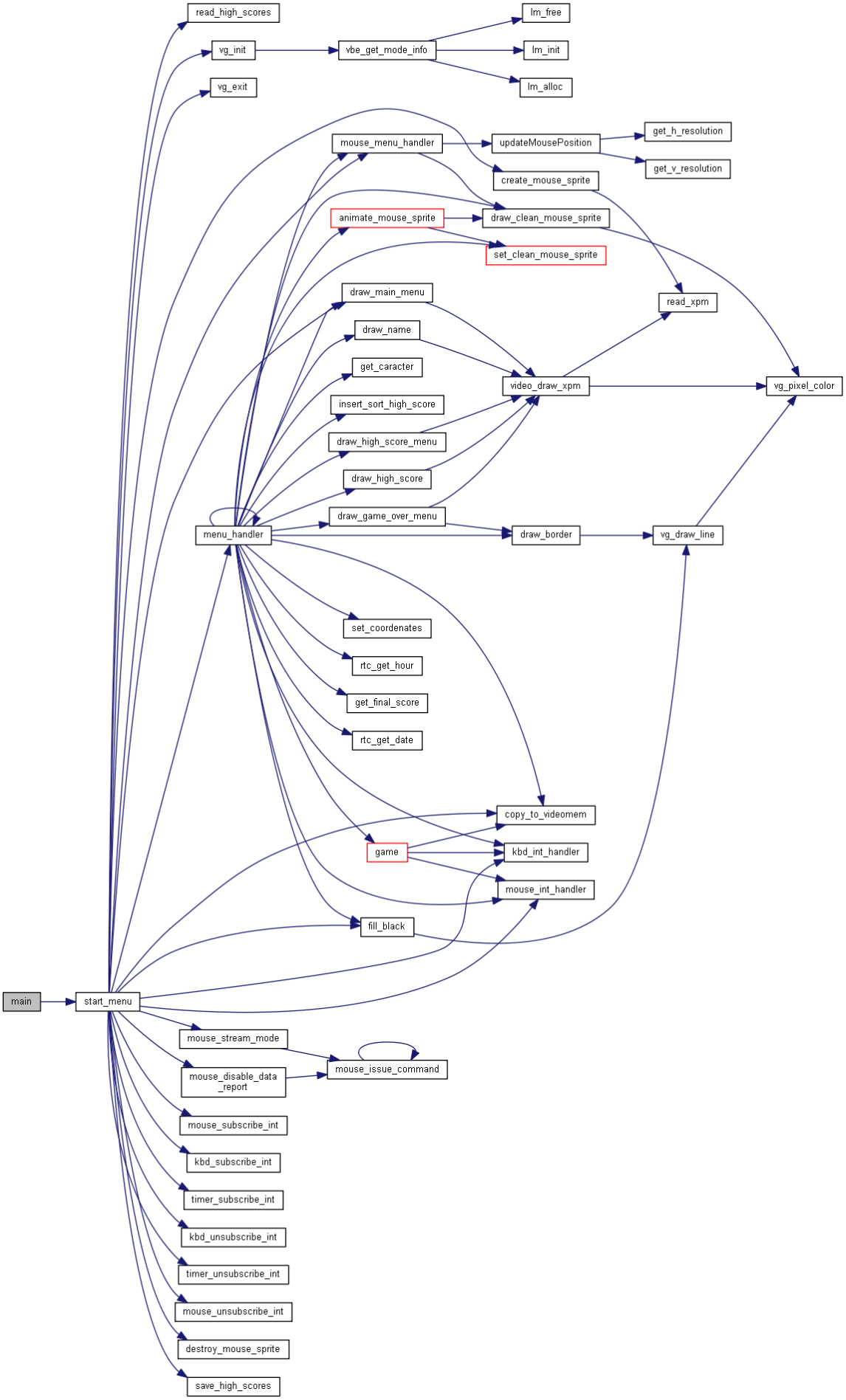
Peso do módulo no projeto: 3%

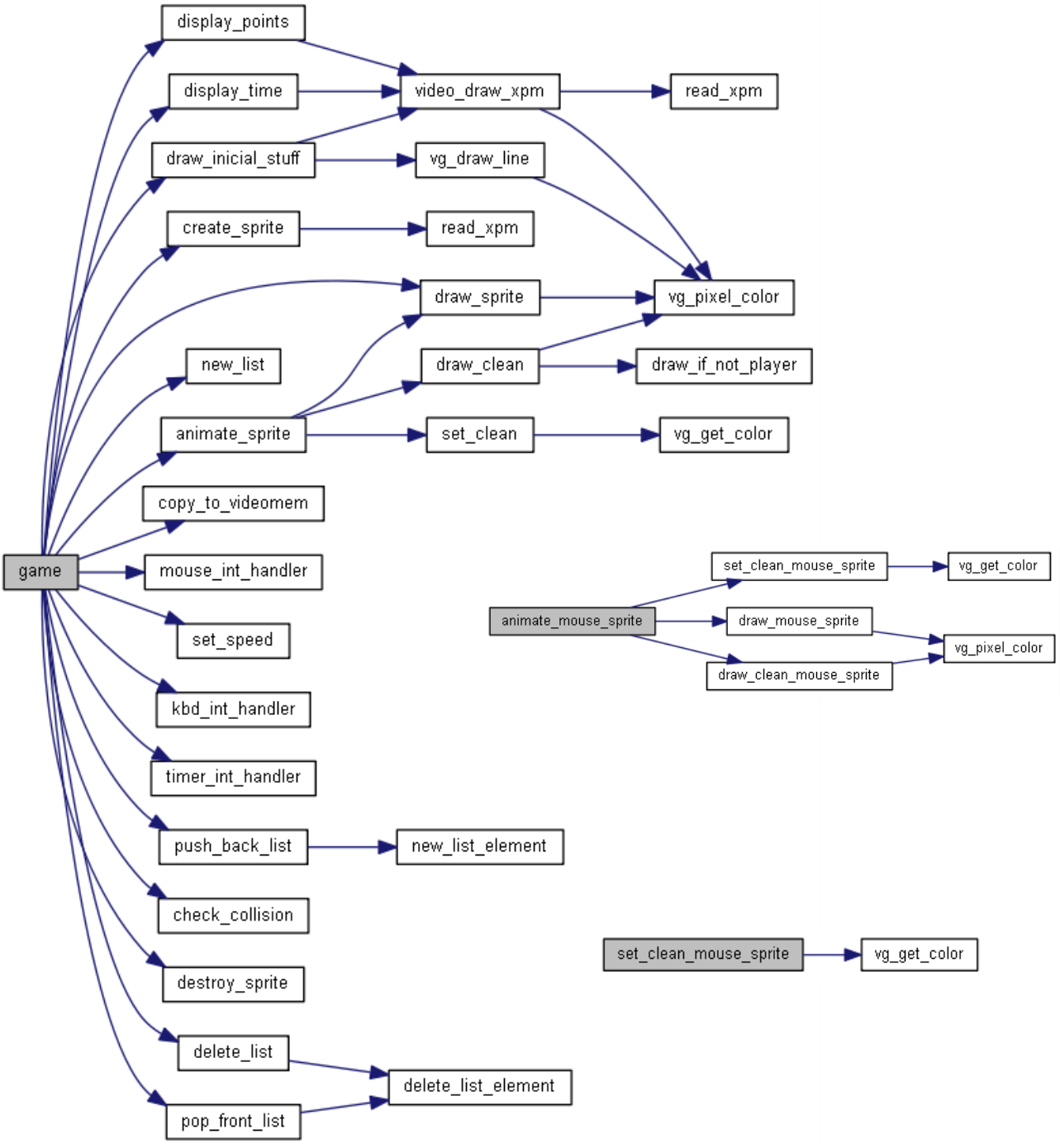
3.13. video_gr.c

Neste módulo está implementado as funções que controlam os dados para o ecrâ no modo gráfico. E também todas a funções de alteração da cor dos pixéis, tais como, a disposição das sprites entre os diferentes menus.

Algumas das funções foram implementadas por um dos membros e outro pelo outro membro, não existiu um só membro como responsável.

