U.PORTO
FEUP FACULDADE DE ENGENHARIA
UNIVERSIDADE DO PORTO

Conquer the moon

Relatório do projeto do laboratório de computadores

Figura 1 – Menu inicial

MIEIC 2.º ano Turma 3 Grupo 2

João Paulo Gomes Torres Abelha – up201706412@fe.up.pt

João Rafael Gomes Varela – up201706072@fe.up.pt

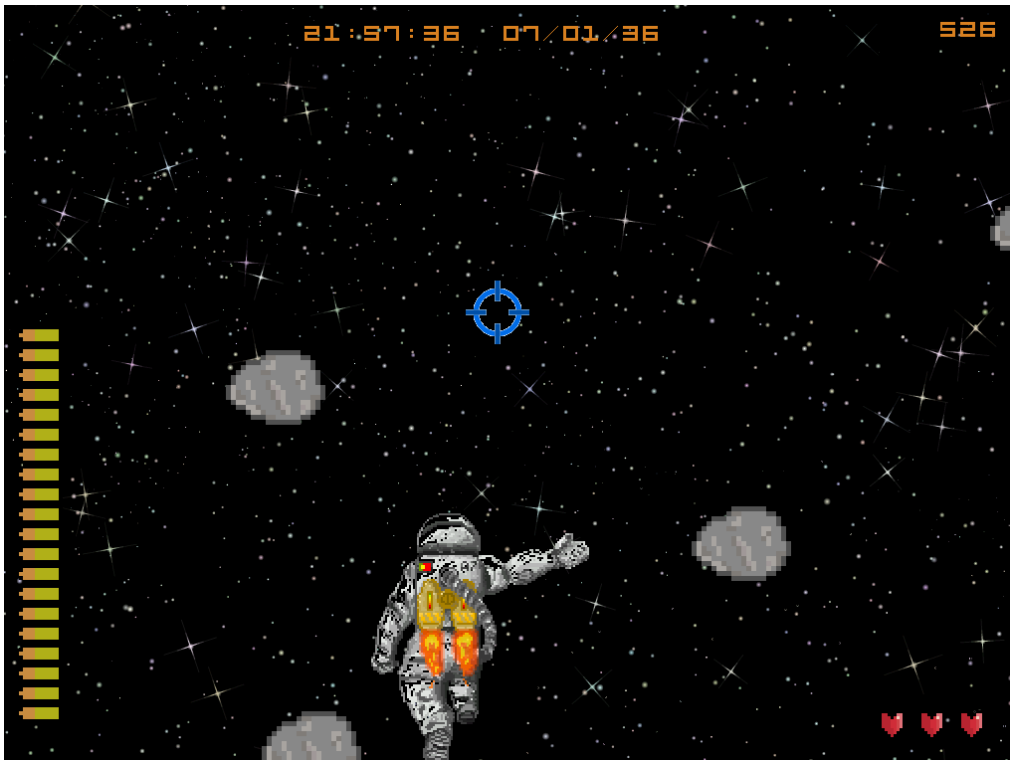
1 – Instruções de Utilização

1.1 – Menu Principal

No início do programa é apresentado ao utilizador o menu principal, a partir do qual ele pode derivar para uma das opções apresentadas, com o auxílio do rato:

- ***"Play Game"*** : Inicia o modo de jogo.
- ***"Leaderboad"*** : Mostra o *leaderboard* que apresenta as cinco melhores pontuações.
- ***"Exit"*** : Termina o programa.

