

“Asteroids” para Minix

Projeto de Desenvolvimento para a cadeira de Laboratório de Computadores

MIEIC 2019/2020

**Turma: 8**

**Grupo: 02**

André Filipe de Soveral Torres Lopes dos Santos nº 200505634

Raul Manuel Fidalgo da Silva Teixeira Viana nº 201208089

Índice

[1. Instruções de Utilização do Programa 3](#_Toc28855773)

[1.1 Menu Principal 3](#_Toc28855774)

[1.2 Ecrã de jogo 3](#_Toc28855775)

[2. Estado do Projeto 4](#_Toc28855776)

[2.1. Resumo dos dispositivos implementados: 4](#_Toc28855777)

[2.2. Descrição detalhada da utilização de dispositivos 4](#_Toc28855778)

[2.2.1. Temporizador 4](#_Toc28855779)

[2.2.2 Teclado 4](#_Toc28855780)

[2.2.3. Rato 5](#_Toc28855781)

[2.2.4. Placa Gráfica 5](#_Toc28855782)

[3. Organização/Estrutura do Código 6](#_Toc28855783)

[3.1. Módulos Existentes 6](#_Toc28855784)

[3.1.1. Módulo “Play” 6](#_Toc28855785)

[3.1.2. Módulo “Graphics” 6](#_Toc28855786)

[3.1.2. Módulo “Keyboard” 6](#_Toc28855787)

[3.1.3. Módulo “Mouse” 6](#_Toc28855788)

[3.1.4. Módulo “Object” 7](#_Toc28855789)

[3.1.5. Módulo “Menu” 7](#_Toc28855790)

[3.2. Grafo de chama de função 7](#_Toc28855791)

[4. Detalhes de implementação 8](#_Toc28855792)

[5. Conclusões 9](#_Toc28855793)

# Instruções de Utilização do Programa

## Menu Principal

Uma imagem com captura de ecrã, monitor

Descrição gerada automaticamenteQuando o programa é iniciado é apresentado o menu inicial:

Figura - Imagem do Menu Principal do Jogo

Neste ecrã, que serve de introdução ao jogo, o utilizador poderá escolher utilizando o rato, uma de duas opções clicando com o botão esquerdo:

* ***Start*:** esta opção fará com que o jogo comece/recomece.
* ***Exit*:** esta opção termina o jogo.

## Ecrã de jogo

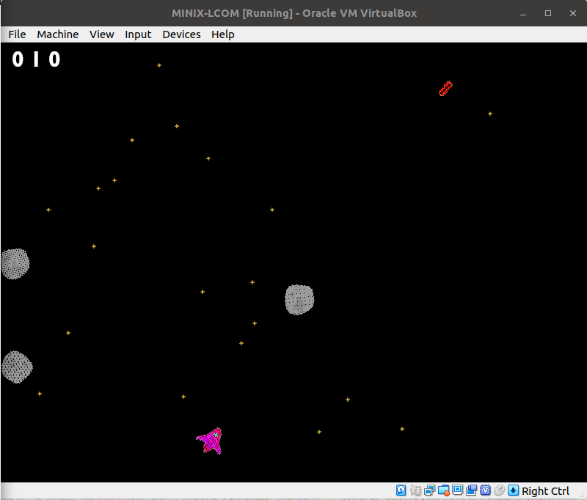
O modo de jogo pode ser representado pela seguinte imagem:

Figura 2 - Imagem do Menu Principal do Jogo

Na imagem em cima pode-se ver uma imagem do jogo em ação. O jogador controla a nave espacial e o objetivo é conseguir destruir o maior número possível de asteroides, evitando-os. Os controlos são os seguintes:

* **W:** Faz com a nave espacial suba no ecrã;
* **A:** Faz com que a nave vá para a esquerda no ecrã;
* **S:** Faz com que a nave desça no ecrã;
* **D:** Faz com que a nave vá para a direita no ecrã;
* **ESC:** Termina o jogo e volta ao ecrã de menu;
* **Movimento do Rato:** Orienta o sentido para o qual a nave está virada;
* **Botão esquerdo do Rato:** Faz a nave disparar as suas munições (ilimitadas).

