

Relatório: Trabalho Prático Laboratório de  
Computadores

Grupo 4 Turma 2

João Luz up201703782 e Francisco Rodrigues up201808909

7 de Janeiro de 2019

# Conteúdo

<b>1 Instruções de Utilização</b>	<b>4</b>
1.1 Menu de Username . . . . .	4
1.2 Menu principal . . . . .	4
1.3 Jogo . . . . .	6
1.4 Menu de High Scores . . . . .	10
<b>2 Estado do projeto</b>	<b>11</b>
2.1 Dispositivos de I/O utilizados . . . . .	11
2.2 Timer . . . . .	11
2.3 Teclado . . . . .	11
2.4 Rato . . . . .	12
2.5 Placa Gráfica . . . . .	12
2.6 RTC . . . . .	12
<b>3 Estrutura do código</b>	<b>13</b>
3.1 Game . . . . .	13
3.2 Graphics . . . . .	13
3.3 i8042 . . . . .	14
3.4 i8254 . . . . .	14
3.5 vbe_macros . . . . .	14
3.6 Timer . . . . .	14
3.7 Keyboard . . . . .	14
3.8 Mouse . . . . .	15
3.9 Proj . . . . .	15
3.10 Menu . . . . .	15
3.11 File . . . . .	16
3.12 RTC . . . . .	16
<b>4 Detalhes de implementação</b>	<b>19</b>
<b>5 Conclusão</b>	<b>22</b>
<b>6 Instruções de instalação</b>	<b>23</b>

# **Lista de Figuras**

1.1	Username . . . . .	4
1.2	Menu do jogo . . . . .	5
1.6	Instruções do jogo . . . . .	5
1.3	Menu do jogo com a opção Start selecionada . . . . .	6
1.4	Menu do jogo com a opção High Scores selecionada . . . . .	7
1.7	Pausa do jogo . . . . .	7
1.5	Menu do jogo com a opção Exit selecionada . . . . .	8
1.8	Jogador Atingido . . . . .	8
1.9	Game over . . . . .	9
1.10	Alien Atingido . . . . .	9
1.11	Game Won . . . . .	10
1.12	High Scores . . . . .	10
3.1	Peso dos diferentes módulos no projeto . . . . .	17
3.2	Call Graph . . . . .	18

# **Lista de Tabelas**

2.1	Tablela de I/O	11
-----	----------------	----

# Capítulo 1

## Instruções de Utilização

### 1.1 Menu de Username

Ao iniciar o programa é apresentado ao utilizador o menu que lhe permite escolher o seu nome de utilizador. O nome de utilizador pode ter apenas três caracteres. Depois de escolher o nome o utilizador deve clicar na tecla **ENTER**. Caso o utilizador se engane ao introduzir o nome pode corrigi-lo clicando na tecla **BACKSPACE**.

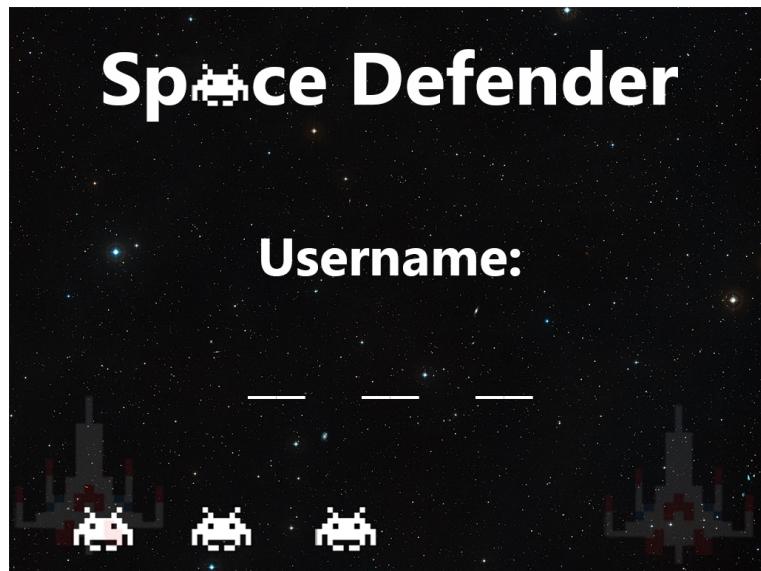


Figura 1.1: Username

### 1.2 Menu principal

Após introduzir o seu nome de utilizador este é direcionado para o menu do jogo. Aqui o utilizador tem a opção de iniciar o jogo, ver os highscores

ou sair do jogo. O utilizador pode escolher a opção que pretende selecionar posicionando o cursor sobre a opção e pressionando o botão esquerdo.



Figura 1.2: Menu do jogo

A partir deste menu podem ainda se apresentadas as instruções do jogo premindo a tecla **H**.