1.2 - Modo de Jogo



O início de jogo começa com um astronauta, já no espaço, em direção à lua, que à medida que avança vai encontrar asteroides dos quais vai ter de se desviar ou destruir, aparecendo com este último a possibilidade de recuperar uma das cinco vidas que tem. Para além disso, encontram-se satélites artificiais que devem ser evitados, uma vez que quando destruídos para alem duma vida também a pontuação é penalizada.
Alguns comandos básicos para controlar o astronauta são:

- **Movimento** – teclas A, S, D, W
- **Recarregar** - até um máximo de 20 balas- R (sendo este interrompido caso haja um disparo)
- **Mira** - movimento do rato
- **Disparo** - de tiros desde a mão até à mira- botão esquerdo do rato

Neste modo, o astronauta vê-se sujeito, como qualquer ser normal, à ação da gravidade e, por isso, não convém ficar quieto! Para além disso, a dificuldade do jogo aumenta com o passar do tempo, o que está diretamente relacionado com a variação da força gravítica.

1.3 - Guardar o *Score*

Quando o astronauta perde as suas cinco vidas, o jogo termina e aparece no ecrã a pontuação do jogador e a possibilidade de escrever um nome (com possibilidade de apagar e de usar um nome, por defeito, *"PLAYER*" quando nada é escrito no nome) que poderá ir fazer parte da *leaderboard*, mediante a pontuação obtida, isto é: o jogador terá de estar no *top* 5. Quando a pontuação é nula, o histórico desse jogo nunca será guardado.

Após guardar o seu nome, o jogador poderá voltar para o menu, clicando no botão MENU, que aparece no canto inferior esquerdo, com o rato.

1.4 - *LeaderBoard*

Nesta secção, podemos ver as cinco melhores pontuações obtidas, acompanhadas com informação adicional, nomeadamente a data e nome com que a pontuação foi guardada. Cada vez que uma pontuação passa ao *top* 5 a última é substituída. Para todo este efeito são escritos e lidos de **ficheiros** as informações dos melhores jogadores sendo o ficheiro atualizado quando saímos do jogo.

1.5 – Sair do Jogo

Quando o utilizador se encontra no menu principal, para abandonar o jogo deve pressionar a tecla *ESC* ou clicar na opção *EXIT*.

2 – Estado do Projeto

Periférico	Função	Interrupções
Timer 0	Atualizar o jogo (frame rate) ,subida de nível e scores	Sim
Teclado	Controlo do astronauta e mudanças de estado	Sim
Rato	Selecionar opções no menu(mudança de estados), controlar mira e disparar tiros	Sim
Placa Gráfica	Parte gráfica do jogo	Não
Real Time Clock	Obter a data e horas atuais e gerar seed	Não

Tabela - Função de cada periférico no projeto

2.1 **- Dispositivos Utilizados**

2.1.1 Timer

O periférico é responsável pela constante atualização do estado jogo, a cada interrupção. É usado para o controlo do *frame rate* (usando a frequência de 60 interrupções por segundo), pela subida de nível e scores e variação da gravidade. Foram usadas funções para subscrever o periférico (implementadas no Lab2), bem como o interrupt handler implementado em *assembly*. São usadas variáveis do tipo *static* para gerir a informação deste, sendo estas acessíveis exteriormente através de "*get functions*". Foi usada uma variável booleana, para processar os eventos, da qual é feita o set e, desse modo, é processada a informação.

2.1.2 Keyboard

O periférico *keyboard* é utilizado para o controlo dos movimentos do astronauta (4 graus de liberdade) e para dar *reload* da arma até serem atingidas o número máximo de balas ou até haver um disparo. Para verificar os *scancodes* das teclas primidas são usadas interrupções. As funções responsáveis são as funções de subscribe e unsubscribe implementadas no decorrer do lab3. O seu interrupt handler esta definido

em *assembly*. Uma outra função que merece destaque é a keyboard_event_handler que é reponsavel pela gestão do movimento do astronauta. Ainda é usada a tecla *ESC* para sair do jogo no menu principal como alternativa à tecla Exit que pode ser selecionada pelo mouse

2.1.3 Mouse

O periférico mouse é usado nos menus e no decorrer do jogo. Nos menus é usado para selecionar as opções sendo uma opção selecionada apenas quando efetuado um clique quando já em cima da tecla. Durante o jogo o rato controla a mira do astronauta (através do seu movimento) e dispara as balas (através de um clique com o botão esquerdo do rato). Este periférico funciona, igualmente com interrupções, sempre que há movimento ou clique no rato. As interrupções são geridas pelo *handler*. Foram implementadas também funções de subscribe e unsubscribe com o intuito de intercetar as interrupções.

