

Faculdade de Engenharia da Universidade do Porto



Mestrado Integrado em Engenharia Informática e Computação

2º ano

Laboratório de Computadores - EIC0020

Ano Letivo 2014/2015

“ Arkanix ”

Professores

Pedro Ferreira Souto
(pfs@fe.up.pt)

Bruno Laranjo
(laranjo@fe.up.pt)

Diego de Jesus
(diegojesus@fe.up.pt)

José Queirós Pinto
(zepinto@fe.up.pt)

Leonel Dias
(leoneldias@fe.up.pt)

Mário Cordeiro
(pro11001@fe.up.pt)

Turma 3, Grupo 3

Estudantes

Diogo Belarmino Coelho Marques, up201305642@fe.up.pt

Pedro Miguel Pereira de Melo, up201305618@fe.up.pt

Índice

1. Manual.....	4
1.1 Menu inicial.....	4
1.2 Menu modo multiplayer	5
1.3 Modo singleplayer	6
1.4 Modo multiplayer	9
2. Informação do Projeto	12
3. Organização/Estrutura do código	13
3.1 arkanix.c	13
3.2 bitmap.c	13
3.3 generate.c	13
3.4 input.c.....	13
3.5 level.c	14
3.6 menu.c	14
3.7 rtc.c	14
3.8 score.c.....	15
3.9 serial.c	15
3.10 sprite.c	15
3.11 ss.c.....	15
3.12 timer.c	16
3.13 vbe.c.....	16
3.14 video.c	16
4. Grafos das chamadas de funções	17
5. Detalhes da implementação.....	19
5.1 Placa de vídeo	20
5.2 <i>Input</i> (rato e teclado)	20
5.3 RTC.....	21
5.4 Timer	21
5.5 Porta série	21
6. Avaliação	22
A. Anexos	23
A1 Tabela de blocos	23
A2 Recomendações de instalação.....	23

Índice de figuras

Figura 1 - menu inicial	4
Figura 2 - botão singleplayer (estado normal e selecionado)	4
Figura 3 - botão multiplayer (estado normal e selecionado)	4
Figura 4 - botões de opção para o controlo da barra	5
Figura 5 - botão “quit” (estado normal e selecionado)	5
Figura 6 - menu modo multiplayer	5
Figura 7 - botão “local game” (estado normal e selecionado)	6
Figura 8 - botão “serial port” (estado normal e selecionado)	6
Figura 9 - botão “<” (estado normal e selecionado)	6
Figura 10 - último nível do modo singleplayer do Arkanix	7
Figura 11 - blocos do jogo	7
Figura 12 - barra do jogador, com respetiva bola	7
Figura 13 - área de jogo	7
Figura 14 - informação do nível a ser jogado	7
Figura 15 - tabela das melhores pontuações	8
Figura 16 - pontuação acumulada pelo jogador	8
Figura 17 - vidas do jogador	8
Figura 18 - ecrã de introdução do nome	8
Figura 19 - partida típica no modo multiplayer	9
Figura 20 - número de vidas e pontuação de um jogador	9
Figura 21 - temporizador do modo multiplayer	10
Figura 22 - ecrã do jogador vencedor	10
Figura 23 - ecrã mostrado numa situação de empate	11
Figura 24 - ecrã mostrado em caso de desistência de um jogador	11

1. Manual

1.1 Menu inicial



Figura 1 - menu inicial

O menu inicial do Arkanix surge quando se inicia a aplicação. Através dele o pode iniciar uma nova partida no modo singleplayer, modificar os controlos da barra ou entrar no menu do modo multiplayer. Existem ao todo quatro botões distintos neste menu:

- ✓ **Botão singleplayer** - inicia uma nova partida no modo *singleplayer*.

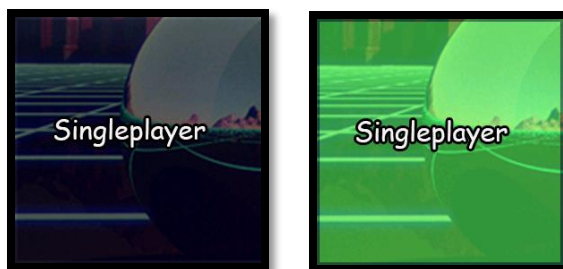


Figura 2 - botão singleplayer (estado normal e selecionado)

- ✓ **Botão multiplayer** - permite aceder ao menu do modo *multiplayer*.

