Universidade do Porto
Faculdade de Engenharia

Laboratório De Computadores

LCOM'S Project Report

Space Invaders

Projeto Realizado Por:

Tomás Fidalgo (up201906743)

João Silva (up201906478)

Índice

1.Instruções de Utilização do Jogo 3
2.Dispositivos usados 7
3.Organização/Estrutura do código 10
4.Detalhes de Implementação 13
5.Conclusões 14

1.Instruções de Utilização do Jogo

1.1 Ecrã Inicial

Figura 1 - Ecrã Inicial

Ao iniciar o jogo, é apresentado ao jogador um ecrã de introdução com informação como o título do jogo, as horas do dia, o HighScore local que o user obteve desde que iniciou o jogo até que o fechou. O jogador a qualquer momento pode avançar levando o cursor a qualquer botão do menu e carregar nele.

1.2 Menu

Como foi referido, o jogador pode a qualquer momento clicar em qualquer botão do menu para avançar do ecrã inicial.

O jogador pode então selecionar uma das 3 opções:

Figura 2 - Menu

New Game: inicia o jogo no modo singleplayer.

Help: Apresenta ao utilizador um ecrã de ajuda em relação a como jogar.

Quit: Fecha o jogo.

1.3 New Game

Quando o utilizador clica no menu na opção New Game carrega-se então o ecrã de jogo e o utilizador é convidado a jogar até que perca, ganhe, ou saia do jogo.

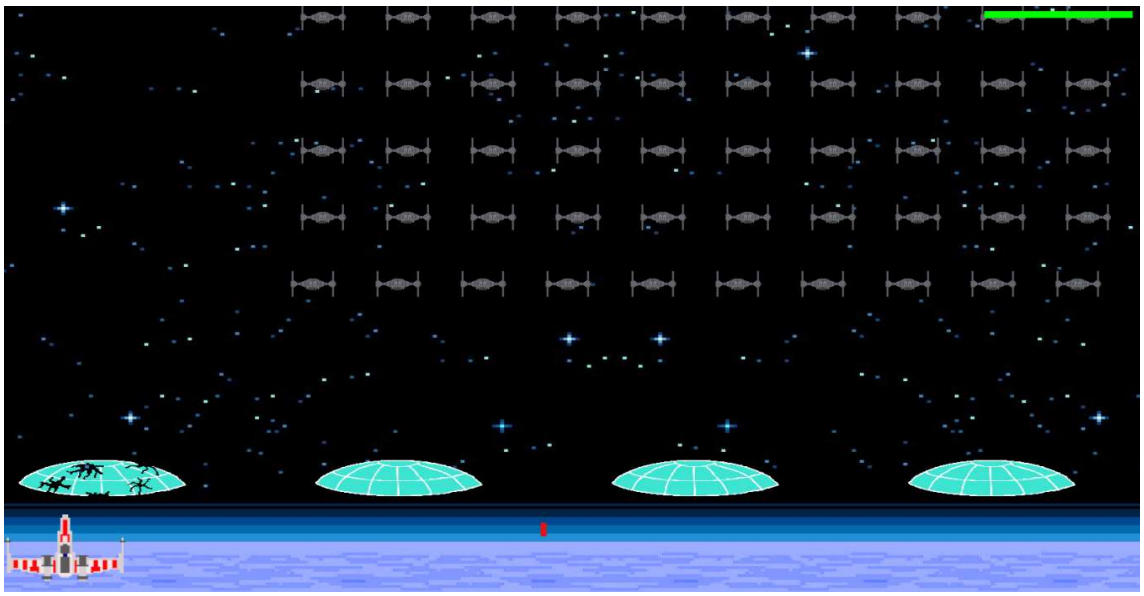

Figura 3- Ecrã de Jogo

Figura 4- Ecrã de Jogo na fase final

Tal como vemos nas figuras 3 e 4, o jogo é constituído, numa fase inicial pela nave espacial que o utilizador controla, pelas naves inimigas e numa fase final por uma nave maior que é o último inimigo a destruir antes de conseguirmos ganhar o jogo.

Figura 5- Ecrã de vitória

O objetivo é então tentar destruir 50 naves inimigas utilizando o rato para disparar contra estas mesmas, antes que as naves inimigas destruam a nave que o utilizador move, sendo que esta tem uma certa quantidade de vida que é possível ver no topo direito do ecrã. Se a vida terminar então o

utilizador perdeu o jogo e capta um score igual à quantidade de naves inimigas que destruiu. No entanto, se o utilizador conseguir abater as 50 naves inimigas chega à fase final e se conseguir abater o último inimigo atirando em direção ao centro vermelho o utilizador ganha o jogo (figura 5).

Se o utilizador pretender sair do jogo prematuramente ou depois de perder/ganhar, basta clicar na tecla escape [ESC].

O movimento e o disparo da nave são feitos com o teclado e o rato respetivamente.

Os controlos a utilizar são:

1. "A" - A nave move-se para a esquerda;
2. "D" – A nave move-se para a direita;
3. [Botão esquerdo do rato] – A nave dispara em direção vertical.