2.1.4 Video Card

Responsavel por toda a parte gráfica do jogo. Foi escolhido o modo 117h em base com uma resolução de 1024 por 768 pixeis, com 16 bits por pixel (disponibilizando 65536 = 2^16 cores). Este modo utiliza RGB 5-6-5 (5 pixeis para cor vermelha, 6 para verde e 5 para azul).
No projeto é usado double buffering de forma a manter uma sucessão de imagens fluídas.
Para a troca de buffers usamos a função copybuffer() que foi implementada usando a função memcpy().
Para a deteção de colisões foi usado um buffer auxiliar em que diferentes objetos têm os seus id's definidos e desse modo vemos se há colisões.
No projeto são usadas fontes e objetos que "se movimentam" (sprites), bem como fontes para números e alfabeto para disponibilizar o nível, pontuação, data e nome do jogador. Recorremos a imagens para texto estático e fonts de cada letra do alfabeto e números para texto dinâmico.
O movimento dos bitmaps é definido por parâmetros velocidade.
Para o fundo do background usamos uma função que a vai reescrevendo dando a noção de movimento.
O que é escrito no no ecrã é controlado por uma máquina de estados implementada na game.c que nos dá informação acerca do estado atual do jogo.

2.1.5 Real Time Clock

Para facilitar a interface, acesso e organização foi criada uma estrutura date que contém cada parâmetro da data: segundos, minutos, horas, dias, mês e ano.
O periférico Real-Time Clock é usada a função de leitura de data e horas. A data e horas e atuais são apresentadas no ecrã enquanto jogamos com o astronauta. Esta é obtida através da função get_date() que retorna uma cadeia de caracteres com esta informação. É feito o poll da data e hora, recorrendo ao uso de assembly: durante o jogo e quando o jogo perde para guardar na leaderboard.

3 – Estrutura do Código

No desenvolvimento do projeto, o código foi dividido em vários módulos, de forma a tornar o código mais estruturado e compreensível. Vários deles foram gerados no decorrer de várias aulas laboratoriais, tendo sido sujeitos a algumas alterações.

Módulos

- Módulos feitos no decorrer de aulas laboratoriais

a) Modulo do timer

Módulo timer

Neste modulo são usadas funções para manipular o timer. Este foi desenvolvido no decorrer da aula laboratorial 2, tendo sido realizado igualmente por ambos os membros. Posteriores alterações que houve foram realizadas com o consenso e simultaneamente por ambos os membros.

Modulo i8254

Este modulo apresenta todas as constantes simbólicas, isto é, macros mais importantes, sendo que este nos foi fornecido já pelos docentes pelo que não é da nossa autoria.

b) Modulo do keyboard

Modulo keyboard

Neste modulo são usadas funções para manipular diretamente o keyboard. Este módulo foi desenvolvido maioritariamente no decorrer da aula laboratorial número 3, tendo sido realizada igualmente por ambos os membros. Contem para alem disso outras funções que gerem a interface com o astronauta, atualizam o estado das teclas que são usadas no jogo, funções de reset do keyboard e get functions.

Modulo i8042

Neste encontram-se as constantes simbólicas usadas para a manipulação do keyboard e também do mouse uma vez que ambos usam o mesmo controller. Este módulo foi desenvolvido no decorrer da aula laboratorial numero três. Aquando da realização do projeto foram acrescentadas outras macros que permitissem reconhecer as teclas quando estamos a controlar o astronautas e quando estamos a escrever o nome do jogador que perdeu, sendo que isto é diferenciado pelo state machine que mediante o estado limita o número de teclas do keyboard.(miudar).

