2018/19

Relatório do Projeto Final

Laboratório de Computadores

Projeto Final de LCOM

TURMA 1 – GRUPO 7

MESTRADO INTEGRADO EM ENGENHARIA INFORMÁTICA E COMPUTAÇÃO

João Pedro Pinheiro de Lacerda Campos UP201704982

Nuno Miguel Teixeira Cardoso UP201706162

Índice

Figuras

1.Tema do trabalho

O objetivo geral do nosso projeto final da unidade curricular é a criação de um jogo do género *side-scrolling shooter*, que tire partido do uso intensivo dos periféricos disponibilizados. Tendo isto em vista, desenvolvemos um pequeno jogo, que permite ao utilizador envolver-se num mundo de dragões, lutando contra inimigos, disparando mísseis e movendo-se pelo ecrã para alcançar a maior pontuação possível.

O jogo foi produzido recorrendo à linguagem de programação C e divide-se em vários módulos, correspondentes aos respetivos periféricos utilizados.

2.Instruções de utilização do programa

2.1. Ecrã Inicial

Figura 1 - Ecrã Inicial

Ao iniciar o programa é apresentado ao utilizador um pequeno menu de três opções evidentes. No canto superior esquerdo, é possível verificar a data atual e um relógio que vão atualizando. O nome do jogo *Nether Dragon* aparece mais ao centro em letras bem legíveis e de cor amarela. Mais a baixo e do lado direito encontra-se uma imagem do Dragão Nether. O utilizador pode percorrer as três opções de escolha através das teclas das setas no teclado, sendo que a opção selecionada no momento adquirirá uma borda vermelha, tal como é exemplificado na figura com o botão "Jogar". Para escolher uma das opções, basta carregar na tecla *ENTER*. De seguida são explicadas as três opções:

Botão que permite o acesso ao jogo propriamente dito.

Botão que permite o aceso ao guia de instruções para o jogo.

Botão que permite sair do programa. Além deste botão, a tecla *ESCAPE* também garante a mesma função.

2.2. Guia de Instruções

Figura 2 - Guia de Instruções

Após escolher a segunda opção, o utilizador é redirecionado para o "Guia de Instruções".

Nesta página é possível visualizar as instruções básicas e execuções necessárias para jogar *Nether Dragon*. Para regressar ao menu principal, tal como é mencionado no guia, é necessário carregar num das setas: para cima ou para baixo, do teclado. Por outro lado, a tecla *ESCAPE* mantém a sua função de fechar o programa. Qualquer outra tecla ou movimento não surtirá efeito algum.

Após regressar ao menu principal, será novamente apresentado ao utilizador o ecrã inicial com as mesmas opções de então. De seguida, após carregar na opção de jogar, o utilizador receberá um ecrã com uma imagem de um fundo, com uma barra preta em baixo onde poderá visualizar o número de vidas e a sua pontuação. A sua pontuação irá aumentar à medida que o tempo passa e também caso destrua inimigos.

