Faculdade de Engenharia da Universidade do Porto

Space Invaders

Projeto de LC - Turma 3 - Grupo 5

Realizado por:
Alexandre Manuel Luz Rodrigues da Costa - up202005319
Ana Beatriz Cruz Fontão - up202003574
Ana Rita Baptista de Oliveira - up202004155
Matilde Maria Amaral Silva - up202007928

Índice

1. Introdução **4**

2. Instruções para utilização **5**
Ecrã inicial 5
Menu 6
Instruções 7
Jogo 8

3. Status do projeto **9**
Graphics card 9
Keyboard 10
Mouse 10
Timer 10
RTC 10

4. Estrutura/organização do código **11**
alien.c 11
alienBullet.c 11
game.c 11
graphic.c 12
keyboard.c 12
menu.c 12
mouse.c 12
proj.c 13
rtc.c 13
ship.c 13
shipBullet.c 13
timer.c 14
utils.c 14
Function call graph 15

5. Detalhes de implementação **16**
Game 16
Menu 16
Ship 16
Alien 17
Ship Bullet 17
Alien Bullet 18
Rondas 18
Double Buffer 18

6. Conclusão **19**

1. Introdução

Para este projeto da cadeira de Laboratório de Computadores (LC), nós decidimos implementar o clássico Space Invaders. O jogo tem apenas modo de Single Player.

O objetivo principal do jogo é andar de um lado para o outro do ecrã e atacar a armada de aliens que rapidamente se aproximam da nossa nave.

O jogo acaba quando a nave é atingida três vezes, voltando ao menu inicial. Se o jogador conseguir matar todos os aliens, passa para a ronda seguinte onde continua a angariar pontuação de acordo com os aliens que vai atingindo.

2. Instruções para utilização

Ecrã inicial

O jogo inicia com o ecrã a anunciar o nome do jogo. Tendo passado 4 segundos o utilizador é apresentado o ecrã do menu principal.