O módulo keyboard foi realizado igualmente pelos dois membros do grupo.

c) Modulo do Mouse

Módulo Mouse

Este módulo é constituído por funções que manipulam diretamente o teclado. Este módulo foi maioritariamente feito no decorrer da aula laboratorial número 4, sendo, realizado igualitariamente pelos dois membros. É constituído por funções como o mouse_handler, subscribe e unsubscribe, sendo ainda constituído por funções de reset do rato, mudando o bitmap do rato, consoante esteja em modo de jogo em no menu e a game_interrupt handler que chama o handler , verifica a existência de erros e executa as mudanças necessárias no bitmap do cursor que é apresentado no ecrã.

Modulo i8042

Controler que é partilhado com o keyboard e que também as constantes simbólicas do mouse.

d) Modulo da placa de vídeo

Modulo vbe

Neste modulo encontram-se funções para manipular as funções VBE. Parte deste módulo foi fornecido no lab5. O restante foi realizado de igual pelos dois membros. Este módulo é importante, uma vez que retorna informação sobre o modo VBE, permitindo preencher uma struct que guarda esta informação, que será importante para a inicialização do modo da placa gráfica.

Modulo video.gr

Neste modulo encontram-se funções que manipulam os buffers de vídeo - vídeo da memoria de vídeo e o buffer usado para double buffer, como second-buffer. Parte deste modulo foi nos fornecido na realização do laboratório número 5, sendo que a sua completação foi realizada de forma igual pelos dois membros.

Módulos feitos pós aulas laboratoriais

a) Modulo rtc

Este módulo serve para manipular o periférico rtc, nomeadamente ler deste a data atual para ser disponibilizada no ecrã no decorrer do jogo. É usada a estrutura date que é constituída por membros constituintes duma data que é atualizado sempre que queremos fazer display no ecrã da data.

b) Modulo score

Este módulo é constituído pela struct Score que é constituído pelo score do jogador, e por bitmaps que apresentam números, letras do alfabeto e alguns símbolos importantes. Apresenta funções que manipulam o valor do score, ora aumentando, ora diminuindo por penalização. Enquanto a máquina de estados esta no estado “playing” é usada a função draw_player_score() e draw_date() na parte superior do ecrã. Possui ainda funções que permitem quando o jogador perde escrever o nome de jogador (permite apagar letras)-manipulação do texto em modo de vídeo imprimindo e apagando dinamicamente, apresentar o score e guardar essa informação, incluindo a data para ser posteriormente vista na leaderboard se a sua pontuação estiver nos 5 melhores. Permite ainda ler e escrever em ficheiros s informação dos top 5 jogadores. Funções de reset, criar, atualizar e eliminar o objeto score.

O ficheiro .h apresenta todas as fontes necessárias para conseguirmos o efeito anteriormente descrito

c) Modulo satellite

Este modulo é constituído pela struct satellite que contem um bitmap, que associa ao satélite a imagem correspondente, um initial_x e initial_y que é calculado de forma random e um last_x e last_y usado para o efeito de explosão de satelites, e por ultimo um booleano exploded para saber quando efetuar um efeito de explosão. Constitui função para calcular uma posição inicial random e função de reset.

d) Modulo menu

Este modulo é constituído pela struct main_menu que tem todos os bitmaps usados no menu. Possui ainda enumerações que permitiram a implementação da state machine que permite saber o estado atual do jogo e quando efetuar alguma transição.

e) Modulo health

Este modulo é consituído pela struct health_point que guarda o numero atual de vidas, havendo um teto, os bitmaps correspondentes e um booleano draw que serve para controlar se é desenhado ou não no ecrã. Este modulo possui ainda função que inicializam a struct, cath_one_life() que verifica que houve uma colisão do astronauta com uma vida, ganhando uma, trata o aparecimento de vida quando destruído um asteroide, bem como trata quando perde uma vida, possui função de reset e terminate da struct health_point.

