U.PORTO

FEUP FACULDADE DE ENGENHARIA
UNIVERSIDADE DO PORTO

Relatório do projeto Asteroids

António Nunes da Cruz (up201603526)
Tiago Araújo Castro (up201606186)

Turma 5 - Grupo 04

Relatório do projeto realizado no âmbito da unidade curricular Laboratório de Computadores do Mestrado Integrado em Engenharia Informática e Computação

2º ano - 2017/2018

Índice

1. Asteroids

1.1) Descrição

O objetivo do jogo é destruir asteroides e os seus fragmentos enquanto se desvia dos mesmos. O jogador controla uma pequena nave triangular que roda e dispara em todas as direções. Cada jogo começa com poucos asteroides no ecrã e vão nascendo mais de forma aleatória com o passar de tempo, incrementando a dificuldade com o aumento do score. Quanto mais tempo aguentar sem colidir com asteroides e mais asteroides destruir, maior será o score final.

1.2) Instruções

Menu Principal:

Após correr o programa aparece o menu inicial com 3 botões:

- Play: Que redireciona o jogador para o ecrã de jogo.
- Highscore: Que permite ao utilizador verificar os melhores resultados
- Exit: Que fecha o programa

Figura 1 - Menu Principal

Play:

O botão de play começa o jogo, a partir deste momento o score vai ser incrementado com o tempo sobrevivido e com o número de asteroides destruídos. O movimento da nave é controlado com as teclas do teclado W/A/S/D para movimentar para cima,baixo,esquerda e direita respetivamente, a nave roda e dispara balas na direção em que o rato está a apontar através do clique do botão do lado esquerdo do rato, podendo inclusive deixar pressionado. Quando uma bala colide com um asteroide, este vai dividir-se em 2 mais pequenos, a não ser que seja do tamanho menor que nesse caso apenas desaparecerá. Mal a nave do jogador colida com um asteroide o jogo terminará e aparecerá o ecrã de fim de jogo.

Figura 2 - Jogo

1. Asteroids

1.2) Instruções

Play:

Quando jogador colide com um asteroide se o seu score for dos 3 melhores então aparecerá um ecrã com o seu resultado que lhe permite inserir o seu nome que será gravado, após pressionar enter, juntamente com a data (Dia/mês/ano-hora:min) em que foi feito.

Caso contrário apenas aparece o seu resultado final, com a opção para voltar para o menu principal.

Figura 3 - New Highscore

Figura 4 - Game Over

Highscore:

Após pressionar no botão highscore o jogador pode ver o nome, o score e a data em que os melhores jogadores conseguiram obter esses resultados. Estes resultados são guardados após terminar o programa num ficheiro de texto e são lidos quando este inicia.

Figura 5 - Highscores

Exit:

O botão de exit termina o programa, gravando os scores efetuados nessa sessão.

2. Estado do Projeto

Dispositivo	**Utilização**	**Interrupções**
Timer	Atualizar o estado do jogo (incrementar o score, criar asteroides) e atualizar o ecrã.	Sim
Teclado	Movimento da nave, escrita do nome para o highscore e opção de término de jogo com a tecla ESC.	Sim
Rato	Navegação dos menus, disparo das balas, e indica a direção da nave e da trajetória das balas.	Sim
Placa Gráfica	Visualização dos menus, estado de jogo e imagens.	Não
Real Time Clock (RTC)	Usado para registar nos highscores a data em que este foi feito.	Sim

A tabela acima representa de forma resumida a função que cada dispositivo tomou no projeto. Abaixo está descrito de forma mais completa estas funcionalidades.

2.0) Timer

O timer é utilizado constantemente uma vez que o estado do jogo é atualizado a cada interrupt:

- **Verificação de colisões** (checkBulletAsteroidCollision() / checkAsteroidShipCollision())
- **Criação e atualização da posição das balas e asteroides** (drawBullets() / drawBuff() / updateBullets() / updateAsteroids() / randomAsteroid())
- **Atualização da direção e posição da nave** (drawSpaceship())

Por outro lado o timer é utilizado também para incrementar e mostrar o score à medida que o jogador consegue sobreviver (drawHudScore()), e aumentar a dificuldade, ou seja a frequência com que nascem mais asteroides.

