U.PORTO

FEUP FACULDADE DE ENGENHARIA
UNIVERSIDADE DO PORTO

Breakout

MIEIC - Laboratório de computadores
2º Ano - 1º Semestre

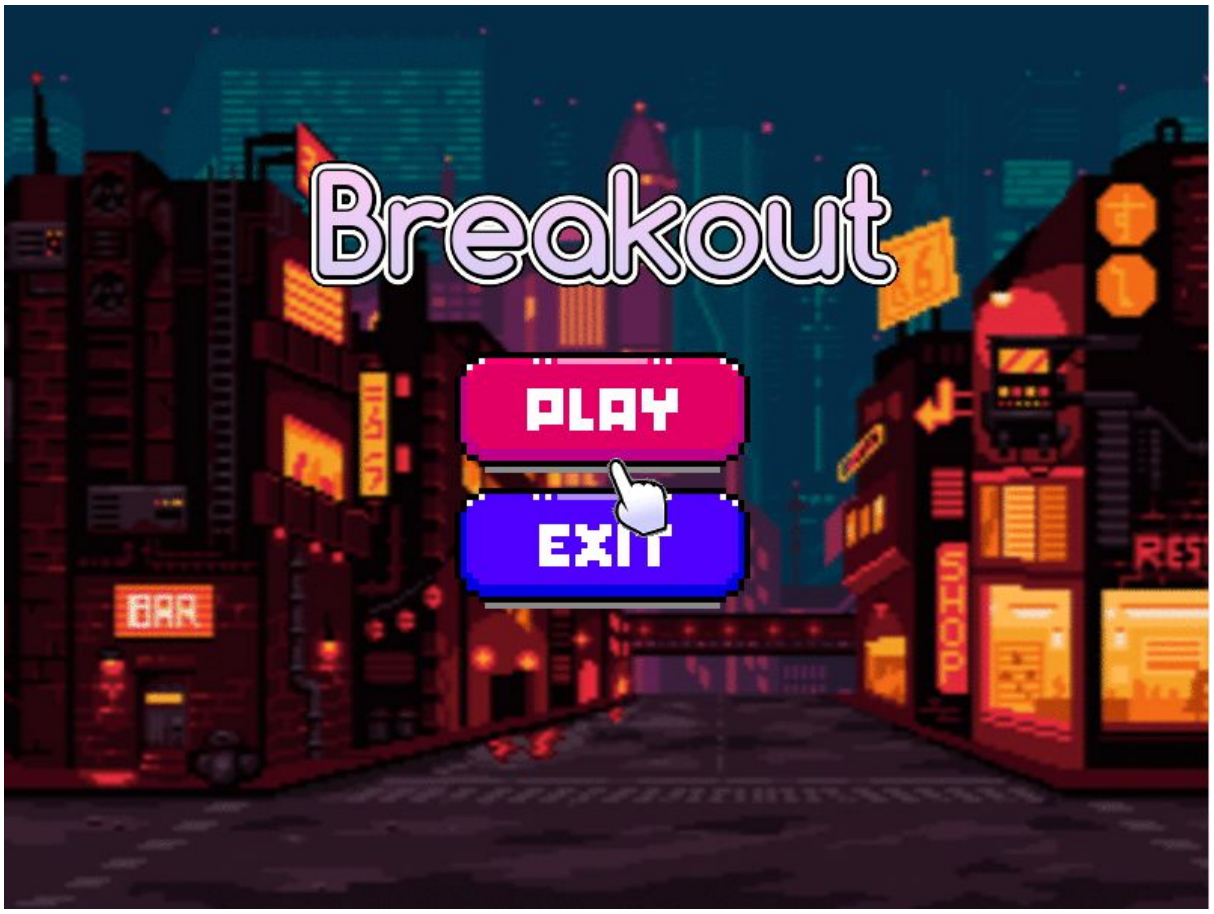
Turma 6 Grupo 7
Pedro Magalhães Moreira Nunes - up201905396
Diogo Alexandre Paredes Azevedo Costa e Sá - up201905383