f) Modulo collision

Este modulo é consittuido por um por um buffer que mapeia os diferentes objetos e trata as colisões dos diferente objetos, sendo atribuído um respetivo id a cada um deles.Possui a função initialize_map_collision() que aloca memória suficiente e inicializa. Possui uma clear_map_collision() que mete os valores todos a zero e ainda a terminate_map_collision() que desaloca a memória alocada anteriormente.

g) Modulo bullet

Este modulo é constituído por a struct bullet que tem como campos um bitmap, um pointer para a próxima bullet e um booleano draw que verifica que é desenhada no ecrã. Ainda é apresentado nestre módulo um uma struct gun que gere as balas e o seu reload. São realizadas neste funções initalize, draws,reset e terminate entre outras relevantes para a implementação destas.

e) Modulo asteroide

Este módulo é constituído pela struct asteroid que contem um bitmap, que associa ao bitmap a imagem correspondente, um initial_x e initial_y que é calculado de forma random e um last_x e last_y usado para o efeito de explosão de asteróides, e,por último, um booleano exploded para saber quando efetuar a animação de explosão.

O módulo possui uma função initialize e ainda uma função para calcular a posição random inicial (random_asteroid()), outra para dar reset nos asteróides (reset_asteroids()), e uma para os terminar (terminate_asteroids()).

f) módulo bitmap

Neste módulo existe uma série de funções para manipulação de bitmaps. Permite-nos ler ficheiros do tipo .bmp (load_bitmap_file())e desenhá-los no ecrã (draw_bitmap()), bem como apagá-los (deleteBitmap()). Para além disso, este módulo contém funções que nos permitem limpar um buffer (clear_buffer()), desenhar o background (paint_background()) e movimentá-lo verticalmente (update_background())
É de destacar que grande parte destas funções foram retiradas do blog do Henrique Ferrolho.

g) módulo game

No módulo Game, existem todas as funções relativas ao processamento de eventos do jogo, a sua atualização, e também do astronauta. Este módulo contém a struct Game com toda a informação relativa ao jogo e com os seus objetos, e a struct leaderboard que representa as estatísticas de um jogador.
O Game está associado a uma state_machine, cujos estados são MENU, LEADERBOARD, PLAYING e SAVESCORE, e eventos que permitem o utilizador navegar entre esses estados (PLAY_BUTTON, EXIT_BUTTON, BACK_TO_MENU_BUTTON, LEADERBOARD_BUTTON, RESET_GAME_BUTTON, LOST_BUTTON).
Deste modo, é neste módulo que são feitas as inicializações dos objetos do Game (initialize_game()), e onde são inicializados os periféricos usados no projeto.

Neste jogo também utilizamos uma uma font para cada número e para cada letra do alfabeto.

Documentação do Projeto
Os módulos do projeto foram documentados, utilizando Doxygen, incluindo os gráficos de chamadas.

