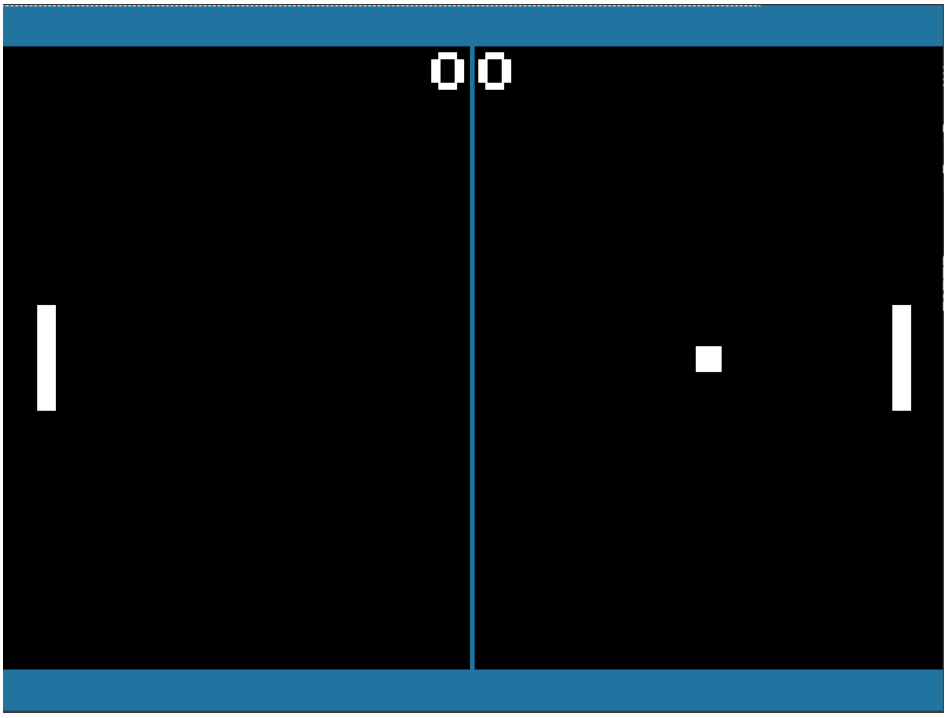
U.PORTO

FEUP FACULDADE DE ENGENHARIA UNIVERSIDADE DO PORTO

AlterPong

Projeto LCOM



Realizado por:

Gonçalo Miranda up202108773

João Sequeira up202108823

Tomás Maciel up202006845

Conteúdo

1. Instruções para o utilizador

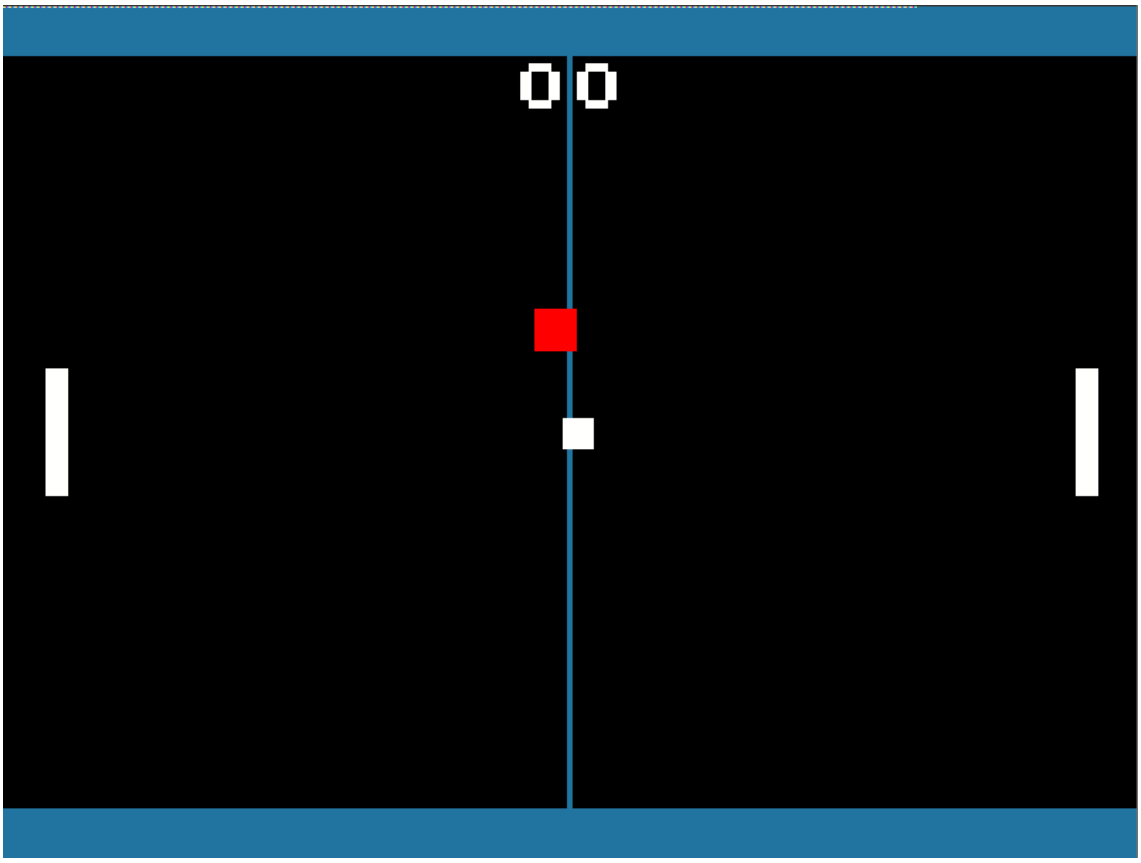
O nosso jogo é muito semelhante ao “Pong”, famoso videojogo lançado em 1972 pela Atari, em que duas barras e um quadrado são os personagens principais. As barras são movidas na vertical e o objetivo do jogo é evitar que a bola toque na borda do lado do jogador e que aconteça isso mesmo no lado adversário. Ao fim de 3 golos, o jogo termina.