1.4 Help

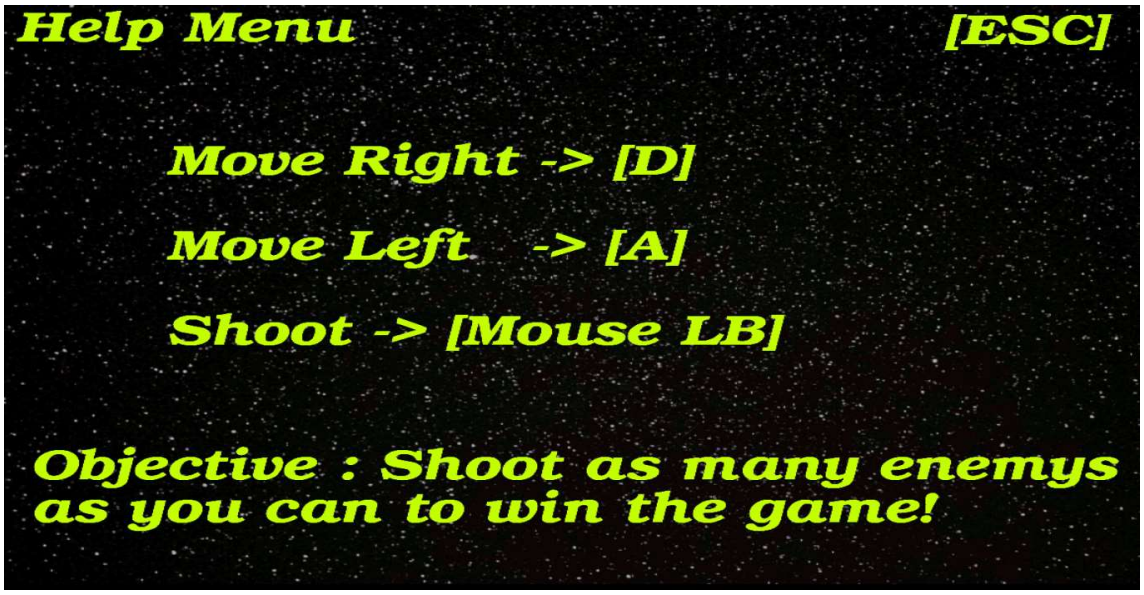

Figura 6- Help

Quando o utilizador clica na opção Help no menu, é apresentado um ecrã de ajuda ao utilizador (figura 6), que lhe permite ver os controlos do jogo

que já foram referidos neste relatório e o objetivo de forma bastante generalizada.

Para fechar este ecrã de ajuda e voltar ao menu basta apenas pressionar a tecla escape no teclado [ESC].

2.Dispositivos usados

Para a implementação deste jogo usamos no total de 5 dispositivos, sendo eles o timer, o teclado, o rato, e a placa gráfica.

Dispositivo	**Funcionalidade**	**Interrupções [y/n]**
Timer	Controlo da frame rate	Y
Teclado	Navegação entre as opções do menu.	Y
Rato	Seleção de opções do Menu e disparo da nave do utilizador.	Y
RTC	Imprimir hora do dia.	Y
Placa Gráfica	Imprimir funcionalidades do menu e o jogo em si.	N

2.2 Timer

O timer é um dispositivo que a sua funcionalidade está frequente no código do projeto, a principal sendo para o controlo da frame rate, esta que controla a forma como tudo é impresso no ecrã, como por exemplo a movimentação de todos sprites neste jogo é possível através do controlo da frame rate que o timer faz. Uma outra funcionalidade semelhante, mas

de certa forma única em que usamos o timer é a movimentação periódica do inimigo final, sendo que ele se move numa duração de 5 segundos e fica estático numa duração de 2 segundos.

Esta funcionalidades estão implementadas maioritariamente na função game(..) no ficheiro game.c. No entanto, o timer é usado em outras funções como load_menu() no ficheiro menu.c.

2.2 Placa Gráfica

A placa gráfica é o motor de toda a nossa movimentação de sprites na implementação do nosso código.

O vídeo mode utilizado neste projeto é o 0x14c.

A resolução é 1152(resolução horizontal) por 854(resolução vertical) e o número de cores é 2^32.

Neste Projeto foi implementado o double buffering, uma técnica para implementar um grafismo que não mostra (ou apresenta menos) falhas.

Existe então a implementação desta técnica da função double_buff() no código.

Como foi referido, a placa gráfica foi utilizada principalmente para mover objetos no decorrer do jogo, assim como para detetar colisões entre tiros e naves.

De forma a criar texto foram utilizados sprites de palavras e para criar todo o tipo de números foram utilizados sprites de números.

A utilização da placa gráfica está presente em funções como color_pixel() no ficheiro graphic.c , print_sprite() no ficheiro sprite.c e em termos de colisões na função check_pixel() no ficheiro graphic.c que é chamada em funções de colisão em game.c.

2.3 Teclado

O teclado é usado principalmente para controlo da nave usando as suas interrupções para captar scancodes que revelam se o utilizador premiu ou não a tecla “A” ou “D”.

Sendo também usado para a navegação entre o menu, o jogo e o separador Help com o premir da tecla [ESC].