Gráficos apresentados no jogo

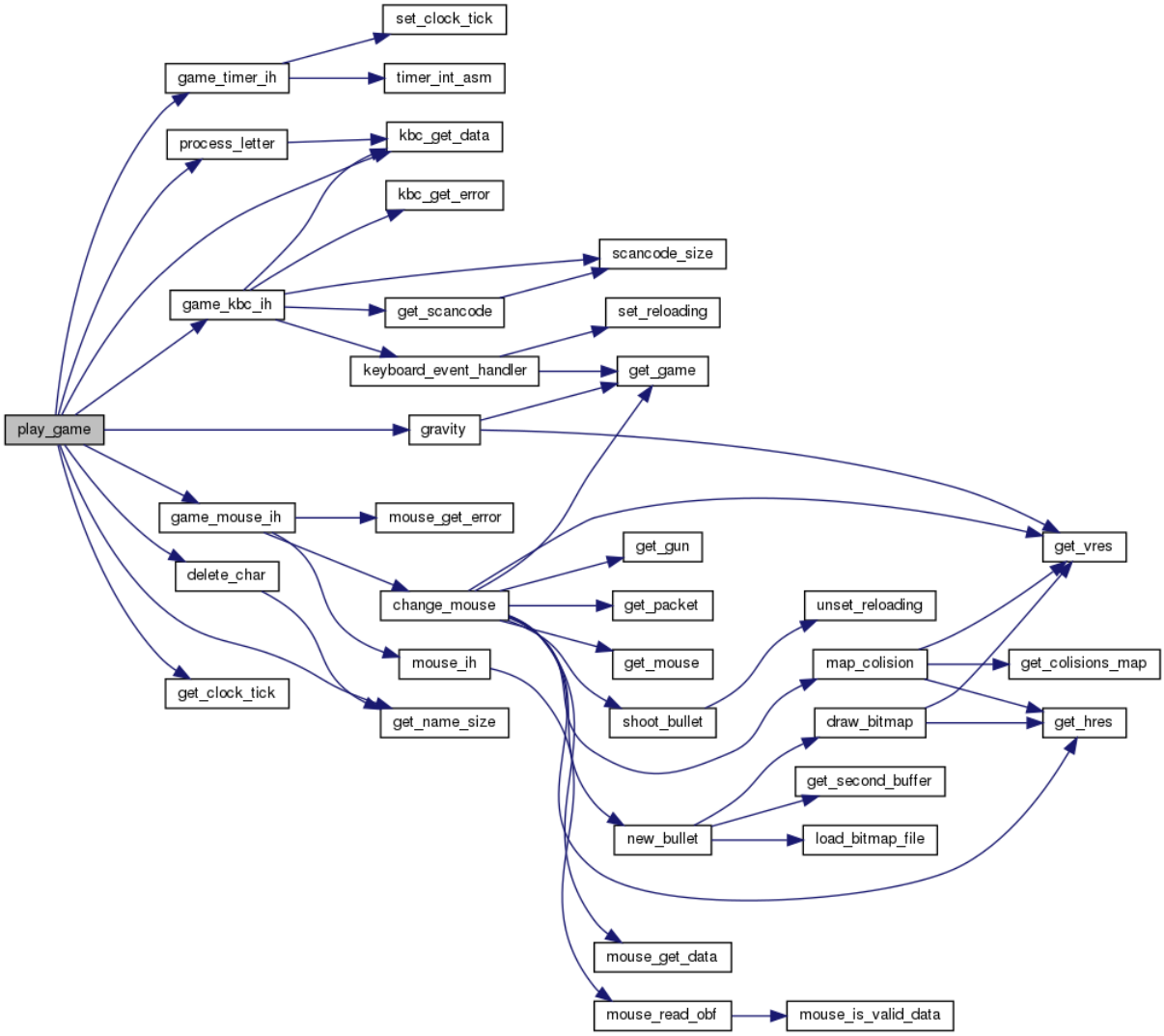
A maior parte dos gráficos foram realizados por nos, e convertidas para bitmaps R5 G6 B5 (16 bits), com o auxilio de um programa GIMP (Gnu Image Manipulation Program).

