Faculdade de Engenharia da Universidade do Porto

LCOM T2G03 - “Super Minix”

Relatório

Mestrado Integrado em Engenharia Informática e Computação

Turma 2 Grupo 3:
Pedro Franco – up201604828
Rui Araújo– up201403263

2 de janeiro de 2018

Indice

Introdução

O projeto, intitulado de "Super Minix" foi realizado, no âmbito da cadeira Laboratório de Computadores, é uma versão do jogo “Super Mário”.

O jogo em si é simples. O jogador tem apenas que com o teclado fazer com que o “Mario” se desloque até ao final do mapa obtendo assim a pontuação máxima, para isso terá de ultrapassar obstáculos (o jogo termina assim que a personagem embata num obstáculo), com o progresso do jogo a velocidade dos obstáculos aumenta, o score final depende de quantos obstáculos ultrapassou.

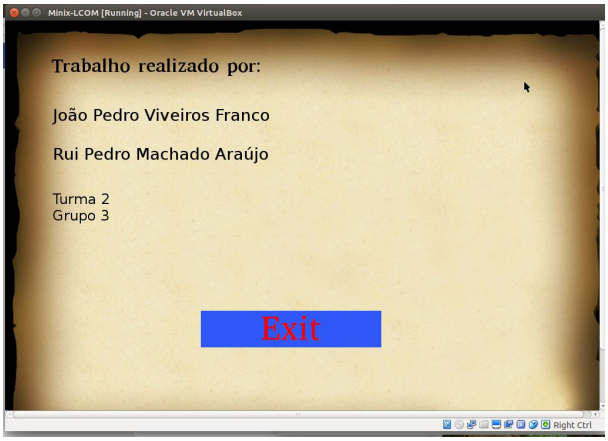
Este projeto permitiu desenvolver conhecimentos adquiridos durante as aulas nomeadamente ao nível de programação de baixo nível, com uso de periféricos.

Instruções de Utilização

Quando se inicia o programa este mostra o menu principal:

A opção “High Scores” leva a uma janela com os melhores resultados alcançados pelos jogadores:

“About” revela a identificação do grupo:



Se for seleccionada a opção de “Start Game” o jogo é iniciado e o ecrã apresentado é o seguinte:

O jogador terá agora de usar o teclado para mover a personagem de maneira a que esta ultrapasse os obstáculos (tubos verdes).

O número no canto superior esquerdo: mostra o score do jogador (quantos obstáculos foram ultrapassados).

Quando o jogo termina é mostrado o seguinte ecrã:

O jogador deve inserir o seu nome recorrendo ao teclado. Com as setas, devem ser seleccionadas as letras pretendidas e prime-se ENTER para passar para a próxima. Depois de inserido o nome é mostrada a tabela de maiores scores (da mesma forma mostrada em cima).

Periféricos Utilizados

Como foi referido na especificação do projeto, utilizamos os seguintes periféricos:

Periférico	**Utilização**	**Interrupções**
Timer	Atualização do jogo	Sim
Teclado	Interação com o Menu de jogo	Sim
Placa Gráfica	Visualização do ecrã de jogo	Não
Rato	Interação com o menu principal	Sim
RTC	Data e hora	Não

Os dispositivos com interrupções foram implementados recorrendo a um ciclo que recebe as interrupções de cada dispositivo e chama os respetivos handlers. A verificação das interrupções é feita no ficheiro main.c.

Timer

Foram utilizadas as interrupções do timer 0 para atualizar o jogo. A estrutura do timer guarda a informação de quantas interrupções teve este dispositivo durante a
execução do programa e se existiu ou não uma interrupção do timer.

É usado na estrutura principal do togo (no ficheiro Menu.c) que contem um timer, permite assim contar o tempo decorrido.

Teclado

As interrupções do teclado são usadas para ler as teclas que o utilizador prime. Isto é usado para fazer o movimento da personagem (em GameState.c, através da função updateGameStateKbd()) e a inserção do nome do jogador no final do jogo.

Rato

As interrupções do rato foram usadas para ler os packets do rato, para, com estes, actualizar a posição e estado do cursor (posição do rato e botão do lado esquerdo.

Usado para escolher opções de menu de navegação através de click com o botão esquerdo no local desejado. Logo usa botões e posição.

No ficheiro Menu.h, a estrutura principal do jogo contem um rato que é desenhado usando a função drawMouse() presente no ficheiro mouse.c e atualizado em updateMouse().

Placa Gráfica

Utilizou-se a placa gráfica para manipular o que era apresentado no ecrã em modo gráfico, escrevendo para a memória de vídeo.

O modo escolhido foi 0x117 que tem dimensões de 1024x768, com modo de cores é 5:6:5 que tem 2^{16} cores.

É utilizado Double buffer tanto no menu inicial como no jogo, para desenho do rato.
(Inspirado no blog: http://difusal.blogspot.pt/2014/08/minixtutorial-3-setting-up-project-with.html).

Existe animação de objetos (com o movimento do Super Mario) mas também dos obstáculos que se deslocam na sua direção. Foi também desenvolvido um teste de colisões entre estes (explicação em Detalhes de implementação).

A função mais importante é drawMenu() que é responsavel por desenhar no ecrã (dependendo depois do menu onde se encontra).

RTC

O RTC é usado para obter a data actual (dia,mes,ano) quando é guardado um highscore. O código referente a isto está situado no ficheiro RTC.c.

Os scores mais altos são depois guardados num ficheiro chamado “./high_score” com a sua respetiva data.

Estrutura de código

Módulos

Bitmap.c

Retirado de http://difusal.blogspot.pt/2014/07/minix-posts-index.html . A fonte-base deste código é (http://forums.fedoraforum.org/archive/index.php/t-171389.html). Serve para mostrar imagens do tipo ".bmp" no ecrã. Estas imagens foram usadas durante todo o jogo.