Menu

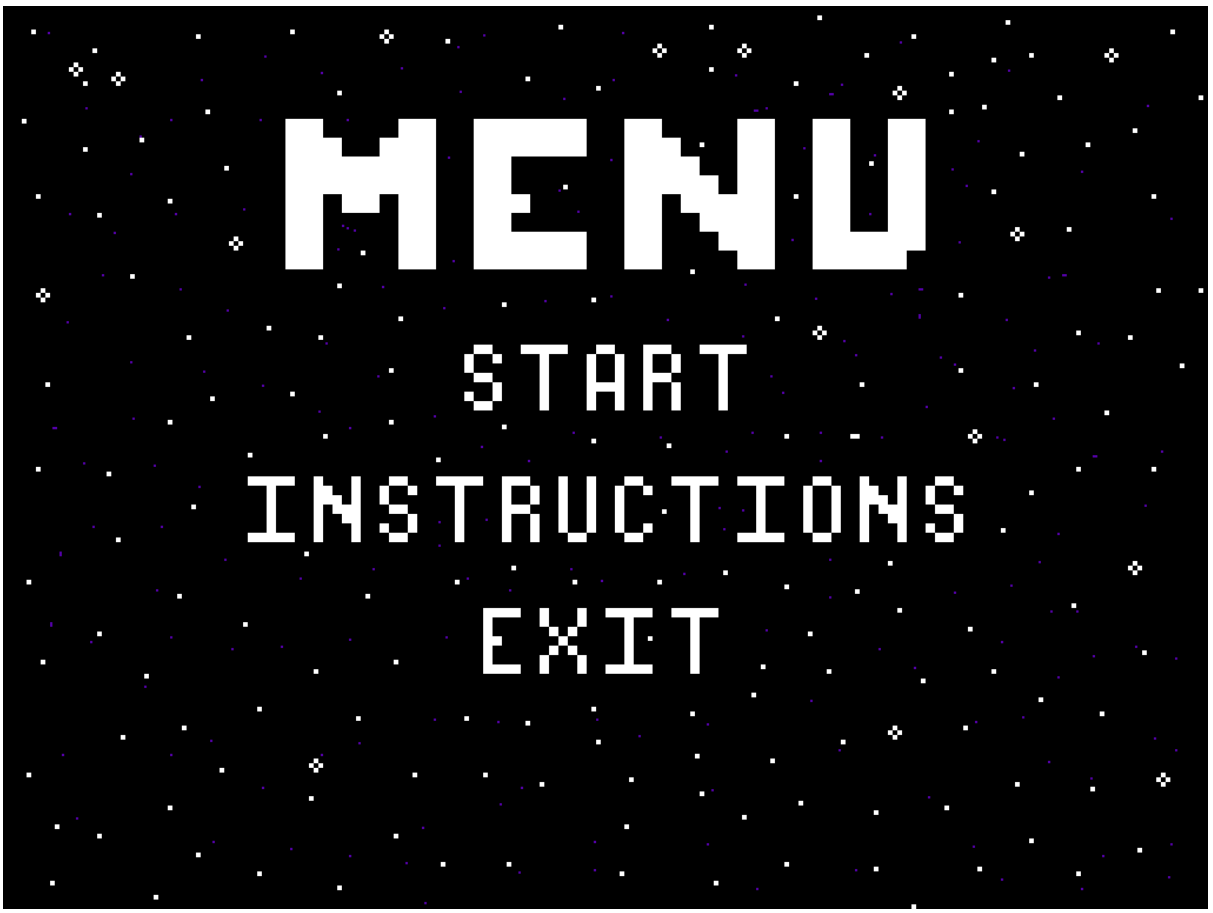


Fig. 2 - Menu principal

A figura ilustra o menu principal onde é possível iniciar o jogo, ver as instruções e sair do jogo.

Usando o cursor, se o utilizador clicar com o botão esquerdo na opção “Instructions”, será apresentado o ecrã que explica o *gameplay* e as regras.

Para sair do jogo, basta o utilizador carregar na opção Exit ou carregar na tecla Esc.

