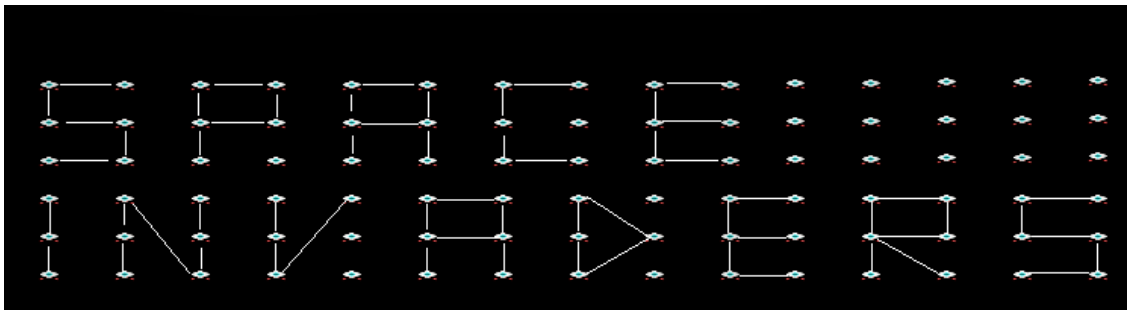
Space Invaders

LABORATÓRIO DE COMPUTADORES



Projeto realizado por:

Bruno Pinto	up201502960
Vítor Magalhães	up201503447

Índice

1. Utilização do projeto

1.1. Início

Para começar o programa, este deve ser chamado com um argumento correspondente ao nome do jogador, como exemplifica a imagem seguinte.

```
# service run `pwd`/Space-Invaders -args "Bruno"
```

1.2. Menu Inicial

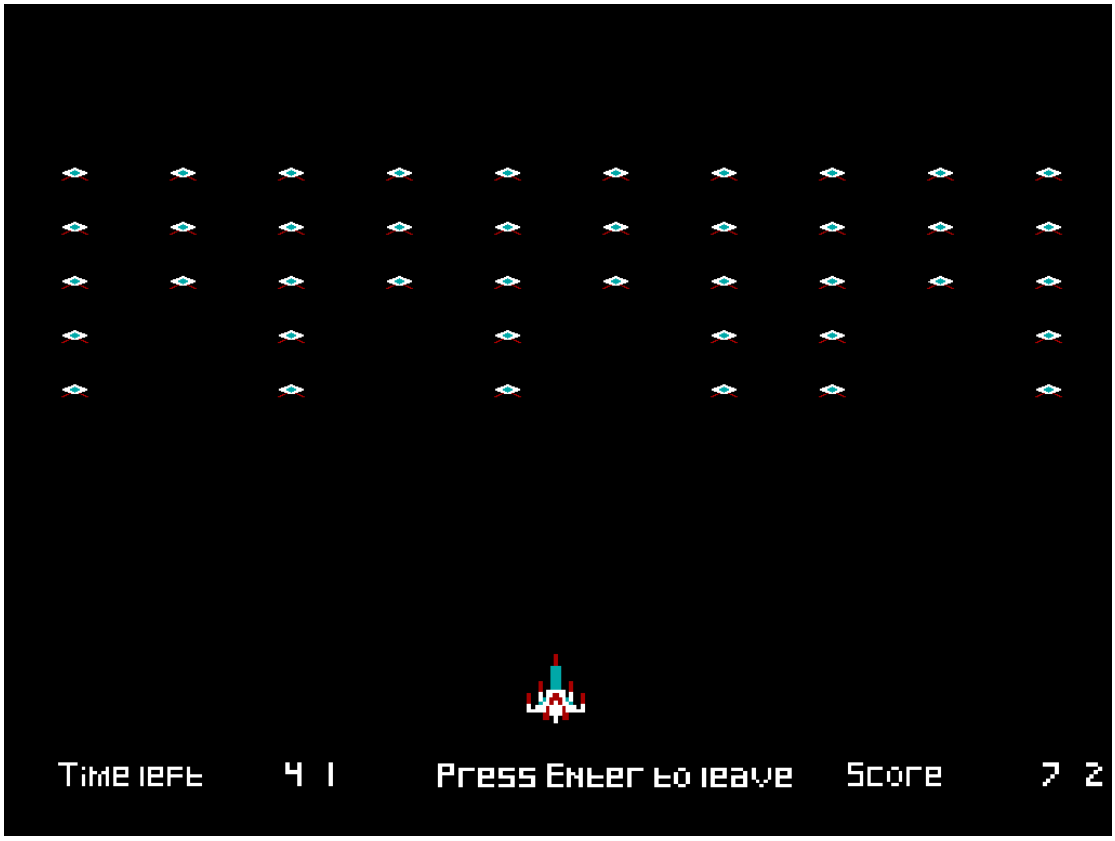
O menu é constituído por duas opções: *New Game* e *Instructions*. Para deslocar a seta para qualquer uma destas opções, basta pressionar as teclas W ou S do teclado. Para selecionar, deve pressionar a tecla *Enter*.