Indice

1. Instruções do utilizador

1.1 Menu principal

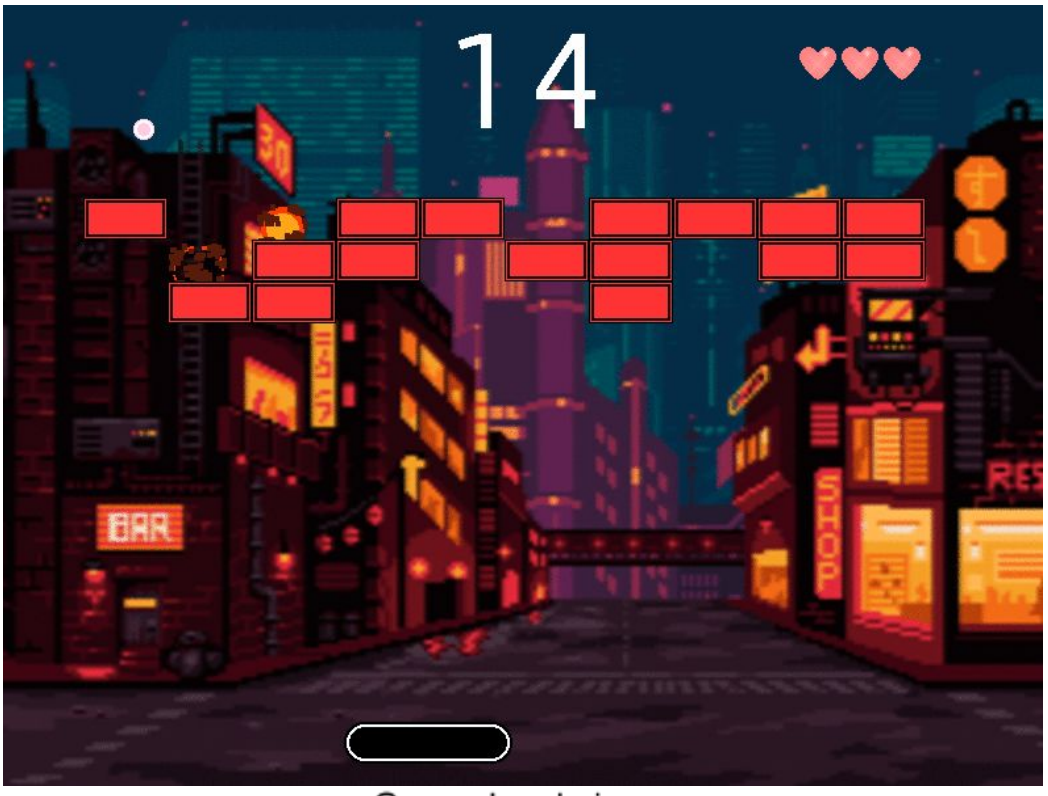
Ao inicializar o projeto é apresentado o menu principal. Neste menu é possível escolher entre jogar o jogo, ou sair da aplicação. A escolha é feita apenas através do rato.



1.2 Jogo

O jogo consiste num clone do clássico Atari, Breakout. O gameplay baseia-se em movimentar um paddle, com o objetivo de defletir uma bola e com isso destruir todos os blocos presentes no jogo, evitando deixar a bola cair.

No jogo é possível ver a barra de vida, que apresenta o número de vidas que o jogador possui (número de corações), os blocos que o utilizador tem que destruir, um score que apresenta o número de blocos destruídos e um paddle que é controlado através das teclas A e D do teclado.



Gameplay do jogo

Caso o jogador consiga destruir os 30 blocos sem perder as 3 vidas, ou caso perca as 3 vidas, é-lhe apresentado o ecrã de game over. Se conseguir destruir os 30 blocos, é lhe apresentado uma mensagem de vitória, caso contrário, uma de derrota

No ecrã de game over é pedido ao utilizador que pressione a tecla ESC para voltar para o menu principal.

