**Faculdade de Engenharia da Universidade do Porto**



**Waffle Lib**

**Projeto Final LCOM - 2018/19 - MIEIC**

**Turma 2MIEIC05 Grupo 1**

**Professor Práticas:** Luís Paulo Gonçalves dos Reis

**Autores**

José Silva, up201705591 ([up201705591@fe.up.pt](mailto:up201705591@fe.up.pt))

Mário Mesquita, up201705723 ([up201705723@fe.up.pt](mailto:up201705723@fe.up.pt))

Porto, 7 de janeiro de 2019

**Índice**

[**Instruções de utilização** 3](#_Toc533805968)

[**Estado do projeto** 4](#_Toc533805969)

[**Organização / Estrutura do código** 5](#_Toc533805970)

[**Detalhes de implementação** 5](#_Toc533805971)

[**Conclusões** 5](#_Toc533805972)

[**Instruções de instalação** 6](#_Toc533805973)

# **Instruções de utilização**

Inicialmente o programa apresenta um ambiente de trabalho vazio onde é possível visualizar a imagem de fundo selecionada.

MOSTRAR SCREENSHOT

No canto superior esquerdo existe um menu que apresenta várias opções ao clicar, cada uma com a sua funcionalidade. FLAAR MAIS SOBRE ISTO E SOBRE OS BUTOES

IMAGEM DO MENU COM BOTOES

As janelas abertas pelo programa podem ser movidas pelo rato, clicando nelas e arrastando. Permitem ser minimizadas, maximizadas e fechadas. Cada janela possui a respetiva referência na taskbar localizada em cima, podendo ser abertas e fechadas clicando aí. É possível também encadear janelas em cima das outras, respeitando a ordem por que foram sobrepostas. De modo a trazer uma janela para primeiro plano, basta clicar nela ou na sua referência e ela entrará em foco.

MOSTRAR JANELAS E FUNCIONALIDADES

No canto superior direito está presente um relógio configurado para mostrar o tempo atual no formato dd:mm:yyyy.

MOSTRAR RELOGIO

Ao deixar o programa inativo durante XXXXXXX segundos, isto é, sem inputs do teclado e rato pelo utilizador, o ecrã é alterado para apresentar um *screensaver*. Este consiste em imagens de waffles, bacon e sumo de laranja em movimento, a colidir entre si e com as margens do ecrã. Com qualquer interação o ecrã volta ao normal.

SHOW THE SCREENSAVER

# **Estado do projeto**

A informação sobre a nossa utilização de periféricos encontra-se resumida na tabela seguinte. As colunas representam o nome do periférico, as funcionalidades para que foi utilizado e se funciona com base em interrupções ou não.

|  |  |  |
| --- | --- | --- |
| Periférico | Utilização | Interrupções |
| Timer | - Controlo da *frame rate*  - Controlo do tempo de inatividade | Sim |
| KBD | - Atalhos para funcionalidades  - Escrita de texto  - Verificação da (in)atividade | Sim |
| Mouse | - Movimento do apontador do utilizador  - Interação com botões  - Controlo das janelas  - Verificação da (in)atividade | Sim |
| Video Card | - Apresentar no ecrã o ambiente de trabalho e todas as suas componentes  - Apresentar o *screensaver* e objetos em movimento | Não |
| RTC | - Obtenção do tempo atual  - Alarmes | Sim |
| UART | - Interação de um utilizador com um ambiente de trabalho remoto | Sim |

TIMER

Utilizamos o timer 0 para controlar a *frame rate* do programa, que definimos ser 30 frames por segundo. Opera internamente a uma frequência de 60 hertz.

É também usado para medir o tempo de inatividade do utilizador, de modo a decidir se deve ser desenhado o ambiente de trabalho ou o *screensaver*.