Como é indicado no ecrã, para sair do jogo, basta pressionar a tecla *ESC.*

1.2.1. Novo jogo

A parte superior do ecrã encontra-se habitada pelos inimigos e, na parte de baixo, encontra-se a nave espacial que o jogador controla.

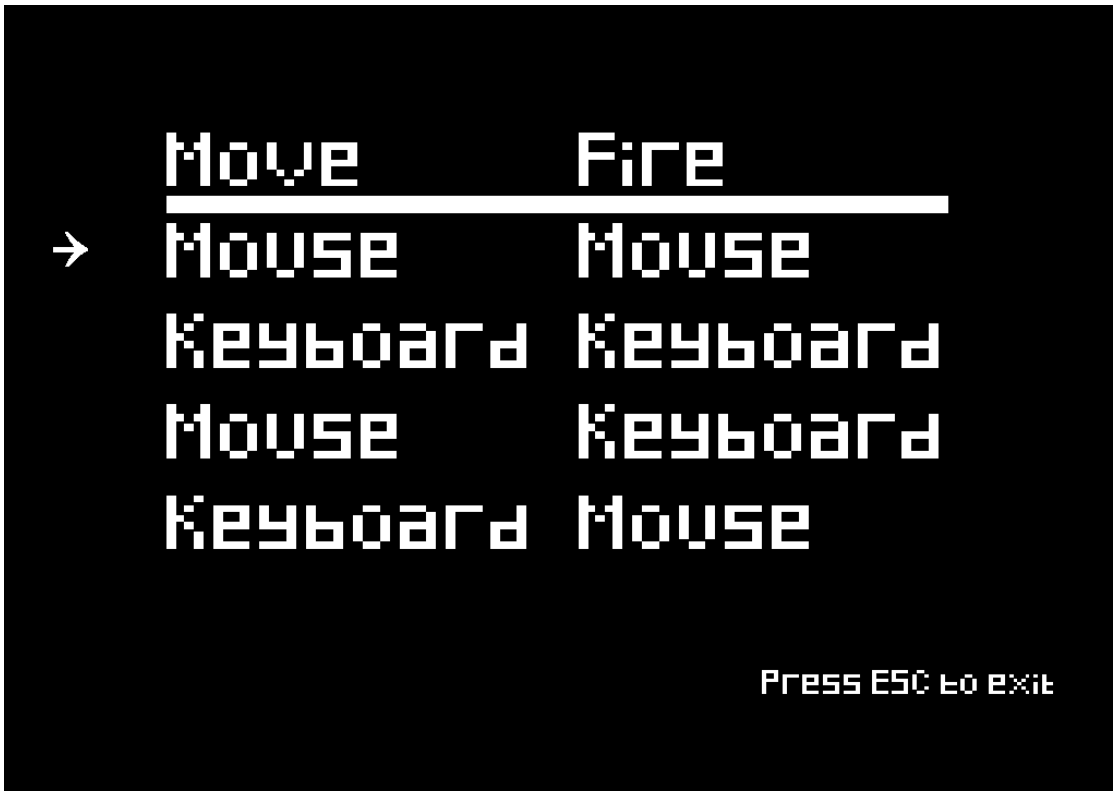
Nos cantos inferiores, encontram-se o tempo restante, que diminui a cada segundo, e a pontuação do jogador, que aumenta à medida que este vai destruindo os inimigos.



1.2.1.1. Controlos

Foi decidido implementar quatro alternativas de jogo: o jogador pode utilizar apenas o rato, apenas o teclado ou uma das duas combinações destes dois, para poder deslocar a nave e disparar.

