Faculdade de Engenharia da Universidade do Porto

Ano letivo 2020/2021

HumanoId

Relatório de projeto

Laboratório de Computadores

Grupo T8G02:

Catarina O. Pires up201907925@fe.up.pt

José E. F. Costa up201907216@fe.up.pt

Índice

1. Instruções de utilização

1.1. Menu Inicial

Ao iniciar o programa, é apresentado o menu principal *(Fig.1)* que contém o logotipo do jogo e várias opções, tais como, jogar e sair do programa. Para além disso, dispõe as horas atuais do dia. A navegação de todos os menus do programa recorre ao uso do rato, em que, ao pressionar o botão esquerdo do mesmo, possibilita a escolha de opções.
Caso o cursor esteja em cima de uma opção, a caixa da opção muda de cor.

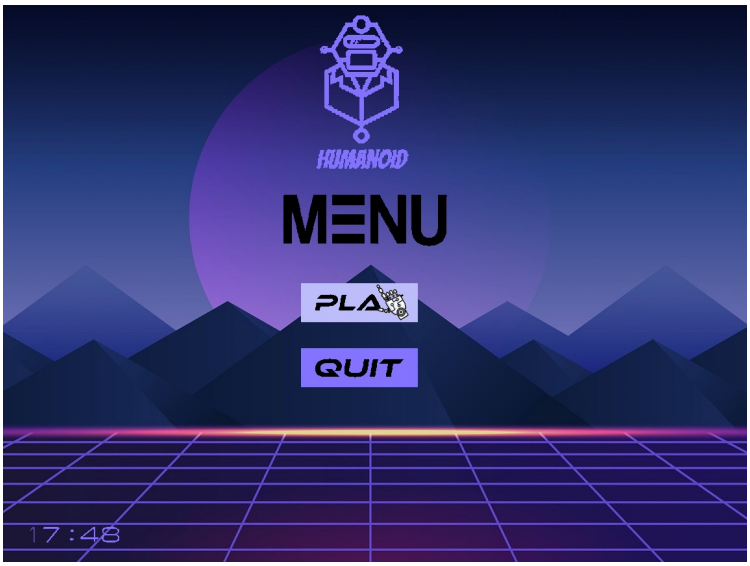


Fig. 1: Menu inicial.

1.2. Menu de escolha do jogo

Selecionando a opção *PLAY* do Menu Inicial, aparecerá um segundo menu no qual estão expostos os mini-jogos constituintes do programa *(Fig.2)*, jogo de reação, jogo de desenho, jogo de passagem do tempo e jogo das teclas. Neste menu, o utilizador poderá escolher qual pretende jogar com recurso ao rato, clicando em cima da imagem do mesmo. Possui também uma opção para voltar para o Menu Inicial (*BACK*).

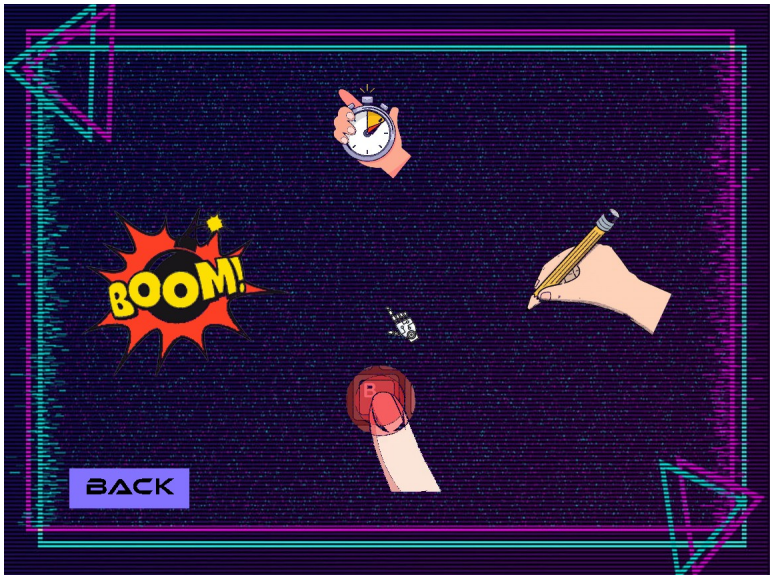


Fig. 2: Menu escolha do jogo.