Jogo

A peça da esquerda é controlada com as teclas ‘W’ (cima) e ‘S’ (baixo) e a peça da direita é controlada com as setas ‘△ ‘ (cima) e ‘▽ ‘ (baixo).

Se for pressionado ESC durante o jogo, retorna-se ao menu inicial.

De destacar que é possível visualizar o resultado do jogo em cima.

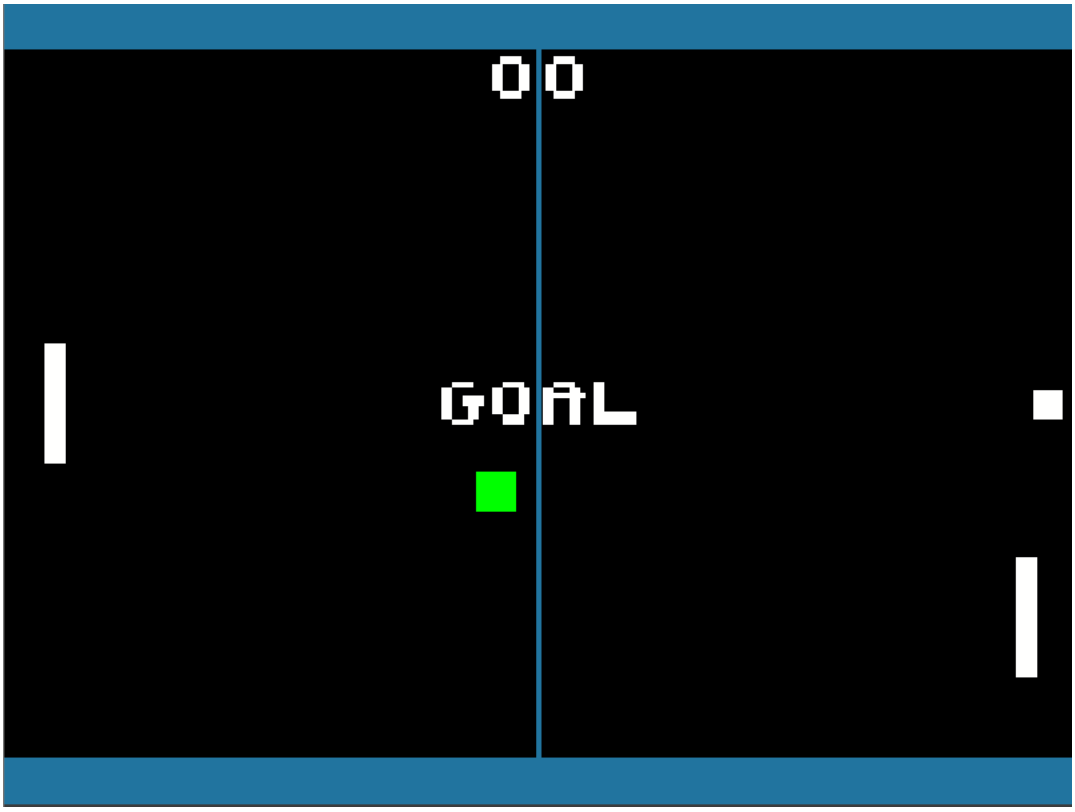


Durante o jogo, poderão aparecer quadrados vermelhos e verdes que são powerups que decidimos implementar e, se forem apanhados, têm a duração de 10 segundos.

Reduz os limites do campo (10 segundos)

Reduz a velocidade da peça adversária para metade (10 segundos)

Quando é marcado um golo (a bola toca numa das bordas, é apresentada uma mensagem que representa o golo).



Ao fim de 3 golos marcados por um dos jogadores, o jogo é então encerrado e esse jogador acaba por sair vencedor.

2. Estado do Projeto

Tabela de dispositivos

Dispositivo	Para quê	Interrupções
Timer	Frame rate, limitar o tempo dos powerups	Sim
Keyboard	Controlar as peças durante o jogo	Sim
Video card	Interface com os jogadores e apresentar o jogo em si e tudo o que acontece	Não

Teclado

O teclado, como já referido anteriormente, é usado para controlar as peças na vertical.

Isto é controlado usando os make codes das teclas em que, quando uma tecla é pressionada (isto é, o scancode é igual ao make code respetivo da tecla) é invocada a função para mover a peça respetiva (move_piece1_down(), move_piece1_up(), move_piece2_down() e move_piece2_up()).

Timer

O timer (timer 0) é utilizado para limitar o tempo em que os powerups aparecem no ecrã e também em que estão ativos, sendo também usado para controlar o frame rate (no nosso caso, 60 FPS).

Video card

A vídeo card apresenta toda a interface com o utilizador e também apresenta o jogo em si e o desenho de todas as unidades que neste participam.

Optamos por usar uma resolução de 800x600 (0x115), em que as cores têm 3 bits por pixel e estão modeladas no modo direto.