2. Estado do projeto

Dispositivos	Utilização	Interrupção
Timer	Controlo da framerate	Y
Teclado	Controlo do *paddle*	Y
Rato	Interação no menu	Y
Placa Gráfica	Desenho do jogo e do menu	N

2.1 Timer

O timer é utilizado para controlar a "frame rate".

Este encontra-se presente na função principal , breakout(), onde acontece a subscrição das interrupções através da função timer_subscribe_int, e a respetiva remoção da subscrição através da função timer_unsubscribe_int().

A frame rate também é definida no início do jogo, com a função timer_set_frequency(), sendo esta estabelecida a 60 frames por segundo.

2.2 Teclado

O teclado é utilizado para controlar o paddle e para interagir com o ecrã de game over.

Este encontra-se presente na função principal, breakout(), onde acontece a subscrição das interrupções através da função keyboard_subscribe_int e a respetiva remoção da subscrição através da função keyboard_unsubscribe_int().

A função kbc_get_scancode() serve para ler o output buffer (por entremeio da função kb_ih()) verificando se tudo o que estamos a fazer é valido.

Por outro lado, a função kbc_verify_scancode() verifica se estamos a ler um scancode de 2 bytes ou de apenas 1 byte.

A movimentação do paddle é feita através das teclas A e D , que movem o mesmo para a esquerda ou para a direita. No final do jogo é utilizada a tecla ESC para voltar para o menu principal.

2.3 Rato

O papel principal do rato é a interação no menu principal. Sempre que for possível uma interação entre o rato e os botões apresentados no menu, a cor lateral dos botões muda e para iniciar essa interação o utilizador apenas precisa de premir o botão esquerdo do rato.

O rato encontra-se presente na função principal, breakout(), onde acontece a subscrição das interrupções através da função mouse_subscribe_int() e a respetiva remoção da subscrição através da função mouse_unsubscribe_int().

É também utilizada a função send_cmd_mouse(), utilizada para inicializar o streaming mode do rato.

As funções analize_buffer() e process_packets() apresentam uma função bastante semelhante as utilizadas no keyboard, sendo a primeira utilizada para ler o output buffer e verificar se tudo o que estamos a ler é valido, e a 2º para processar os packets que estamos a ler.

A função parse_packet() é apenas chamada aquando da conclusão de um packet, e a sua função é passar toda a informação processada na função process_packets() para a struct packet.

2.4 Placa Gráfica

A gráfica é utilizada para desenhar sprites e animated sprites.

Esta encontra-se presente em todas as funções de desenho, tais como draw_sprite(), draw_brick(), draw_ball(),animate_sprite(),draw_background().

É também utilizado o método de double_buffering através da função double_buff(),sendo esta a única função que escreve na VRAM. As funções para escrever requerem a utilização da função pixel_set_color(), função esta que altera a cor do pixel no segundo buffer.

Uma função desenvolvida durante os labs utilizada é a vg_read_xpm(), função esta que lê e desenha automaticamente uma imagem xpm.

O modo utilizado foi 0x115, modo este de 24 bits com uma resolução 800x600 e este é inicializado através da função vge_set_mode().

3. Estrutura do código

3.1 Timer

Com este módulo é possível subscrever as interrupções do timer e alterar a respetiva frequência.

O módulo foi desenvolvido durante os labs.

Desenvolvido por : Pedro Nunes (80%) e Diogo Sá (20%)

3.2 Teclado

Com este módulo é possível subscrever e cancelar a subscrição das interrupções do teclado e processar scancodes.

O módulo foi desenvolvido durante os labs.

Desenvolvido por : Pedro Nunes

3.3 Rato

Com este módulo é possível subscrever e cancelar a subscrição das interrupções do rato, ler e processar os packets assim como enviar comandos para o rato.

O módulo foi desenvolvido durante os labs.