Figura 1.6: Instruções do jogo



Figura 1.3: Menu do jogo com a opção Start selecionada

### 1.3 Jogo

O objetivo do jogo, é derrotar a nave inimiga disparando sobre ela e evitando os seus disparos, tentando obter a maior pontuação possível.

O jogo é constituído por três fases, com diferentes níveis de dificuldade que se distinguem pelo número de vidas da nave inimiga, pela sua velocidade e pelo impacto dos seus disparos sobre o jogador. O jogador é recompensado com 20 pontos sempre que um disparo seu acerta na nave, e com 50 pontos sempre que o número de vidas da nave inimiga (que é decrementado a cada impacto) atinge o valor zero. É, no entanto, penalizado sempre que não acerta no inimigo (até um valor mínimo de zero pontos) ou é atingido por um dos seus disparos, se for atingido quando a sua pontuação é zero o jogador perde o jogo sendo levado para o menu *Game Over* (“fim do jogo”). A pontuação obtida durante o jogo pode ser visualizada no canto superior esquerdo do ecrã à medida que o seu valor varia.

Para controlar as ações do personagem, o jogador dispõe do teclado e do rato. Por um lado, teclas **A** e **D** provocam o movimento da nave para a esquerda e para a direita respetivamente. Por outro, o botão esquerdo do rato é usado para disparar e o movimento lateral do rato com o botão direito premido “ativa” um escudo protetor que impede a penalização de um disparo inimigo.

Quando o número de vidas de um dos inimigos chega a zero, este desaparece, e o jogador tem de se desviar de dois asteroides que se aproximam da sua posição, a colisão com um asteroide leva uma penalização de 10 pontos que pode levar ao fim do jogo. Se os asteroides “saírem” para fora da tela, ou atingirem a nave sem que o jogo acabe, o jogador avança para fase seguinte ou ganha o jogo se ultrapassar a terceira fase levando ao menu de *You Won* (“venceu”). Neste ultimo caso, o seu nome, a sua pontuação, e a data são



Figura 1.4: Menu do jogo com a opção High Scores selecionada

registados num ficheiro que pode ser consultado na próxima sessão através do menu *Highscore* (“pontuação mais alta”).

Em qualquer momento durante uma partida, o jogador pode pressionar a tecla **P** para alternar entre o menu *Pause* (“pausa”) e o jogo. Pode também pressionar a tecla **ESC**, sendo levado para o menu principal perdendo todo o seu progresso até ao momento.