Para desenharmos todas as unidades do jogo (peças, bola, powerups, mensagem de golo e resultado), usamos ficheiros XPM que eram depois desenhados no ecrã com a função draw_xpm (xpm_map_t xpm, int x, int y). Os únicos elementos do campo que não desenhamos desta forma foram os limites de lado e a linha do meio campo em que recorremos à função vg_draw_rectangle (uint16_t x, uint16_t y,

uint16_t width, uint16_t height, uint32_t color) que já tínhamos implementado anteriormente nas aulas práticas no lab5.

3. Organização/Estrutura do código

kbc.c – 2%

Este ficheiro contém apenas uma função que efetua a leitura do registo de estado do KBC e que aproveitamos do Lab3 que já tínhamos feito na aula prática.

keyboard.c – 5%

Este ficheiro contém as funções que lidam com as interrupções do teclado (susbcribe e unsubscribe) e também o “interrupt handler” do dispositivo. À semelhança do kbc.c foi inspirado no trabalho que já tínhamos feito na aula prática do Lab3.

vídeo_gr.c – 15%

Este ficheiro é bastante importante no nosso projeto pois é com ele que desenhamos todo o campo e as peças que este contém. Temos funções que permitem inciar a gráfica no modo pretendido e também funções que desenham no ecrã. Além disso, também implementamos as funções responsáveis pelo double buffering. Parte do código deste módulo foi reaproveitado do Lab5.

timer.c – 4%

Mais uma vez neste módulo reaproveitamos o trabalho desenvolvido no Lab2 e temos implementadas funções que lidam com as interrupções do timer e também o “interrupt handler” deste que incrementa apenas o irq_timer_counter, e também aproveitamos a função que define a frequência do timer para podermos definir a frequência do nosso jogo.

utils.c – 2%

Este ficheiro contém as funções já desenvolvidas no Lab2 exatamente iguais uma vez que são funções gerais e, portanto, não sofreram nenhuma alteração. Permitem-nos obter o LSB e MSB de um valor de 16 bits e efetuar uma leitura de um registo de um valor de 8 bits.