Peso do módulo no projeto: 20%


read_high_scores
vg_init
vg_exit
vbe_get_mode_info
lm_free
lm_init
lm_alloc
mouse_menu_handler
updateMousePosition
get_h_resolution
get_v_resolution
create_mouse_sprite
animate_mouse_sprite
draw_clean_mouse_sprite
set_clean_mouse_sprite
draw_main_menu
read_xpm
draw_name
get_caracter
video_draw_xpm
vg_pixel_color
insert_sort_high_score
draw_high_score_menu
draw_high_score
draw_game_over_menu
menu_handler
draw_border
vg_draw_line
set_coordenates
rtc_get_hour
get_final_score
rtc_get_date
copy_to_videomem
game
kbd_int_handler
mouse_int_handler
fill_black
main
start_menu
mouse_stream_mode
mouse_disable_data_report
mouse_issue_command
mouse_subscribe_int
kbd_subscribe_int
timer_subscribe_int
kbd_unsubscribe_int
timer_unsubscribe_int
mouse_unsubscribe_int
destroy_mouse_sprite
save_high_scores



Start menu, *menu_handler* e *game* são as funções que chamam driver_receive.

4. Detalhes de Implementação

As colisões foram algo difícil de implementar no nosso projeto por se tratar de um jogo com perspetiva 3D, podendo assim existir objetos no mesmo pixel que estejam por cima/por baixo de outro, aumentando assim a dificuldade de calcular colisões e de calcular o que mostrar ao jogador (o que está mais próximo/por cima).

Para além disso, tornou-se algo difícil "apagar" as sprites quando elas se moviam do seu local anterior, porque ao ter o nosso jogo dois tipos de objetos a movimentarem-se (jogador e obstáculos) estes deixavam por vezes lixo do outro quando se moviam depois de se intersetarem sem colidirem (passando por cima/baixo).

O facto de o jogo ser teoricamente infinito, com número de obstáculos aleatório e com velocidade crescente, fez-nos perceber que não poderíamos utilizar nenhuma estrutura de dados tradicional de C para guardar os objetos.
Por isso tivemos necessidade de criar uma lista, o que nos criou alguma dificuldade por ser bastante diferente do exemplo de queue demonstrando nas teóricas, por não se "basear em array" mas sim em pointer-to-next. Esta linked list permitiu-nos ter um numero variável de obstaculos e indo eliminando-os (conseguindo assim poupar memória) durante o jogo à medida que saem do ecrã.

Na implementação da state machine tornou-se algo díficil, no que se refere à ordem de atualização dos menus. Ou o rato deixava rasto, ou ao mudar de menu o rato pintava o que estava na sua posição do menu anterior no novo menu. Também por vezes em mudanças entre menus o evento da state machine estava errado que fazia com que o programa deixasse de funcionar, o que nos levou a adicionar um novo evento à state machine *Do_Nothing*.

5. Conclusão

Um dos maiores problemas de LCOM é, na nossa opinião, é a carga temporária. Todas as semanas perdemos tempos de estudo de outras cadeiras para perceber e conseguir fazer os labs, por isso é da nossa opinião que o número de créditos da UC deve ser aumentado ou então a dificuldade e a carga temporal devem ser reduzidas. Apesar disto LCOM foi uma cadeira interessante do ponto de vista de desenvolver em nível baixo, diretamente com os periféricos e ser capaz de construir algo útil e interessante com isso.

Outro grande problema que encontramos com LCOM foi o facto de termos praticamente 0 experiência de trabalho em C e pouco conhecimento da linguagem (apesar das suas similaridades com C++) e pensamos que deveria ser dada mais enfase na aprendizagem desta linguagem nesta (apesar de LCOM não ter que ver com conhecimento desta linguagem) ou noutra UC prévia.

O maior ponto positivo de LCOM foi a possibilidade de explorarmos por nós mesmos os tópicos da cadeira, sendo apenas nos dadas orientações e dada a liberdade de obter informação e conseguir executar por nós mesmos. Para além disso, o facto de partirmos praticamente do 0 e fazermos nós todo um projeto interessante é algo que traz bastante satisfação quando somos capazes de o por em prática.

Anexo: Instruções de Instalação

Para além da instalação normal, é necessário copiar o ficheiro “liblm.a” presente no módulo ficheiros para a pasta src do projeto e assegurar que o ficheiro “high_score.txt” está em “/home/lcom/proj/src” (ou mudar o path no código na chamada das funções read_high_scores e save_high_scores presentes no inicio e no fim, respetivamente, da função start_menu no ficheiro menu.c).