# Estado do Projeto

## 2.1. Resumo dos dispositivos implementados:

|  |  |  |
| --- | --- | --- |
| **Dispositivo** | **Utilização** | **Interrupções** |
| Temporizador | Medir intervalos de tempo para mover objetos pelo ecrã | Sim |
| Teclado | Controlar o movimento da nave espacial pelo ecrã e sair abruptamente do programa | Sim |
| Rato | Determinar orientação da nave espacial e escolha de opções no menu | Sim |
| Placa Gráfica | Mostrar todos os elementos de jogo ao jogador, animação e deteção de colisões | Não |

## 2.2. Descrição detalhada da utilização de dispositivos

### 2.2.1. Temporizador

#### Descrição de Utilização

As interrupções do *timer* são utilizadas para criar os intervalos de tempo que permitem fazer a animação dos diferentes componentes no ecrã e o movimento dos mesmos, e ainda para contabilizar determinados tempos de desenho, como por exemplo as explosões.

#### Funções em que é utilizado:

* menu();
* play();

As funções do *timer* encontram-se implementadas nos ficheiros **timer.c** e as suas respetivas macros no ficheiro **i8254.h**.

### 2.2.2 Teclado

#### Descrição de Utilização

O teclado é utilizado para o controlo da nave espacial no jogo e para sair abruptamente do jogo pressionando na tecla ESC.

#### Funções em que é utilizado

* process\_ship\_speed(Object \*ship, Keys key);
* menu();
* play();

As funções do teclado estão implementadas nos ficheiros **keyboard.c** e as suas respetivas macros no ficheiro **i8042.h**.

### 2.2.3. Rato

#### Descrição de Utilização

O rato é utilizado em duas instâncias diferentes: menu e jogo. A sua utilização é a seguinte:

* **Menu:**  No ecrã de menu, o rato é utilizado para controlar um cursor que se movimenta pelo ecrã, controlado pelo movimento do rato, sendo que esse cursor pode ser utilizado para selecionar uma das opções através de um clique com o botão esquerdo do rato quando este está sobre um dos botões presentes no ecrã.
* **Jogo:** Durante a execução do jogo, os movimentos do rato gerem a orientação da nave espacial, sendo que esta última está sempre a apontar para a localização do ponteiro do rato. O botão esquerdo do rato é utilizado para fazer com a nave dispare as suas munições na direção para o qual ela está orientada.

#### Funções em que é utilizado

* menu();
* play();

As funções do rato estão implementadas nos ficheiros **mouse.c** e as suas respetivas macros no ficheiro **i8042.h**.

### 2.2.4. Placa Gráfica

#### Descrição de Utilização

Neste projeto, apesar de toda a biblioteca desenvolvida durante os laboratórios estar preparada para funcionar em qualquer tipo de modo gráfico disponível, é definido apenas um modo gráfico. Esta decisão foi tomada essencialmente devido ao grande número de *xpm’s* que necessariamente teriam de ser criados (cada *xpm* teria que ser recriado, com as dimensões apropriadas a cada modo gráfico).

Assim, o modo gráfico utilizado é o modo 0x115, que possui as seguintes características:

* **Resolução de Ecrã:** 800x600.
* **Modelo de Cor:** Direto.
* **Bits por pixel (R:G:B):**  24 (8:8:8).

Foi implementado um *double buffering* e ainda a deteção de colisões através do método da caixa.

O *double buffering foi implementado* da criação de um *buffer* extra*,* onde são escritos todos os pixéis da imagem do écran. Posteriormente esse buffer é copiado para a memória gráfica da forma mais rápida possível, através da função *copyBuffToRam().* Desta forma foi possível reduzir o *flickering* de forma total, sendo que o jogo corre de uma forma muito fluida.

Para a deteção de colisões foi implementada uma função *checkCollisions()* que deteta as colisões entre os *sprites* através da aplicação de uma caixa á sua volta. Se as caixas se sobrepuserem é confirmada uma colisão.