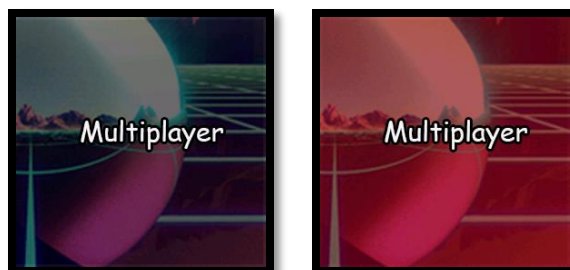


Figura 3 - botão multiplayer (estado normal e selecionado)

- ✓ **Botão controlo** - possui dois estados, *Mouse* (por defeito) e *Keyboard* que indicam qual o dispositivo a utilizar para controlar a barra durante o jogo no modo singleplayer. Quando pressionado, transita de estado e muda a sua aparência, permitindo assim ao utilizador jogar com o dispositivo que lhe é preferido e verificar a sua escolha antes do início da partida.



Figura 4 - botões de opção para o controlo da barra

- ✓ **Botão “Quit”** - encerra a aplicação quando pressionado.



Figura 5 - botão “quit” (estado normal e selecionado)

Sempre que o utilizador coloca o rato sobre qualquer um dos botões do menu iniciar, estes mudam de cor de fundo.

1.2 Menu modo multiplayer

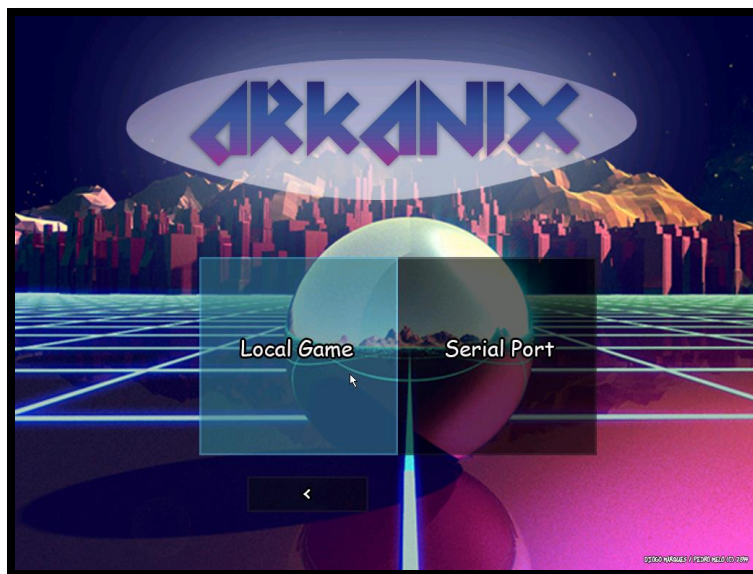


Figura 6 - menu modo multiplayer

O menu do modo multiplayer permite iniciar uma nova partida no modo multiplayer; **do tipo local**, em que ambos os jogadores se enfrentam no mesmo computador ou **do tipo em série**, no qual ambos os jogadores se enfrentam em computadores diferentes. No menu do modo multijogador existem ao todo três botões distintos:

- ✓ **Botão “Local Game”** - inicia um novo jogo do tipo local no modo multiplayer.

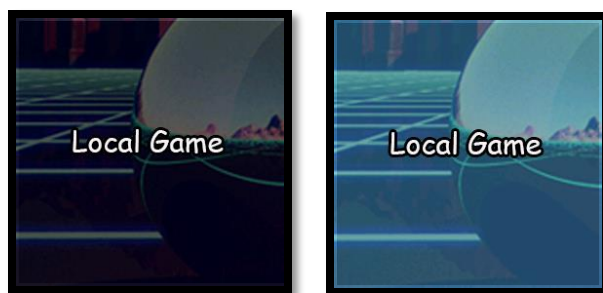


Figura 7 - botão “local game” (estado normal e selecionado)

- ✓ **Botão “Serial Port”** - inicia um novo jogo do tipo em série no modo multiplayer.

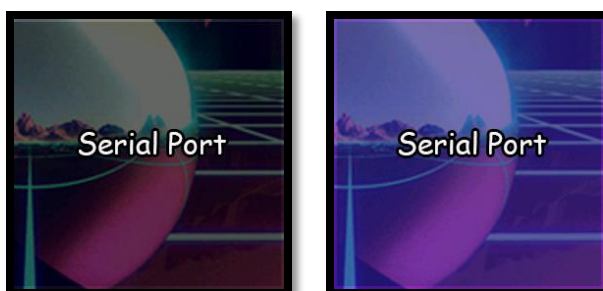


Figura 8 - botão “serial port” (estado normal e selecionado)

- ✓ **Botão “<”** - permite sair do menu do modo multiplayer, regressando ao menu inicial.