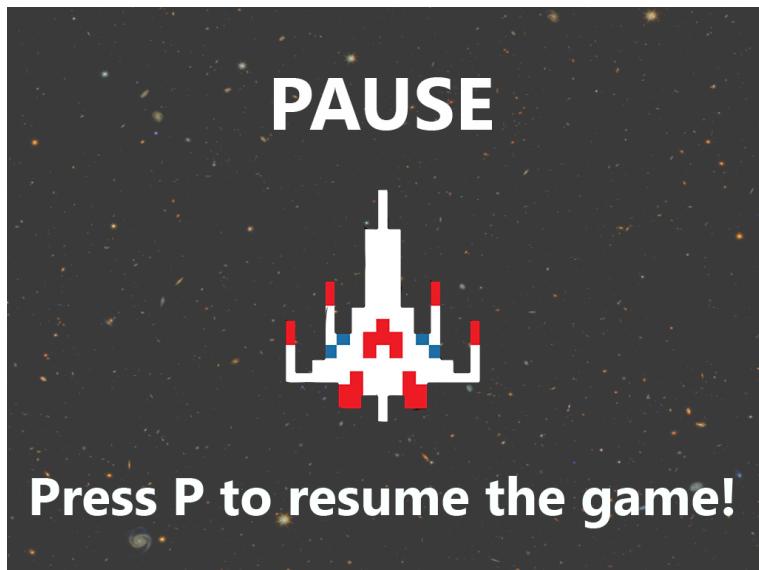


Figura 1.7: Pausa do jogo



Figura 1.5: Menu do jogo com a opção Exit selecionada

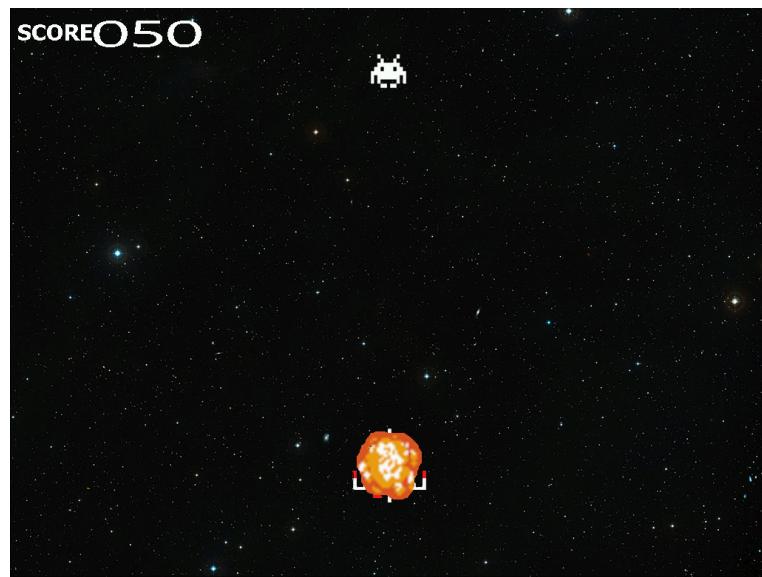


Figura 1.8: Jogador Atingido



Figura 1.9: Game over

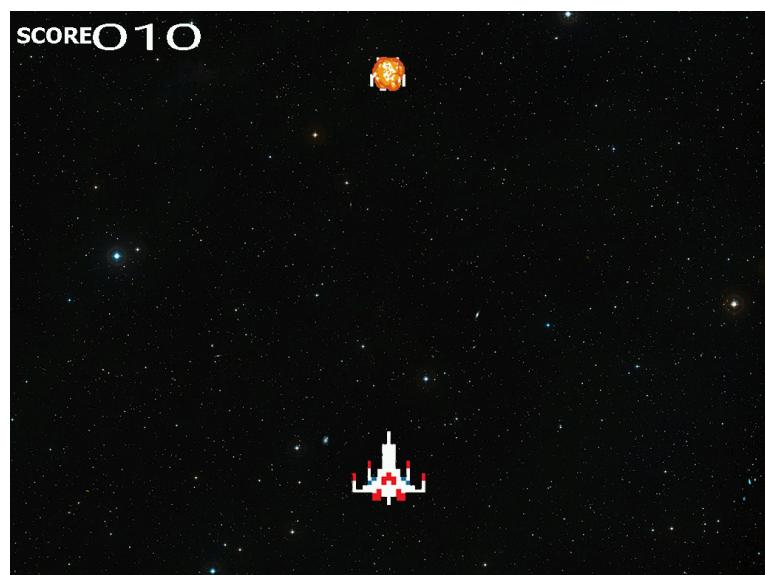


Figura 1.10: Alien Atingido

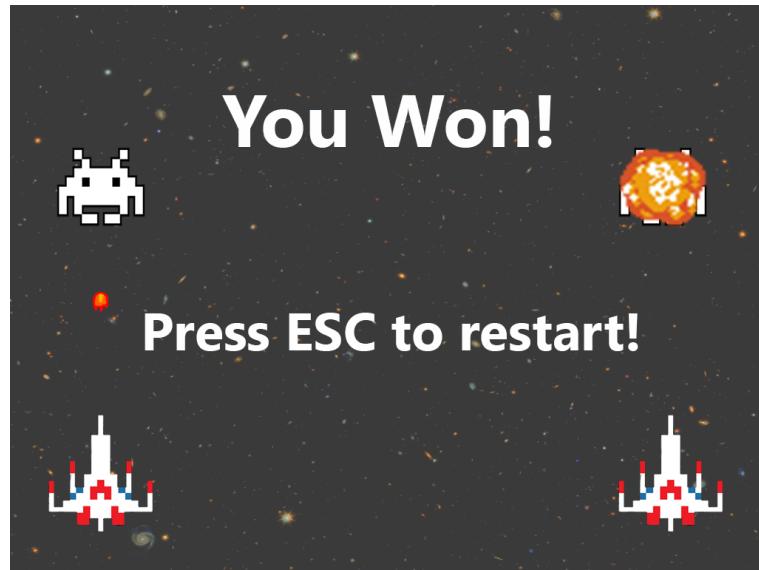


Figura 1.11: Game Won

#### 1.4 Menu de High Scores

Neste menu o utilizador pode observar as três melhores pontuações obtidas no jogo.