1.3. Jogo de Reação

1.3.1. Menu Descrição do jogo

Se o escolhido for o Jogo de Reação, serão apresentadas as descrições e explicações sobre como o jogar e a que dispositivos recorrer *(Fig.3)*. Para iniciar o jogo, o utilizador terá de selecionar a opção *PLAY.* Possui também uma opção para voltar para o Menu escolha do jogo (*BACK*).

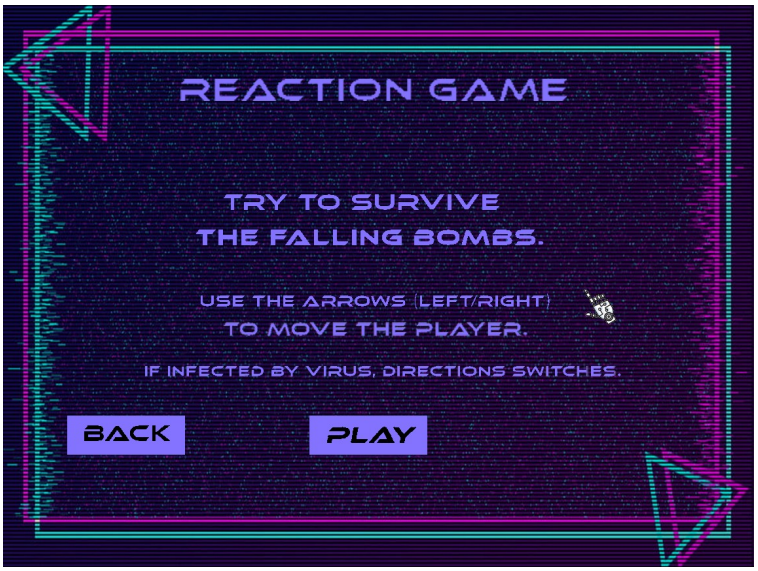


Fig. 3: Descrição do jogo de reação.

1.3.2.Jogo

Após a escolha do Jogo de Reação *(Fig.4)*, o utilizador terá de se desviar das bombas que estão a cair recorrendo às setas direita e esquerda do teclado. Aparecerão também objetos aos quais chamamos de Vírus, se o jogador entrar em contacto com os mesmo, ficará infetado, isto é, os seus sentidos ficarão alterados num período de 8 segundos. Se conseguir fugir às bombas durante um intervalo de tempo de 1 minuto (cronômetro apresentado no canto inferior esquerdo), irá concluir o jogo, saindo vitorioso.

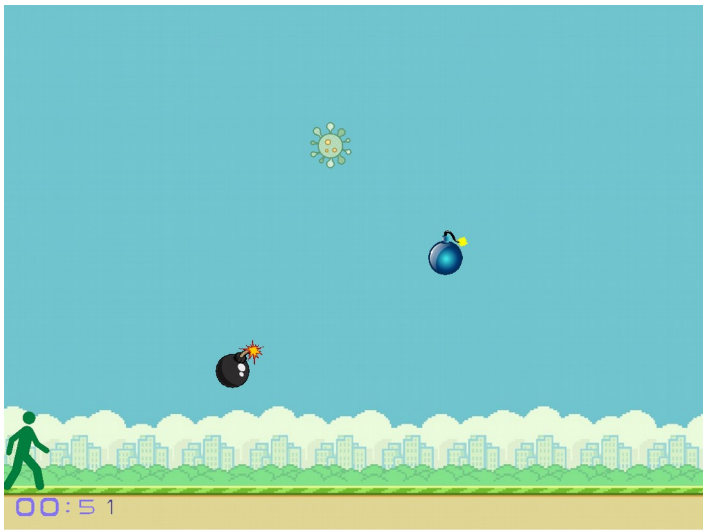


Fig. 4: Jogo de reação.