Figura 3 - Barra de pontuação e vidas

2.3. Jogabilidade

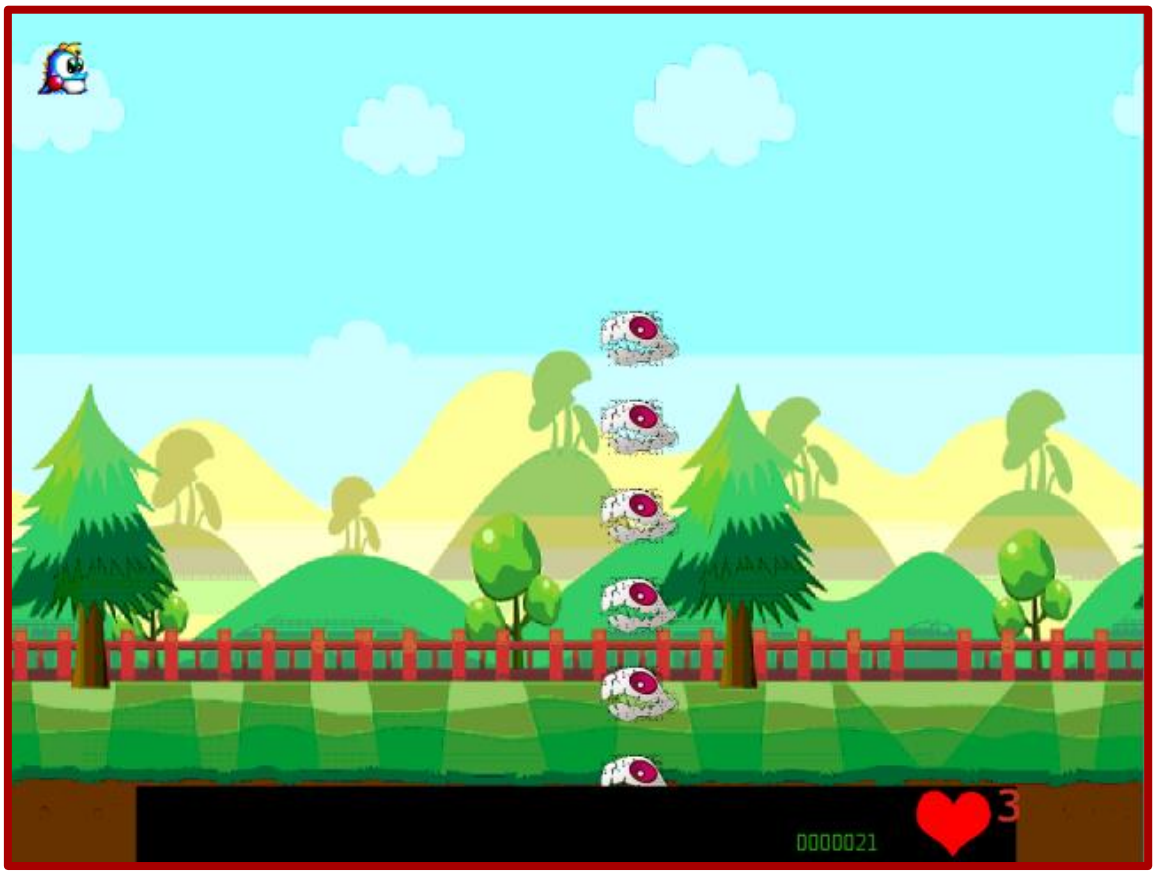

Figura 4 - Ecrã do Jogo

O utilizador movimenta o dragão *Nether* ao longo da tela através do movimento do rato ou das setas do teclado. Deverá evitar colidir com os inimigos, caso contrário irá explodir e perder uma vida. Quando quiser, para se defender, poderá lançar mísseis através da boca do dragão. Para tal, basta carregar na tecla *SPACE* do teclado ou, se preferir, no botão esquerdo do rato. Mantendo o *SPACE* ou o botão esquerdo do rato premidos durante aproximadamente dois segundos, o míssil azul transforma-se num míssil maior e amarelo. Enquanto que os mísseis azuis e pequenos se destroem ao colidir com um inimigo, os mísseis amarelos e maiores percorrem toda a tela até atingirem a borda do ecrã, destruindo todos os inimigos no seu caminho. Porém, enquanto que se podem efetuar múltiplos disparos azuis, quando é disparado um míssil amarelo o utilizador fica impossibilitado de disparar um novo míssil até que o anterior chega ao fim da largura da tela.

O jogo tem três níveis de dificuldade, regidos pela pontuação. O jogo começa com o nível fácil, onde é lançada apenas uma linha de inimigos de cada vez. Quando a pontuação alcança um determinado patamar, os inimigos começam a ser lançados em duas linhas de invasão. Por fim, após o dobro dos pontos anteriores, dá-se início ao nível difícil, onde além das duas linhas de inimigos, é também lançado um outro inimigo capaz de aparecer numa posição aleatória. Cada míssil azul causa um dano de um valor nos inimigos. As vidas dos inimigos podem variar entre uma e duas durante os níveis fácil e médio, sendo sempre três no nível difícil.