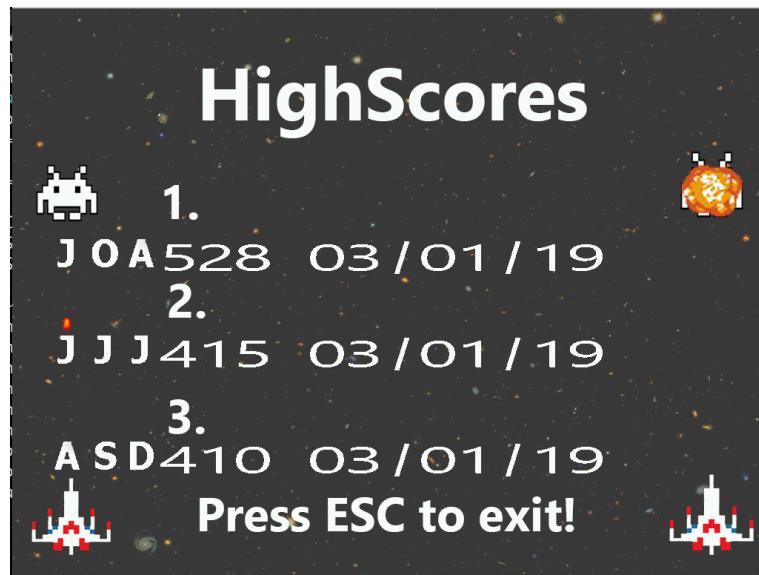


Figura 1.12: High Scores

# Capítulo 2

## Estado do projeto

### 2.1 Dispositivos de I/O utilizados

I/O devices	Utilização	Interrupts?
Timer	Animações	Sim
Teclado	Input de texto e controlo da personagem	Sim
Rato	Navegação no menu do jogo e controlo da personagem	Sim
Placa Gráfica	Desenho dos gráficos do jogo	Não
RTC	Leitura da data atual	Não

Tabela 2.1: Tablela de I/O

### 2.2 Timer

A principal função do timer é a atualização periódica do estado do jogo nomeadamente o desenho dos gráficos do mesmo. Isto faz do timer um dos dispositivos de I/O mais importantes (senão o mais importante) do jogo uma vez que as ações de outros dispositivos como o teclado e o rato só podem ser observadas através das interrupções do timer que provocam o seu desenho no ecrã.

- timer.c;

### 2.3 Teclado

Este dispositivo é utilizado, em conjunto com o rato, na lógica do jogo para alternar entre os seus estados. Nos ”menus” de *Instruções*, *Game Over* e *You Eon* a tecla **ESC** deve ser premida para voltar ao menu principal do jogo. Adicionalmente, durante o jogo o mesmo comando pode ainda ser executado

para desistir. Por fim pode ser pressionada a tecla **P** para entrar e sair do "menu" de *Pause*.

Para além disto o teclado é ainda usado durante o jogo para controlar o movimento da personagem nomeadamente premindo a tecla **A** para a deslocar para a esquerda e a tecla **D** para a movimentar para a direita.

- keyboard.c;
- game.c - *void move\_ship()*, *void show\_letter\_byte()*;

## 2.4 Rato

O rato, tal como o teclado, é usado na lógica do jogo. É com o rato que o utilizador navega através do menu principal podendo escolher entre iniciar um novo jogo, ver os *High Scores* ou sair do programa.

Adicionalmente, o rato é usado durante o jogo para disparar, premindo o botão esquerdo, e para ativar o "escudo protetor" da personagem. Para isto deve ser realizado com o rato um gesto horizontal com o botão direito do rato premido.

- mouse.c;
- menu.c - *void update\_position()*;

## 2.5 Placa Gráfica

A Placa Gráfica é um dispositivo extremamente importante pois é utilizado para desenhar os gráficos do jogo. O modo gráfico que utilizamos foi o modo **0x117** com resolução 1024x768 e com **270** cores. Por fim, as imagens que escolhemos para o programa são em formato **XPM**, modo RGB(5:6:5).

Para garantir a fluidez do jogo optamos por implementar a técnica de *Double Buffering* que copia de uma só vez o estado do ecrã para a memória virtual procurando garantir a sincronia das animações.

O uso da placa gráfica é também importante para a verificação de colisões entre objetos. Esta é feita recorrendo ao conteúdo da memória virtual.

São ainda usadas imagens de letras e números (fontes) para a apresentação de elementos escritos durante o jogo.

- graphics.c;
- game.c - *void drawJogo()*;

## 2.6 RTC

O RTC é usado para ler a data atual.

Assim, decidimos usar esta informação para guardar um registo das datas e associá-las às pontuações obtidas pelos jogadores nas diferentes partidas. Esta informação é depois utilizada para apresentar as três melhores pontuações obtidas no "menu" do *High Scores*.

- RTC.c;

# Capítulo 3

## Estrutura do código

### 3.1 Game

É neste módulo que são desenvolvidas as principais características do jogo, definidas principalmente pelas structs **Jogo**, **Player**, **Alien**, **Asteroid** e pela máquina de estados `game_state`, assim como algumas variáveis globais sendo todas estas inicializadas, utilizadas e manipuladas pelas várias funções desenvolvidas no módulo.