1.4. Jogo de desenho

1.4.1.Menu Descrição do jogo

Se o escolhido for o Jogo de Desenho, serão apresentadas as descrições e explicações sobre como o jogar e a que dispositivos recorrer *(Fig.5)*. Para iniciar o jogo, o utilizador terá de selecionar a opção *PLAY.* Possui também uma opção para voltar para o Menu escolha do jogo (*BACK*).

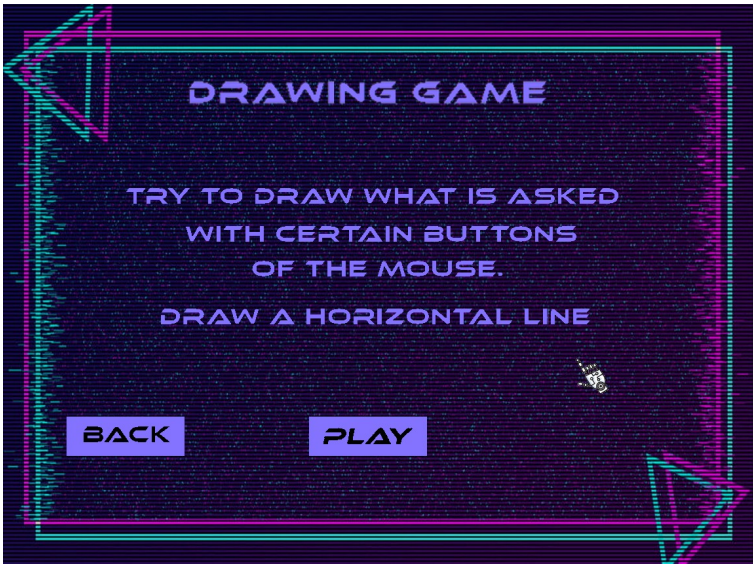


Fig. 5: Descrição do jogo de desenho.

1.4.2.Jogo

Após a escolha do Jogo de Desenho *(Fig.6)*, o utilizador terá de desenhar figuras/formas recorrendo ao rato e seus botões. Se conseguir desenhar uma linha horizontal, pressionando o botão esquerdo do rato e , com uma certa tolerância para y, passará no jogo. Para o jogador ter uma ideia do que está a desenhar, iremos desenhar o rasto.

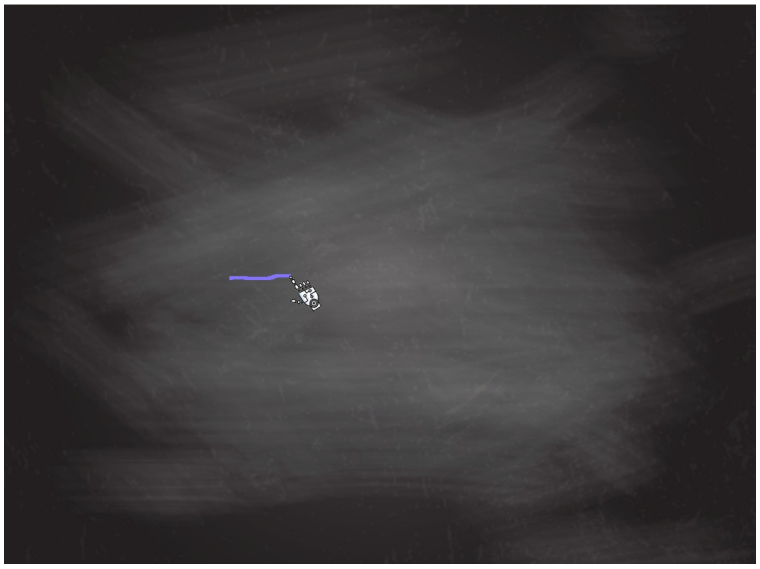


Fig. 6: Jogo de desenho.

1.5. Jogo da passagem do tempo

1.5.1.Menu Descrição do jogo

Se o escolhido for o Jogo do Tempo, serão apresentadas as descrições e explicações sobre como o jogar e a que dispositivos recorrer *(Fig.7)*. Para iniciar o jogo, o utilizador terá de selecionar a opção *PLAY.* Possui também uma opção para voltar para o Menu escolha do jogo (*BACK*).