ball.c – 25%

Este módulo contém tudo o que está relacionado com movimento, colisões e desenho da bola durante o jogo que está a ocorrer e também quando a bola toca na borda do lado de um dos jogadores (é golo).

entity.c – 5%

Apesar de este módulo possuir apenas uma função, é um módulo muito importante no nosso projeto sem o qual não seria possível desenhar quase nada do nosso jogo. A única função nele presente é responsável por desenhar figuras que estão em ficheiros xpm.

interrupts.c – 20%

Este ficheiro é responsável por lidar com as interrupções em geral e com tudo o que acontece quando se dá uma de qualquer dispositivo. Nele criamos duas funções em que uma subscreve as interrupções de todos os dispositivos e a outra dessubscreve todas as interrupções (por questões de organização) e depois temos a função principal interrupts() que então lida com cada interrupção de cada dispositivo. Também temos a função handle_keys() que basicamente apenas regista o estado de cada tecla (se foi pressionada ou largada).

main.c – 10%

Este ficheiro contém o loop responsável pelo nosso jogo e também funções que tratam do início e fim da execução do jogo no MINIX.

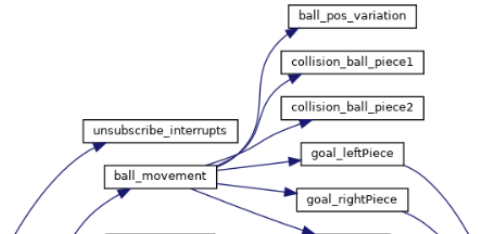
piece.c – 6%

Este é o módulo responsável pela lógica do movimento das peças dos jogadores e também pelo desenho destas na posição pretendida.

powerups.c – 6%

Este módulo lida com a lógica dos powerups, sendo responsável pela escolha deles e desenho dos mesmos (draw_powerup()) e também por detetar a colisão da bola com o powerup e também que jogador é que vai beneficiar da atuação do powerup (catch_powerup()).

Function Call Graph



4. Detalhes da Implementação

Tópicos abordados nas aulas

Podemos abordar o tema do double buffering que foi algo que abordamos nas aulas teóricas, mas a nível prático não tivemos a oportunidade de trabalhar nisso. Na realização do nosso projeto, essa foi a parte decisiva uma vez que depois de percebermos isso conseguimos avançar com o projeto, mas até que percebêssemos a lógica (que é um pouco vaga apenas com a teoria) ainda acabamos por perder algum tempo, e sendo um tema interessante seria bom termos oportunidade de o trabalhar nas aulas práticas.

Tópicos não abordados nas aulas

Dentro dos tópicos que não abordamos nas aulas, gostaria de destacar de facto as colisões dos objetos. Inicialmente estávamos a verificar se haviam colisões usando =, mas isto não resultava, uma vez que, por exemplo, o movimento da bola por frame não é de uma unidade em x e sim de 5. Por isso, paras as colisões tivemos de usar os sinais de comparação < e > e verificar as colisões não num número certo, mas sim num intervalo de 5 unidades neste caso. Seria interessante termos uma base sobre isto uma vez que no curso não é assim um tema muito abordado e, no entanto, é algo muito útil de conhecer minimamente.

Outro tópico que penso que não temos um conhecimento muito profundo e tivemos de rever e aprender um pouco por nós é a modularidade de classes.

Inicialmente, estávamos só a escrever código no main relativo à lógica do nosso jogo, e só depois é que começamos a dividir em módulos e a criar ficheiros para cada parte do jogo. Penso que seria interessante dar-nos uma base de como começar a implementação de um projeto assim por módulos ao invés de fazermos como a maioria das pessoas que faz tudo no main e só depois pensa nos módulos.

5. Conclusões

A nossa ideia inicial neste projeto era outra que não conseguimos implementar, diria que por falta de tempo e sobreposição de projetos de outras cadeiras. No entanto, no fim, pensamos que fizemos um bom trabalho neste projeto e a única coisa que teríamos de mudar para alcançar o objetivo inicial seria ao invés de termos uma vista aérea do jogo termos uma vista em primeira pessoa.

Precisamente este facto seria aquilo que gostaríamos de adicionar ao nosso programa assim como trocar as barrinhas que os jogadores controlam por algo mais animado como um boneco ou algo do género.

Com a realização do projeto, conseguimos obter um conhecimento profundo do funcionamento dos dispositivos que já tínhamos abordado nas aulas práticas e vê-los em ação na prática num projeto de larga escala como foi este.