É também aqui que é feita a maioria das chamadas de funções relativas aos periféricos implementados. A manipulação destas funcionalidades, fez com que fosse possível, associar inputs do rato e do teclado fornecidos pelo utilizador às ações do personagem principal assim como criar animações de objetos que apresentam um comportamento autónomo, utilizando o **timer**. Por fim os efeitos deste conjunto de funções são mostrados ao utilizador através das funcionalidades da placa gráfica que permite obter uma representação visual dos acontecimentos do jogo.

As funções principais deste módulo (indispensáveis ao funcionamento do mesmo) são:

- `inicio()`, `playerInit()`, `alienInit()` e `asteroidInit()` que inicializam o jogo e os seus objetos do jogo;
- `kbd_read()` e `move_ship()` que permitem alterar a posição do personagem utilizando o teclado;
- `player_fire()` que associa os inputs do rato aos disparos da nave;
- `drawJogo()` que é responsável pelo retorno de informação visual para o utilizador associando os acontecimentos provocados pelo mesmo às mudanças que ocorrem de forma aparentemente autónoma;

Este módulo foi desenvolvido em conjunto com igual contribuição de ambos os elementos do grupo (50%/50%).

### 3.2 Graphics

Neste módulo, são implementadas as funções que permitem o uso das capacidades da placa gráfica. Estas funções realizam o mapeamento da video

memory, a técnica de *Double Buffering*, o desenho das imagens nas coordenadas desejadas e a verificação de colisão entre objeto recorrendo ao conteúdo na memória – que permite a colisão entre pixéis de cores específicas ignorando a transparência das imagens.

Este módulo é assim constituído por 5 funções:

- vbe\_info() e map\_vram() que realizam o mapeamento;
- double\_buffering() que copia conteúdo para a video\_mem;
- vg\_draw\_xpm() que coloca na posição desejada do double buffer uma imagem;
- check\_colision() que verifica a colisão de um objeto de dimensões especificadas pelos argumentos;

Este módulo foi desenvolvido em conjunto com igual contribuição de ambos os elementos do grupo (50%/50%).

### **3.3 i8042**

Desenvolvido durante as aulas práticas (LABS) 3, 4 e 5 e posteriormente adaptado às necessidades do trabalho. Contém definições importantes relativas ao teclado e ao rato.

### **3.4 i8254**

Desenvolvido durante a aula prática (LABS) 2 e posteriormente adaptado às necessidades do trabalho. Contém definições importantes relativas ao timer.

### **3.5 vbe\_macros**

Desenvolvido durante a aula prática (LABS) 5 e posteriormente adaptado às necessidades do trabalho. Contém definições importantes relativas à placa gráfica.

### **3.6 Timer**

Desenvolvido durante a aula prática (LABS) 2 e posteriormente adaptado às necessidades do trabalho.

### **3.7 Keyboard**

Desenvolvido durante as aulas práticas (LABS) 3 e posteriormente adaptado às necessidades do trabalho.

### **3.8 Mouse**

Desenvolvido durante as aulas práticas (LABS) 4 e posteriormente adaptado às necessidades do trabalho. Neste módulo foram elaboradas funções para o projeto que gerem os inputs do rato e permitem utilizar o rato como cursor no menu principal, para disparar e proteger a nave do jogador utilizando não só os botões como o deslocamento do rato. Foi neste módulo também implementada uma máquina de estados para verificar o gesto específico que o jogador deve realizar para concretizar a ação de proteger o personagem.

São de especial destaque as funções:

- `check_line()` `set_mouse_events()` - importantes para a análise da variação dos inputs provenientes do rato e do seu efeito tanto na máquina de estados do rato como na do menu principal;
- `protect_tolerance()` – verifica se o gesto realizado respeita uma tolerância (que procura um deslocamento aproximadamente horizontal) previamente definida para averiguar se o gesto surte algum efeito;

Este módulo foi desenvolvido em conjunto com igual contribuição de ambos os elementos do grupo (50%/50%).

### **3.9 Proj**

Este módulo, constitui, tal como o módulo Game, um dos mais importantes pois é também responsável por “conciliar” as várias interrupções dos periféricos implementados chamando as funções necessárias dos diferentes módulos. É constituído principalmente por dois ciclos de interrupções de distintos um para o menu principal e para os menus neste contidos e outro para as ocorrências durante a partida.

São assim importantes realçar as funções:

- `menu_interrupt_loop()` – ciclo de interrupções que devem ocorrer durante a navegação dos menus;
- `(interrupt_loop)()` - ciclo de interrupções que devem ocorrer durante o jogo;
- `(proj_main_loop)()` – que inicializa o programa e que permite a alternância entre os diversos estados da máquina de estados e a chamada do respetivo ciclo de interrupções;

Este módulo foi desenvolvido em conjunto com igual contribuição de ambos os elementos do grupo (50%/50%).