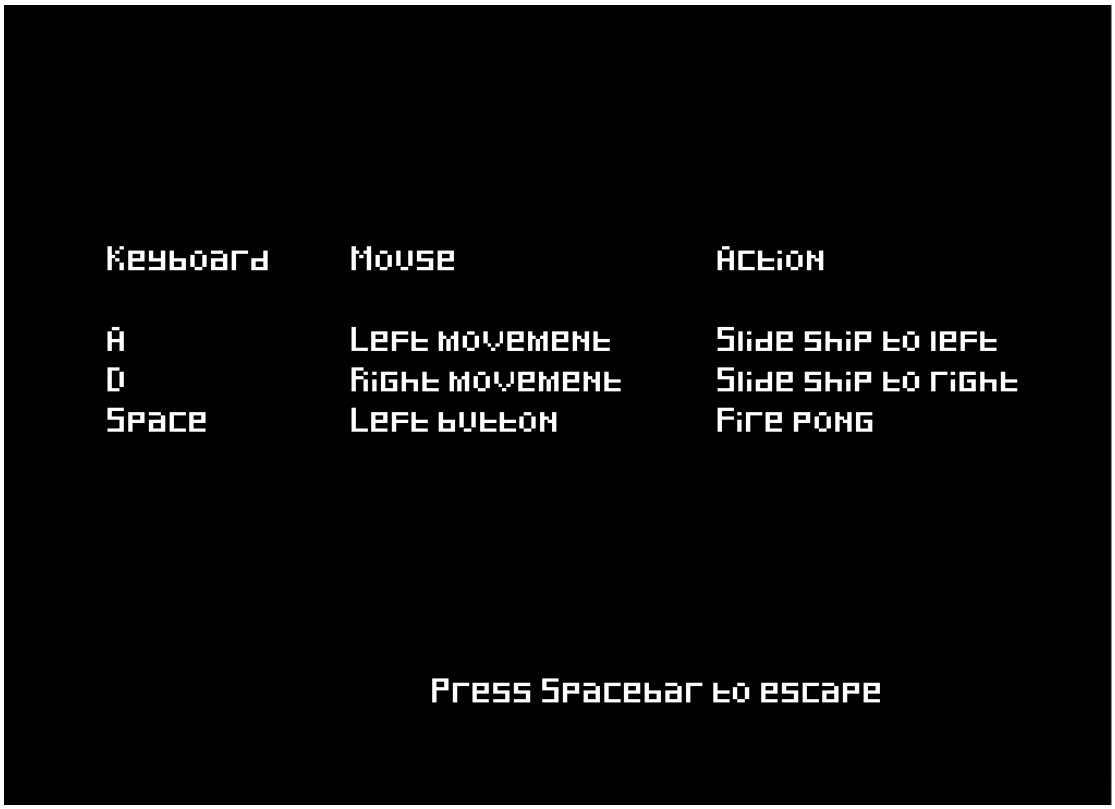
Para escolher a opção, basta usar as teclas W e S para navegar no menu e a tecla *Enter* para selecionar.



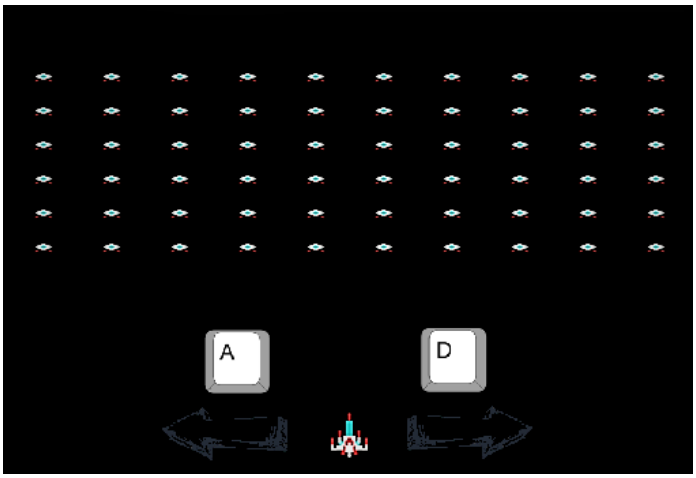
1.2.2. Instruções

A opção *Instructions* oferece a lista de controlos da personagem, identificando o dispositivo (teclado ou rato) e a ação correspondente ao uso desse dispositivo.