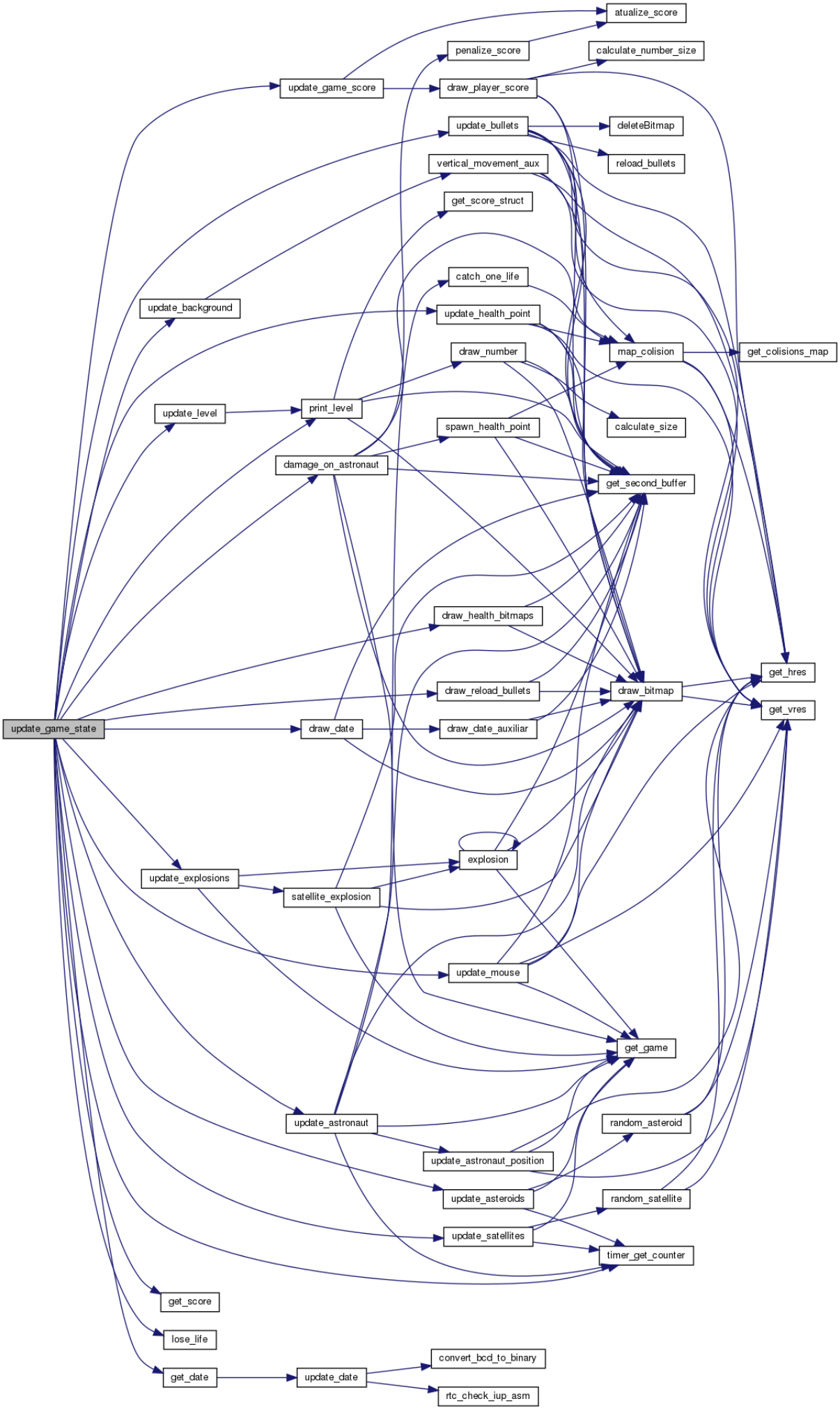
As restantes foram retiradas da net, com a devida permissão para uso.

Peso do código de cada aluno

Em todos os módulos o peso de ambos foi praticamente igual, considerando em todos os módulos 50% / 50%.