2. Estado do Projeto

2.1) Teclado

A função principal do teclado é controlar o movimento da nave durante o modo de jogo(updateSpaceship()), porém é também utilizado para escrever o nome do jogador que bate um novo highscore (keyHandler() / printName()) e para sair do modo de jogo através da tecla ESC.

2.2) Rato

O rato é utilizado ao longo do programa todo, inicialmente é criada a sua estrutura e depois a sua posição e o botão do lado esquerdo são alterados a cada interrupção (getMouse()). O botão do lado esquerdo não só é usado para clicar nos menus como também para disparar as balas quando se encontra no modo de jogo.

2.3) Placa Gráfica

A placa gráfica é usada para comunicar com o utilizador através dela representa-se tudo no ecrã. Foi utilizado o modo gráfico 0x117 e a fim de evitar problemas com flickering foi utilizada a técnica de double buffering, ou seja primeiro desenhar para um buffer paralelo e depois de esta imagem estar pronta é que se copia a informação para o buffer principal.
A única função da VBE utilizada foi a vbe_get_mode_info().
A deteção de colisões foi feita através das imagens (bitmaps) e foram utilizados bitmaps com rotação.
A placa gráfica é atualizada a cada interrupção do timer.

2.3) Real Time Clock RTC

O RTC é usado para obter a hora e data a cada momento. Foram subscritos os interrupts do RTC de forma que estes são handled “mandando” a informação dos registos para um objeto da struct do tipo Date onde fica armazenado o ano, o mês, o dia, a hora, o minuto e o segundo de cada interrupt. É subscrito o modo de update interrupts e a cada chamada do handler é limpa a flag do registo C.

3. Organização do código

Nesta secção será feita um resumo da forma como o código foi estruturado e dividido entre ficheiros/módulos.

3.0) Main

Este módulo pode ser considerado a “raiz” do jogo uma vez que irá chamar todas as funções e é a partir deste que será executado o programa. Neste existe um ciclo principal que funciona como uma state machine, em que dependendo do estado irá chamar diferentes funções, sejam elas para inicar o jogo ou um menu. Este ciclo apenas termina quando o utilizador pressiona no botão EXIT no menu principal.

Peso no projeto: 7% Aluno responsável: Tiago Castro (70%)

3.1) Asteroid

Este módulo está responsável por todas as implementações relacionadas com asteroids.
Um “asteroid” é uma struct contendo:

- ❏ Size: que varia entre 1-3, todos os asteroids inicialmente são criados com tamanho 3 e após colidirem com uma bala são destruídos e criam 2 asteroides de tamanho inferior, caso este seja de tamanho 1 então desaparece.
- ❏ map: que é um pointer para o Bitmap do asteroide que vai depender do tamanho
- ❏ xpos/ypos: que são ints correspondentes à sua posição no eixo dos x e dos y
- ❏ xspeed/yspeed: que são os valores da velocidade em x e em y
- ❏ indx: que corresponde ao índice em que este se encontra no array de “asteroids”

Neste módulo existem as funções de criação/destruição/atualização de posição/desenho/ de asteroides juntamente com outras funções auxiliares.

