L.EIC – Licenciatura em Engenharia Informática e Computação

Projeto de LC (Laboratório de Computadores) -

“SPACE RACE”

Turma 4 – G04

António Ferreira – up202004735
Henrique Ferreira – up202007459
João Maldonado – up202004244
José Ribeiro – up202007231

10 de junho de 2022

Índice

- Manual de Instruções.....3
  - Menu.....3
  - Jogo.....4
  - Fim de Jogo.....6
- Estado do Projeto.....6
  - Timer.....7
  - Keyboard.....7
  - Mouse.....8
  - Video Card.....8
  - RTC.....9
- Organização/Estrutura do Código.....10
  - timer.....10
  - keyboard.....11
  - mouse.....11
  - video.....11
  - UI.....12
  - main.....12
  - utils.....12
  - rtc.....13
  - astro.....13
  - boost.....13
  - cursor.....13
  - file.....13
  - player.....14
  - points.....14
  - shots.....14
  - game.....14
  - Function Call Graph.....16
- Detalhes de Implementação.....17
- Conclusões.....18

Manual de Instruções

Como ideia principal para este projeto, foi decidido criar uma adaptação do jogo “Space Race”, cujo objetivo principal é que o player, durante 40 segundos, tente chegar ao topo do ecrã tantas vezes quanto possível, desviando-se simultaneamente dos diversos asteroides que flutuam pelo espaço. Para facilitar esta tarefa, em certos momentos aparecem boosts que aumentam a velocidade da nave. No final, a pontuação será o número de vezes que o jogador conseguiu atingir o topo do ecrã.

▪ Menu

Após a execução do código do projeto, é apresentado ao utilizador um **Menu Principal** em que se evidenciam duas opções:

- **‘Play’**, que permite ao user começar o jogo;
- **‘Quit’**, que termina o programa.

Fig. 1 – Menu Principal

É de referir que, de forma a escolher uma das opções em questão, forneceu-se a possibilidade de se estas serem escolhidas com o rato,

pressionando o botão esquerdo em cima de cada palavra, ou com o teclado, recorrendo às teclas ‘w’ e ‘s’ para trocar entre as opções e à tecla do ‘espaço’ para selecionar a opção desejada. Também é de notar que é dada prioridade à escolha efetuada pelo rato, uma vez que, se o rato estiver posicionado num dos botões, o teclado não consegue alterar a opção selecionada.

- Jogo

Quando o utilizador seleciona a opção de jogar, é de seguida apresentado o ecrã principal do jogo. Neste, é possível verificar uma nave cujo movimento será controlado pelo user, recorrendo às teclas ‘w’, ‘d’, ‘s’ e ‘a’ do teclado.

Por outro lado, evidencia-se a presença de 8 asteroides, cuja função principal é impedir a nave de chegar ao topo do ecrã. Estes possuem um movimento horizontal, sendo que o sentido desse movimento depende da paridade do índice de criação, enquanto a rapidez destes é um valor aleatório.