Figura 5 - Personagem do utilizador

Figura 6 - Inimigo

Figura 7 - Explosão do Dragão

2.4. Menu de pausa e fim do jogo

Além de todas as funcionalidades apresentadas anteriormente, se durante o jogo o utilizador sentir a necessidade de parar, a tecla *ESCAPE* cede-lhe essa oportunidade. Ao pressionar esta tecla abre-se um menu de pausa com duas opções: "Jogar" ou "Sair". A opção de "Jogar" permite continuar o jogo, enquanto que a opção de "Sair" leva ao regresso ao menu principal. Este menu funciona de forma semelhante ao menu principal, na medida em que a escolha das opções se processa através do teclado, das setas e do *ENTER*. Ao colocar o jogo em pausa o utilizador poderá ainda ver a data e hora do início da sua sessão de jogo e a data e hora atuais. Caso o utilizador pretenda sair do jogo sem ter perdido todas as suas três vidas e regressar ao menu principal, se optar por jogar novamente, retoma o jogo anterior. A sessão é guardada num ficheiro ao sair do jogo.

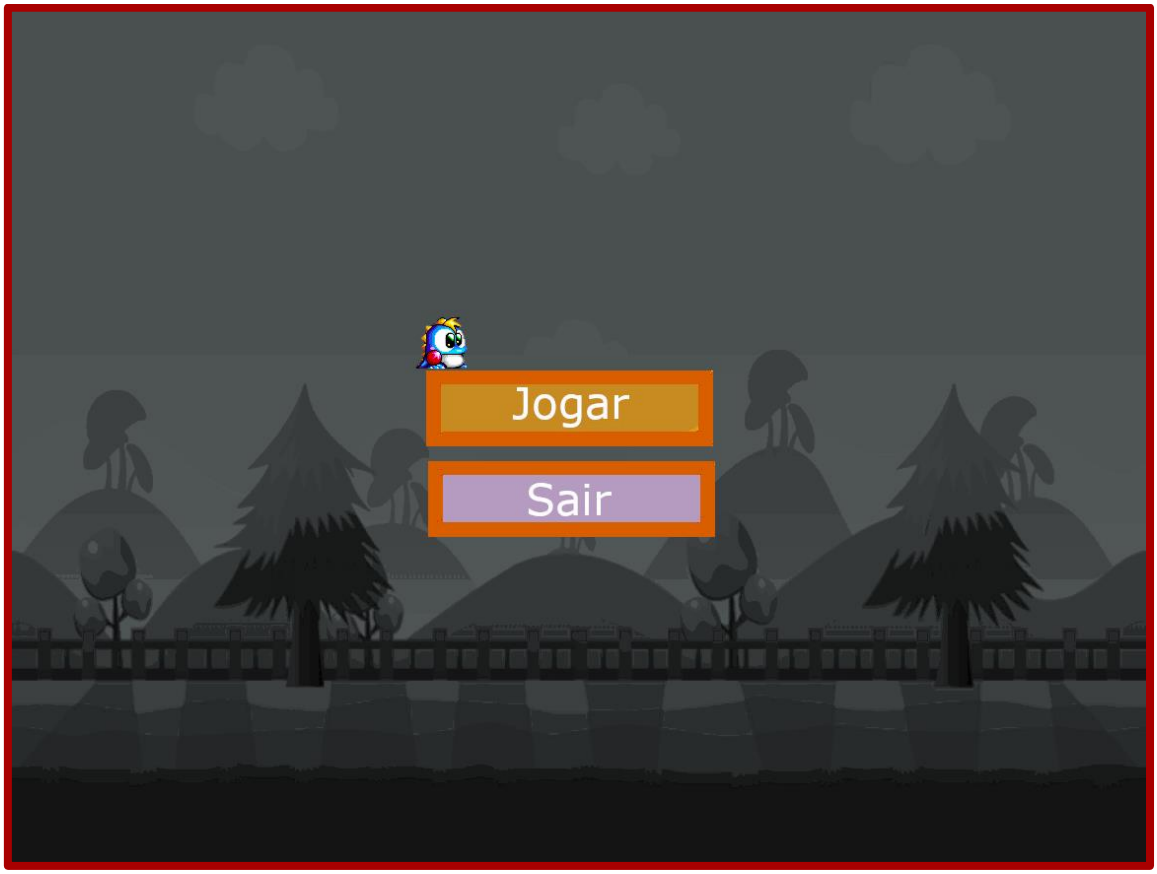

Figura 8 - Menu de pausa

Existe ainda o menu de *game over*, que será apresentado ao utilizador caso este chegue ao fim das suas três vidas. O utilizador perde vidas de cada vez que colide com um inimigo e explode. Enquanto a explosão persistir, o utilizador não poderá perder mais vidas, nem disparar mísseis, mas poderá mexer-se pela tela. Caso o dragão esteja a preparar um míssil e morrer durante essa preparação o tiro será cancelado. Neste menu é igualmente apresentada a data e a hora como no menu de pausa. Para sair do menu *game over* basta carregar numa das setas para cima ou para baixo, ou então na tecla *ESCAPE*.

Figura 9 - Menu de fim do jogo

3.Periféricos e suas funcionalidades

Neste projeto procuramos tirar o máximo partido de todas as funcionalidades dos periféricos que nos foram apresentados em aula. Em todos recorremos ao modo por interrupções e não por *polling*. De seguida é apresentada uma tabela onde a primeira coluna representa o periférico utilizado e a segunda coluna representa a sua função no projeto.