Peso no projeto: 10% Aluno responsável: António Cruz (100%)

3.2) Bitmap

Este módulo foi uma adaptação do código feito pelo antigo aluno Henrique Ferrolho, foi grande parte feita de forma semelhante, porém foram feitas pequenas adaptações.
Foram usados short pointers em vez de char, uma vez que fazia mais sentido a nosso ver já que a cor seriam 2 bytes, e também foi alterada a função de drawBitmap() a fim de poder ignorar uma certa cor e assim ter imagens com fundo “invisivel”.
URL para o blog: http://difusal.blogspot.pt/2014/09/minixtutorial-8-loading-bmp-images.html

Peso no projeto 8% Aluno responsável: António Cruz (100%)

3. Organização do código

3.3) Collision

Este módulo contém todas as funções responsáveis por detetar colisões entre Bitmaps no jogo, ou seja Balas-Asteroide/ Asteroide-Nave. Estas duas funções percorrem os arrays de balas e asteroides e verificam a cada interrupção do timer se existiu colisão.

Peso no projeto: 7% Aluno responsável:António Cruz(100%)

3.4) EndGameState

Este módulo é responsável pelo ecrã pós-jogo onde está presente o score obtido pelo jogador no seu jogo. Haverá uma variável global da struct “EndGameState” contendo:

- ❏ done: Uma flag que indica se o state está completo ou não
- ❏ exitButton: Struct com a informação do botão
- ❏ background: Bitmap do fundo do estado

Peso no projeto: 2% Aluno responsável: Tiago Castro(100%)

3.5) Game

Este módulo é o responsável pelo modo de jogo, irá ter uma variável global da struct “game”, que contém:

- ❏ asteroids[100]: Um array de apontadores para todos os asteroides em jogo.
- ❏ ship: Um apontador a nave em jogo
- ❏ bullets[100]: Um array de apontadores para todas as balas em jogo.
- ❏ score: Contendo o score atual do jogador

Quando o jogo começa será inicializada a struct game, com todos os asteroides e balas a NULL e será feito load de todas as imagens utilizadas para o mesmo. Na função startGame() existe um ciclo onde a cada interrupção do timer / teclado / rato / rtc , serão atualizados o ecrã / movimento da nave / cursor do rato / data respetivamente, este ciclo será quebrado quando existir colisão com um asteróide ou quando o jogador pressiona na tecla ESC.

Peso no projeto: 7% Aluno responsável: António Cruz (80%)

3. Organização do código

3.6) Graphics

Este módulo é responsável pela implementação da struct Rectangle que possui a informação dum botão retangular (borders) constituída por:

- ❏ x1: início do retângulo em x
- ❏ y1: início do retângulo em y
- ❏ x2: fim do retângulo em x
- ❏ y2: fim do retângulo em y

Peso no projeto: 3% Aluno responsável: Tiago Castro (100%)

3.7) HighscoreState

Este módulo é responsável por dar display dos highscores obtidos, ou seja, nome, score e data (que por sua vez contém ano, mês, dia, hora e minuto), tendo um botão para voltar ao menu. Está implementado neste modulo também a leitura e escrita nos ficheiros dos highscores. Haverá uma variável global da struct “HighscoreState” contendo:

- ❏ done: Uma flag que indica se o state está completo ou não
- ❏ exitButton: Struct com a informação do botão
- ❏ background: Bitmap do fundo do estado

Também há 3 variáveis globais da struct “HighestScore” que tem a informação de um highscore, contém:

- ❏ score: Score do highscore
- ❏ date: Data em que o score foi obtido (contém ano, mês, dia, hora, minuto e segundo)
- ❏ name: nome do user

Peso no projeto: 6% Aluno responsável: Tiago Castro (70%)

3.8) Keyboard

Este módulo é responsável pela implementação do keyboard, tendo sido implementadas algumas funcionalidades dos labs, contendo o handler em assembly.

