U.PORTO

FEUP FACULDADE DE ENGENHARIA
UNIVERSIDADE DO PORTO

Relatório do Projeto

LCOM 2019/2020

Tema: “Games”

Elementos:

Diogo Santos up201806878

Marcelo Reis up201809566

Turma 2

Índice

Instruções de Usuário……3

Estado do Projeto……6

Estrutura do Código……10

Function Call Graph……14

Detalhes de Implementação……15

Conclusões……16

Instruções de Usuário

Menu Principal

Quando iniciámos o programa aparece-nos esta tela. Aqui é possível observar os elementos do grupo, o tema do projeto, a turma e no jogo em que nos inspirámos para o desenvolver.

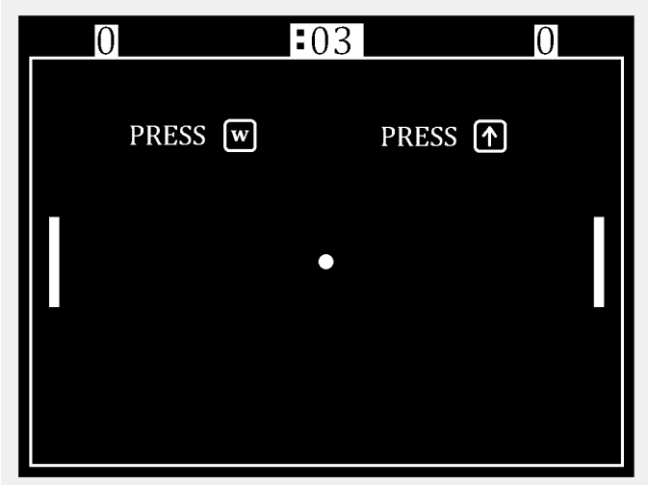
Uma vez aqui é possível selecionar com o rato, a opção “Start Game” que inicia o jogo, a opção “Rules” que abre o menu “Rules” ou então a opção “Exit” que sai do jogo.

Menu Rules

Quando selecionámos a opção para abrir o menu “Rules”, aparece-nos esta tela, na qual encontramos explicadas as regras do jogo, e os seus respetivos controlos.

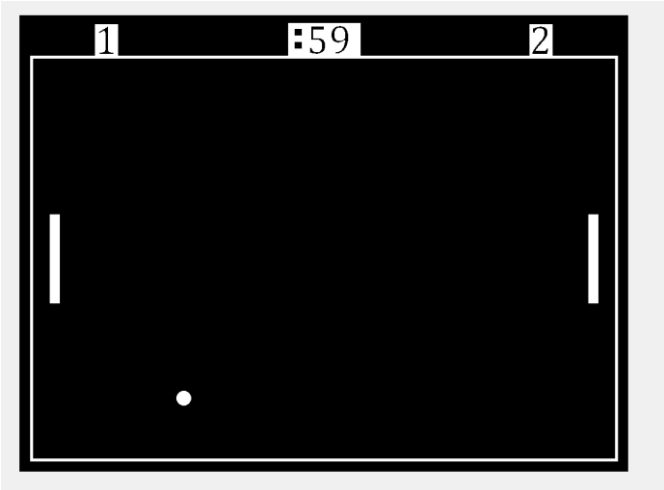
Para deixarmos este menu clicamos na tecla “Esc” que nos leva de volta para o menu principal.

Modo de Preparação para o Jogo



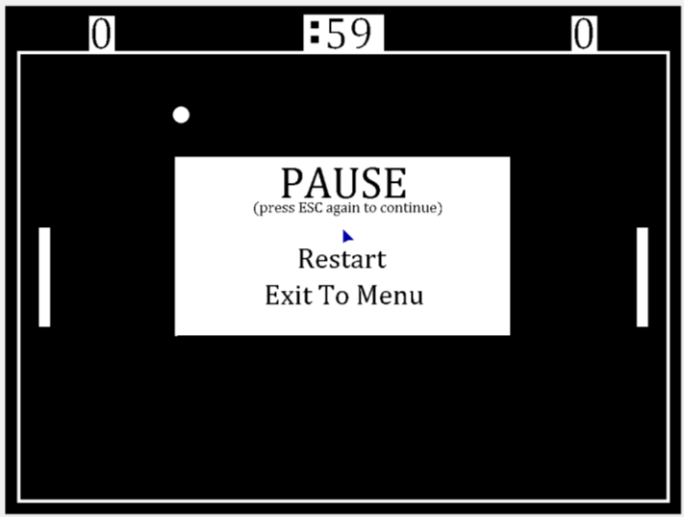
Quando selecionámos a opção “Start Game”, aparece-nos esta tela, na qual é necessário que os jogadores cliquem em simultâneo nas teclas “W” e “seta para cima” para dar início ao jogo.