Instruções

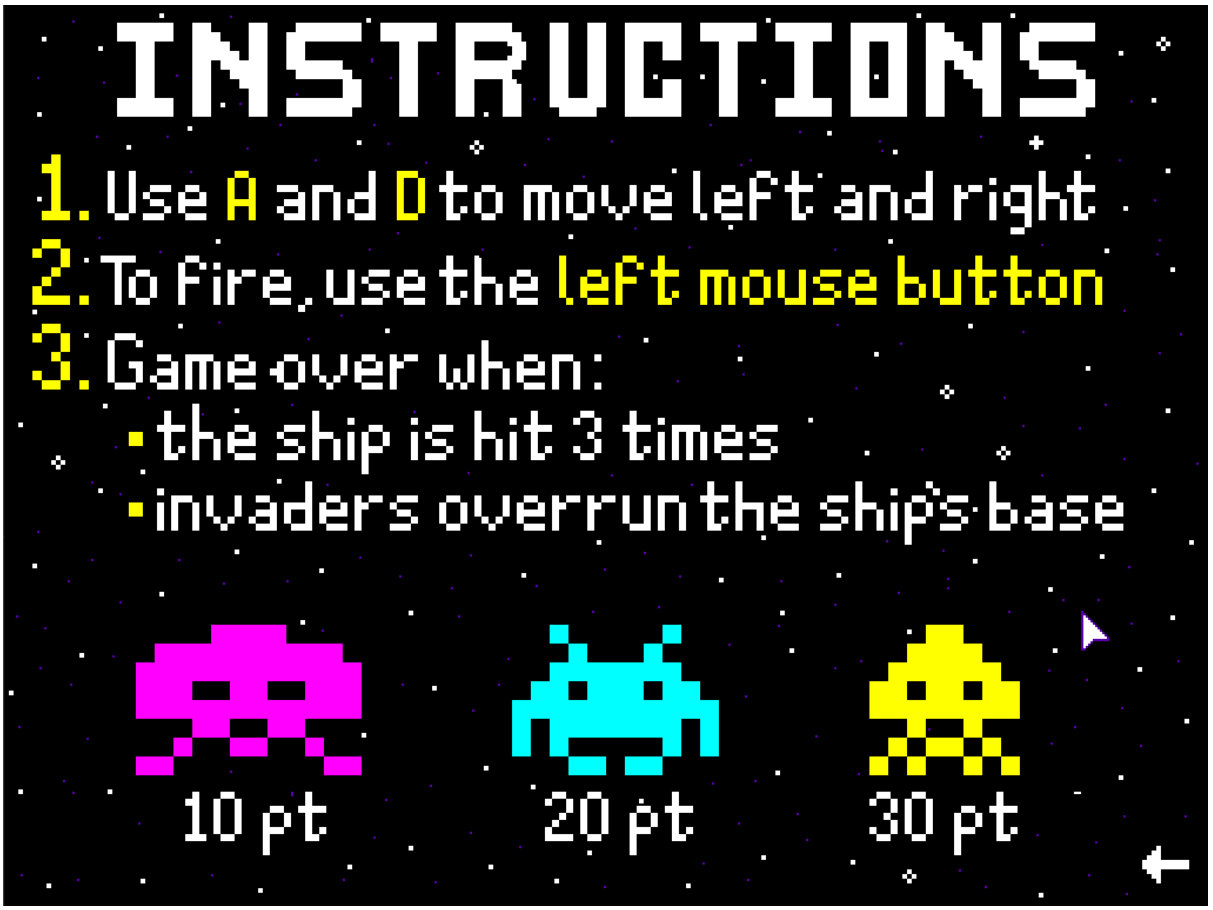


Fig. 3 - Ecrã de Instruções

Este ecrã mostra as teclas a ser usadas para movimentar a nave, o botão do rato a usar para mandar balas, a pontuação que o utilizador arrecada se atingir um alien em específico, e os possíveis cenários de término do jogo.

Para voltar ao menu principal, basta usar o cursor para clicar na seta de retorno que se encontra no canto inferior direito do ecrã.