Periférico	Função no projeto
Timer	Gerir a velocidade do movimento de todos os objetos. Temporizar a geração de inimigos. Temporizar e gerir as incrementações da pontuação. Contar o tempo que os botões estão premidos para gerar mísseis maiores. Contar o tempo de explosão.
Keyboard	Escolha das opções nos menus. Mover o dragão ao longo da tela. Gerar balas e disparar. Colocar o jogo em pausa. Sair do jogo. Navegar entre menus.
Mouse	Mover o dragão ao longo da tela. Gerar balas e disparar.
Video card	Apresentar na tela todos os objetos: fundo, inimigos, dragão, mísseis pequenos, mísseis grandes, menus, botões, vidas, pontuação, data e hora.
Real Time Clock	Dar a data e hora atuais.

Figura 10 - Periféricos e suas funções

3.1. *Timer*

Tal como já foi mencionado anteriormente, o *Timer* ou Contador foi usado para gerir a velocidade de movimento de todos os objetos. No ciclo de interrupções da função escape_or_move(), a cada interrupção do *Timer* recebida é executada uma outra função draw_buffer() que chama várias outras funções que tratam do movimento do jogador (move_player()), dos inimigos (move_enemy()) e dos mísseis (move_shot()). Estas funções, por sua vez, fazem somar à posição atual do xpm a velocidade contida nos membros dados *speed* de cada *struct,* promovendo o movimento dos objetos.

A cada interrupção do *Timer* é incrementada uma variável *counter*. A cada duzentas e cinquenta interações é chamada a função enemies_handler() que gera uma onda de inimigos. Deste modo, é através do *Timer* que se consegue temporizar a chegada de novos inimigos. Além disso, caso nenhum dos menus esteja aberto, a cada interrupção do *Timer* é incrementada a variável *score,* levando ao incremento da pontuação adquirida pelo utilizador.

Existem ainda outras duas variáveis *time_dead* e *time_pressed* que são geridas por este periférico de modo a contar o tempo que o dragão está em modo de explosão até voltar ao normal e a contar o tempo que determinado botão é pressionado de forma a que sejam gerados mísseis maiores.