Modo de Jogo



Quando o jogo está a decorrer aparece-nos esta tela em que a circunferência no meio do campo é a bola os dois retângulos um de cada lado são as barras que cada jogador controla sendo que elas só se podem mover para baixo e para cima. No topo da tela também é possível observar no centro o tempo a descontar sendo que começa em 60 segundos. Em cada lado no topo é possível observar a pontuação de cada jogador. O jogo acaba quando um dos jogadores conseguir alcançar três pontos. Se ao fim de 60 segundos, nenhum jogador tiver marcado ponto, o ponto é atribuído ao jogador que se encontrar mais longe da bola. Nesta situação de jogo é ainda possível colocá-lo em pausa selecionando a tecla “Esc” abrindo assim o “Menu de Pausa”.

Menu de Pausa



Neste menu é possível selecionar com o rato a opção “Restart” em que o jogo é reiniciado ou então a opção “Exit To Menu”, em que se volta para o menu principal.

Podemos ainda se quisermos voltar ao jogo clicando novamente na tecla “Esc”.

Menu de Vitória

Quando o jogo acaba aparece um destes menus conforme tenha ganho o jogador 1 ou jogador 2. Em qualquer um dos dois podemos selecionar com o rato a opção “Play Again” em que podemos voltar a jogar ou então selecionámos “Exit To Menu” em que voltámos ao menu principal.

Estado do Projeto

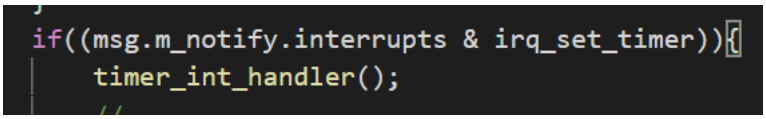
Dispositivo	Funcionalidade	Int.
Timer	Usado para controlar o tempo de jogo e o frame rate.	Y
KBD	Usado para controlar as barras de cada jogador.	Y
Mouse	Usado para selecionar as opções dos menus.	Y
Video Card	Usado para mostrar os gráficos do jogo.	N

Timer

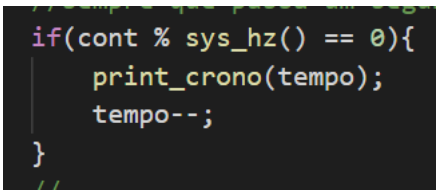
Ao longo do projeto foram várias as vezes em que usámos o timer através de interrupções.

Como já foi referido utilizamo-lo para controlar o tempo de cada jogo, a cada 60 interrupções do timer decrementamos um segundo.

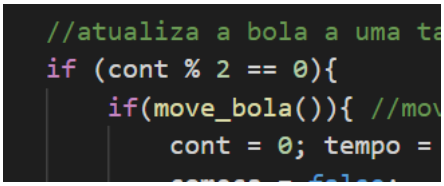
Processo: A cada interrupção do timer chamamos uma função "timer_int_handler()" que apenas incrementa um contador, e assim era possível verificar se tinha passado um segundo através deste pedaço de código que incluo aqui abaixo e que faz parte da função "game()".



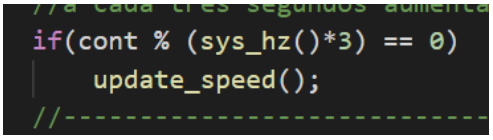
Aqui "cont" é a variável a incrementar e "tempo" é o tempo em que se encontra o jogo desde o instante inicial: 60.



Utilizámos também o timer para o frame rate em que a cada duas interrupções do timer atualizamos a imagem do jogo.