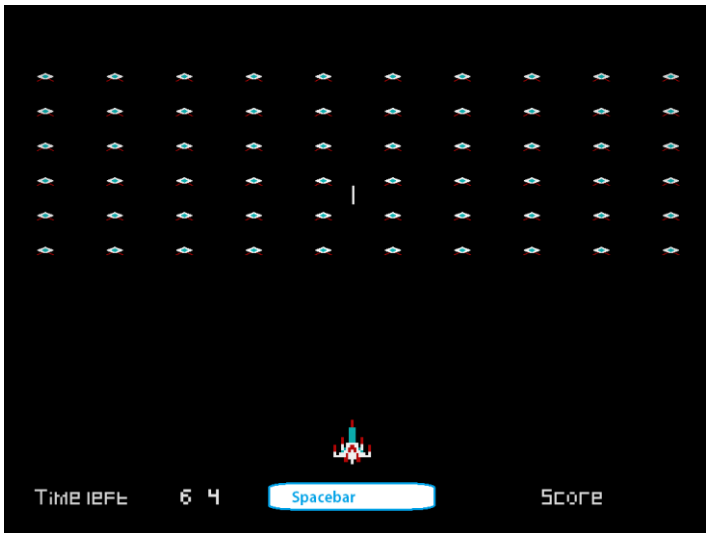
Para sair deste menu, basta pressionar a tecla Barra de Espaços do teclado.



Como mencionado anteriormente, para se **movimentar,** basta pressionar na tecla A e D do teclado ou então mexendo no rato, para a direita e para a esquerda.



Para **disparar** um laser, pressiona-se na barra de espaços ou no botão esquerdo do rato.



Deve-se ter em atenção que, como referido, existem quatro modos de controlo do jogo e as indicações devem ser respeitadas com cuidado.

1.3. O jogo

O objetivo do jogo é destruir todos os inimigos presentes no ecrã. Para destruí-los, usa-se uma nave espacial, equipada com munições.

A nave espacial desloca-se para a direita ou para a esquerda, com os botões *A* e *D* do teclado ou com o movimento do rato. A nave dispara um laser para destruir os inimigos, usando a barra de espaços do teclado ou o botão esquerdo do rato.

Se o jogador destruir todos os inimigos antes de qualquer um deles atingir a parte inferior do ecrã, onde se encontra a nave, aparecerá no ecrã uma mensagem a dizer: ***You Win!*** Caso contrário, aparecerá, no ecrã, uma mensagem a dizer: ***You Lose!***

Quer ganha, quer perca, é mostrado no ecrã a pontuação total do jogador.

Como informações para o utilizador, são apresentados no ecrã o tempo restante até a fila superior atingir a base do ecrã e a pontuação obtida até ao momento.

2. Estado do Projeto

Ao contrário daquilo que foi referido na especificação, foi possível implementar não só os quatro dispositivos mencionados (*timer*, placa gráfica, *keyboard* e *mouse*), como também o *Real Time Clock,* ou RTC.

Dispositivo	**Utilidade**	**Interrupções?**
Timer	Supervisionar os *frames* e atualizar os dados do jogo, como o movimento do laser, o tempo restante e o resultado.	Sim
Teclado	Movimento horizontal da nave e/ou disparar laser.	Sim
Rato	Movimento horizontal da nave e/ou disparar laser.	Sim
Placa Gráfica	Apresentação do espaço de jogo.	Não
Real Time Clock	Guardar data e hora do fim do jogo.	Não

O *timer*

O *timer* foi pensado de modo a controlar o *framerate* do jogo e a atualizá-lo. O movimento descendente dos inimigos também é controlado pelo timer, tal como a atualização do tempo restante, a pontuação total e o movimento do laser disparado pela nave. Não foi necessário alterar os dados do timer durante o desenvolvimento do projeto.

As funções relativas a este dispositivo estão no ficheiro *"game.c"*, que inclui o ciclo das interrupções (e a atualização dos dados referidos acima), *"devices.c"* que inclui as subscrições dos dispositivos, e *"timer.c"*, que inclui a implementação do *timer*.

"game.c": *enemy_movedown, no_enemies e start_game.*

"devices.c": *devices_subscriptions e devices_unsubscriptions.*

"timer.c": *timer_subscribe_int, e timer_unsubscribe_int.*

O teclado

O teclado é usado para funções relativas à navegação dos menus e à deslocação da nave espacial. Permite o movimento horizontal da mesma, assim como o disparo do laser que destrói os inimigos. Como foi mencionado anteriormente, o movimento é realizado através das teclas A e D do teclado e o disparo do laser é feito através da barra de espaços.