### **3.10 Menu**

Neste módulo, são implementadas funções que tornam possível a utilização do rato como ferramenta para navegar os menus. A função principal deste módulo é `updatePosition()` que em função da posição do rato e do clique do botão esquerdo alterna entre os diferentes valores da máquina de estados abordados no capítulo 4.

Este módulo foi desenvolvido em conjunto com igual contribuição de ambos os elementos do grupo (50%/50%).

### 3.11 File

O modulo File, relativo aos ficheiros, é fulcral no registo com datas das pontuações das partidas dos diferentes jogadores. Este registo é possível devido à utilização de uma lista ligada, tópico abordado no capítulo 4, que permite em cada nodo guardar as informações como o nome do jogador, a sua pontuação e, recorrendo ao modulo RTC, a data da partida.

Assim as funções mais importantes neste módulo são:

- write\_to\_file() e read\_from\_file() que escrevem e leem ficheiros respetivamente;
- usersInit() e freeUsers que inicializam a struct **Users** (alocando a memória necessária) e libertam essa mesma memória respetivamente;
- adduser() que adiciona um novo utilizador à lista ligada users de forma a manter a lista ordenada por ordem crescente de *scores*.

Este módulo foi desenvolvido em conjunto com igual contribuição de ambos os elementos do grupo (50%/50%).

### 3.12 RTC

O módulo do RTC, foi desenvolvido para que seja possível datar as pontuações de cada partida. Para tal foram implementadas funções as seguintes funções:

- read\_reg() - que retorna o valor de um registo do RTC pedido por argumento, esta função tira proveito de uma outra providenciada pelos professores wait\_valid\_rtc();
- get\_Year(), get\_Month(), get\_Day(), que devolvem o ano, mês e dia atuais respetivamente;

Este módulo foi desenvolvido em conjunto com igual contribuição de ambos os elementos do grupo (50%/50%).