Fig. 7: Descrição do jogo do tempo.

1.5.2.Jogo

Após a escolha do Jogo do Tempo *(Fig.8)*, o utilizador deverá mentalmente esperar um intervalo de tempo indicado pelo computador e carregar no botão do meio do rato aquando pensar ter passado esse intervalo.

Fig. 8: Jogo do tempo.

1.6. Jogo das teclas

1.6.1.Menu Descrição do jogo

Se o escolhido for o Jogo das Teclas, serão apresentadas as descrições e explicações sobre como o jogar e a que dispositivos recorrer *(Fig.9)*. Para iniciar o jogo, o utilizador terá de selecionar a opção *PLAY.* Possui também uma opção para voltar para o Menu escolha do jogo (*BACK*).

Fig. 9: Descrição do jogo das teclas.

1.6.2.Jogo

Após a escolha do Jogo do Tempo *(Fig.10)*, o utilizador deverá, com uma mão, tentar pressionar as combinações de
teclas apresentadas no ecrã.

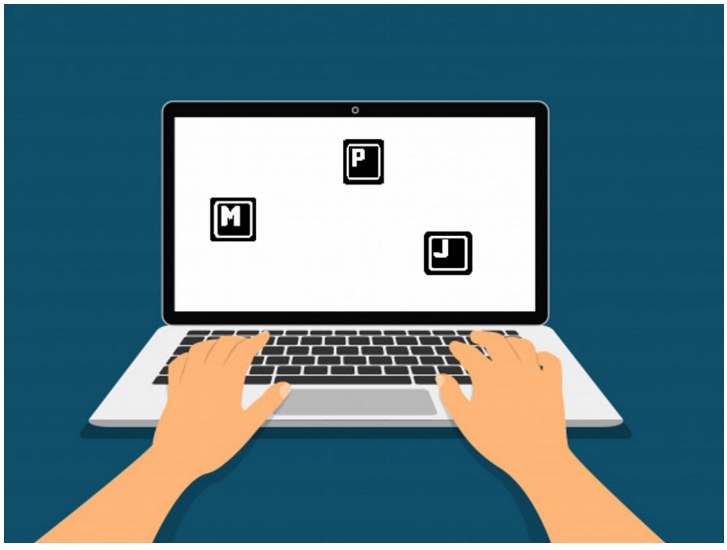


Fig. 10: Jogo das teclas.

1.7. Menu jogo perdido

Findando um jogo e não ter tido sucesso, leva o utilizador ao Menu do jogo perdido (*Fig. 11*). Este menu possui uma opção para voltar para o Menu Inicial do jogo (*BACK*). No caso específico de ter perdido o Jogo do Tempo, mostramos os segundos que o jogador realmente contou (*Fig. 12*), considerando apenas se tiverem 2 casas, caso contrário, não mostra a mensagem.

Fig. 11: Menu jogo perdido.

Fig. 12: Menu jogo perdido no caso do Jogo do Tempo.

1.8. Menu jogo ganho

Findando um jogo e saindo vitorioso, leva o utilizador ao Menu do jogo ganho (*Fig. 12*). Este menu possui uma opção para voltar para o Menu Inicial do jogo (*BACK*).

Fig. 12: Menu jogo ganho.

2. Estado do projeto

Dispositivo	Funcionalidade	Usa interrupções?
Timer	Calcula tempos de reação. Atualização de imagens e *Sprites*. Movimento de objetos e verificação de colisões.	✔
Teclado	Reconhecimento de teclas específicas para controlo de jogador em Jogo de Reação e deteção de teclas para Jogo das Teclas.	✔
Rato	Navegação no menu e reconhecimentos de formas e cliques no Jogo de Desenho.	✔
Placa gráfica	Permite visualização de imagens (menus, imagens de fundo, adereços de jogos, …).	Não se aplica
RTC	Alarmes com determinados intervalos de tempo para término de jogos. Apresentação do tempo atual no Menu Inicial.	✔

2.1. Timer

O timer é utilizado para:

- Controlo do desenho e alteração dos buffers da placa gráfica, bem como controlo da alteração de posição dos sprites
- Uso específico no jogo de contagem do tempo para a contagem do tempo.
- Utilizado no jogo de reação para desfazar o aparecimento dos diferentes sprites.