Por último usámo-lo também para atualizar a velocidade da bola a cada três segundos.

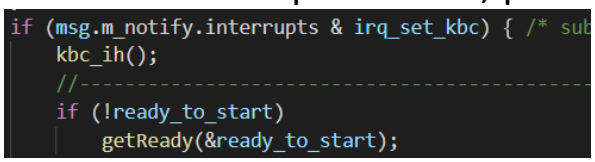


Tudo isto que falei aqui foi feito na função “game()”.

KBD

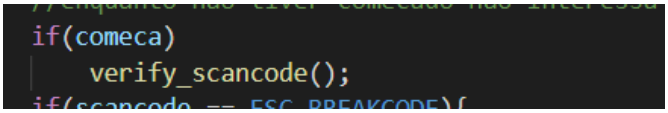
O keyboard como anteriormente foi descrito, é utilizado para a movimentação das barras que se encontram em cada um dos lados do campo, por parte dos utilizadores (jogador1 e jogador2).

Para iniciar o jogo é necessário que os dois jogadores estejam a pressionar em simultâneo as teclas “W” e “seta para cima”, para verificarmos essa ocorrências criamos a função “getReady()”.



Esta verifica os makecodes e os breakcodes recebidos, e se verificar que as duas teclas estão premidas a variável booleana, passada como primeiro argumento, é alterada para “true”, dando início ao jogo.

Iniciando o jogo é necessário tratar dos movimentos verticais da barra, para tal temos a função “verify_scancode”.



Esta função, à medida que vai recebendo makes e breakcodes, modifica as variáveis booleanas “UP_PLAYER1” e ”UP_PLAYER2”, que tomam valor “true” quando os utilizadores clicam no “W” e na “seta para cima”, e as ”DOWN_PLAYER1”e ”DOWN_PLAYER2” que ficam “true” quando clicam no “S” ou na “seta para baixo”.

As quatro variáveis que são alteradas na função anterior são usadas logo a seguir para se poder chamar a função “move_barra()”, responsável pelo visível movimento da barra no decorrer do jogo.

```
//---------------------------------------------------------
//          MOVIMENTA A BARRA
//---------------------------------------------------------
//verifica se algum dos jogadores esta a movimentar a barra
if(UP_PLAYER1 && BarraEsq.y>75){
    move_barra(&BarraEsq,UP_PLAYER1);
}
if(DOWN_PLAYER1 && BarraEsq.y<595){
    move_barra(&BarraEsq,UP_PLAYER1);
}
if(UP_PLAYER2 && BarraDir.y>75){
    move_barra(&BarraDir,UP_PLAYER2);
}
if(DOWN_PLAYER2 && BarraDir.y<595){
    move_barra(&BarraDir,UP_PLAYER2);
}
```

Por fim, a função “move_barra()” recebe como primeiro argumento uma struct que contém as coordenadas ou da barra esquerda ou da barra direita, sendo estas enviadas por referência para posteriormente serem alteradas à medida que se movimentam.

E como segundo argumento a variável que indica se a algum dos jogadores pretende mover a barra para cima.

Mouse

Como já referido utilizámos o rato em stream mode e ativámos o data reporting. Desta forma consegui-mos usar as interrupções do rato em cada menu para verificar a posição do rato na tela e também para saber se foi selecionado o botão direito. A cada interrupção do rato nós chamámos “mouse_ih()” que lê cada byte recebido e tratá-lo da forma adequada.

```
}
if (msg.m_notify.interrupts & irq_set_mouse) {
    mouse_ih();
```

Usámos o mouse na função “main_menu()”, na função “winner()” e na função “menu_pause()”.