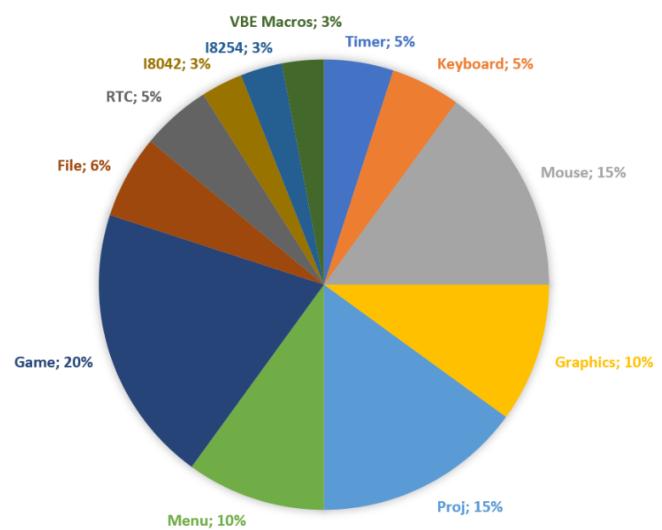


Figura 3.1: Peso dos diferentes módulos no projeto

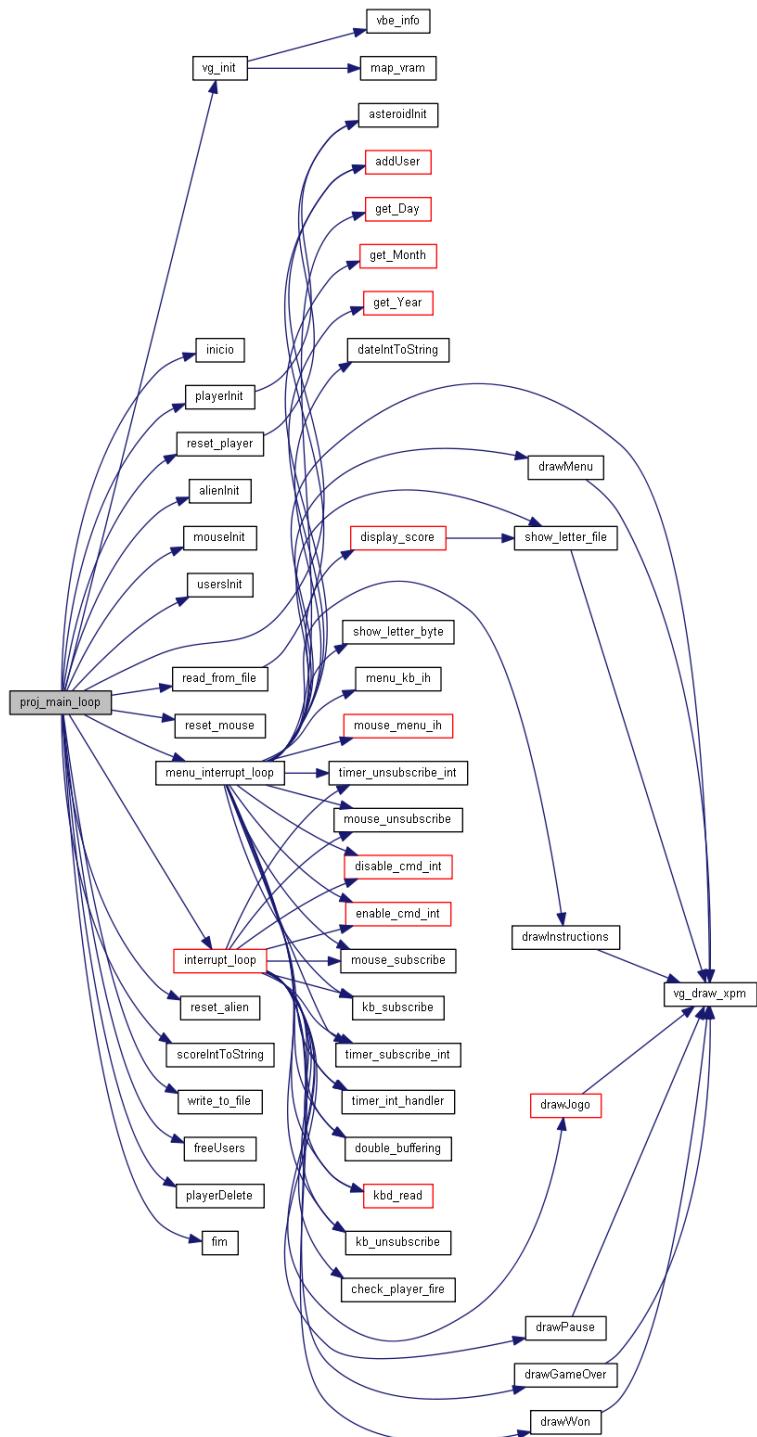


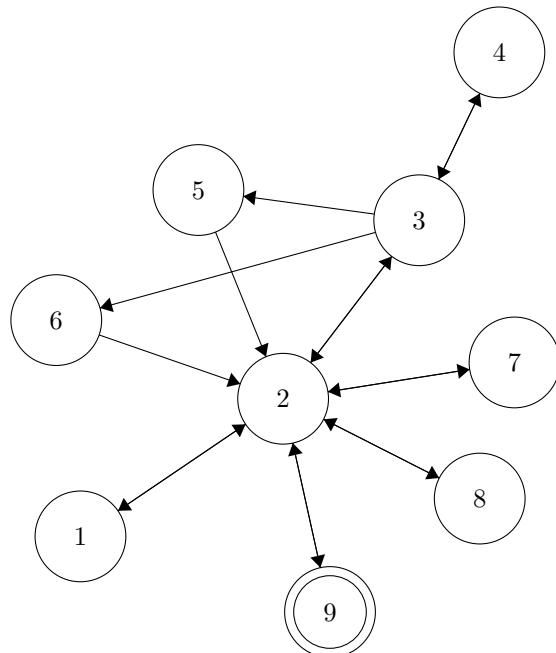
Figura 3.2: Call Graph

## Capítulo 4

# Detalhes de implementação

A estrutura para este programa foi desenvolvida recorrendo a uma máquina de estados de modo a obter uma melhor organização e compreensão do código bem como para facilitar o desenvolvimento da sequência lógica do jogo. Cada estado tem por isso uma condição de saída ou de passagem para o próximo estado. Tudo isto garante uma suave transição entre os vários estados do jogo.

```
typedef enum {
    MAIN_MENU, SCORE, NAME, GAME, PAUSE, OPTIONS, GAME_OVER, WON, INSTRUCTIONS, COMP
} game_st;
```



1. NAME
2. MAIN\_MENU

3. GAME
4. PAUSE
5. WON
6. GAME\_OVER
7. SCORE
8. INSTRUCTIONS
9. COMP

Foi ainda empregue no programa uma outra máquina de estado cujo objetivo é estruturar os vários estados do rato e as transições entre eles. Desta forma foi possível conceber de forma simples um algoritmo que reconhece gestos do rato.

A leitura de imagens no formato **XPM** foi feita a partir das funções disponibilizadas pelos docentes no *LCF*.