Peso no projeto: 3% Aluno responsável: Tiago Castro (70%)

3. Organização do código

3.9) MainMenuState

Este módulo é responsável por iniciar o user no programa dando-lhe opção de sair do mesmo, jogar ou ver os highscores. O ecrã inclui as 3 opções e é criada uma variável global da struct “MainMenuState” que contém:

- ❏ done: Uma flag que indica se o state está completo ou não
- ❏ playButton: Struct com a informação do botão de jogo (leva ao game)
- ❏ highscoreButton: Struct com a informação do botão dos highscores (leva ao HighscoreState)
- ❏ exitButton: Struct com a informação do botão de saída
- ❏ next: indica o estado que sucederá o menu (1-game 2-highscore 3-exit)
- ❏ background: Bitmap do fundo do estado

Peso no projeto: 3% Aluno responsável: Tiago Castro (100%)

3.10) Mouse

Este módulo é responsável pela implementação do mouse, tendo sido implementadas algumas funcionalidades dos labs. Tudo relacionado com o mouse está contido neste módulo e até algumas funcionalidades do jogo. Contém uma variável global mouse da struct mouse que contém:

- ❏ x: posição no eixo do x do cursor
- ❏ y: posição no eixo do y do cursor
- ❏ xSign: sentido do movimento no eixo do x
- ❏ deltaX: movimento a incrementar ao mouse no eixo do x na proxima atualização da VRAM (conjunção dos movimentos obtidos pelos interrupts do mouse anteriormente)
- ❏ deltaY: movimento a incrementar ao mouse no eixo do y na proxima atualização da VRAM (conjunção dos movimentos obtidos pelos interrupts do mouse anteriormente)
- ❏ speedMultiplier: componente não implementada que consistia em o user poder alterar a velocidade do cusor
- ❏ leftButton: flag que indica se o botao esquerdo esta a ser pressionado
- ❏ leftButtonReleased: flag que indica se o botao esquerdo foi solto
- ❏ middleButton: flag que indica se o botao do meio esta a ser pressionado
- ❏ rightButton: flag que indica se o botao da direita esta a ser pressionado
- ❏ bmp: bitmap do mouse

Peso no projeto: 8% Aluno responsável: Tiago Castro(70%)

3. Organização do código

3.11) NewHighscoreState

Este módulo é responsável pelo ecrã pós-jogo quando é obtido um novo highscore e onde está presente o score obtido pelo jogador no seu jogo. Neste estado o user da input do nome que pretende a que o highscore fique associado e pressiona “ENTER” quanto pretender seguir em frente. Haverá uma variável global da struct “NewHighscoreState” contendo:

- ❏ background: Bitmap do fundo do estado

Peso no projeto: 10% Aluno responsável: Tiago Castro (100%)

3.12) RTC

Este módulo é responsável pela implementação do RTC (subscrição e handle), contendo também um handler em assembly e a struct Date constituída por:

- ❏ year: ano
- ❏ month: mês
- ❏ day: dia
- ❏ hour: hora
- ❏ minute: minuto
- ❏ second: segundo
- ❏

Peso no projeto: 7% Aluno responsável: Tiago Castro (100%)

3.13)Spaceship

Este módulo possui uma struct “spaceship” e outra “bullet” e está responsável pelas balas e pela nave no modo de jogo. A struct Spaceship tem:

- ❏ map: Um apontador para o seu Bitmap
- ❏ xpos/ypos: A sua posição nos eixos do x e y
- ❏ xspeed/yspeed: As suas velocidades em ambos os eixos

As funções relativas à criação/destruição/movimento/rotação da nave encontram se neste módulo.

A struct Bullet possui:

- ❏ map: Um apontador para o seu Bitmap
- ❏ xpos/ypos: A sua posição nos eixos do x e y
- ❏ xspeed/yspeed: As suas velocidades em ambos os eixos
- ❏ indx: Que corresponde à sua localização no array de balas

Aqui encontram-se as funções relativas à criação/destruição/atualização/disparo de balas.

Peso no projeto: 10% Aluno responsável: António Cruz(80%)

3. Organização do código

3.14) Timer

Este módulo foi importado do lab 2 e o seu principal uso é atualizar as imagens representadas no ecrã juntamente com o score e dificuldade do jogo

Peso no projeto: 4 % Aluno responsável: Antonio Cruz (70%)

3.15) VBE

Este módulo foi também importado das aulas dos labs e apenas é utilizada a função vbe_get_mode_info()

Peso no projeto 1% Aluno responsável: Tiago Castro(100%)

3.16) Video_gr

Este módulo foi importado das aulas passadas, porém foram feitas alterações a fim de corresponder com as funções do Bitmap e também para implementar a técnica de double buffering.