As funções relativas à utilização do teclado estão presentes no ficheiro *"game.c",* que inclui o ciclo das interrupções (que deteta os botões utilizados), "*print.c*", que inclui as funções necessárias para a implementação dos menus, *"devices.c",* que inclui as subscrições do dispositivo, e *"keyboard.c",* que inclui a implementação do *keyboard*.

"game.c": *start_game*.

"devices.c": *devices_subscriptions e devices_unsubscriptions.*

"*print.c":* *print_menu_kbd_or_mouse, print_menu e print_instructions.*

"keyboard.c": *kbd_subscribe, kbd_unsubscribe e kbd_scan.*

O rato

O rato é utilizado, tal como o teclado, para funções relativas à nave espacial. O movimento horizontal do rato permite deslocar a nave e o clique no lado esquerdo o disparo do laser.

As funções que foram necessárias para a utilização do rato estão no ficheiro *"game.c",* que inclui o ciclo das interrupções e as funções que deslocam a nave e disparam o laser, *"devices.c",* que inclui as subscrições do dispositivo, e *"mouse.c"*, que contém a implementação do *mouse*.

"game.c": *start_game e fire_pong.*

"devices.c": *devices_subscriptions e devices_unsubscriptions.*

"mouse.c": *mouse_subscribe, mouse_unsubscribe e mouse_scan.*

A placa gráfica

Sendo um dispositivo obrigatório, foi o primeiro a ser implementado. É utilizado o modo gráfico 0x105, cuja resolução é 1024 * 768, com 64 cores.

É usada para apresentar a componente gráfica do jogo: a nave, os inimigos, o laser, o tempo restante, a pontuação total (e final) e as mensagens de vitória ou derrota; a placa é atualizada segundo o conceito de *double buffering*.

Para uma utilização com sucesso, foi necessária a implementação da *video card*, feita nos ficheiros *"video_gr.c",* de forma a que o Minix entre em modo gráfico; *"devices.c"*, para a subscrição do dispositivo; *"xpm.c",* para a criação e destruição dos XPM criados; "*print.c"*, para a criação do menu principal; "numbers_pixmap.h", que contém os XPMs que representam números; "pixmap.h", que guarda os XPMs restantes, e *"game.c",* que contém funções necessárias para a deteção de colisão entre XPMs e movimentos dos XPM utilizados.

"game.c": *is_enemy, is_ship, enemy_movedown, start_game.*

"devices.c": *devices_subscriptions e devices_unsubscriptions.*

"video_gr.c": *vg_init, vg_exit, getVideoMem, getHRes, getVRes, getBitsPerPixel, vbe_get_mode_info.*

*"**print.c":** print_menu e print_instructions.*

"xpm.c": *object_creator, buffer_destructor, read_xpm e title_creator.*

Real Time Clock

Sendo um dispositivo opcional, dependente da utilização do rato, foi o último a ser implementado.

É utilizado para ler a data e a hora do computador, imediatamente após o fim do jogo; estas informações são guardadas, juntamente com o nome do utilizador, no ficheiro de texto *"highscores.txt"*.

As funções utilizadas estão contidas nos ficheiros *"devices.c",* que inclui a subscrição do dispositivo, e *"rtc.c",* que contém a implementação do *Real Time Clock*.

"devices.c": *devices_subscriptions e devices_unsubscriptions.*

"rtc.c": *rtc_subscribe, rtc_unsubscribe, read_reg, convertBCDtoBinary, getDay, getMonth, getYear, getSec, getMin e getHour.*

3. Organização e Estrutura do Código

Ao longo do desenvolvimento do projeto, a separação do mesmo em vários ficheiros aconteceu de forma natural. Assim, no final, o projeto é composto pelos seguintes ficheiros apresentados:

"*devices.c*"

Faz as subscrições de todos os dispositivos utilizados e cancela as subscrições desses dispositivos no fim da execução.

Responsável: Bruno Pinto (60%) e Vítor Magalhães (40%) Peso no projeto: 05%

"*file_handler.c*"

Contém a função *write_score,* que obtém o tempo e data do RTC e guarda-os, juntamente com o nome do user pedido como argumento, ao correr o programa, num ficheiro de texto.