Implementação na pasta ‘timer’, no ficheiro i8254.c.

2.2. Teclado

O teclado é utilizado para controlo do programa no:

- Movimento do jogador no Jogo de Reação, reconhecimento das teclas ‘<-’ (esquerda) e ‘->’ (direita). (*check_movement_r_l* e *assemble_directions_r_l*)
- Reconhecimento das teclas ‘P’, ‘M’ e ‘J’, em simultâneo, no Jogo das Teclas. Detenção de teclas diferentes às pretendidas do jogo. (*assemble_keys*)

Implementação na pasta ‘*kbc*’, no ficheiro *keyboard.c*.

2.3. Rato

O rato é utilizado, com recurso ao seu deslocamento e botões, para:

- Interação com os menus. (*check_collision_options*)
- Uso no jogo de passagem do tempo, como modo para o jogador indicar o final do jogo (uso do botão do meio).
- Uso no jogo de desenho, para o reconhecimento de uma linha horizontal (uso do botão direito de de deslocamento do rato). (*gestureDetection_draw_process_state_H*)

Implementação na pasta '*kbc*', no ficheiro *mouse.c*, *gestureDetection*.c e *i8042*.c.

2.4. Placa Gráfica

Modo Gráfico	0x14C
Resolução	1152 x 864
Modo das cores	Modo Direto
***Bits per pixel* (R:G:B)**	32 ((8:)8:8:8)
Número de cores	$2^{32}-1=4294967295$

Para este dispositivo implementamos:

- *Page flipping* e *Triple Buffering*
  Foi implementado triple buffering (*video_map_vram_mem*) com recurso à função de *page flipping (video_flip_page)* da VBE (0x07), melhorando significativamente o desempenho dos gráficos, principalmente quando um elevado número de sprites, ou *sprites* de grande dimensão, estão "em cena".
- *VBE functions*
  As funções 0x01, 0x02 e 0x07 foram usadas, a 0x07 tal como foi atrás indicado, foi usada para o *page flipping*, a função 0x01 para obter informações sobre o modo gráfico usado (*video_get_mode_info*), neste caso o modo 0x14C, e a função 0x02 para a mudança entre o modo de texto (0x003) e modo 0x14C. (*video_change_mode*)

Implementação na pasta '*video*', no ficheiro *video.c*.

- Imagens e imagens animadas *(Sprites)*, e sua escrita no ecrã.
- Algoritmo de deteção de colisões entre *Sprites* com recurso a *arrays* para detetar apenas as que eram do nosso interesse, usado no Jogo de Reação. (*check_collisions_sprite*) Colisão detetada a partir das dimensões das imagens dadas por um retângulo limitador da mesma.

- Deteção de colisão entre ponto e retângulo usada para interpretar a interação do rato nos menus. Ponto da posição inicial do *Sprite* do cursor do rato e retângulos das opções. (*check_collision_options*)
- Para apresentarmos texto no ecrã, recorremos à apresentação de imagens com texto.

Implementação na pasta '*video*', no ficheiro *extImages.c.*

2.5. Real Time Clock (RTC)

O RTC, com uso a interrupções de alarme, é utilizado para:

- Determinar duração do Jogo de Reação, aquando da interrupção findando um intervalo de tempo, o jogo termina. Para esta funcionalidade, recorremos à leitura do tempo (hora, minutos e segundos) para gerar alarmes para um intervalo de tempo após o tempo atual. (*rtc_calculate_finish_alarm* e *rtc_config_alarm*)
- Leitura do tempo e sua apresentação no Menu Inicial (*Fig. 13*). (rtc_get_value e *display_time_menu*(em *extIMages.c*))

Implementação na pasta '*rtc*', constituída por *rtc.c* e *rtc.h*. Interrupções tratadas em proj.c no *loop* do *driver receive*.