Além destas funcionalidades, o *Timer* regula ainda em cada iteração a gestão das execuções caso a variável *data* seja um scancode da tecla *SPACE,* a atualização do buffer (update_buffer()), a gestão da temporização da apresentação dos menu, das pontuações, das datas e da verificação das colisões (verifyCollision()) e a criação de novos mísseis.

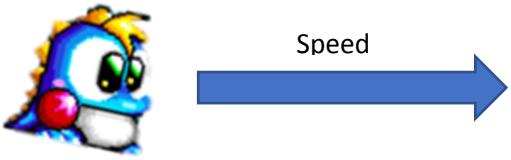

Figura 11 - Funções do Timer

3.2. *Keyboard*

Os dados provenientes do *keyboard* são analisados no código pela função kbd_data_handler(), que analisa o tamanho dos *scancodes* (um ou dois *bytes*) e gere os bytes, colocando-os num *array*.

O *keyboard* ou teclado teve o seu papel principal na navegação entre menus. Através de um ciclo *switch case* foi possível atribuir uma funcionalidade a cada tecla. Nos menus, as teclas das setas para cima e para baixo são essenciais para a escolha do botão correspondente à escolha. Esta escolha está no código transformada numa variável *option*, cujo valor começava inicialmente a zero e, consoante a seta pressionada, ia aumentando (seta para baixo) ou diminuindo (seta para cima) o seu valor. A escolha é selecionada através da tecla *ENTER*.

Além desta funcionalidade, o mesmo *switch case* permite que quando em modo de jogo, o utilizador possa movimentar o seu personagem ao longo da tela com as setas para cima, baixo, lado esquerdo e lado direito, ou ainda disparar mísseis com a tecla *SPACE* e colocar o jogo em pausa ou sair com a tecla *ESCAPE*.

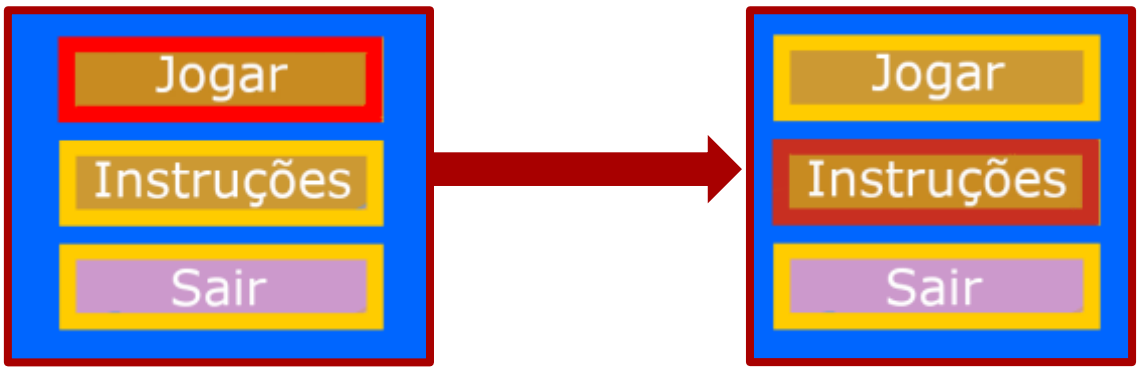

Figura 12 - Escolha de opções

3.3. *Mouse*

Os dados provenientes do *mouse* são analisados no código pela função mouse_data_handler(), que se encarrega de colocar cada *byte* de um *packet* na respetiva posição de um *array* e atualizar os membros dado de uma *struct*.

Essencialmente, o *mouse* ou rato é usado no projeto para tornar o jogo mais transversal e dar rapidez de movimentos ao utilizador. Com o rato, o utilizador pode mover de forma mais rápida e intuitiva o seu personagem e disparar mísseis carregando no botão esquerdo. Para tal implementamos uma pequena máquina de três estados, correspondentes ao estado inicial de repouso, ao estado de criação de um míssil e ao estado de libertação desse míssil. Através destes estados, o rato desempenha as mesmas funcionalidades que o teclado executa durante o modo de jogo.