Responsável: Bruno Pinto Peso no Projeto: 02%

"game.c"

Contém a função *start_game* que faz o ciclo das interrupções de três dispositivos (timer, teclado e mouse). Inclui também as funções que permitem o movimento inimigo, o disparo da nave, a deteção de colisões e a função *init* que inicializa o jogo, as variáveis e o ambiente do jogo.

Responsável: Bruno Pinto (50%) e Vítor Magalhães (50%) Peso no projeto: 40%

"keyboard.c"

Este módulo, relativo ao teclado, contém as funções que subscrevem e cancelam as subscrições do teclado. Contém também a função *kbd_scan,* que deteta a tecla que foi premida (que é usada para mover a nave e disparar o laser).

Reponsável: Bruno Pinto (50%) e Vitor Magalhães (50%) Peso no projeto: 05%

"mouse.c"

Este módulo, relativo ao rato, contém as funções que subscrevem e cancelam as subscrições do rato. Contém também a função *mouse_scan* que deteta os pacotes enviados pelo rato. Estes pacotes são analisados em *"game.c",* usados para atualizar o movimento da nave e/ou disparar o laser.

Responsável: Bruno Pinto (50%) e Vítor Magalhães (50%) Peso no projeto: 05%

"print.c"

Imprime no ecrã, enquanto o Minix está em modo gráfico, os menus, a pontuação, o tempo restante, a mensagem final (vitória ou derrota) e um número.

Responsável: Bruno Pinto (40%) e Vítor Magalhães (60%) Peso no projeto: 10%

"rtc.c"

Este módulo, relativo ao Real Time Clock, contém as funções que subscrevem e cancelam as subscrições do RTC. Contém também funções necessárias para ler a data e a hora, usando a função *read_reg.*

Responsável: Vítor Magalhães Peso no projeto: 05%

"spaceinvaders.c"

É o módulo mais pequeno e inclui apenas uma função: *main,* que chama a função de subscrição dos dispositivos, inicializa o jogo, chamando *print_menu,* cancela as subscrições e chama a *write_score*).

Responsável: Bruno Pinto (70%) e Vítor Magalhães (30%) Peso no projeto: 08%

"timer.c"

Este módulo, relativo ao timer, contém as funções que subscrevem e cancelam as subscrições do timer.

Responsável: Bruno Pinto (50%) e Vítor Magalhães (50%) Peso no projeto: 05%

"utilities.c"

É o ficheiro com os dois utilitários usados para conversão: uma função para converter de BCD para binário e outra para converter complemento para dois.

Responsável: Bruno Pinto (50%) e Vítor Magalhães (50%) Peso no projeto: 05%

"vídeo_gr.c"

Este módulo, relativo à placa gráfica, contém as funções que permitem a entrada e saída do modo gráfico, assim como outra que é utilizada para obter informações acerca do mesmo. Contém também funções para obter alguns valores retirados da função de inicialização do modo gráfico usado e a função responsável por copiar o *buffer* para a *video_mem (double buffering)*.