Foram implementadas funções para gerir a subscrição de interrupções, **timer\_subscribe\_int()** e **timer\_unsubscribe\_int()**. Além disso, destaca-se a função **timer\_int\_handler(),** que deve ser chamada a cada interrupção do timer. Esta última incrementa um contador que regista quantas interrupções passaram, baseado na sua frequência interna.

KBD

O teclado é utilizado para … FALAR DE ATALHOS E ESCREVER TEXTO

Em adição, o teclado, juntamente com o rato, controla a (in)atividade do utilizador, de modo a permitir decidir se deve ser desenhado o ambiente de trabalho ou o *screensaver*. Com qualquer interação, o tempo de inatividade é reduzido a 0.

São utilizadas funções para subscrição de interrupções (**keyboard\_subscribe\_int()** e **keyboard\_unsubscribe\_int())**, para gerir interrupções (**keyboard\_ih()**) e para processar scancodes (**opcode\_available()**).

Mouse

O rato é o meio principal pelo qual o utilizador interage com o programa, através de um cursor que pode ser movimentado no ecrã. Este permite clicar e interagir com os diversos botões e elementos do programa, seja para abrir menus, selecionar opções, arrastar / focar / abrir / fechar / minimizar / maximizar janelas.

Além disso, juntamente com o teclado, controla a (in)atividade do utilizador, de modo a permitir decidir se deve ser desenhado o ambiente de trabalho ou o *screensaver*. Com qualquer interação, o tempo de inatividade é reduzido a 0.

Relativamente ao mouse foram implementadas funções relativas à subscrição de interrupções (**mouse\_subscribe\_int()**  e **mouse\_unsubscribe\_int()**), para gerir cada interrupção (**mouse\_ih()**) e para processar os pacotes do rato (**assemble\_mouse\_packet()** e **parse\_mouse\_packet()**).

Video Card

O nosso programa opera no modo **0x14C**, com uma resolução de **1152x864** e **32 bits** por pixel em **RGB (8)-8-8-8** (8 bits para cada componente azul, verde e vermelha, e 8 bits reservados).

Utilizamos a técnica de **double buffering** para desenhar no ecrã, sendo … ADSADASDAS FALAR MAIS DISTO

As imagens e objetos no programa são **xpm**s, fazendo uso da função load\_xpm() fornecida pelos professores da cadeira para os converter em pixmaps.

Recorremos a um xpm com todos os símbolos da tabela ASCII desde ‘!’ até ‘~’ para representar a nossa **fonte**, tendo este sido preparado por nós. Ao longo do nosso programa utilizamo-la para apresentar informação em diversos locais.

No ambiente de trabalho são verificadas várias **colisões**, seja entre o rato e as janelas / botões ou entre as janelas e as margens do ecrã. No entanto, destaca-se o *screensaver* que entra em ação quando o utilizador permanece inativo durante XXXXXXX segundos. Neste surgem objetos de waffles, bacon e sumo de laranja em movimento e a colidir, tendo sido implementada **colisão pixel a pixel.**

RTC

O RTC é utilizado para obter o tempo atual e apresentá-lo na nossa taskbar, no canto superior direito, sob a forma de um relógio. A informação encontra-se no formato dd:mm:yyyy.

São utilizadas as interrupções de *update*, de modo a atualizar o relógio sempre que o tempo muda (a cada segundo).

Foram implementadas funções para gerir as subscrição de interrupções (**rtc\_subscribe\_int()**  e **rtc\_unsubscribe\_int()**), para ativar as interrupções necessárias (neste caso apenas as de *update*, **rtc\_enable\_update\_int()**) e para gerir cada interrupção (**rtc\_int\_handler()**). Esta última verifica se a fonte da interrupção foi um *update* dos registos e atualiza o relógio em caso afirmativo.

UART

O dasdasd sadsadas das

# **Organização / Estrutura do código**

Ao longo dos laboratórios durante o semestre e agora durante a realização do projeto, fomos organizando o código em módulos para torná-lo mais fácil de ler e interpretar. Uma das decisões que tomamos foi organizá-lo em pastas, agrupando módulos relacionados.