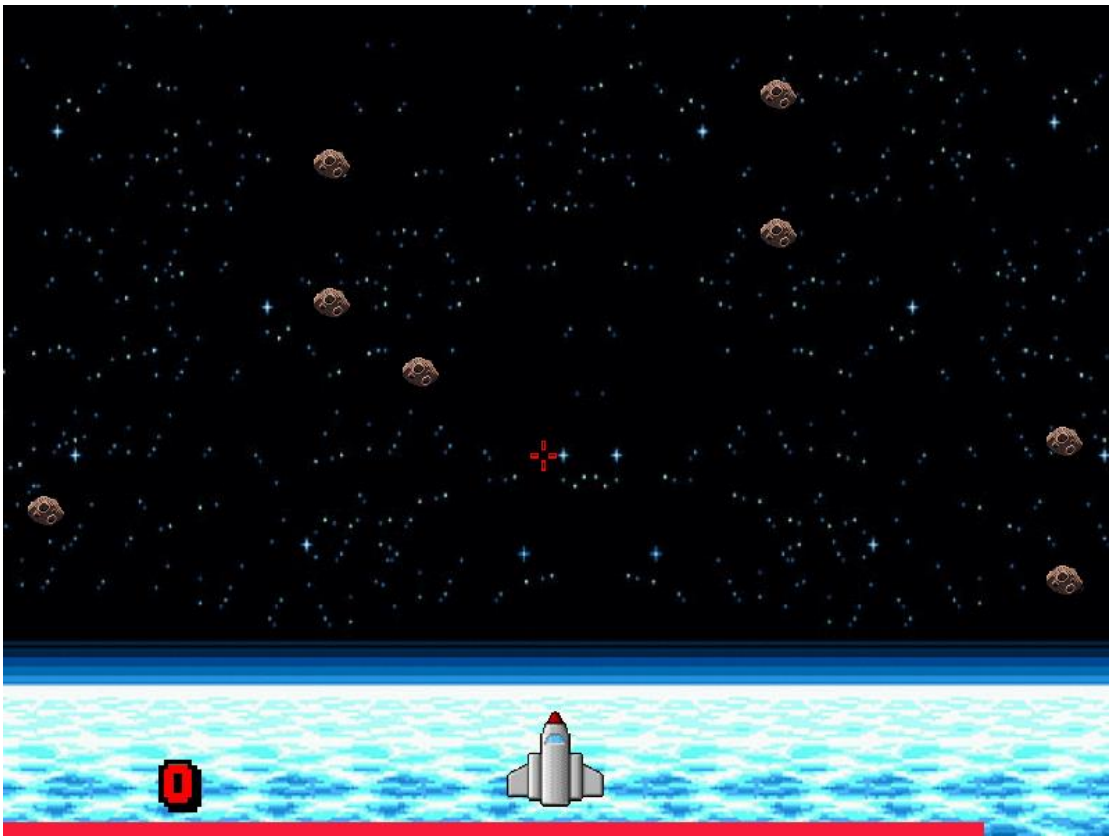
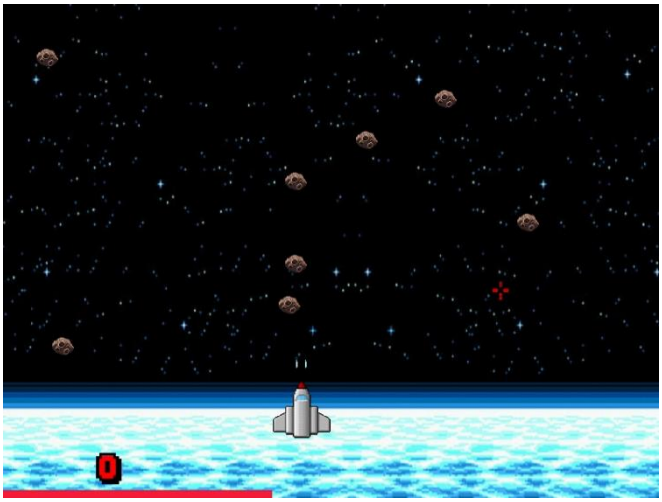
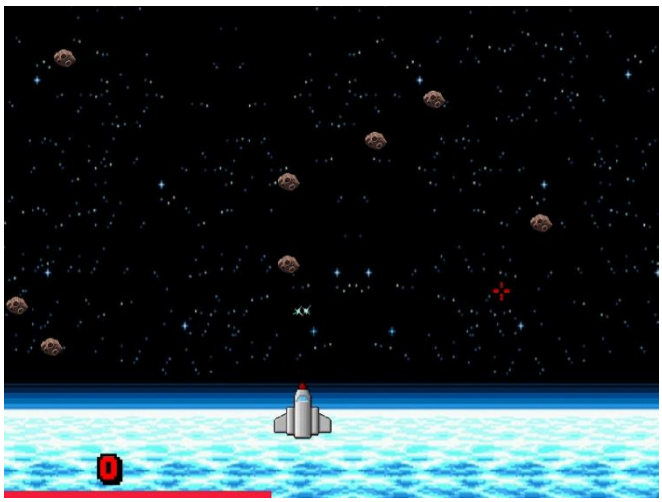


Fig. 2 – Ecrã do jogo

Para além das features referidas, na figura apresentada também é de destacar a presença de uma barra cujo comprimento vai diminuindo ao longo do tempo, sendo que funciona como um indicador do tempo restante de jogo, e de um valor no canto inferior esquerdo que representa o número de pontos do jogador que vai sendo atualizado à medida que este ganha pontos.

Com isto em consideração, também foi decidido fornecer a possibilidade de o utilizador se defender dos astros utilizando o rato. Ao clicar no botão esquerdo deste, a nave lança um míssil que, ao acertar num asteroide, destrói-o por completo, facilitando a sua tarefa de se desviar destes inimigos.