Fig. 13: Apresentação do tempo atual no Menu Inicial.

3. Organização e estrutura do código

3.1. Timer

3.1.1.i8254.c

Módulo implementado de modo a configurar cada timer individualmente (apesar de apenas ser utilizado o *timer* 0 ao longo do projeto), com capacidade de mudar a frequência de cada timer.

Peso no projeto: 5%
Participação: Catarina: 50% José:50%

3.2. KBC

3.2.1.i8042.c

Módulo implementado para atender às informações provenientes das linha IRQ do KBC, como, o *interrupção handler*, onde lê as informações do *output register (kbc_ih)*. Relativamente ao rato, possui funçẽs que determinam se o byte vem deste dispositivo e se tem o *bit* 3 ativo. (*kbc_is_mouse_packet* e *kbc_is_first_mouse_packet)*

Peso no projeto: 5%
Participação: Catarina: 50% José:50%

3.2.2.keyboard.c

Módulo implementado com o intuito de receber os *bytes* provenientes do teclado e interpretá-los em ações no jogo, como as setas para o Jogo de Reação, em que codifica as teclas em direções do *Sprite* (*check_movement_r_l*) e reconhecimento de teclas variadas para o Jogo das Teclas (*assemble_keys*). Estas funções anteriormente referidas, utilizam um algoritmo pensado por nós que guarda num *array* os *makecodes* que são relevantes e, no caso do Jogo das Teclas, se esse *array* deixar de ter um valor a zero, significa que completou com sucesso, já no Jogo de Reação, interpreta os valores diferentes de zero em sentidos do movimento do jogador.
Este módulo contém também uma função que analisa se um *scancode* se trata de um *makecode* ou *breakcode* (*is_make_code*).

Peso no projeto: 10%
Participação: Catarina: 75% José:25%

3.2.3.mouse.c

Módulo capaz de, após receber os *bytes* lidos do KBC, previamente organizados num *mouse_packet_raw*, processar as informações lidas do rato, organizando-as num *mouse_packet_processed*, bem como configurá-lo para utilização no *minix* (comando *MOUSE_ENABLE_DATA_REP_STR*).

Peso no projeto: 10%
Participação: Catarina: 25% José: 75%

3.2.4.gestureDetection.c

Recorrendo a eventos, detetados por nós na função *gestureDetection_detect_event*, e máquinas de estados (*horizontal_state*), este módulo, base do jogo de desenho, reconhece quando uma linha horizontal é desenhada ao mesmo tempo que o botão esquerdo do rato é premido (*gestureDetection_draw_process_state_H*).

Peso no projeto: 5%
Participação: Catarina 75 % José:25%

3.3. Placa Gráfica

3.3.1.video.c

O módulo de gráficos foi implementado com um cuidado redobrado, de modo a assemelhar-se a uma implementação orientada a objetos, de modo a tornar as interações com a placa gráfica, normalmente muito penosas, do ponto de vista do programador, mais simples. Os dados da placa gráfica, como informações sobre o modo e os comandos associados foram então guardados numa *struct video_instance* sobre a qual operam funções capazes de controlar a resposta da placa gráfica. Numa primeira implementação, antes do conhecimento acerca da necessidade de que as funções criadas tivessem de ser obrigatoriamente usadas (requerimento do compilador), foram usados *function pointers*, tornado a experiência de programação orientada a objetos em C, aproximada àquela que existe em linguagens de mais alto nível como C++. Ao longo do processo de refinamento foram successivamente implementados vários números de *buffers* (1, 2 e 3), pelo que acabamos por acordar que 3 *buffers* ofereciam o melhor desempenho, sem taxar negativamente a máquina. Foram também aqui desenvolvidas

funções de desenho de *sprites* e objetos no ecrã, como é o caso da função *draw_pixel*, base de todo o desenho de gráficos, e a função *draw_rectangle*, usada para o desenho de botões.