**Periféricos**

Os módulos descritos sob esta secção dizem respeito a periféricos e, à exceção do *rtc* e do *serial port*, foram desenvolvidos durante o decorrer dos Laboratórios de LCOM. Para o projeto apenas foram adicionadas algumas funcionalidades necessárias e / ou refatorizado o código.

* **Timer**

**Módulo ‘timer’**

Este módulo contém código relativo à manipulação dos timers (que no nosso caso apenas usamos o timer 0), com funções para subscrição de interrupções, handlers e métodos para alterar o funcionamento interno. Foi desenvolvido durante o Laboratório 2 e ambos os membros contribuíram igualmente na sua criação.

**Módulo ‘i8254’**

Este módulo contém as constantes e macros definidas para a programação do timer. Foi nos fornecido durante o Laboratório 2, e, portanto, não foi desenvolvido por nós.

**Módulo ‘timer\_user’**

Este módulo contém algumas constantes e macros adicionais que consideramos ser úteis para a programação do timer, tendo sido desenvolvido durante o Laboratório 2. Ambos os membros contribuíram igualmente na sua criação.

* **KBC**

**Módulo ‘kbc’**

Este módulo contém código relativo à manipulação do kbc, seja para gerir interrupções ou enviar / receber comandos. Foi desenvolvido durante o Laboratório 3 para o teclado e, mais tarde, refatorizado para permitir usá-lo também com o rato. Ambos os membros contribuíram igualmente na sua criação.

**Módulo ‘i8042’**

Este módulo contém as constantes e macros definidas para a programação do teclado e rato. Foi desenvolvido durante os Laboratórios 3 e 4 e ambos os membros contribuíram igualmente na sua criação.

**Módulo ‘keyboard’**

Este módulo contém código relativo à manipulação do teclado, seja para subscrever e gerir interrupções ou processar *scancodes*. Foi desenvolvido durante o Laboratório 3 e ambos os membros contribuíram igualmente na sua criação.

Imagens e gráficos pipis

**Módulo ‘mouse’**

Este módulo contém código relativo à manipulação do rato, seja para subscrever e gerir interrupções ou processar *packets*. Foi desenvolvido durante o Laboratório 4 e ambos os membros contribuíram igualmente na sua criação.

Imagens e gráficos pipis

* **Video Card**

**Módulo ‘vbe’**

Este módulo contém código relativo à manipulação da VBE, com funções para inicializar o modo de vídeo, alterar o modo de vídeo, desenhar no ecrã e obter / alterar configurações. Foi desenvolvido durante o Laboratório 5 e ambos os membros contribuíram igualmente na sua criação.

* **RTC**

**Módulo ‘rtc’**

Este módulo contém código relativo à manipulação do RTC, com funções para gerir e processar interrupções, alterar o funcionamento do periférico e ler / escrever nos seus registos. Encontram-se também neste módulo todas as constantes e macros definidas para a sua programação. Foi desenvolvido maioritariamente pelo Mário Mesquita.

* **Serial Port**

**Módulo ‘serial\_port’**

dasdsadasdas

**Waffle Lib**

Os módulos descritos sob esta secção foram desenvolvidos para o projeto e, portanto, são específicos para este contexto.

* **Window**

**Módulo ‘window’**

Este módulo contém código relativo à manipulação dos timers (que no nosso caso apenas usamos o timer 0), com funções para subscrição de interrupções, handlers e métodos para alterar o funcionamento interno. Foi desenvolvido durante o Laboratório 2 e ambos os membros contribuíram igualmente na sua criação.

**Módulo ‘context\_menu’**

Este módulo contém as constantes e macros definidas para a programação do timer. Foi nos fornecido durante o Laboratório 2, e, portanto, não foi desenvolvido por nós.