Video Card

Utilizámos o video card basicamente para tudo no projeto. Como o nosso projeto não exigia nem muita resolução nem muitas cores usámos o modo 0x105, cuja resolução é: 1024x768; modo indexado com 256 cores e usamos double buffering(função responsável por isto é a função copia_buffer(), que é chamada em várias funções tais como: “menu_pause()” , “main_menu()”, , “winner()”, “game()” e “menu_rules()”). Em relação à movimentação de objetos temos: “move_barra()” que move a barra e “move_bola()” que move a bola; tivemos de verificar também a colisão dos objetos através de: “checkLateralBarColision()” que verifica a colisão da bola com as barras, “checkLineColision()” que verifica a colisão da bola com as linhas laterais e “checkGoal()” que verifica a colisão com as linhas de fundo; todas estas três funções são invocadas em “move_bola()”.

```
checkLateralBarColision();
checkLineColision();
if(checkGoal())
    return true;
```

No que diz respeito a VBE functions usámos apenas a função 0x02 para colocarmos o modo 0x105. Sendo que para isso primeiramente lemos o endereço de memória base do vídeo em “read_adress()” e depois a configuração do modo foi feita em: “graphic_mode()”.

```
int (graphic_mode)(uint16_t mode){

  if(read_adress(mode) != 0)
    return 1;
```

Estrutura do Código

proj.c

Contém apenas o código fornecido pelos professores só que na função “main()” invocámos a função “game()”. Logo por isto o peso deste módulo é muito pequeno, pode-se mesmo dizer insignificante, por isso consideramos com peso praticamente nulo.

timer.c

Neste ficheiro estão contidas as funções relacionadas com o timer. “timer_int_handler()”: incrementa o contador já referido acima; “timer_subscribe_int()”: subscreve as interrupções do timer; e “timer_unsubscribe_int()”: faz o oposto, ou seja, dá “unsubscribe” das interrupções do timer. Esta parte representa cerca de 10% do projeto e o código foi desenvolvido no lab2 por ambos sendo que cada um fez 50% do código lá contido.

mouse.c

Neste ficheiro estão contidas as funções relacionadas com o rato. “mouse_ih()”:lê, a cada interrupção deste, o byte presente no outputbuffer e trata-lo de forma adequada, “subscribe_mouse()”: subscreve as interrupções do rato;“unsubscribe_mouse()”: dá “unsubscribe” das interrupções do rato; “inputBufferFull()”: verifica se é possível escrever no inputbuffer, ou seja, se está vazio; “outputBufferFull()”: verifica se o outputBuffer tem alguma coisa para ler, ou seja, se está cheio; “enable_reporting()”: ativa o data reporting; e “reset_mouse1()”: configura o rato para atuar nas suas configurações iniciais. Esta parte representa cerca de 20% do

projeto e o código foi desenvolvido no lab4 em conjunto, sendo que cada um fez 50%.

keyboard.c

Neste ficheiro estão contidas as funções relacionadas com o teclado. “kbc_ih()”: trata de ler o scancode enviado pelo teclado a cada interrupção deste e trata-lo de forma conveniente; “kbc_subscribe()”: subscreve as interrupções do teclado; “kbc_unsubscribe()”: dá “unsubscribe” das interrupções do teclado; e “kbc_Enable()”: ativa as interrupções no outputubuffer do teclado. Esta parte representa igualmente 20% do projeto e o código foi desenvolvido no lab4 por ambos os membros de forma igual.

vídeo.c

Neste ficheiro estão contidas as funções relacionadas com o vídeo card. “graphic_mode()”: configura o vídeo para atuar no modo passado como argumento; “read_adress()”: lê o endereço base do vídeo; “draw_mouse()”: desenha o rato sendo que se a cor presente no seu xpm for branca não desenha; “draw_xpm()”: desenha um determinado xpm passado como argumento; “clean_xpm()”: coloca tudo a preto relativo a um determinado xpm também passado como argumento; e por último “copia_buffer()”: responsável por passar o conteúdo de memória auxiliar para a memória mapeada em “read_adress()”. Este módulo representa também 20% do projeto. Tal como nos outros módulos este código foi desenvolvido em conjunto no lab5 de forma igual.