Peso no projeto: 15%
Participação: Catarina: 20% José: 80%

3.3.2.extImages.c

Foram criados dois tipos de “imagens externas”. Por imagens externas entende-se que se originaram com recurso a ficheiros. Existem portanto imagens estáticas *Images* e imagens animadas *Sprite*. A principal diferença entre ambas é que um *Sprite* tem a si associada uma velocidade, que lhe permite mover-se no espaço do ecrã. De facto o tipo *Sprite* pode ser interpretado como um “subtipo” do tipo *Image*. Sobre estes dois tipos operam as funções de moviento e de desenho, necessárias ao correto funcionamento do programa.

Peso no projeto: 10%
Participação: Catarina: 70% José: 30%

3.4. Real Time Clock (RTC)

3.4.1.rtc.c

Este módulo foi implementado com o objetivo de apresentarmos o tempo atual e existência de alarme. Para concretizarmos os objetivos pretendidos, criamos funções de escrita (*rtc_write*) e leitura (*rtc_read*) para obtermos os valores de determinados campos (*rtc_get_value*) e tratarmos das configurações (*rtc_config_alarm*). Com a finalidade da existência de alarmes, criamos uma função que alterava o valor de uma *flag* em determinado registo, dependendo se a queria ativar ou desativar (*rtc_set_flag*) e uma função que tratava das interrupções vindas do rtc (*rtc_ih*), analisando a sua proveniência (*alarme*, *renovação* ou periódicas). Funções úteis como a conversão de BCD para decimal e vice-versa, encontram-se no módulo *utils.c*.

Peso no projeto: 5%
Participação: Catarina: 100% José: 0%

3.5. Comuns

3.5.1.counters.c

Esta biblioteca foi criada com o intuito de reunir alguns counters

necessários ao decorrer do programa. Foi, no entanto não extensivamente usada devido à dificuldade em ser usada. A sua implementação baseia-se em dois *arrays* de *pointers* para valores que guardam as contagens, um deles com as contagens ativas e outro com as contagens em pausa ou paradas, sendo possível reaver valores para qualquer um dos elementos dos arrays em qualquer momento.
Peso no projeto: 5%
Participação: Catarina: 0% José: 100%

3.5.2.interrupts.c

Aqui usaram-se *arrays* de *interrupt_info,* tipo que guarda a linha de IRQ e o *hook id* de cada interrupção, permitindo, mais facilmente subscrever e desativar interrupts para um dado dispositivo, e, no caso do encerramento do programa, desativar os interrupts de todos os dispositivos.
Peso no projeto: 5%
Participação: Catarina: 25% José: 75%

3.5.3.utils.c

Este módulo contém funções úteis e gerais para o desenvolvimento das funcionalidades do programa, tais como, a leitura (*util_sys_inb*) em registos, obtenção dos bits mais e menos significativos de um valor (*util_get_LSB* e *util_get_MSB*), obtenção do mínimo entre 2 valores (*utils_min*) e funções de conversão de BCD para decimal e vice-versa (*bcd_to_decimal*, *decimal_to_bcd*).