Figs. 3.1 e 3.2 – Destruição de um Asteróide

Por fim, como já foi referido anteriormente, em certos períodos de tempo aparece no ecrã um boost temporário que o utilizador tem de tentar apanhar, carregando com o botão direito do rato em cima dele, de forma a andar mais rápido durante 2 segundos.

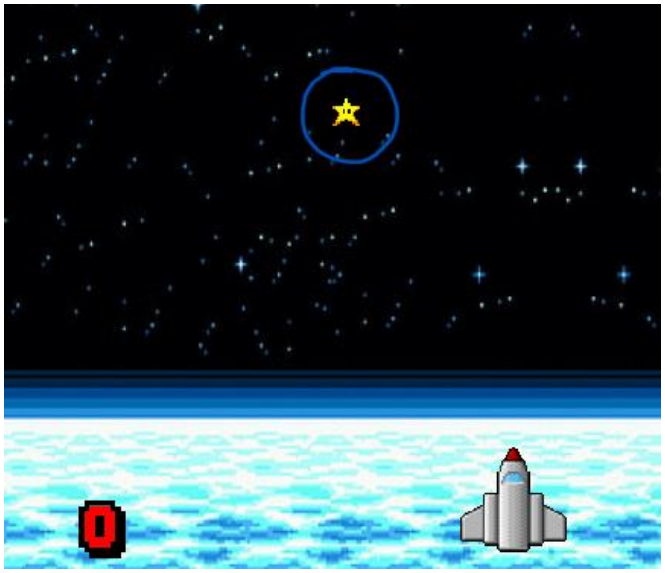
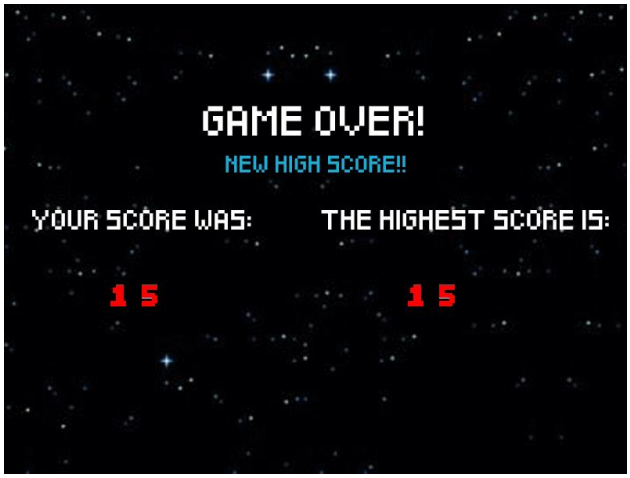


Fig. 4 – Boost

- Final de Jogo

Após o jogo acabar, é apresentado ao utilizador um ecrã que inclui a sua pontuação e o recorde atual de pontos obtidos, sendo que, se o score obtido pelo jogador for superior a esta highscore, o valor desta última será então atualizado. O jogador depois tem a possibilidade de voltar a jogar, carregando nas teclas do 'esc' ou do 'enter', que o levarão novamente para o menu principal.



Figs. 5.1 e 5.2 – Menus de final de jogo

Estado do Projeto

Device	What for	Interrupt/Polling
Timer	Controlar o frame rate do jogo Controlar o tempo do jogo Atualizar a tela do jogo	I
KBD	Selecionar uma opção no menu principal Controlar o movimento da nave	I
Mouse	Selecionar uma opção no menu principal Controlar os disparos da nave Apanhar um boost	I
Video Card	Desenhar as imagens de fundo e os sprites utilizados	NA
RTC	Mudar a imagem de fundo Lançar um boost para o ecrã	I & P

Para além dos dispositivos apresentados, também estava inicialmente planeada a inclusão da Porta Série. Todavia, devido a falta de tempo e

uma superestimação das nossas capacidades, não foi possível incluir o device no projeto.

Timer

O timer constitui um dispositivo de grande importância neste projeto, sendo de destacar os seus interrupts. Quando ocorre uma interrupção do timer, o ecrã do jogo será atualizado, de forma a apresentar todos os sprites nas suas posições mais recentes ou o desenho de novos sprites no ecrã do jogo. Também é de referir a existência de um counter (‘game_over_counter’) que será decrementado sempre que ocorre uma interrupção deste device, pelo que a sua função essencial é determinar quando é que acaba o jogo. Por outro lado, este dispositivo também está responsável por definir, na parte de setup do jogo, a frame rate deste.