Diagrama de Chamada das funções mais relevantes




update_game_state
update_game_score
atualize_score
penalize_score
calculate_number_size
draw_player_score
update_bullets
deleteBitmap
reload_bullets
vertical_movement_aux
get_score_struct
catch_one_life
update_background
update_health_point
map_colision
get_colisions_map
draw_number
update_level
print_level
calculate_size
spawn_health_point
damage_on_astronaut
get_second_buffer
draw_health_bitmaps
get_hres
draw_reload_bullets
draw_bitmap
get_vres
draw_date
draw_date_auxiliar
explosion
update_explosions
satellite_explosion
update_mouse
get_game
update_astronaut
random_asteroid
update_astronaut_position
update_asteroids
random_satellite
update_satellites
timer_get_counter
get_score
lose_life
get_date
update_date
convert_bcd_to_binary
rtc_check_iup_asm

Detalhes de implementação

No projeto foram realizadas funções por “layers”, de forma a separar o código que interage mais com os periféricos (grande parte realizada nas aulas laboratoriais) e outras que utilizam estsa funções quase de forma abstrata e por ultimo uma ultima camada que gere toda a informação, chamando funções das camadas anteriores, funcionando estas como abstrações. Com isto, pretendeu-se estruturar o código da melhor maneira possível.

Foi usado no projeto uma state machine com o intuito de analisar os eventos gerados a cada interrupção e desse modo os eventos são tratados à medida. Deste modo, o desenvolvimento do jogo é facilitado, sendo mais facilmente definido o que é necessário fazer em cada estado.

Outro cuidado que tentamos ter foi a da implementação de structs que funcionam como classes, sendo as suas instanciações os objetos. Estes possuem ainda funções que os contruem, destroem, métodos get, set dos seus campos. Desta maneira, a interface com o programa é simplificada e fica de forma mais organizada. A struct Game funciona como o objeto do jogo, funcionando como uma charneira com todos os outros objetos do jogo (por exemplo os asteroides, as vidas, etc).

Para a identificação de uma colisão entre determinados objetos de forma mais precisa possível, mapeamos todos os objetos no ecrã com um determinado id e vemos a cada interrupção de algum desses ids colide. Para este algoritmo é usado um buffer auxiliar que e feito o memset para zero, sendo os seguintes objetos, preenchidos em seguida.

Cada evento é gerado a cada tick do timer, isto é, 60 vezes por segundo (60 FPS), em que os objetos, ou melhor, os bitmaps a que estes estão associados são desenhados num segundo buffer, buffer auxiliar, que será copiado para um buffer principal e serão, então, desenhados o ecrã a cada tick do timer.

Na “class” bullets utilizamos ainda o conceito de listas ligadas, alocando ou desalocando memoria sempre que uma bala é criada uma vez que o numero de balas do astronauta tem um valor dinâmico, apesar de ter um teto de vinte balas.

Apesar do RTC ser um periférico relativamente simples, devemos ter algum cuidado especial para evitar aparecimento de erros. No projeto fazemos poll da data e da hora, tendo especial cuidado a ler, verificando a flag UIP do registo A, isto para verificar se não esta a ocorrer alguma atualização. Para além disso, deve-se ter cuidado durante a leitura, uma vez que as interrupções devem estar desativadas.

Codigo Assembly

Foi realizado código assembly no timer para o interrupt handler, no keyboard para o interrupt handler e ainda no rtc para verificar a flag uip.

Conclusão

A cadeira de LCOM foi a mais desafiante até agora, as também gratificante. Deste modo há, portanto aspetos mais positivos e outros negativos.

Os aspetos positivos e que achamos que deverão ser mantidos são: a metodologia usada, em forma de Labs e a quantidade de informação que nos ensinam e a maturidade que nos é dada, quer pela elevada autonomia, como pela grande quantidade de informação

Contudo, há alguns aspetos que poderiam ser abordados doutra maneira. Apesar de nos irmos habituando ao modo como tudo funciona, o início acaba por ser um grande choque por ser uma cadeira muito diferente de todas as outras até à data. Gostaríamos que os labs saíssem a tempo para aprendermos com os erros ou sabermos o que devemos manter. Os créditos da cadeira não refletem o trabalho necessário comparado com outras cadeiras. Seria algo positivo se alguns temas como ler, escrever em ficheiros, ou que alguns temas fossem abordados com maior profundidade a organização e estruturação em diferentes camadas, usando states machine no projeto.