Jogo

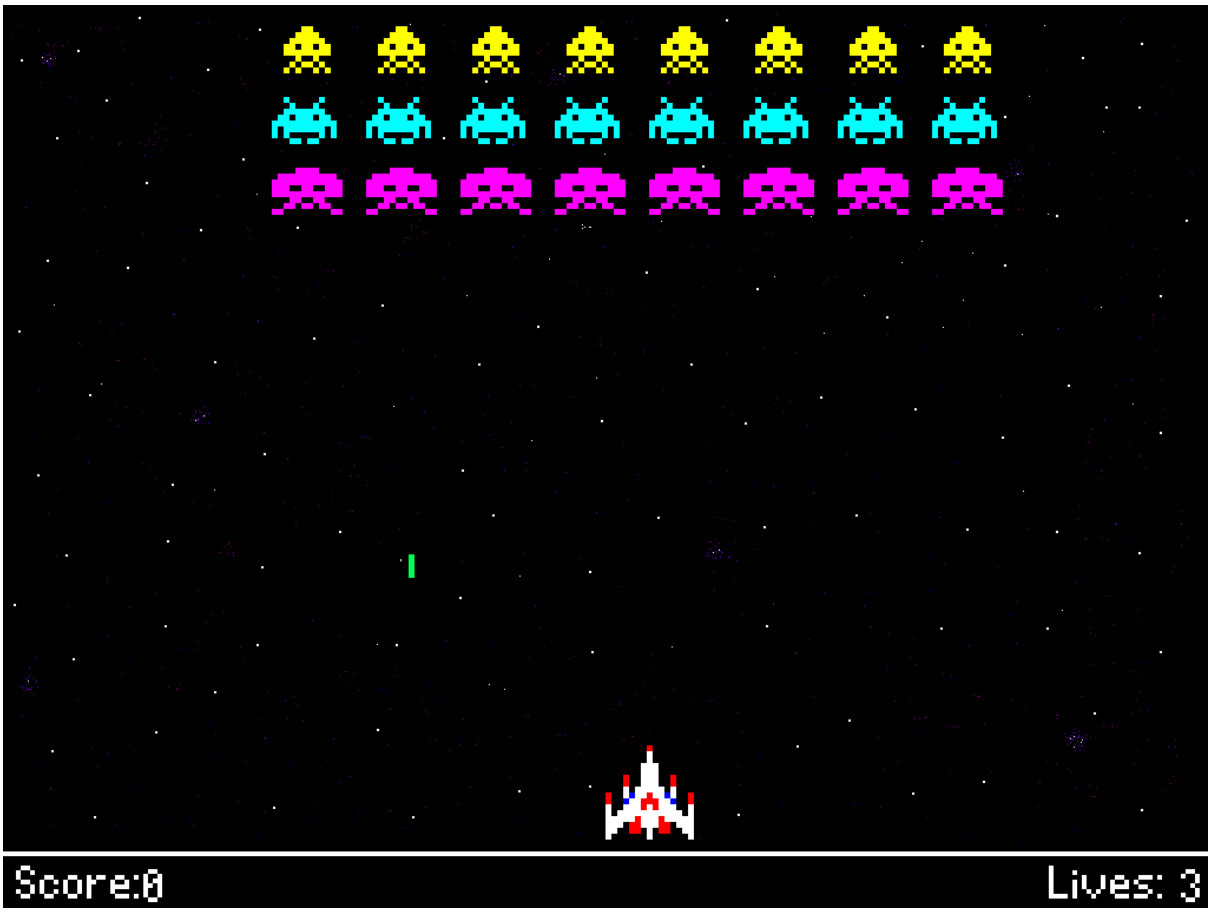


Fig. 4 - Ecrã de jogo

Quando o utilizador escolhe a opção *START* no menu principal será apresentado o ecrã de jogo, onde poderá desfrutar da nostalgia que este jogo oferece!

Ao entrar no jogo, o utilizador vê um grupo de 24 aliens distribuídos por 8 colunas e três linhas. Existem três categorias de aliens com cores, aparência e pontuações diferentes, tal como apresentado no menu das instruções.

No canto inferior esquerdo é possível ver o indicador do *Score* que mostra a pontuação angariada pelo jogador ao longo do jogo. Já no canto inferior direito é possível observar o número de vidas que restam ao jogador. O jogo é inicializado com a pontuação a 0 e as vidas a 3.

3. Status do projeto

Faltam implementar as seguintes funcionalidades:

- RTC - controlar a aceleração dos aliens (feito com o Timer)

Neste projeto utilizamos os dispositivos seguintes:

Dispositivo	Propósito	Interrupt/polling
Timer	Controlar o frame rate, cronometrar o ecrã inicial de apresentação do jogo	Interrupt
Keyboard	Movimento da nave, saída do jogo a qualquer momento (ESC)	Interrupt
Mouse	Navegação no menu, tiros da nave	Interrupt
Graphics Card	Menu e *display* do jogo	N/A
Real Time Clock(RTC)	Controlar a aceleração dos aliens	Interrupt

- **Graphics card**

Para este projeto usamos a placa gráfica no modo 0x105, cuja resolução 1024x768. Foi utilizado o modo de cores indexado que pode utilizar até 256 cores. Foi usada também a técnica de double buffer que melhora a apresentação do jogo, nomeadamente no que toca ao movimento da nave, dos aliens e das balas. Para isso foi elaborada a função displayScreen onde usamos a função memcpy para copiar o conteúdo do segundo buffer para o buffer que irá ser visualizado no ecrã.

Utilizámos as funções vg_get_mode_info e vg_set_mode para obter informação sobre o modo e inicializar o mesmo. Para mostrar os diferentes elementos do jogo no ecrã foi usada a função drawXpm implementada nos labs.

As imagens utilizadas neste projeto foram todas desenhadas pela Ana Rita Oliveira.

- **Keyboard**

O teclado foi utilizado para controlar o movimento da nave. Ao usar a tecla A, a nave movimenta-se para a esquerda e ao clicar na tecla D, a nave movimenta-se para a direita. Além disso, a tecla Esc permite ao utilizador terminar o programa a qualquer momento.

Para este dispositivo foram utilizadas as funções elaboradas no lab3 e implementada a função updateShipPosition, cujo propósito é atualizar o valor do x da nave.

- **Mouse**

A posição do rato foi utilizada para saber se o rato está sobre algum item do menu e, no caso de estar, destacar essa opção. O botão esquerdo é utilizado para disparar durante o jogo e para selecionar a opção desejada no menu.

As funções do mouse, elaboradas no lab4, foram utilizadas principalmente no game_loop, com o objetivo de chamar a função updateMouse que muda a posição do mouse sempre que necessário.

- **Timer**

Neste projeto, reutilizamos as funções de timer elaboradas nos labs. As interrupções do timer, foram utilizadas para controlar o movimento dos aliens e também para desenhar os elementos do jogo.

- **RTC**

O RTC, embora tenha sido implementado, não chegou a ser utilizado, uma vez que não foi possível colocá-lo a trabalhar. O objetivo era utilizar as suas funções para aumentar a velocidade dos aliens sem necessidade de utilizar o timer.

As funções do RTC, mais especificamente o rtc_ih (interrupt handler) seria chamado no game_loop a cada interrupção do RTC de forma a aumentar a velocidade dos aliens.