O timer é usado nas seguintes funções:

- timer_int_handler()
- timer_set_frequency(uint8_t timer, uint32_t freq)
- timer_get_config(uint8_t timer, uint8_t *st)

Keyboard

O keyboard é utilizado no projeto com a função principal de movimentar a nave. Quando ocorre uma interrupção deste dispositivo, para além da leitura dos scan codes das teclas pressionadas, é utilizada uma struct ‘keyboard_state’ que armazena o estado das 4 teclas utilizadas para o movimento do jogador, isto é, dependendo se o scan code da tecla utilizada for um ‘make’ ou ‘break’ code, esta será considerada como ‘down’ (pressionada) ou não, respetivamente. Como consequência, é possível obter um movimento da nave bastante fluído e que pode ocorrer em direções diagonais. Para além disso, as interrupções do teclado também possuem um papel na seleção das opções do menu principal e no evento de voltar a jogar o jogo quando o user está no ecrã final.

O keyboard é utilizado nas seguintes funções:

- kbc_ih()
- read_from_kbc()
- issue_command()

- issue_arg()

E os seus dados são utilizados nas seguintes funções:

- state_scancode()

Mouse

Juntamente com o timer, as interrupções do rato possuem uma grande importância para este projeto. Por um lado, como já foi referido, ao clicar no botão esquerdo deste, a nave consegue disparar mísseis capazes de destruir asteroides. Por outro lado, com o botão direito do rato, o utilizador consegue capturar um boost quando coloca o cursor sobre este. Também é de notar que o rato é usado no menu principal do jogo, sendo que o user pode selecionar uma opção clicando no botão esquerdo.

O mouse é utilizado nas seguintes funções:

- mouse_ih()
- mouse_get_packet()
- mouse_enable_data()
- mouse_disable_data()
- mouse_issue_cmd()
- mouse_read_data()

Video Card

A placa gráfica é uma parte fundamental do projeto, uma vez que sem ela não poderíamos visualizar a grandiosidade do nosso jogo. Esta é, portanto, utilizada para reproduzir visualmente todos os contextos gráficos inerentes ao projeto, nomeadamente o menu inicial, o jogo em si e todos os elementos que o constituem (nave, asteroides, boosts, plano de fundo etc..), e o menu de fim de jogo.

O modo de vídeo utilizado é o 0x115 com as seguintes características:

-Resolução: 800x600 (vídeo mode 0x115);

-Modo de cor: Direct color;

-Bits por pixel: 24(8:8:8);

É também utilizado Double buffering e as colisões são efetuadas de modo que se a cor que estiver no buffer for diferente da do background haverá uma colisão.

A placa gráfica é utilizada nas seguintes funções:

- vg_init()
- my_vbe_get_mode_info()
- vg_draw_rectangle)()
- vg_draw_hline()
- copy_buffer()
- vg_draw_pixel()
- vg_draw_sprite()
- vg_draw_astro()
- vg_destroy_sprite()

E os seus dados são utilizados nas seguintes funções:

- initialize_game_sprites()
- initialize_game_background()
- initialize_menu_background()
- initialize_astros()
- initialize_boost()
- initialize_cursor()
- initialize_player()
- initialize_points()
- initialize_shots()
- draw_scores()
- capture_boost()
- draw_astros()
- draw_shots()
- draw_player()
- draw_points()
- draw_boost()
- capture_boost()
- menu_option_sprite()
- vg_draw_frame()

- RTC

O Real Time Clock é usado no projeto com dois propósitos. Por um lado, no início do jogo, serão lidos os minutos do RTC, pelo que dependendo da paridade desse valor, a imagem de fundo escolhida será diferente. Por outro lado, sempre que ocorrer uma interrupção deste dispositivo, os segundos apresentados por este serão lidos, pelo que se estes forem um valor divisível por 10, então um boost de duração de 3 segundos será desenhado no ecrã, para que o utilizador o tente apanhar e, consequentemente, a velocidade da nave aumente durante 2 segundos.

O RTC é usado nas seguintes funções:

- rtc_ih()
- get_mins()
- get_secs()

Organização/Estrutura do Código

O projeto encontra-se dividido nos seguintes módulos:

timer –

A estrutura deste módulo é bastante semelhante ao do lab2, pelo que possui funções responsáveis pela subscrição de interrupções, alteração da frequência do Minix e obtenção da configuração do timer.