**Módulo ‘wnd\_elements’**

Este módulo contém algumas constantes e macros adicionais que consideramos ser úteis para a programação do timer, tendo sido desenvolvido durante o Laboratório 2. Ambos os membros contribuíram igualmente na sua criação.

**Módulo ‘wnd\_state\_machine’**

Este módulo contém as constantes e macros definidas para a programação do timer. Foi nos fornecido durante o Laboratório 2, e, portanto, não foi desenvolvido por nós.

**Módulo ‘wnd\_tskbar’**

Este módulo contém algumas constantes e macros adicionais que consideramos ser úteis para a programação do timer, tendo sido desenvolvido durante o Laboratório 2. Ambos os membros contribuíram igualmente na sua criação.

* **Screensaver**

**Módulo ‘screensaver’**

Este módulo contém código relativo ao *screensaver*, seja para desenhar o fundo e elementos no ecrã, para detetar e resolver colisões entre elementos ou para determinar as novas posições dos elementos. Foi desenvolvido pelo Mário Mesquita.

Relativamente ao screensaver, estão também definidos os ficheiros waffle\_xpm.h, bacon\_xpm.h, orange\_juice\_xpm.h e screensaver\_background.h, que contêm os xpms utilizados para desenhar os elementos e fundo do screensaver.

* **Font**

**Módulo ‘letters’**

Este módulo contém código relativo à manipulação de texto pelo nosso programa. Dispõe de funções para inicializar a fonte utilizada, para imprimir no ecrã um símbolo ou para imprimir no ecrã palavras (horizontal ou vertical).

Relativamente à fonte, está também definido um ficheiro font.h que contem o xpm de toda a fonte utilizada.

**Módulo ‘messages’**

**Módulo ‘waffle’**

**Outros**

Os módulos descritos sob esta secção são tais que não se enquadram nas secções anteriores. Dizem respeito a funções utilitárias ou ao código necessário para correr o projeto.

**Módulo ‘util’**

Este módulo contém código relativo a funções utilitárias para facilitar ou simplificar a programação de outros módulos. Foi desenvolvido igualmente pelos dois membros.

**Módulo ‘proj’**

Este módulo contém o código necessário para correr o programa, sendo aqui onde se encontra a função **main()** e o **driver\_receive()** loop. Trata da configuração dos periféricos inicial, gestão das interrupções e configuração dos periféricos final. Foi desenvolvido igualmente pelos dois membros.

# **Detalhes de implementação**

Ao longo de todo o semestre, durante os Laboratórios e o projeto, esforçamo-nos por manter o **código organizado e comentado**, de modo a permitir que seja **facilmente interpretado e reutilizado** em outros contextos. Os ficheiros do nosso projeto encontram-se organizados em pastas, agrupando módulos relacionados. Esta abordagem permitiu que facilmente conseguíssemos integrar o código relativo aos periféricos, desenvolvido nos laboratórios, no nosso projeto, com poucas ou nenhumas alterações.

No mesmo tópico, definimos **macros e constantes simbólicas** para todos os valores que utilizamos no nosso código, de modo a que, em qualquer altura, consigamos identificá-los pelo o que representam. Deste modo, por exemplo, mesmo alguém não familiarizado com o KBC saberá que 0x64 permite aceder ao registo de controlo do dispositivo.

Destaca-se também a distribuição por **camadas de abstração**, separando as funcionalidades de baixo e alto nível que interagem entre si. Um exemplo desta separação é o módulo do KBC, um dispositivo que é usado tanto pelo teclado como pelo rato. Com efeito, os dois periféricos mencionados partilham o mesmo código, chamando as funções de baixo nível definidas para o KBC.

Pixel by pixel collisions

Serial port com HEADEr, TRAILER e ACKS

State machine

Double buffering

Assembly ?

Double linked list para windows

Mais cenas fixes

# **Conclusões**

- course evaluation

- what went wrong

- what went well

# **Instruções de instalação**

- MAYBE?

THIS IS OPTIONAL