Peso no projeto: 5%
Participação: Catarina: 50% José: 50%

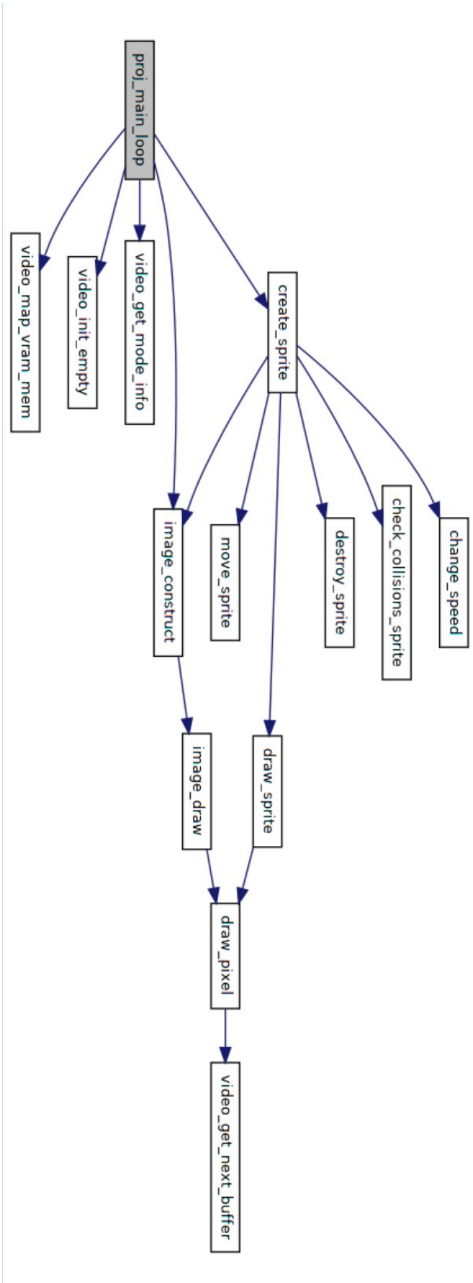
3.6. Proj.c

Este módulo é responsável pelo controlo, praticamente total, do jogo. Aqui é feita a configuração dos dispositivos e respetivos *interrupts* bem como é aqui que se encontra o loop do *driver_receive,* para o atendimento dos diversos interrupts e respetiva execução condicional de código, associado a cada módulo de controlo / jogo. Dada a natureza modular do nosso projeto existiu a necessidade de criar uma máquina de estados para o controlo da execução de código em função do módulo e interrupção recebida, pelo que se recorreu a um

enumerated type, *module*, e a um *array* de interrupt flags alteradas quer por cada módulo interessado no *interrupt*, caso esteja ativo, quer pelo *interrupt handler* do dispositivo. É neste contexto, após o atendimento de interrupções que se encontra, distribuído por cada módulo, a lógica e geração de frames de cada jogo.

Peso no projeto: 20%
Participação: Catarina: 60% José: 40%

4. Diagrama de funções



5. Detalhes da implementação

Neste projeto procuramos pôr em prática e mostrar o que aprendemos nesta unidade curricular ao longo do semestre. Desta forma, o nosso objetivo foi incluir todos os aspetos dados.

Aspetos como *event driven* e máquinas de estado, revelaram-se úteis mais especificamente para o módulo do rato e no Jogo de Desenho, onde era preciso identificar o evento atual do rato para, através da máquina de estados, reconhecer o estado em que se encontrava.

Revelou ser bastante útil o uso de *structs* e *function pointers*, no sentido de nos aproximarmos da orientação de objetos em C.

Para a geração de *frames*, usamos o dispositivo *Timer*, para a cada interrupção, desenhar os *Sprites* e *Imagens* nas posições pretendidas, bem como mudar a página em uso pela placa gáfica.

Algo que não abordamos nesta unidade curricular e que era imprescindível termos neste projeto era a deteção de colisões, tanto a nível de *sprites*, para o Jogo de Reação, como a nível de um ponto com um retângulo, para facilitar o reconhecimento da escolha de opções do utilizador nos menus.

Numa fase inicial do projeto, atendíamos apenas as interrupções que não era necessário utilizar, dependendo do módulo, o que causava problemas ao transitar para um módulo que onde só queríamos que dispositivos específicos diferentes funcionassem, caso notório principalmente no rato e teclado, dado partilharem o mesmo controlador1. Para corrigirmos este aspeto, passamos a atender todas as interrupções, para não deixarmos dispositivos bloqueados devido a interrupções não atendidas, mas analisando os dados de apenas aquelas que fazem sentido em cada um dos módulos.

6. Conclusão

Ambos somos da mesma opinião que a unidade curricular aborda assuntos interessantes e que, o projeto, revelou ser uma parte que, vendo o produto final, nos satisfaz muito.
Concordamos que, relativamente ao peso dado a esta unidade curricular no semestre, não corresponde ao trabalho investido nela uma vez que requer muito esforço e tempo para fazermos as atividades e trabalhos com sucesso.
Inicialmente, por ser algo totalmente diferente do que estávamos habituados, sentimo-nos um pouco perdidos mas, no decorrer do semestre, essa sensação foi desaparecendo.

7. Apêndice

Para executar o programa, é unicamente necessário invocar ”l*com_run proj*”, sem argumentos.