Peso no projeto: 5%

Desenvolvido por: Henrique Ferreira e José Ribeiro

keyboard –

Quanto a este módulo, a estrutura e o conteúdo deste não diferem de forma significativa das funções desenvolvidas para o lab3. Para além da das típicas funções de subscrição de interrupções, também se evidencia funções de leitura do output buffer e da escrita de comandos e argumentos para os respetivos registos.

Peso no projeto: 7%

Desenvolvido por: António Ferreira e João Maldonado

mouse –

Relativamente a este módulo, as funções presentes já tinham sido implementadas no lab4, pelo que se evidencia a capacidade de ativar e desativar o 'data reporting', de ler packets de data e construí-los e de subscrição e handling de interrupções.

Peso no projeto: 8%

Desenvolvido por: Henrique Ferreira

video –

Este módulo possui um conjunto de funções relacionadas com a parte gráfica, sendo que o peso deste para o projeto é bastante elevado. Primeiramente, é de destacar as duas funções responsáveis pela inicialização do modo gráfico no projeto – vg_init() e my_vbe_get_mode_info(), sendo esta última uma versão pessoal da função get_vbe_mode_info() fornecida no lab5. Tendo isto em consideração, para além destas e das funções desenvolvidas no lab referido, que por sua vez possuem a funcionalidade de desenhar um retângulo ou de fazer load e desenhar xpms, também foram criados outros métodos que desempenham funcionalidades específicas ao projeto em questão, como a deteção de colisões entre sprites. Por fim, é de referir a criação de uma função essencial para a utilização de 'double buffering' – copy_buffer().

Peso no projeto: 15%

Desenvolvido por: António Ferreira, João Maldonado e José Ribeiro

UI –

Juntamente com o ‘video’, este módulo revela uma grande importância no projeto, sendo que de certa forma pode ser visto como uma extensão do módulo anterior. Neste foram desenvolvidas funções cuja responsabilidade principal é o desenho e atualização dos diversos sprites e componentes gráficos do jogo, sendo de destacar a função vg_draw_frame(), essencial para desenhar a frame do jogo

Peso no projeto: 15%

Desenvolvido por: António Ferreira, João Maldonado e José Ribeiro

main –

Neste módulo são feitas as subscrições e, no fim, o cancelamento das mesmas de todos os dispositivos utilizados (à exceção da placa gráfica que tem um funcionamento diferente). Além disto, são feitas invocações das funções principais (menu(), play() e end()) e também a inicialização das imagens de fundo do menu e do ecrã de jogo.

Peso no projeto: 11%

Desenvolvido por: António Ferreira, João Maldonado, José Ribeiro e Henrique Ferreira

utils –

Os métodos presentes neste módulo são referentes à leitura de um endereço para uma variável de 8 bits e a obtenção do byte mais significativo e do menos significativo.

Peso no projeto: 1%

Desenvolvido por: Henrique Ferreira

rtc –

Neste módulo verifica-se a presença de funções simples relacionadas com a subscrição de interrupções e leitura das informações do RTC.

Peso no projeto: 2%

Desenvolvido por: António Ferreira e José Ribeiro

astro –

O módulo relativo aos asteroides possui uma struct responsável pela organização das informações relativas a um astro e funções relacionadas com a inicialização e atualização das posições dos asteroides.

Peso no projeto: 3%

Desenvolvido por: António Ferreira e João Maldonado

boost –

Módulo relativo ao boost que inclui a estrutura que contém os dados do mesmo e funções para desenhar e para inicializar.

Peso no projeto: 2%

Desenvolvido por: José Ribeiro

cursor –

Módulo referente ao cursor que inclui a estrutura que contem os dados do mesmo e a função para inicializar os dados da estrutura mencionada.

Peso no projeto: 2%

Desenvolvido por: José Ribeiro e Henrique Ferreira

file –

Módulo onde são feitos todos os acessos a ficheiros. Contém funções para ler o highscore do ficheiro e para atualizar o mesmo.

Peso no projeto: 1%

Desenvolvido por: João Maldonado

player –

Módulo referente ao jogador, sendo que inclui uma struct do mesmo e função inicializadora. Neste módulo também está presente a função que desenha, consoante a posição anterior e teclas premidas, o jogador no ecrã.