3.4. *Video Card*

A função da *Video Card* ou placa gráfica é a de mostrar na tela todos os objetos e fundo, ou seja, a parte visual do projeto. No projeto usamos o modo de vídeo 0x14B de resolução 1152x864 pixels, em *direct color mode* de 256x256x256 = 16777216 cores. Implementamos *second buffering* de modo a diminuir conflitos.

Através das funções iniciadas pelo prefixo *draw* ou *print* (*draw_xpm, draw_menu, draw_tecla, draw_buffer, draw_buffer_background, draw_xpm_noclip, print_number, print_date)* conseguimos assegurar o movimento de objetos como o dragão, os inimigos, as balas e o fundo, animações como o dragão abrir a boca e a explodir e a bola azul a se transformar numa bola amarela maior, e colisões entre os inimigos, as bolas e o dragão, além das colisões do dragão com as margens.

A placa gráfica permitiu ainda a apresentação das pontuações, datas, horas e menus no ecrã, bem como as animações de mudança de botão selecionado.

Figura 13 - Placa gráfica: animações

3.5. *Real Time Clock*

O *Real Time Clock* é usado para obter a data e hora atuais, de modo a possibilitar a apresentação ao utilizador da data e hora de início da sessão de jogo, bem como a data e a hora atuais tanto no menu principal, como no menu de pausa e de fim do jogo.

São as funções *get* que permitem a passagem para variáveis destes valores e, posteriormente, permitem à placa gráfica mostrar ao utilizador a data e hora pretendidas.

Figura 14 - RTC

4.Estrutura do código

O código encontra-se estruturado por módulos, correspondentes essencialmente aos vários periféricos. No projeto desenvolvemos principalmente o módulo gráfico e o módulo do *RTC*, uma vez que os restantes módulos já tinham sido implementados ao longo do semestre durante os *LABS,* sofrendo apenas ligeiras alterações.

4.1. Módulo do *Timer*

Este módulo representa cerca de 20% do peso do projeto. O ficheiro *timer.c* contém, essencialmente, uma função capaz de estabelecer o valor da frequência de determinado contador de entre os três existentes (*timer_set_frequency()*), uma função para subscrever as interrupções do contador 0 (*timer_subscribe_int()*) e outra para dessubscrever (*timer_unsubscribe_int()*), uma função que incrementa o contador responsável pela contagem do tempo (*timer_int_handler()*), uma função que permite obter a configuração de um contador (*timer_get_conf()*) e outra que permite mostrar essa configuração (*timer_display_conf()*). Em termos do que cada um fez neste módulo, uma vez que não houve alterações significativas, mantém-se a autoavaliação apresentada no *LAB 2*.

50% - João Campos / 50% - Nuno Cardoso.

4.2. Módulo do *Keyboard*

Este módulo representa cerca de 20% do peso do projeto. O ficheiro *kb.c* contém, tal como o anterior, funções de subscrição (*kb_subscribe_int()*) e de dessubscrição (*kb_unsubscribe_int()*) de interrupções. A função *sys_inb_cnt()* destina-se a executar a função *sys_inb()* e incrementar um contador, de modo a ser possível contar o número de vezes que é utilizada. A função *kbc_ih()* destina-se a tratar da gestão das interrupções e a função *kbd_data_handler()* verifica se os *scancodes* são de um ou dois *bytes* e coloca-os num *array* na posição devida. Existem ainda duas funções *util_get_LSB()* e *util_get_MSB()* que servem para obter, respetivamente, os *bits* menos e mais significativos de uma variável de dois *bytes*. A função *enable_interrupts()* ativa as interrupções do teclado, através de comandos enviados para o registo de *status*.

Em termos do que cada um fez neste módulo, uma vez que não houve alterações significativas, mantém-se a autoavaliação apresentada no *LAB 3*.

50% - João Campos / 50% - Nuno Cardoso.