4. Estrutura/organização do código

- **alien.c**

Código relativo aos objetos *Alien,* inclui um construtor, funções que alteram as coordenadas do mesmo e *getters.*

Peso do módulo no projeto: 5%

Este módulo foi escrito pela Matilde Silva.

- **alienBullet.c**

Código relativo aos objetos *AlienBullet,* inclui uma função que cria a bala, uma função que atualiza as coordenadas da mesma e outra que verifica a colisão entre uma bala de alien e a nave pilotada pelo utilizador.

Peso do módulo no projeto: 5%

Este módulo foi escrito pela Ana Beatriz Fontão.

- **game.c**

Contém a função game_loop, que, por sua vez, contém o loop driver_receive principal.

Peso do módulo no projeto: 60%

Neste módulo a divisão de tarefas foi:

- Alexandre Costa - Atualização do valor score e das vidas, utilização e handling do RTC
- Ana Beatriz Fontão - Movimento da nave, handling do mouse, handling do keyboard, tiros da nave, tiros dos aliens e movimento dos aliens
- Ana Rita Oliveira - Atualização do valor score e das vidas, utilização e handling do RTC
- Matilde Silva - Movimento dos aliens

- **graphic.c**

Código relativo ao Graphics card. Contém funções que inicializam o video mode desejado e desenham XPMs.

Peso do módulo no projeto: 10%

Este módulo foi adaptado das aulas práticas, pelo que contém igual contribuição de todos os membros do grupo.

- **keyboard.c**

Código relativo ao Keyboard. Contém funções que subscrevem os interrupts do Keyboard e lidam com esses mesmos interrupts.

Peso do módulo no projeto: 2%

Este módulo foi adaptado das aulas práticas, pelo que contém igual contribuição de todos os membros do grupo.

- **menu.c**

Código relativo ao menu principal, inclui a declaração do menu que contém as localizações de cada opção do menu, e as funções que desejam os XPMs relativos ao menu.

Peso do módulo no projeto: 3%

Este módulo foi escrito pela Ana Beatriz Fontão.

- **mouse.c**

Código relativo ao mouse, inclui uma função que subscreve as interrupções do mouse, outra que lida com as mesmas, além de funções relativas ao desenho do mouse no ecrã.

Peso do módulo no projeto: 2%

Este módulo foi escrito por todos os elementos do grupo.

- **proj.c**

Módulo fornecido pelos professores.

Peso do módulo no projeto: 1%

- **rtc.c**

Código relativo ao RTC, inclui uma função que subscreve as interrupções do RTC, outra que lida com as mesmas, além de funções que leem o status do RTC.

Peso do módulo no projeto: 2%

Este módulo foi escrito pela Ana Rita Oliveira e pelo Alexandre Costa.

- **ship.c**

Código relativo ao objeto *Ship,* inclui um construtor, funções que alteram as coordenadas da mesma. Este módulo inclui ainda as funções relativas ao *score* e às vidas disponíveis.

Peso do módulo no projeto: 2%

As funções relativas à nave foram escritas pela Ana Rita Oliveira e as restantes pelo Alexandre Costa.

- **shipBullet.c**

Código relativo aos objetos *ShipBullet,* inclui uma função que cria balas, uma função que atualiza as coordenadas das mesmas e outra que verifica a colisão entre uma bala de *ship* e um qualquer alien.

Peso do módulo no projeto: 2%

Este módulo foi escrito pela Ana Beatriz Fontão.

- **timer.c**

Código relativo ao timer 0, inclui uma função que subscreve as interrupções do timer e outra que lida com as mesmas.

Peso do módulo no projeto: 5%

Este módulo foi adaptado das aulas práticas, pelo que contém igual contribuição de todos os membros do grupo.