Podemos encontrar a implementação da primeira funcionalidade na função game() no ficheiro game.c e a segunda também na função game() do mesmo file assim como na função handle_help() no ficheiro Menu.c.

2.4 Rato

Utilizamos o rato para fazer primeiramente a movimentação do cursor no menu com o uso da posição do rato, para detetar colisões com as opções do menu e detetar o premir do rato no jogo para disparar, isto com o uso dos botões do rato.

Especificamente, usamos as interrupções do rato para conseguirmos processar os pacotes enviados pelo controlador e assim de acordo com os eventos do rato podemos definir o estado do programa.

Estas funcionalidades podem ser verificadas na check_collisions() do ficheiro menu.c , game() no game.c e move_cursor() no sprite.c.

2.5 RTC (Real Time Clock)

Neste projeto, o RTC é usado apenas para imprimir a hora do dia no menu, assim, foi preciso subscrever às interrupções do RTC e ativar interrupções de fonte de atualização da hora. Tendo feito isto, podemos ler os registos das horas, minutos e segundos do RTC para a impressão da hora.

Esta implementação do RTC pode ser verificada nas funções rtc_ih() e rtc_read_reg() no ficheiro rtc.c. As interrupções do RTC verificam-se na função load_menu() no ficheiro menu.c.

3.Organização/Estrutura do código

Módulos

Teclado

A maior parte do teclado foi implementado no lab3, assim, o teclado neste projeto torna se uma versão do lab3, em que é possível receber scancodes e escrever comandos para o KBC.

Implementado já no lab3.

Rato

Maior parte do código implementado no lab4, não existe diferença significativa com a versão do lab, funcionalidades parecidas com a test_gesture() , e tal como no lab4 utiliza se uma array para guardar o pacote de dados enviado pelo rato.

Implementado já no lab4.

Timer

Praticamente semelhante ao lab2.

Implementado no lab2.

RTC

As funcionalidades implementadas permitem ativar e desativar o RTC, ler registos do rtc, ativar as interrupções de atualização de hora,

colocar numa estrutura de dados como um array os dados lidos, neste caso as horas, os minutos e os segundos.

Implementado por João Silva.

Gráfica

Em termos de funções implementadas com relação à gráfica é tudo à versão do lab5, incluindo funções da VBE. A notar é a implementação do double buffering e algumas funções de desenho.

Implementado por: Tomás fidalgo (double buffering, funções de desenho), João Silva (funções de desenho).

Sprite

Semelhante ao código retirado das transparências de LCOM, class sprite.h (por jcard@fe.up.pt). No entanto, existem funções implementadas como a move_cursor() e funções de posição de objetos.

Implementado por: Tomás fidalgo (funções de posição), João Silva (move_cursor()).

Menu

O Menu controla o estado do programa, é responsável por redefinir certas variáveis, verificar colisões e tratar de cada uma das opções do menu.

Implementado por: João Silva

Game

Controla toda a dinâmica de jogo, seja criar os sprites do jogo, a sua movimentação, gestão do score, verificar colisões entre vários objetos e verificar se o jogo acabou ou não.

Implementado por: João Silva (colisões, gestão do score, verificar se o jogo acabou), Tomás Fidalgo (todas as funções de movimento).

Enemies

O módulo enemies gere e guarda tudo relacionado com os inimigos identificados a partir de structs, implementa também todas as funções de impressão e movimento de inimigos.

Implementado por: Tomás Fidalgo

Spaceship

O módulo spaceship guarda toda a informação relacionada com a nave que o utilizador controla dentro de uma struct e implementa as funções de impressão e movimento da nave.

Implementado por: Tomás Fidalgo

Peso relativo dos Módulos

Módulo	Peso Relativo
Teclado	5%
Rato	5%
Timer	5%
RTC	1%
Gráfica	10%
Sprite	5%
Menu	20%
Game	45 %
Enemies	2%
Spaceship	2%

Function Call Graphic

4.Detalhes de Implementação

Utilização dos Pacotes do Rato

No menu o mindset era criar código que respondesse aos eventos do rato dado que no lab4 foi explorado o conceito de state machine e de event driven code. O objetivo era de facto que segundo um determinado evento do rato, seja o deslocamento ou o premir do botão, o estado do programa mudava, por isso a implementação de um tipo enumerado que contém opções do Menu, com o tipo enumerado torna-se fácil o desenho de uma pequena state machine que possibilita dinâmica ao nosso jogo.

Colisões

Colisão de Sprites é uma parte bastante importante do nosso código. Não foi abordada nos labs por isso teríamos de decidir a melhor técnica para implementar esta funcionalidade. Primeiramente, de forma intuitiva, verificar a colisão de coordenadas x e y entre sprites era o objetivo, porém surgiu a ideia depois de pesquisa de verificar se um determinado sprite tem uma determinada cor a impor-se sobre ele o que faria muito mais sentido devido a nossa utilização de sprites com cor única.

5.Conclusões

Um ponto positivo sobre a cadeira em geral são os labs que obrigam de certa forma a preparação do projeto o que facilita muito a implementação e o “learning” de muitos conceitos essenciais para o projeto final.