Peso no projeto: 4% Aluno responsável: Antonio Cruz(80%)

3. Organização do código

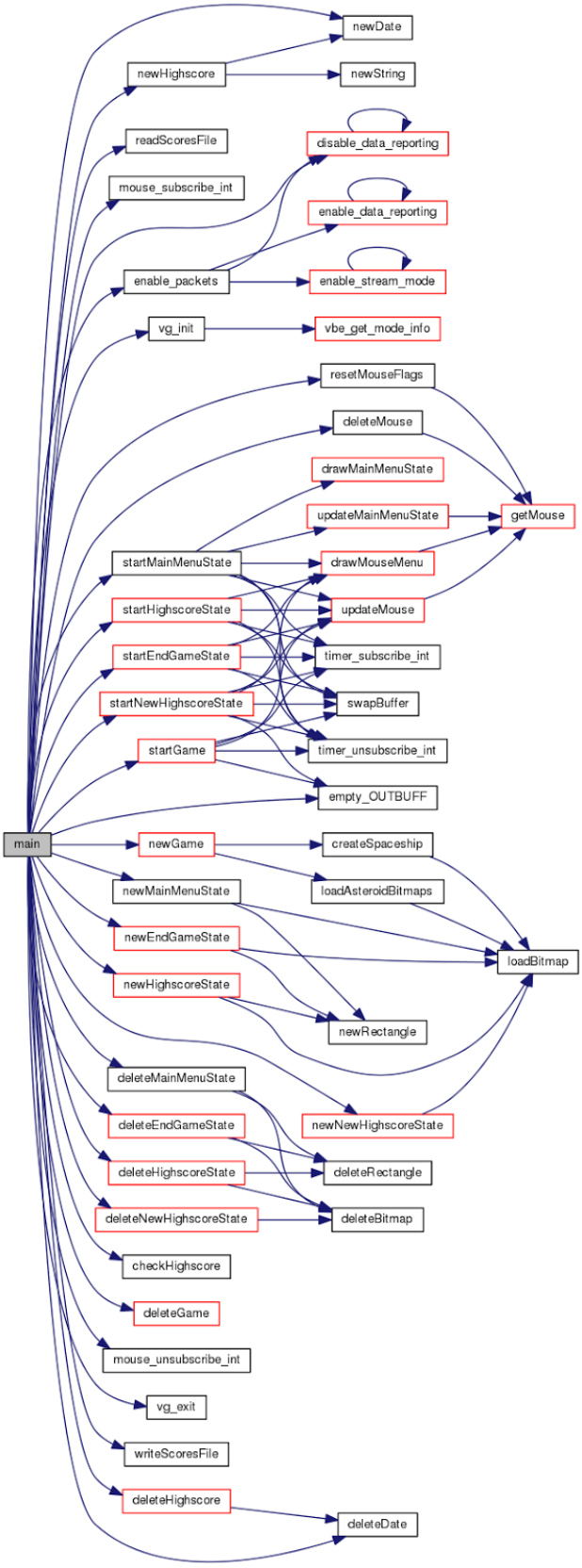

Diagrama de Chamada de Funções

4. Detalhes da Implementação

O programa foi desenvolvido recorrendo a uma máquina de estados que possui 6 estados diferentes, sendo que o jogo apenas se pode encontrar num de cada vez (Exit, Menu, Game,Highscore,New_Highscore,End_Game).

Quando o programa é corrido este encontra se no estado Menu, onde o utilizador navega por este com o rato. Se for pressionado o botão do lado esquerdo do rato em cima do botão Play o estado mudará para Game e começará o jogo.