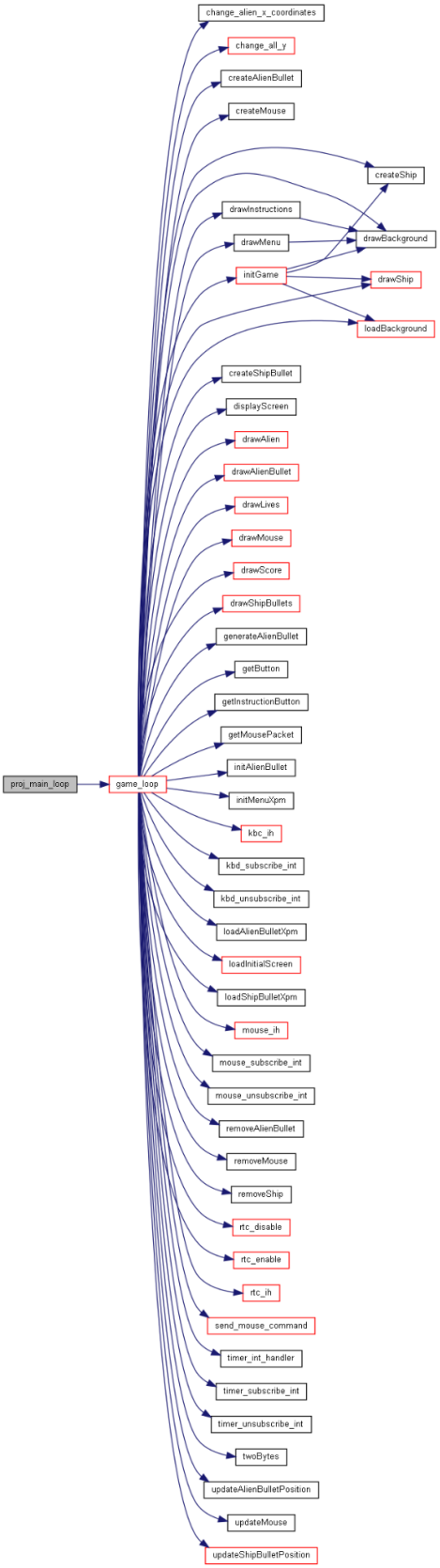
- **utils.c**

Módulo com funções multi-usos, úteis para vários aspetos do projeto.

Peso do módulo no projeto: 1%

Este módulo foi adaptado das aulas práticas, pelo que contém igual contribuição de todos os membros do grupo.

Function call graph



Para uma visão mais detalhada da imagem aceder à pasta doc e abrir o ficheiro: function_call_graph.png

5. Detalhes de implementação

Neste projeto usamos uma *object oriented approach* fazendo uso do tipo de dados struct, para atingir tal fim.

Game

No ficheiro game.c, a função game_loop possui o ciclo de interrupções (já fornecido nos labs da cadeira) onde são feitas as interações entre os diferentes elementos do jogo.

Foi criada a máquina de estados (enum) game_state que contém os seguintes valores: *MENU_DISPLAY, INSTRUCTIONS_DISPLAY, PLAYING, NEXT_ROUND* e *EXIT*.

Estes valores auxiliam o desenvolvimento do jogo já que contém o estado atual do jogo.

Menu

No ficheiro menu.h foi definida uma enum (Button) com os valores *START_BUTTON*, *INSTRUCTIONS_BUTTON*, *EXIT_BUTTON* e *INITIAL*. Estes valores pretendem avaliar a posição do cursor do rato no ecrã relativamente às opções que estão a ser mostradas e são obtidos na função getButton.

Com a função drawMenu, desenhamos o menu no ecrã, sendo que pode ou não ter uma opção selecionada (dependendo da posição do rato).

Ship

No ficheiro ship.h foi definida a struct Ship onde guardamos a posição da nave no ecrã (x e y), a velocidade (speed), a imagem da nave (img), o número de vidas que restam (lives) e a pontuação atual (score). É também declarada a struct keyActivity que contém a atividade da teclas pressionadas ou libertadas pelo utilizador.

Alien

No ficheiro alien.h foi definida a struct Alien onde é guardada a informação da posição e dimensão do alien. É também guardado um valor booleano (alive) que fica a falso se tiver ocorrido uma colisão e um inteiro (value) onde está guardado o valor do alien (pontuação que o jogador ganha quando o atinge).

Para guardar os aliens foi criado um array de 24 aliens que é inicializado na função initGame no ficheiro game.c.

O movimento dos aliens é feito nas interrupções do timer e foi usada uma variável (changeDir) para auxiliar a mudança de direção.

Ship Bullet

No ficheiro shipBullet.h foi definida a struct ShipBullet com a posição, a velocidade e a imagem da bala. Possui também um valor booleano (hitAlien) que fica a verdadeiro caso tenha existido uma colisão entre a bala da nave e um alien.