O algoritmo para o desenho das mesmas foi desenvolvido pelos estudantes e foi tomado o cuidado para implementar transparente nas imagens, ou seja, a parte transparente das imagens (que era interpretada pelas funções do *LCF* e preenchida com uma cor específica) é ignorada quando se "desenha" as imagens na video memory.

```
if(pic[p] != (unsigned char) 0x0588 && pic[p+1] != (unsigned char) 0x0588)
{
    *(doubleBuff +temp) = pic[p];
    *(doubleBuff + temp + 1) = pic[p+1];
}
```

A verificação de colisões é feita verificando se á sobreposição de pixéis, ou seja, ao "desenhar"na memória virtual é verificado se o espaço de memória que queremos alterar tem uma determinada cor (a cor do "shot"). Se isto suceder podemos concluir que há colisão entre a imagem do "tiro" e a imagem da personagem (ou a imagem do inimigo dependendo da situação).

```
bool check_colision(unsigned char* pic,int x, int y,int width, int height)

(...)

if(((uint8_t)*(doubleBuff +temp) == 0x00 && (uint8_t)*(doubleBuff +temp + 1) == 0xF8)
    && pic[p] != (unsigned char) 0x0000 && pic[p+1] != (unsigned char) 0x0000)
{
    return true;
}
```

A implementação dos *High Scores* foi feita com o recurso á escrita em ficheiros e à criação de uma lista ligada que guarda a informação sobre os utilizadores.

```

typedef struct lligada {
    char name[3];
    char score[3];
    char date[8];
    struct lligada *next;
} *Users;

```

No início de cada jogo o ficheiro "users.txt" é lido pelo programa e a sua informação é guardada numa lista ligada recorrendo às funções "usersInit" e "read\_from\_file". No final do jogo o utilizador atual é guardado na lista ligada "users" com o auxílio da função "addUser". Esta função guarda a informação do jogador de forma ordenada na lista ligada de acordo com o seu score.

```

void addUser(Users users, char* name, char* date, char* score){
    Users l = users;
    Users pt, ant;
    pt = l;
    ant = malloc(sizeof(struct lligada));

    (...)

    while( l->next != NULL && scoreToInt(l->next->score) > scoreToInt(score) ){
        l = l->next;
    }
    ant->next = l->next;
    l->next = ant;

    users = pt;
}

```

Por fim, após o jogador premir a opção **EXIT** do meu principal todo o conteúdo da lista ligada "users" é escrito no ficheiro mantendo a sua ordem. A apresentação dos *High Scores* é feita lendo as três primeiras ocorrências de "users" e imprimindo-as no ecrã de forma facilmente comprehensível.

# Capítulo 5

## Conclusão

Com este trabalho, foi nos dada a oportunidade de aprofundar os conhecimentos adquiridos ao longo da cadeira e de ver não só as suas aplicações práticas, como também a forma como se complementam entre si, levando a um melhor domínio do funcionamento dos periféricos que não se limita apenas aos aspectos teóricos.

Relativamente à cadeira, achamos importante apontar não só para os aspectos positivos como também para os aspectos a melhorar. Por um lado, trata-se de uma cadeira fundamental, dado que nos permitiu desenvolver diversas capacidades tanto de trabalho autónomo e de grupo como também de programação no âmbito das linguagens C e Assembly e da sua utilização simultânea, algo que ainda não tinha sido abordado durante o curso. É também relevante o facto de as aulas práticas serem fulcrais para obter estas competências não sendo, no entanto, suficientes. O que nos leva ao próximo ponto: os aspectos negativos da disciplina.

Desde a primeira aula prática, os alunos têm de rapidamente se tornar autodidatas e adaptar-se de uma forma relativamente brusca aos novos conceitos que terão de aplicar. A isto, acrescenta-se a dificuldade de gerir o tempo investido que a disciplina exige e o tempo do qual dispõem. Por fim, o facto de não termos noção da nossa prestação ao longo dos LABS tornou-se rapidamente uma adversidade, não só porque não sabíamos em que aspectos deveríamos trabalhar mais com o pouco tempo que tínhamos, mas também porque tanto os seguintes LABS como o próprio projeto foram construídos com base no código desenvolvido nestas aulas. Podemos, portanto, concluir que se tratou de uma experiência maioritariamente positiva, mas com espaço para melhorias.

Resta apenas, dar a entender que tanto o projeto, como o relatório foram realizados de forma equilibrada pelos dois membros do grupo.

## Capítulo 6

# Instruções de instalação

Uma vez que o programa necessita de ler e escrever em ficheiros o *path* para os mesmos deve ser especificado como um argumento ao correr o programa. Este *path* pode variar dependendo da pasta para onde foi feito o *checkout* do SVN.

```
lcom_run proj "/home/lcom/labs/proj/src/users.txt"
```

**IMPORTANTE:** ”/users.txt” tem de estar sempre especificado no fim do *path*.

Pode tambem ser necessário executar o comando **lcom\_conf add conf/proj** dentro da pasta /proj para dar as permissões necessárias ao programa.