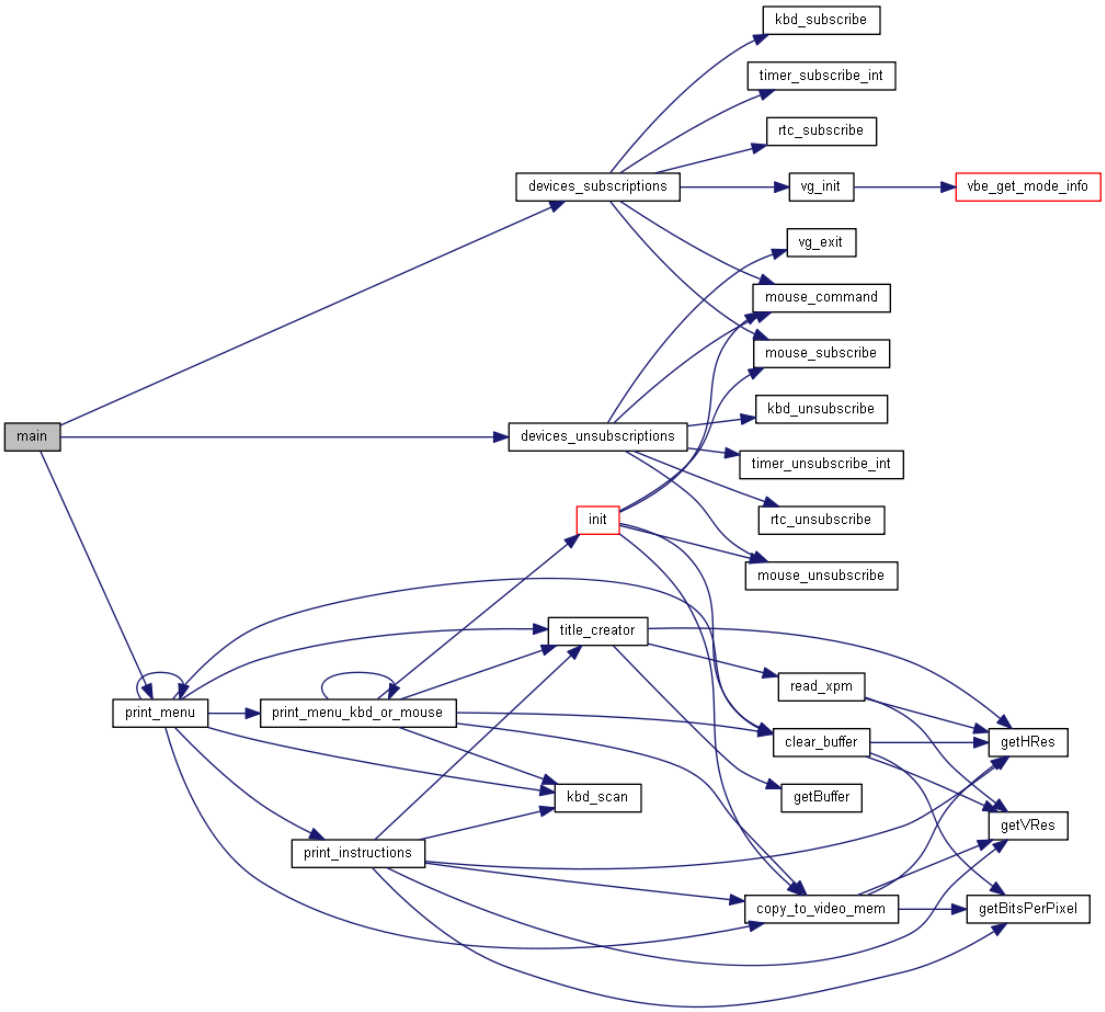
Responsável: Bruno Pinto Peso no projeto: 05%

“xpm.c”

Permite a criação e destruição de XPM’s, sejam eles imagens como a nave e os inimigos ou títulos. Inclui também a função *read_xpm*, fornecida pelo Professor João Cardoso.

Responsável: Bruno Pinto (65%) e Vítor Magalhães (35%) Peso no projeto: 05%

3.1. Diagrama de Chamadas de Funções



4. Implementação

Para o desenvolvimento do *Space Invaders*, foram necessários vários conceitos cobertos nos *labs* feitos ao longo do semestre, tais como máquinas de estado e interrupções. No entanto, tivemos que simular o *lab* relativo ao *Real Time Counter* e experimentar o *double buffering* por nossa conta, para além de obter mais algum conhecimento da linguagem C que ainda não tinha sido adquirido no decorrer das aulas desta Unidade Curricular.

4.1. Implementação de *Structs*

Antes da aula teórica de Programação Orientada a Objetos com C, já se tinha pensado, em grupo, na utilização de *structs* para definir vários conceitos pertencentes ao programa.

4.1.1. Nave Espacial

Foi criada a *struct Ship_t,* que contém as coordenadas da nave, a sua altura e largura, e a matriz de caracteres que compõe a sua imagem. Estes parâmetros foram fundamentais na criação e movimento da nave.

A maior dificuldade sentida, nesta implementação, foi no movimento da nave através do rato. Foi difícil ter o rato a funcionar, pois este dava alguns erros, nomeadamente latência na resposta.

4.1.2. Inimigos

Paralelamente à implementação da nave espacial, foi utilizada uma *struct* para definir cada inimigo. *Enemy_t* inclui os mesmos parâmetros da *Ship_t*, aos quais se juntam a pontuação de cada inimigo (que difere de inimigo para inimigo) e uma *flag* que indica se este foi ou não destruído.