Dentro do estado jogo, irão nascer asteroides nos limiares do ecrã de forma aleatória e a sua frequencia irá aumentar com o passar do tempo. O utilizador poderá controlar a nave através do teclado e do rato. Uma implementação que se demonstrou bastante trabalhosa e que significou aprendizagem autónoma foi a forma de fazer com que as balas fossem na direção do ponteiro do rato, e que o bitmap da nave fosse alterando consoante a posição do rato também.
A implementação das balas e dos asteroides foi feita de forma bastante semelhante, recorrendo a uma struct cujo núcleo contém um xpos/ypos e xspeed/yspeed, algo que foi adaptado do que aprendemos no lab 5 com o movimento das sprites.
Na verdade, grande parte do projeto foi feito recorrendo a structs que tornou o seu desenvolvimento muito mais fácil e intuitivo.
Após ter sido detectada uma colisão da nave com um asteroide, dentro do nosso ciclo inicial será verificado se o score obtido foi bom o suficiente para ir para a tabela de Highscores, se sim o novo estado será New_Highscore, caso contrário será End_Game.

Uma vez no New_Highscore, o user dá input do seu nome (foi utilizado um pointer para caracteres), podendo visualizar o seu input a medida que este o faz, foram utilizados bmps de cada letra de forma a obter este resultado, voltando para o menu quando o user clicar no “ENTER”. A string começa inicialmente toda a zeros (com 10 zeros, tendo sido alocados 11 espaços de memoria para conter também o caracter nulo) e é modificada a medida que o keyboard recebe interrupts, sendo apenas contabilizados interrupts que contenham uma letra do abecedário, do back_space (para remover o último caracter) e do enter (para terminar o input).

No caso de o resultado obtido não ser um highscore, o user é encaminhado para o End_Game onde apenas é mostrado o seu resultado e podendo voltar ao menu clicando com o botão esquerdo do mouse no botão de saída.

Se no menu tiver sido pressionado o botão do highscore o programa passará para o Higscore onde serão dispostos os 3 highscores que estão dispostos por ordem. Mais uma vez, o utilizador poderá voltar ao menu pressionando o botão de saída com o botão esquerdo do rato.

Se no menu tiver pressionado o exit, o programa passará para o estado Exit e será terminado o programa.

5. Conclusões

LCOM revelou-se uma experiência gratificante apesar de a nosso ver é sem dúvida a cadeira que requer dedicar mais trabalho e tempo o que não se acaba por refletir nos créditos obtidos.

Inicialmente acaba por ser um choque uma vez que a forma como a informação nos é transmitida acaba por ser difícil de absorver, os handouts de facto possuem toda a informação necessária, e agora olhando para eles fazem realmente todo o sentido, porém lembro me da enorme dificuldade que tivemos a entender inicialmente. A nosso ver sinto que no início de cada aula prática deveria haver 30 min onde o professor das práticas fala de cada periférico, já que o ambiente das aulas práticas é diferente do das aulas teóricas e podemos começar a ler os handouts mal recebemos o conhecimento.

Um aspeto negativo que se tornou relativamente recorrente foram os pequenos erros / desatualizações encontradas nos handouts, o que se refletiam em alguma confusão durante o estudo.

Gostamos bastante do lab5 relativo à placa gráfica uma vez que a partir deste é que começamos a ter uma ideia de como organizar e executar o projeto. Algo positivo também a realçar foi a forma como esta cadeira nos permitiu aprender a programar em C e a melhorar a forma como estruturamos um programa de início ao fim.

6. Apêndice

Instruções de instalação:

- ❏ Download da pasta Asteroids para /home/lcom
- ❏ Comando cd Asteroids
- ❏ Comando “su”
- ❏ Comando “sh install.sh”
- ❏ Comando “sh compile.sh”
- ❏ Comando “sh run.sh”

Estes comandos devem-se aos recursos do jogo (imagens bmp).