Figura 9 - botão “<” (estado normal e selecionado)

Sempre que o utilizador coloca o rato sobre qualquer um dos botões do menu iniciar, estes mudam de cor de fundo.

1.3 Modo singleplayer

No modo singleplayer, o jogador terá de ultrapassar três níveis distintos. Para tal necessita de controlar uma barra e uma bola, de forma a destruir todos os blocos presentes em cada um dos níveis. O utilizador possui um total de três vidas, com as quais terá de ultrapassar todos os níveis do Arkanix. O jogador perde uma vida sempre que não conseguir desviar a bola com a barra, deixando-a tocar no fundo da área de jogo.

O jogador possui ainda uma pontuação, que vai aumentando consoante o número de blocos que destruir.

Na figura seguinte pode ver-se o último nível do Arkanix:



Figura 10 - último nível do modo singleplayer do ArkaniX



Figura 13 - área de jogo

A componente principal deste modo é a **área de jogo** (figura à esquerda), estando toda a ação da partida aqui confinada. No topo da área de jogo encontram-se os **blocos** a eliminar pelo jogador.



Figura 11 - blocos do jogo

Na parte inferior da área de jogo encontra-se a **barra** e a **bola**, inicialmente presa. O utilizador deve libertar a bola na posição que lhe parecer estrategicamente mais conveniente, necessitando apenas de pressionar a tecla *Enter* ou o botão esquerdo do rato consoante esteja a utilizar o teclado ou o rato, respetivamente.



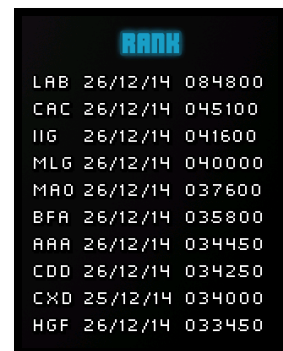
Figura 12 - barra do jogador, com respetiva bola

No centro da área de jogo pode ainda ver-se uma **etiqueta** que indica o número do respetivo nível a ser jogado.



Figura 14 - informação do nível a ser jogado

No lado direito da área de jogo, o jogador pode consultar a **tabela de melhores pontuações** (figura à direita), onde se mostram as melhores classificações no modo singleplayer até ao momento da partida. Ao lado de cada pontuação é possível ver-se o nome do jogador que a obteve bem como a data desse feito.



RANK		
LAB	26/12/14	084800
CAC	26/12/14	045100
IIG	26/12/14	041600
MLG	26/12/14	040000
MAO	26/12/14	037600
BFA	26/12/14	035800
AAA	26/12/14	034450
COO	26/12/14	034250
CXD	25/12/14	034000
HGF	26/12/14	033450

Figura 15 - tabela das melhores pontuações

Acima da tabela de pontuações o jogador pode consultar a sua **pontuação** em todos os instantes da partida (figura à direita). A pontuação de um jogador incrementa sempre que a bola atinge um dos blocos na área de jogo. Aos blocos de diferentes cores está atribuída uma diferente pontuação (ver anexo).



Figura 16 - pontuação acumulada pelo jogador

Acima da área de jogo, no canto superior esquerdo pode ver-se as restantes **vidas do jogador** (figura à direita). O jogador começa sempre uma partida com três vidas, únicas para os três níveis. O jogador perde uma vida sempre que a bola tocar no fundo da Área de Jogo sem ser defletida pela barra. Quando isto acontece o número de vidas é também atualizado no ecrã.

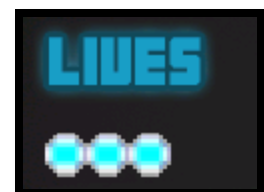


Figura 17 - vidas do jogador

Se a pontuação do jogador estiver entre as melhores, mesmo que não tenha conseguido superar todos os níveis, terá direito a inserir uma nova pontuação da tabela de melhores pontuações, sendo mostrado o ecrã seguinte.