Peso no projeto: 3%

Desenvolvido por: António Ferreira e João Maldonado

points –

Módulo referente aos pontos do jogador, sendo que inclui uma struct essencial para a organização das informações do mesmo e as funções para inicializar os dados da estrutura mencionada e desenhá-lo no ecrã.

Peso no projeto: 1%

Desenvolvido por: António Ferreira e João Maldonado

shots –

Módulo referente às balas disparadas pela nave. Contém a estrutura das balas e das suas animações, onde está contida a sprite e a image, e também a função inicializadora das estruturas mencionadas.

Peso no projeto: 4%

Desenvolvido por: Henrique Ferreira

game –

Este módulo também é de grande relevância para o projeto, sendo que este é composto pelas principais funções para o funcionamento do jogo, que, por sua vez, incluem diversas chamadas a funções dos módulos apresentados. Nestas funções também é de referir a presença de ciclos do driver_receive(), destacando assim a sua importância.

As funções referidas anteriormente são:

- **Menu():** nesta função são feitas as inicializações dos elementos utilizados no menu. De seguida, dentro do ciclo do driver_receive(), são tratados os diversos interrupts de modo a que seja possível mudar a opção escolhida e confirmar a seleção, com o teclado e com o rato. Também é utilizado o timer para manter a interface gráfica atualizada.

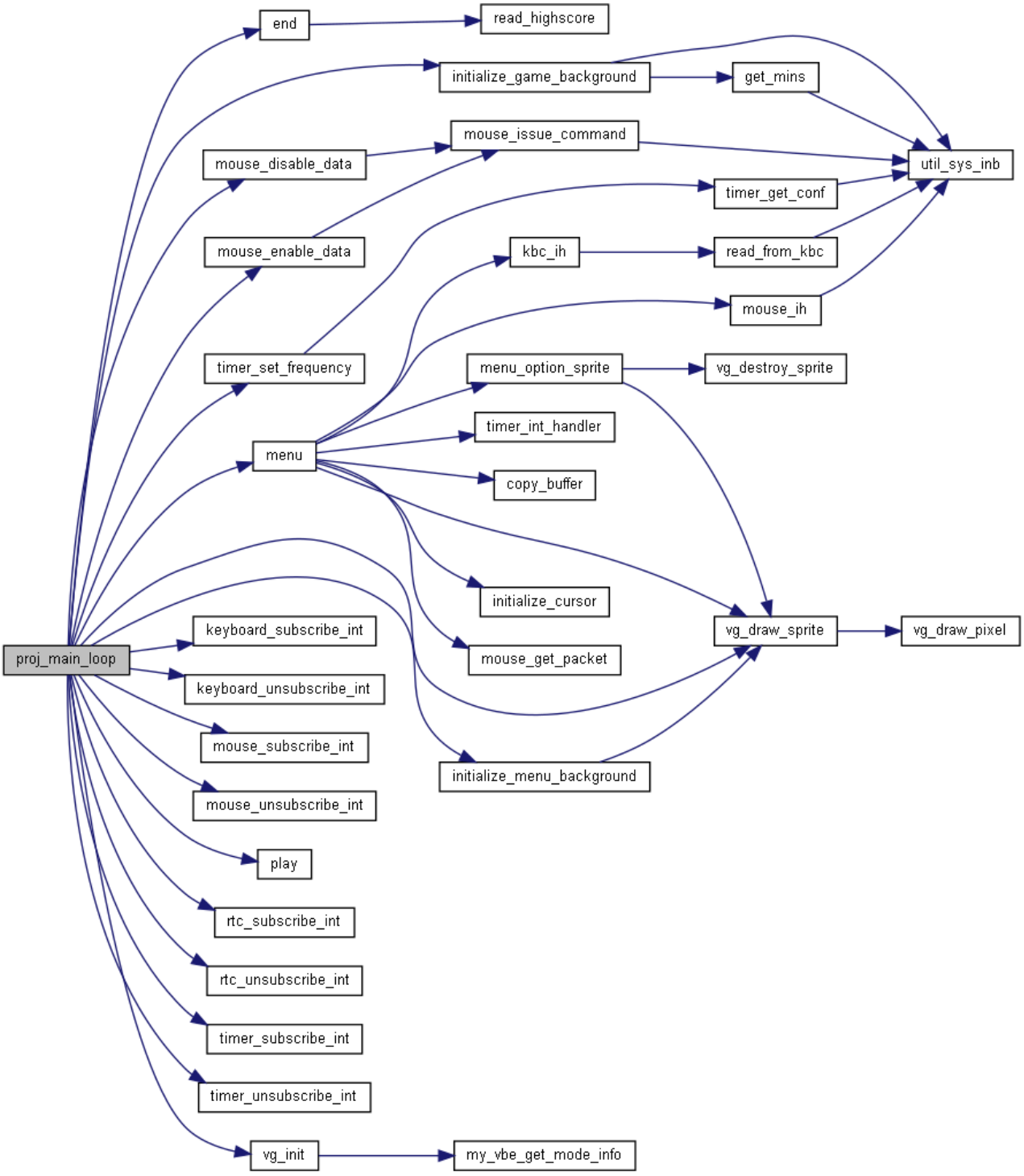
- **Play():** De certa forma, esta função corresponde ao coração do jogo. Para além da inicialização dos diversos sprites e elementos utilizados, é incluído o ciclo do driver_receive(), que possui diversas funcionalidades, como o desenho da frame do jogo e atualização do valor das coordenadas dos sprites, recorrendo às interrupções do timer; o movimento do jogador, usando os interrupções do teclado; o disparo de balas como consequência dos interrupts do rato; e o aparecimento de um boost para o player apanhar quando o valor dos segundos lido devido a uma interrupção da RTC apresentar é múltiplo de 10.