Desenvolvido por : Pedro Nunes

3.4 Placa Gráfica

Com este módulo é possível inicializar o modo gráfico, utilizar double buffering, “pintar” os pixels , ler ficheiros “xpm” e limpar o ecrã.

O módulo foi totalmente desenvolvido durante os labs, porém ligeiramente adaptado para o projeto.

Desenvolvido por : Pedro Nunes

3.5 Animate Sprite

Com este módulo é possível inicializar sprites animadas, processar sprites animadas já criadas e destruir essas mesmas sprites.

Desenvolvido por : Pedro Nunes

3.6 Sprite

Permite criar sprites a partir de xpms , desenhá-las e destruí-las.

Desenvolvido por : Pedro Nunes

3.7 Background

Permite criar um buffer para o background, desenhar o respetivo background, e limpar tudo o que tenha sido desenhado no mesmo.

Desenvolvido por : Pedro Nunes

3.8 Ball

Todas as mesmas funcionalidades da sprite porém possui uma funcionalidade extra que guarda as coordenadas antigas da mesma. Isto é utilizado para calcular as colisões no breakout() loop.

Desenvolvido por : Pedro Nunes

3.9 Bricks

Módulo idêntico à sprite, apenas possui um parâmetro extra que verifica se determinado brick já foi destruído.

Desenvolvido por : Pedro Nunes

3.10 Game

Módulo que contém o loop principal do jogo. É possível detectar colisões entre o cursor e os diversos botões, entre a bola e os diversos bricks, desenhar um score no ecrã e apresentar um ecrã de game over.

É neste módulo que ocorre a inicialização de todas as sprites do jogo.

Desenvolvido por : Pedro Nunes

Peso de cada modulo

Timer	5%
Keyboard	10%
Mouse	10%
Video	10%
Animate Sprite	5%
Sprite	5%
Background	10%
Ball	5%
Bricks	5%
Game	35%

4. Detalhes de implementação

4.1 Máquina de estados

O loop principal do jogo consiste numa máquina de estados com 3 estados possíveis, o menu, o jogo e o de saída.

Os estados são alterados no menu principal, sendo que no menu pode ser escolhido entre o jogo, e a saída.

Por outro lado, no jogo apenas podemos escolher o menu.

4.2 Programação orientada a objetos

Foi usada programação orientada a objetos nos módulos sprite, animsprite, bricks e ball.

Os últimos dois são bastante idênticos porém o módulo bricks apresenta um parâmetro extra para verificar se determinado brick já foi destruído, enquanto que o módulo ball apresenta dois parâmetros que guardam as posições antigas da bola, que, como já foi dito são utilizadas para calcular o movimento relativo da bola consoante uma colisão.

4.3 Deteção de colisões

No projeto foi utilizada a deteção de colisões entre cursor-butões, butões-bola e butões-paddle.

No geral, a deteção de colisões apresenta-se funcional, porém o cálculo da posição da bola apresenta um pequeno bug quando atinge dois bricks diferentes no mesmo frame.

4.4 RTC

A implementação do RTC foi iniciada, porém, devido a constrangimentos de tempo não foi incorporada no projeto.

Porém o protótipo desenvolvido permite subscrever as interrupções, remover a subscrição e obter a data.