Foi criada uma matriz de inimigos e uma função que atualiza o movimento dos inimigos, aumentando o valor do parâmetro *y* a cada *ENEMY_DOWN_TIME* segundos (*ENEMY_DOWN_TIME* é uma macro definida em *"variables.h"*)

4.1.3. Laser/Pong

A última *struct* a ser definida foi a *Pong_t*, que representa o laser que a nave utiliza para destruir os inimigos. Os parâmetros são equivalentes ao da *Ship_t*.

Para que a sua utilização fizesse sentido no contexto do jogo, foi importante definir a colisão entre laser e inimigos.

4.1.3.1. Colisão entre laser e inimigos

A deteção da colisão entre laser/pong e inimigo é feita a partir de condições matemáticas, que envolvem a posição do canto superior direito de cada objeto, e a sua altura e largura. Caso haja colisão, o inimigo é apagado e fica com a flag *destroyed* na struct *Enemy_t* a 1.

4.2. Implementação de tempo restante e pontuação

Durante a projeção do jogo, foi decidido que este seria acompanhado por uma constante atualização do tempo que o jogador ainda tem para destruir todos os inimigos e a pontuação que este vai obtendo à medida que elimina cada inimigo.

O tempo restante decrementa um segundo a cada sessenta interrupções do timer (que correspondem a um segundo, visto usarmos sessenta *frames* por segundo).

A pontuação é incrementada cada vez que o jogador elimina um inimigo. Cada inimigo, dependendo da sua posição, tem valores de pontuação diferentes e, por cada inimigo derrubado, os seus pontos são acrescentados aos pontos do jogador.

Estas duas funcionalidades são apresentadas no ecrã com uso da placa gráfica. Foram feitos dez XPMs, correspondentes a todos os algarismos decimais para que, recursivamente, pudesse ser apresentado o algarismo.

4.3. Implementação de Máquinas de Estado

4.3.1. Menus

Uma das abstrações importantes para a realização deste projeto foi a projeção do Menu Principal, que inclui, como referido anteriormente, duas opções: Novo jogo e Instruções.

Para que este funcionasse, foi implementando uma máquina de estados que inclui, numa variável do tipo *typedef enum,* estas duas opções.

Dentro da opção *Novo Jogo*, há um novo menu com as opções de jogar com o rato, com o teclado ou com uma junção dos dois. Para o seu funcionamento, foi criada outra máquina de estados para fazer a separação.

4.4 Implementação do ficheiro de texto *Highscores.txt*

No final foi criado um ficheiro de texto onde é guardada a pontuação do jogador.

Este ficheiro de texto inclui o nome do jogador, a pontuação final e a data e hora do fim do jogo.

O nome é um argumento na chamada da execução do jogo. Para registar a data e hora, foi usado o *Real Time Clock* e funções relativas ao para que estas informações pudessem ser obtidas e guardadas no ficheiro de texto *highscores.txt*.

```
1 Date=21/12/16 Hour=12:08 Player=Briguel Score=78
2 Date=21/12/16 Hour=12:08 Player=Briguel Score=18
3 Date=21/12/16 Hour=12:09 Player=Briguel Score=60
4 Date=21/12/16 Hour=12:09 Player=Briguel Score=252
5
```

5. Conclusões

Em primeiro lugar, consideramos que a cadeira de Laboratório de Computadores devia ser precedida por uma outra, através da qual se aprendesse, detalhadamente, a linguagem C.

Por outro lado, encontra-se a grande complexidade relacionada com o assunto dos periféricos que necessitaria de um apoio mais aprofundado. Acrescente-se a isso o facto de os alunos terem necessidade de aprender esta linguagem de programação de uma forma individualizada, levando a que o desempenho e o trabalho, exigido noutras cadeiras, seja prejudicado, como sucede, por exemplo, em Algoritmos e Estruturas de Dados, cujo número de ECTS é superior ao número de ECTS de Laboratório de Computadores.

Para além disso, o reduzido número de responsáveis pelas aulas práticas, professor e monitor, influencia negativamente no desenvolvimento e desempenho dos *labs* e do projeto de cada grupo, visto que cada turma é constituída por mais de uma dezena de alunos.

No entanto, apesar das dificuldades mencionadas acima, consideramos que o projeto final, ao permitir uma maior liberdade de escolha dos temas, proporciona, aos alunos, uma motivação positiva para o curso.