game.c

Neste módulo estão presentes as funções reponsáveis por gerir o jogo. “load_xpm()”: carrega todos os xpm logo de início de forma a evitar carregar mais que uma vez o mesmo xpm e desta forma não temos de esperar que um xpm carregue no meio do jogo; “entra()”: é responsável por subscrever todas interrupções e dar os enables necessários; “sai()”: faz o contrário ou seja dá unsubscribe a todas as necessárias e reconfigura os dispositivos; “print_crono()”: imprime o temporizador; “draw_field()”: responsável por desenhar o campo no início e depois de ter ocorrido um ponto; “reboot_field()”: reinicializa as coordenadas de cada componente do campo; “checkGoal()”: verifica se ocorreu colisão entre a bola e a linha de cada lado e portanto se ocorreu ponto; “checkLineColision()”: verifica se ocorreu colisão entre a bola e as linhas superiores e inferiores; “checkLateralBarColision()”: verifica se ocorreu colisão entre a bola e as barras; “move_bola()”: movimenta a bola; “move_barra()”: movimenta a barra; “geraXY()”: gera movimento aleatório para a bola numa fase inicial; “update_speed()”: atualiza a velocidade da bola; “verify_scancode()”: verifica se a tecla premida é válida e se sim atualiza as convenientes variáveis; “verify_pause_mouse_position()”: verifica a posição do rato no menu de pausa; “verify_menu_mouse_position()”: verifica a posição do rato no menu principal; “verify_winner_mouse_position()”: verifica a posição do rato no menu winner; “menu_pause()”: responsável pelo menu de pausa; “menu_rules()”: é responsável pelo menu das regras; “main_menu()”: é responsável pelo menu principal; “winner()”:

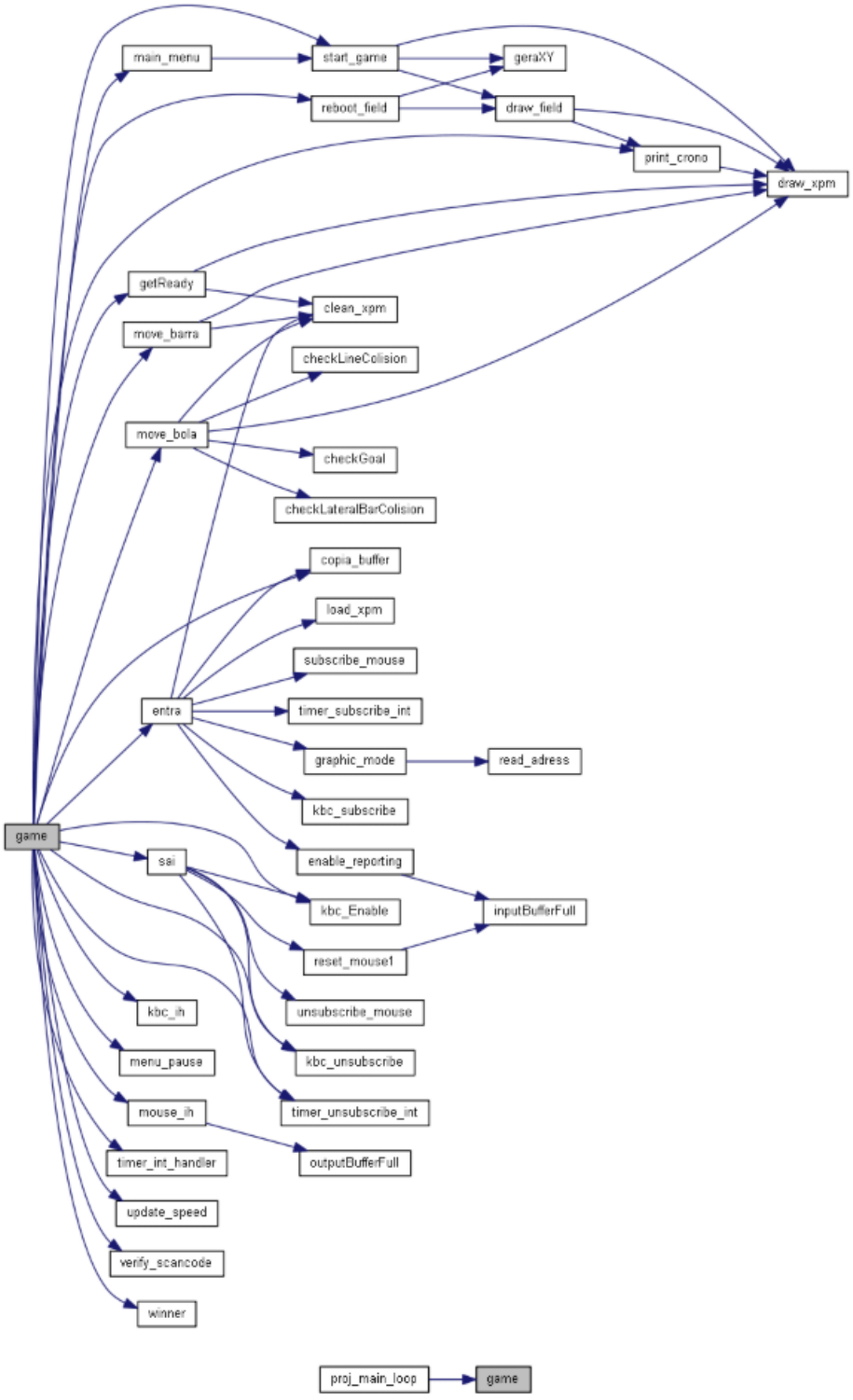
que é responsável pelo menu winner; “getReady()”: é responsável por verificar se os jogadores se encontram prontos para iniciar o jogo; “start_game()”: coloca cada xpm com as suas respetivas coordenadas iniciais; e por último a mais importante na nossa opinião “game()”: responsável por gerir o jogo.