- **End():** Por fim, nesta função, é desenhado o ecrã de game over com a pontuação obtida e highscore. Dentro do ciclo do driver_receive() é esperado que seja premida a tecla ‘enter’ (para jogar de novo) ou a tecla ‘esc’ (para fechar o jogo).

Peso no projeto: 20%

Desenvolvido por: António Ferreira, Henrique Ferreira, João Maldonado e José Ribeiro

Function Call Graph –



Detalhes de Implementação

No sentido de manter o código organizado e modular, foi decidido a implementação de structs (player, points, astro, etc.). Por sua vez, as structs permitem o armazenamento de diversas informações relativas a um determinado elemento, como a sua posição ou sprite, de forma bastante organizada, contribuindo de certa forma para uma melhor compreensão do código. Tendo isto em conta, é de afirmar que estas estruturas contribuem para uma abordagem de programação orientada para objetos.

Relativamente à RTC, este dispositivo revelou-se como sendo bastante simples de desenvolver, sendo que foi optado por implementá-lo com métodos de polling e de interrupções, cujo foco principal de ambos se baseia na leitura das informações dos seus registos.

Também é de destacar o método de deteção de colisões utilizado no projeto. Como os sprites utilizados possuem uma hit box retangular, a deteção de colisões recorrendo exclusivamente à verificação das posições de cada elemento não seria a melhor opção, pois levava a colisões inesperadas. Sendo assim, decidiu-se implementar um método baseado na comparação dos valores dos bytes presentes nos dois buffers utilizados. Ou seja, sabendo que o desenho dos diversos componentes do jogo é sequencial, se, ao desenhar-se um asteroide, por exemplo, o valor que se encontra no buffer auxiliar não corresponder a uma cor da imagem de fundo, mas sim a uma cor da nave, então admite-se que aí ocorre uma colisão.

Conclusões

Tendo tudo o que foi referido em consideração, podemos afirmar que conseguimos alcançar diversas conquistas, sendo de destacar a demonstração dos nossos conhecimentos obtidos nesta cadeira ao longo do semestre e a programação de dispositivos que, no início deste curso, nos pareciam conceitos bastante complexos e impossíveis de compreender.

Ao longo da execução deste projeto, foram levantados uma variedade de problemas que contribuíram para um atraso na execução deste. O que merece uma maior atenção é a implementação do rato, uma vez que este se revelou como sendo um dispositivo até complexo de desenvolver e de incluir no projeto, pelo que o número de erros causados por ele, especialmente na parte dos mísseis da nave, foi a principal causa dos nossos problemas.

Sendo assim, para concluir, apesar de estarmos satisfeitos com o produto final deste projeto, há um conjunto de funcionalidades que, apesar de não termos definido inicialmente, gostaríamos de implementar no nosso programa, como, por exemplo, a rotação da nave utilizando o rato e a introdução de vários níveis. Por outro lado, como já foi referido, contrariamente ao plano inicial, não conseguimos implementar a porta série, pelo que a funcionalidade de multiplayer seria uma outra feature que gostaríamos de adicionar no futuro.