Adicionamos a função drawBitmapTransparent() em que a cor transparente é o branco. Permite que o branco seja ignorado aquando do desenho no ecrã.

Membro responsável: Pedro Franco /Rui Araujo
Peso relativo: 5%

GameState.c

É o estado de jogo quando se escolhe "Play Game". Neste módulo estão implementadas as funções que controlam o jogo propriamente dito e o atualiza. A estrutura deste estado inclui os bitmaps de fundo do ecrã, o score, o número de tubos gerados pelo jogo e a personagem.

Membro responsável: Pedro Franco / Rui Araujo
Peso relativo:25 %

Kbd.c

Módulo do teclado. Contém as funções de "subscribe "unsubscribe"do teclado, bem como as funções que permitem ler o KBC.

Membro Responsável: Rui Araujo

Peso relativo: 5%

Mario.c

Modulo da personagem principal. Estão implementadas as funções que criam, atualizam, desenham e apagam o “super mario”. A estrutura é composta pela sua posição inicial (x e y), o seu tamanho, o seu bitmap, o seu limite inferior entre outros.

Quando o jogador salta ou se move as coordenadas são atualizadas.

Membro responsável: Pedro Franco / Rui Araujo
Peso relativo: 10%

Menu.c

Este é o módulo base do jogo. É onde se encontram os handlers para as interrupções, onde os dispositivos são subscritos e onde se atualiza e desenha o jogo de acordo com o estado atual.

A estrutura é composta pelas imagens de fundo, todos os menus, pelo rato, timer, score bem como as variáveis dos periféricos.

Membro responsável: Pedro Franco
Peso relativo: 25 %

Mouse.c

Este módulo contém as funções que inicializam, atualizam, desenham e apagam o rato, bem como as funções para escrever para o rato, ativar o stream mode e fazer subscribe/unsubscribe.

A estrutura do rato é constituída pela sua posição, flags dos botões direito e esquerdo estão ou não pressionados,um array com o pacote recebido, bem como se é preciso desenhar o rato.

Membro responsável: Pedro Franco/ Rui Araujo
Peso relativo:5%

Pipe.c

Módulo que cria, atualiza, desenha e destrói a estrutura tubos/obstáculos. A estrutura contem as suas coordenadas e uma flag que indica se foi ou não ultrapassado.

Membro responsável: Pedro Franco / Rui Araujo
Peso relativo: 5%

RTC.c

Neste módulo estão implementadas as funções que retiram informação do RTC, mais concretamente, a data atual.

Membro responsável: Pedro Franco
Peso relativo: 5 %

Vbe.c

Neste módulo está implementada a função que vai buscar a informação sobre o modo de vídeo atual.

Membro responsável: Pedro Franco
Peso relativo: 5%

Video_gr.c

As funções para entrar e sair do modo gráfico estão neste módulo, bem como as funções para “pintar” o ecrã de uma determinada cor e desenhar quadrados ou linhas no ecrã.

Membro responsável: Rui Araujo

Peso relativo: 5%

Score.c

Módulo que cria a estrutura score no fim do jogo. É a estrutura que guarda o nome do jogador, o seu score e a data do jogo.

Membro Responsavel: Pedro Franco
Peso relativo:5%

Timer.c

Aqui implementaram-se as funções que criam, atualizam e apagam a estrutura do Timer, bem como as funções subscribe/unsubscribe. Esta estrutura é composta por uma flag, que indica se existiu uma interrupção do timer e um atributo que contém a informação de quantas interrupções do timer já existiram.

Membro responsável: Rui Araujo
Peso relativo: 5%

Detalhes de Implementação

O trabalho está organizado por camadas, desde a implementação base de periféricos, à camada superior que usa esses periféricos e a camada superficial que utiliza as funções que usam os periféricos.

Ainda quanto à organização de código, implementámos as nossas estruturas no estilo de object oriented C, utilizando tipos de dados abstratos. Cada estado e cada elemento do jogo tem a sua própria “classe” definida.

Quanto a implementação de máquinas de estado e código “event-driven” , usámos uma flag na estrutura Menu (que tem um valor diferente dependendo do estado de jogo/menu em que o jogo se encontra), que depois funciona como uma máquina de estados já que o que é mostrado no ecrã depende desse valor (menu).

O RTC foi implementado sem recurso às interrupções. Sempre que o jogo termina recorre-se a polling para ir buscar a data atual.

A animação dos objetos é obtida através de um update (em que são alteradas as suas posições, aquando de um número definido de interrupções). De notar que com o decorrer do jogo a velocidade dos obstáculos aumenta.

Para verificar colisões entre a personagem e os obstaculos é usada uma função checkCollision() (em GameState.c) que testa se algum dos 4 cantos do “super mario” se encontra dentro de algum Pipe.

Conclusões

Permitiu desenvolver conhecimentos de programação em C, de baixo nivel, melhorar o conhecimento sobre periféricos e o seu uso, bem como aprender sobre o sistema operativo linux.

Pelo lado negativo, os labs são por vezes demasiados longos, com funções menos importantes, para a fase do projeto. Torna essa fase da unidade curricular mais difícil cansativa.

Instalação

Baseamo-nos nos scripts disponibilizados pelo Henrique Ferrolho (em http://difusal.blogspot.pt/2014/08/minixtutorial-3-setting-up-project-with.html) e alterámos de modo a ser compatível com o nosso programa.

Para correr o programa deve, dentro do directório projeto, executar os seguintes comandos, em modo root), por esta ordem:

sh install.sh

sh compile.sh

```
sh run.sh
```