Ainda, não neste ficheiro mas no game.h, está declarada uma struct direcionada para os xpm que é responsável por guardar as suas coordenadas iniciais, as suas coordenadas atuais e pelo seu deslocamento em cada eixo. Este módulo é o mais importante e por isso atribuímos lhe um peso de 27%, o código deste módulo foi desenvolvido ao longo destas últimas semanas por ambos os membros.

figuras.h

Este ficheiro contém apenas a declaração de cada xpm, que desenvolvemos usando o gimp, por isso atribuímos-lhe um peso de apenas 3%. Tal como todo o trabalho, este módulo foi desenvolvido em conjunto.

Function Call Graph



Detalhes de Implementação

No nosso jogo utilizámos programação orientada a objetos, sendo que uma grande parte dos elementos do jogo é uma struct. A bola, as barras, a pontuação dos jogadores, o tempo de jogo e o rato são structs “Coordenadas”, com as coordenadas da sua posição no ecrã e, no caso da bola, também com o deslocamento em Y e em X necessário para o deslocamento da mesma.

Relativamente a geração de frames, a cada 60 interrupções é atualizado o tempo de jogo. O movimento da bola é realizado a uma taxa de 30 frame rate. A cada interrupção do timer é copiado do buffer auxiliar para a memória principal, usando assim double buffering.

Na realização do projeto também tivemos em atenção a ocorrência de colisões, nos menus (principal, pausa, winner) entre o rato e as opções (“Start Game”, “Rules”, “Exit”, “Restart”, “Play Again” e “Exit to Menu”) e entre a bola e os elementos do jogo como as barras e os limites do campo. No primeiro caso se o rato passar por cima das opções estas ganham uma borda, mudando assim o seu aspeto. Já o segundo caso se a bola colidir com as barras ou com os limites, superior e inferior, ela é refletida com um ângulo de 90^{o}. Caso colida com as laterais, tanto a esquerda como a direita, é incrementada uma das pontuações e a bola é recolocada no centro campo.

Conclusões

Nós achamos que tanto as aulas teóricas como as práticas foram enriquecedoras, apesar de as práticas trazerem mais benefícios, pois foi através delas que se tornou muito mais fácil entender a matéria dada nas teóricas. Porém, devido à estrutura do nosso horário, as aulas teóricas decorriam no mesmo dia que as práticas só que posteriormente, causando dificuldade na aprendizagem, visto que a matéria que trabalhávamos nas aulas práticas era a que tinha sido lecionada na semana anterior.

Uma melhoria que propomos é a junção da informação necessária para a realização dos "labs" num só local, porque existe informação importante nos slides das aulas teóricas que não constam nos guiões.

Em relação ao trabalho, sentimos alguma dificuldade na união de todos os periféricos num só "programa", na criação das imagens utilizadas no jogo (p.e o campo de jogo, menu principal, …) e também na deteção de colisões.

Para finalizar, gostávamos de realçar que ambos os membros do grupo trabalharam de igual forma, tendo sido o trabalho igualmente dividido pelos dois membros.