Foi desenvolvida uma estrutura de dados *Object* que funciona como uma espécie abstração de classe para os objetos a serem desenhados no ecran. Esta estrutura é responsável por guardar todos os dados de um objeto, que asseguram o seu desenho e a sua movimentação no ecran, tendo funções associadas para a sua modificação. Está implementada no ficheiro **object.c.**  As respetivas macros encontram-se no respetivo ficheiro **object.h.**

As imagens utilizadas no menu foram criadas por nós, enquanto que as utilizadas no modo de jogo foram todas retiradas da internet, nos sites referidos na bibliografia, com exceção das estrelas que também foram criadas por nós.

As funções relacionadas com a placa gráfica estão implementadas nos ficheiros **graphics.c** e as respetivas macros declaradas em **graphics.h.**

#### Funções em que é utilizado

* menu();
* play();

# Organização/Estrutura do Código

## Módulos Existentes

### Módulo “Play”

Este é o módulo principal em que o “game loop” corre. Este módulo interage com os outros módulos de maneira a poder correr o jogo em si. Este módulo gere a interação com os diferentes componentes e também a interação entre o jogador e o jogo.

Sendo este o módulo principal do jogo, este terá um peso mínimo de 50%.

### 3.1.2. Módulo “Graphics”

Este módulo serve principalmente como intermediário entre o programa e a placa de vídeo. As diferentes funções executam métodos diferentes para poder “desenhar” os objetos no ecrã e gerir o que neste é desenhado em modo gráfico.

Este módulo tem um peso de INSERIR

### Módulo “Keyboard”

Este módulo gere a interação do programa com o teclado, sendo que este módulo tem as funções e drivers necessários para o utilizador poder utilizar o teclado neste programa desenvolvido para Minix.

Este Módulo tem um peso de INSERIR

### Módulo “Mouse”

Este módulo gere a interação do programa com o rato, sendo que este módulo tem as funções e drivers necessários para o utilizador poder utilizar o teclado neste programa desenvolvido para Minix.

Este Módulo tem um peso de INSERIR

### Módulo “Object”

Este módulo tem como objetivo a criação e gestão de diferentes “objetos” (structs em C). As funções são usadas para criar, destruir e mover objetos de diferentes tipos.

Este módulo tem um peso de INSERIR

### Módulo “Menu”

Este módulo tem como função a criação de um Menu básico para o programa para o utilizador poder começar o jogo e sair do programa.

Este módulo tem um peso de INSERIR

## Grafo de chama de função

# Detalhes de implementação

## Interrupções

Optou-se por colocar todas as variáveis necessárias ao ciclo que intercepta as interrupções como variáveis estáticas do ficheiro *game.c*. Desta forma foi possível criar dois ciclos de interceção de interrupções, um no menu e outro para o modo de jogo utilizando as mesmas variáveis. Inicialmente houve algum receio de que houvesse implicações negativas neste desenho ao passar do modo de menu para o modo de jogo e vice-versa, mas acabou por funcionar muito bem.

## Deteção de Colisões

A deteção de colisões implicou algum trabalho e retrocesso. Isto porque fomos aconselhados a detetar a colisão entre os objetos ao nível da cor, ou seja, ao desenhar um pixel ir detetando se nesse pixel já tinha sido desenhado outro objeto. Esta abordagem revelou-se inapropriada pelo facto de que a nossa versão da função de desenho: *draw\_xpm()* já ter sido pensada para incluir outras informações no retorno e na chamada da função *coloring()* que faz a alteração de cor do pixel. Desta forma não seria trivial implementar as alterações necessárias para a deteção de colisão ao nível do pixel.

Assim optou-se por realizar a deteção de colisão por caixa, visto tratar-se de uma implementação mais simples, embora mais rudimentar e imprecisa. Em cada ciclo de desenho, é calculada uma caixa para cada objeto, caixa essa que confina as extremidades do objeto, e depois todas as caixas são comparadas para detetar as colisões. Ao ser detetada inicia-se o desenho de uma explosão no centro do espaço entre esses dois objetos.