4.3. Módulo do *Mouse*

Este módulo representa cerca de 15% do peso do projeto. O ficheiro *mouse.c* contém funções que não foram utilizadas neste projeto, mas que foram necessárias para o *LAB 4.* Assim, apresentaremos aqui apenas aquelas que foram necessárias para a boa execução do projeto. Este módulo contém funções de subscrição (*mouse_subscribe_int()*) e de dessubscrição (*mouse_unsubscribe_int()*) de interrupções. A função *mouse_ih()* destina-se a tratar da gestão das interrupções e a função *mouse_data_handler()* encarrega -se de colocar cada *byte* de um *packet* na respetiva posição de um *array* e atualizar os membros dado de uma *struct packet.* As funções *enable_data_report()* e *disable_data_report()* destinam-se a ativar e desativar a receção de dados provenientes do rato.

Em termos do que cada um fez neste módulo, uma vez que não houve alterações significativas, mantém-se a autoavaliação apresentada no *LAB 4*.

40% - João Campos / 60% - Nuno Cardoso.

4.4. Módulo da *Video Card*

Este módulo representa cerca de 30% do peso do projeto. É neste módulo que se encontra a função principal (*escape_or_move()*) que origina o jogo. Esta função contém o ciclo de interrupções e os modos de jogo e menus. No ficheiro graphi.c existem várias funções relacionadas com jogabilidade propriamente dita:

- vg_init() – estabelece o modo gráfico da consola.

- ret_graphics_mode() – função auxiliar chamada pela função vg_init().

- draw_xpm_noclip() – função que desenha xpm capazes de ultrapassar as boradas do ecrã.

- number_to_array() – passar um número para um *array de chars.*

- print_number() – função que imprime números no ecrã.

- print_date() – função que imprime as datas no ecrã com o formato dia/mês/ano hora/minuto/segundo.

- draw_xpm() – função que desenha xpms no ecrã.

- update_buffer() – função que chama *memcpy()* para copiar o *second_buffer* para o

video_mem assegurando a atualização dos dados.

- sin_pattern() – função que calcula a posição do inimigo segundo uma função seno.

- cross_pattern() – função que calcula a posição do inimigo segundo funções afins que mudam de declive quando colidem com as bordas.

- move_enemy(), move_player(), move_shot() – funções que asseguram o movimento dos inimigos, jogador e mísseis, respetivamente.

- draw_buffer_background() – desenha o fundo no second_buffer.

- draw_buffer() – desenha (por ordem) fundo, jogador, inimigos, mísseis e barra inferior.

- verifyCollision() – verifica colisões entre mísseis e inimigos.

- draw_menu() – desenha xpms relativos a menus.

- draw_tecla() – desenha a tecla que se encontra selecionada no momento pelo utilizador.

- remove_life() – retira vidas ao jogador e torna-o incapaz de disparar enquanto estiver morto.

- enemies_handler() – função que faz a gestão da geração de inimigos e da dificuldade atual do jogo.

- update_highscore() – função que verifica num ficheiro as pontuações mais elevadas e atualiza-as caso a nova pontuação seja superior a uma delas.

- save_game() – guardar a sessão de jogo.

- load_game() – carregar o jogo a partir do ficheiro.

- escape_or_move() – função principal do programa, responsável pelo controlo do jogo, pelo que trata de todas as interrupções dos periféricos, da gestão dos eventos do teclado e do rato. Esta função é responsável por executar a tarefa necessária consoante determinado evento ocorrido durante a execução do programa.

As funções deste módulo, em especial, foram todas desenvolvidas cooperativamente.

60% - João Campos / 40% - Nuno Cardoso.

4.5. Módulo do *RTC*

Este módulo representa cerca de 10% do peso do projeto. O ficheiro *rt.c* contém funções de subscrição (*rtc_subscribe_int()*) e de dessubscrição (*rtc_unsubscribe_int()*) de interrupções. As funções *set_rtc_int()* e *unset_rtc_int()* servem, respetivamente, para ligar e desligar as interrupções do *RTC*, enquanto que a *clear_reg_c()* limpa as *flags* no registo C. A função *convert_BCD()* converte uma variável em BCD para Binário e retorna-a. Além destas funções, contém ainda funções de prefixo *get* que permitem obter os valores dos dias, meses, anos, horas, minutos e segundos.