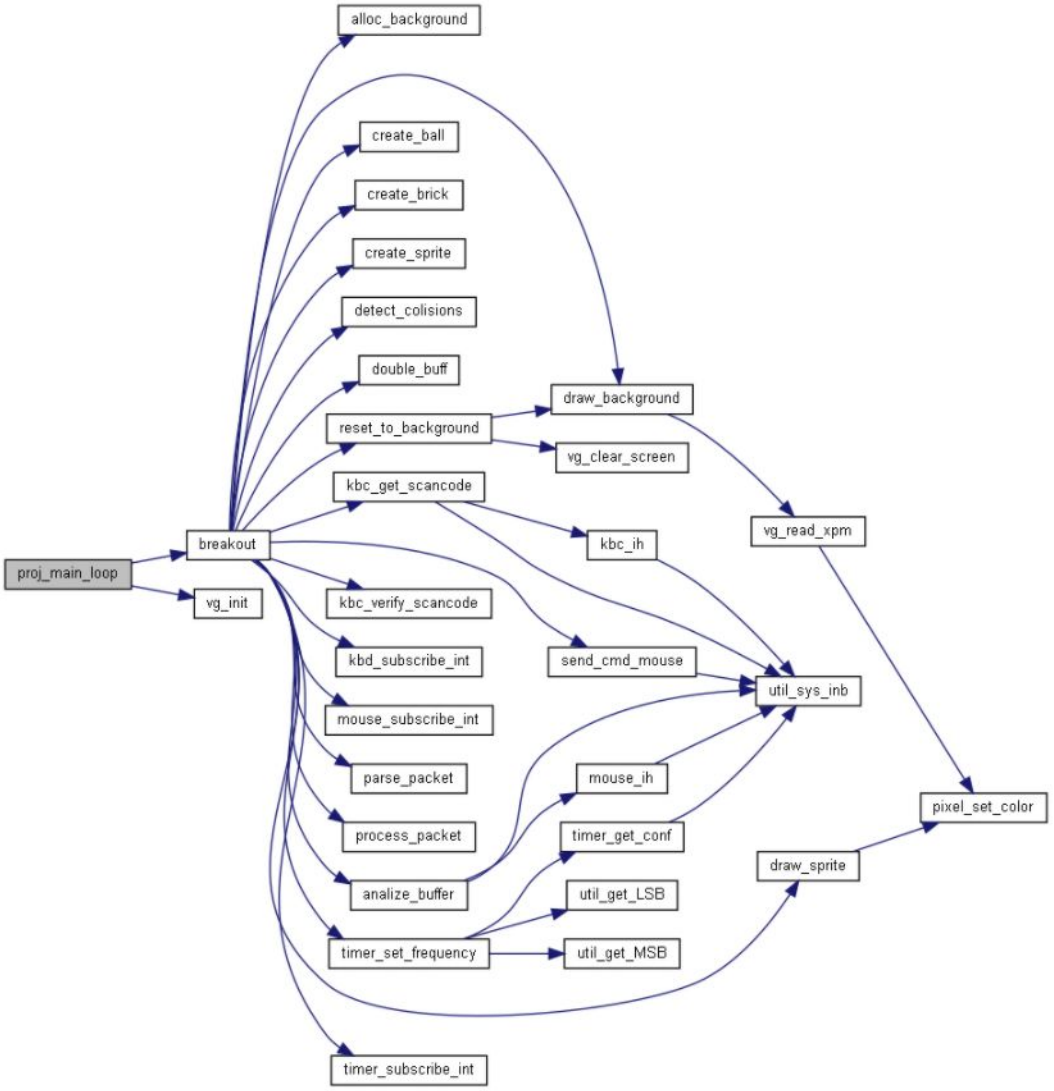
4.5 XPMs utilizados

Todos as sprites e animações utilizadas foram retiradas do website https://opengameart.org/ e do https://itch.io/game-assets , sendo posteriormente adaptadas e transformadas em xpm através do uso do GIMP.

Sempre que era pedido, era atribuído o crédito no código.

As fontes, por outro lado, foram feitas manualmente no GIMP.

4.6 Function call graph



5. Conclusões

No geral, achei a unidade curricular bastante interessante e trabalhosa. A meu ver tanto os guiões como os slides não eram de fácil leitura, sendo ligeiramente confusos, dado que a maior parte do tempo despendido era gasta a tentar compreender ambos.

Por outro lado, acho que a estrutura dos labs está excelente, sendo o término da unidade curricular um projeto, é algo bastante interessante(mesmo desgostando de projetos) , pois temos a chance de implementar aquilo que trabalhamos durante o semestre inteiro.

Por fim, acho que aulas sobre colisões poderiam ser feitas, pois é algo que bastantes projetos utilizam e que pode ter um cariz trabalhoso.