## Alternância de processamento

Tanto no menu como no modo de jogo foi implementada uma forma de aproveitar o tempo de processador, tentando distribuir o processamento exigido para desenhar cada *frame* em duas fases. O ciclo principal do menu e do modo de jogo têm uma fase de processamento do desenho no écran que acontece a cada 5 interrupções do *timer*. Como o timer está programado para realizar interrupções à frequência do sistema, 60 Hz, isso corresponde a aproximadamente a 83ms. Por outro lado, todo o processamento que implica o cálculo da nova posição de cada objeto em função da sua velocidade e a nova posição em função da sua rotação é efetuado com o mesmo espaçamento de 83ms, mas de forma desfasada, sendo que desta forma o tempo de processamento é mais bem aproveitado.

## Rotação e desenho dos objetos

Os objetos são desenhados no ecrã através da chamada da função *draw\_xpm* para objetos do modo de jogo e *draw\_menu* para objetos do menu. Esta função está encarregue de chamar posteriormente *coloring* para desenhar cada pixel do objeto.

A função *coloring* recebe ente outros argumentos o resultado do cálculo do seno e do cosseno das coordenadas do pixel a desenhar, sendo desta forma que cada pixel é desenhado numa nova posição em conformidade com o angulo de rotação para ele definido naquele momento. Retorna ainda informação acerca de o pixel ter sido desenhado dentro do ecrã. Isto é muito importante pois objetos que estejam a ser desenhados fora do ecrã são tornados inativos e podem dessa forma ser reciclados e utilizados posteriormente quando para isso houver necessidade.

Tanto a função *draw\_xpm* como a *draw\_menu* recebem o retorno de *coloring* e avaliam quando tornar o objeto inativo, retornando também essa informação para as funções que realizaram a chamada.

A única diferença entre estas duas funções reside no facto de *draw\_menu* não fazer qualquer cálculo em relação à rotação dos objetos. Isto aconteceu por questões de eficiência, uma vez que os objetos do menu são bastantes grandes e acarretam bastante processamento.

## *Double Buffering*

Para a redução do fenómeno de *flickering* foi implementado um *double buffering*. Ao mesmo tempo que a memória gráfica foi mapeada para o programa foi também criado um buffer com exatamente o mesmo tamanho.

Assim o desenho dos objetos, realizado através da função *coloring,* foi efetuado no buffer local. Apenas quando todo o ecrã se encontra desenhado é que é feita uma cópia de todo o conteúdo do buffer local para a memória gráfica de uma forma o mais rápida possível.

Antes de começar a desenhar no ecrã este é apagado através da chamada da função *clearScreen.*

# Conclusões

A realização deste trabalho levou a que fossem realmente atingidos os objetivos da unidade curricular, sendo assimilados e compreendidos os conhecimentos necessários à utilização de interfaces de hardware, desenvolvimento de software de baixo nível e programação em C.

A unidade curricular apresenta uma dificuldade um pouco acima das restantes deste semestre. Não tanto pelos conteúdos, mas pela forma como está estruturada/definida. Implica muitas horas de dedicação, trabalho de *self-learning* fora das horas de aula, principalmente porque os labs estão estruturados de uma forma pouco “top-down”, são demasiado intrincados. Isso leva a que durante as aulas práticas surjam muitas dúvidas e o professor não consiga atender aos alunos todos.

O teste é também algo difícil, e, na nossa opinião, deveria ter mais testes unitários, de forma a que as notas fossem mais distribuídas. Isto porque da forma como a avaliação é realizada o resultado acaba por ser “tudo ou nada”, 0 ou 20, em cada função a desenvolver (de uma forma caricaturada, obviamente). A avaliação do teste deveria ser mais incremental.

Por fim, achamos que deveria haver mais feddback durante o semestre, por exemplo, poderiam ser publicadas algumas notas de alguns labs, de forma aos alunos se aperceberem de como estão a evoluir.

# Bibliografia

1. Nave: <https://www.kindpng.com/imgv/ToTioi_spaceship-clipart-images-clipart-cute-rocket-clipart-hd/>
2. Meteorito:<https://www.pngjoy.com/preview/c6x8w7v7z4o1x0_asteroid-asteroid-clipart-png-download/>
3. Projétil Laser: <https://www.festivalclaca.cat/view/117646/>