Este módulo foi implementado na totalidade pelo aluno João Campos.

4.6. Restantes módulos

Os módulos *enemy, player e shot* representam cerca de 5% do peso do projeto. Nos ficheiros *source* deste módulo tratamos da criação e eliminação dos diferentes objetos do jogo.

50% - João Campos / 50% - Nuno Cardoso.

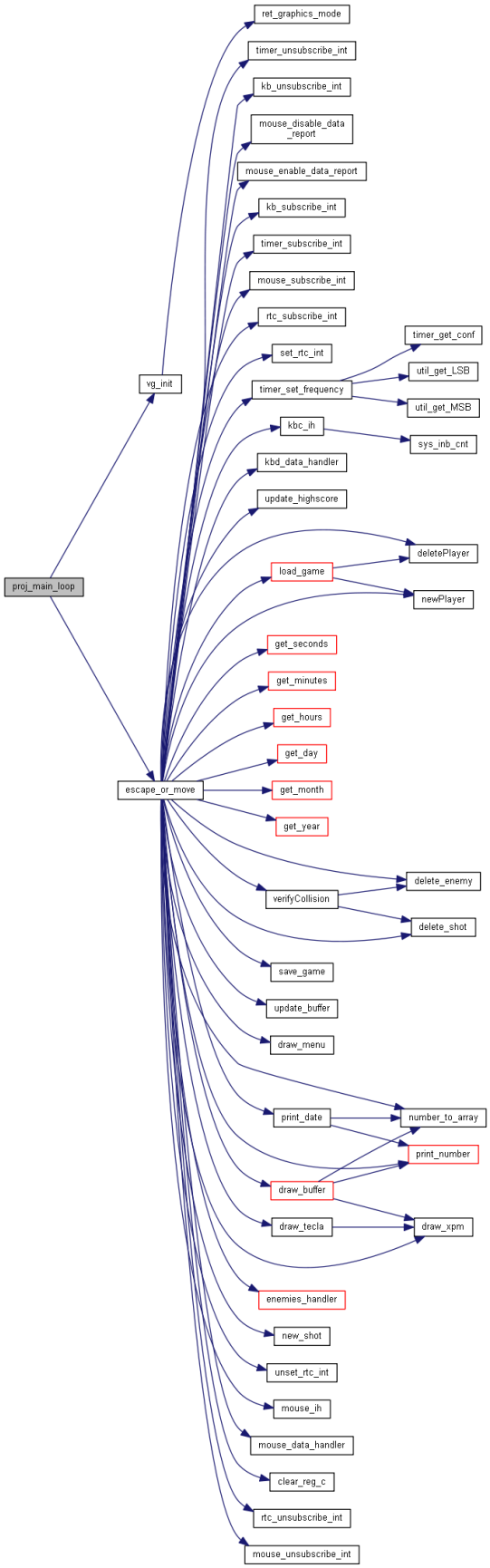

Figura 15 - Gráfico de Funções

5.Detalhes de Implementação

Para tornar mais legível o código, criamos uma máquina de três estados para o rato, correspondentes ao estado inicial de repouso, ao estado de criação de um míssil e ao estado de libertação desse míssil.

De forma a guardar as pontuações mais altas e a sessão de jogo utilizamos ficheiros. Visto que não nos foi ensinado como trabalhar com ficheiros em linguagem C, vimo-nos obrigados a aprender através de pesquisas na internet.

6.Conclusões

Concordamos que uma das falhas na disciplina de Laboratório de Computadores está no facto de a *serial port* não ser abordada nas aulas práticas, pois consideramos ser um periférico de difícil implementação.

Apesar deste pequeno problema, consideramos que no geral a disciplina está bem organizada.