No jogo possuímos um array de 7 ShipBullets que é inicializado com a função initShipbullets. Uma nova bala apenas pode ser criada se existir algum lugar vago para ela no array. Esta verificação é feita na função createShipBullet.

Para verificar se existe alguma colisão entre uma bala e um alien é usada a função verifyAlienAndBulletCollision. Dá-se uma colisão quando as coordenadas x e y da bala estiverem entre os valores de x e y do alien.

Caso ocorra uma colisão com um alien, o valor da pontuação da nave é incrementado pelo *value* do alien atingido. Este aumento pode ser visualizado no canto inferior esquerdo do ecrã.

Alien Bullet

Tal como nas balas da nave, definimos a struct Alien Bullet com o valor de x e y da bala, a imagem e um valor booleano (active) que toma o valor verdadeiro quando a bala atinge o valor máximo de y (720).

Ao contrário das balas da nave, apenas existe uma bala de aliens no jogo. O alien que lança a bullet é escolhido aleatoriamente através da função generateAlienBullet que por sua vez usa a função rand(). Esta função apenas gera o índice do alien que irá lançar a bala e apenas na função game_loop é lançada a bala. Este alien tem de estar vivo, isto é, tem de ter o valor *alive* a verdadeiro.

Para verificar a existência de colisões entre a nave e a bala dos aliens, verificamos, tal como nas colisões com os aliens, a sobreposição das coordenadas da nave e da bala. Caso ocorra uma colisão, o número de vidas da ship diminui um valor. Esta diminuição pode ser verificada no canto inferior direito do ecrã.

Se o número de vidas da nave chegar a zero, o utilizador volta para o menu, onde poderá voltar a jogar ou sair do jogo.

Rondas

Para que o jogo não acabe sempre com a mesma pontuação e dificuldade, integramos rondas. O jogador passa para a ronda seguinte quando estiver no estado PLAYING e conseguir atingir todos os aliens que vê no ecrã. Aí o estado do jogo passa para NEXT_ROUND. Neste estado, todos os aliens são reiniciados enquanto que a nave continua com os mesmos valores anteriores.

Para aumentar a dificuldade de cada ronda, a velocidade dos aliens é aumentada, o que faz com que consigam chegar mais rapidamente à nave.

Double Buffer

Para que o movimento dos elementos do jogo fosse mais fluido, utilizamos a técnica de double buffering. Para esse propósito, desenhamos todos os elementos do jogo por cima do background no endereço video_mem e, quando já tivermos todos os elementos desenhados, com a função displayScreen (no ficheiro graphic.c) copiamos a memória inicial para o endereço display_mem. Quando a função mencionada é chamada no game_loop, o utilizador passa a ver todos os elementos do jogo.

6. Conclusão

Ao longo do projeto deparamo-nos com algumas dificuldades, nomeadamente o funcionamento com o Minix (ultrapassar e resolver bugs no código sem um debugger), a implementação do RTC, organização de código e na gestão de tempo.

Quanto à implementação do RTC, consideramos que os powerpoints relativamente ao assunto, especificamente as partes sobre as interrupções periódicas, não são tão informativos como os relativos aos outros tópicos(mouse, keyboard, timer, etc.), o que levou a uma dificuldade elevada na sua implementação.

Infelizmente não conseguimos cumprir todos os nossos objetivos para este trabalho. Para além do que implementamos, nós gostaríamos de ter implementado:

- O RTC na sua forma mais completa e utilizá-lo para periódicamente aumentar a velocidade dos aliens
- Outra entidade, o UFO, que passaria no rapidamente topo do ecrã e daria uma quantidade mistério aleatória de pontos ao player (dentro de certos limites), tal como existe no jogo original
- Bunkers, estruturas que protegiriam o player de um certo número de balas dos aliens antes de serem destruídos.
- O leaderboard que pudesse ser acessado a partir do menu inicial e que guardasse a pontuação dos player com o highest scores
- Um menu de pausa que, quando o player clica-se na tecla ESC lhe desse a opção de continuar ou de voltar ao menu.

Apesar de todas as nossas dificuldades, estamos satisfeitos com o resultado obtido e consideramos que a influência que este projeto teve no nosso entendimento de dispositivos de I/O e do funcionamento do kernel foi positiva. Foi também importante no nosso desenvolvimento enquanto estudantes deste curso e será, certamente, importante para o nosso futuro.