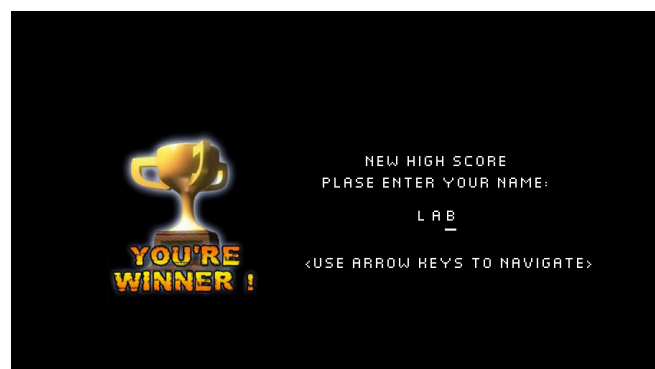


Figura 18 - ecrã de introdução do nome

O jogador deverá então utilizar o teclado para inserir o seu nome. A pontuação será automaticamente inserida na tabela de pontuações, assim que carregar na tecla *Enter*.

Quando uma partida no modo singleplayer termina o utilizador é redirecionado novamente para o menu iniciar. Deve-se notar que o estado do Botão Controlo assume sempre o valor de defeito (*Mouse*) após cada partida (independentemente do modo jogado).

O jogador pode abandonar o modo singleplayer pressionando a tecla *ESC* do teclado ou o botão direito do rato caso, se estiver a utilizar, respetivamente, o teclado ou o rato para controlar a barra.

1.4 Modo multiplayer

No **modo multiplayer**, é possível defrontarem-se dois jogadores.

Assim sendo, existem em simultâneo duas áreas de jogo iguais (*splitscreen*), independentes uma da outra, correspondendo cada uma a um jogador diferente. A mecânica de jogo no modo multiplayer é igual à do modo singleplayer, sendo cada barra controlada do mesmo modo que no modo singleplayer.

Uma novidade deste modo é a presença de um **temporizador** (em segundos) que vai decrementando no decorrer da partida. O tempo de uma partida no modo *multiplayer* é 90 segundos.

Na figura seguinte pode ver-se uma partida do Arkanix no modo multiplayer:

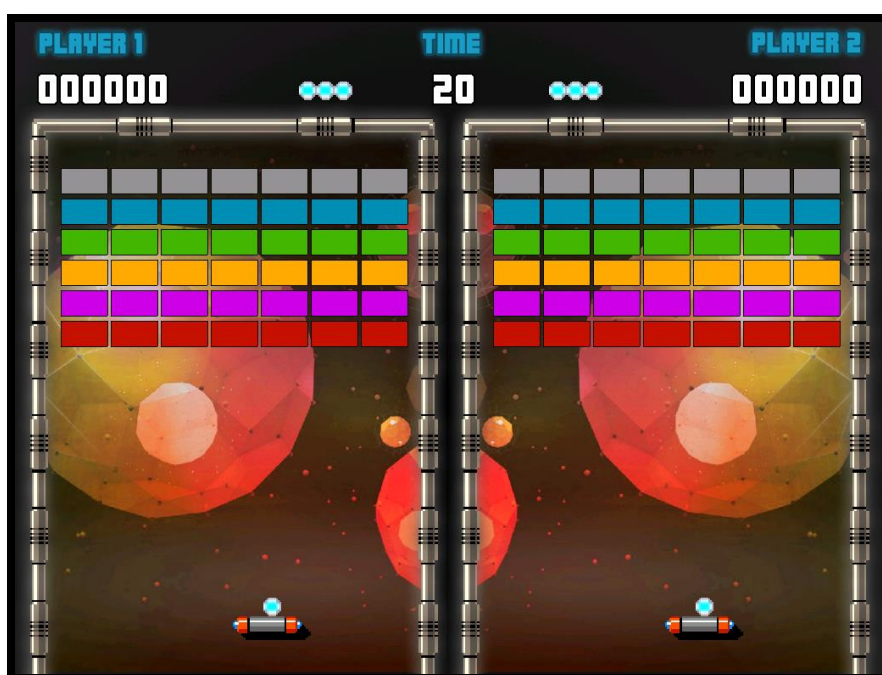


Figura 19 - partida típica no modo multiplayer

Acima de cada uma das áreas de jogo pode ver-se as pontuações de cada jogador bem como o respetivo número de vidas (em semelhança ao modo singleplayer).



Figura 20 - número de vidas e pontuação de um jogador

É possível ver entre as áreas de jogo o temporizador, constantemente atualizado e que delimita o tempo máximo até ao fim da partida.



Figura 21 - temporizador do modo multiplayer

É sagrado vencedor o jogador que no menor tempo conseguir uma maior pontuação (destruindo assim um maior número de blocos).

Sempre que se chega ao fim de uma partida surge o ecrã do vencedor.

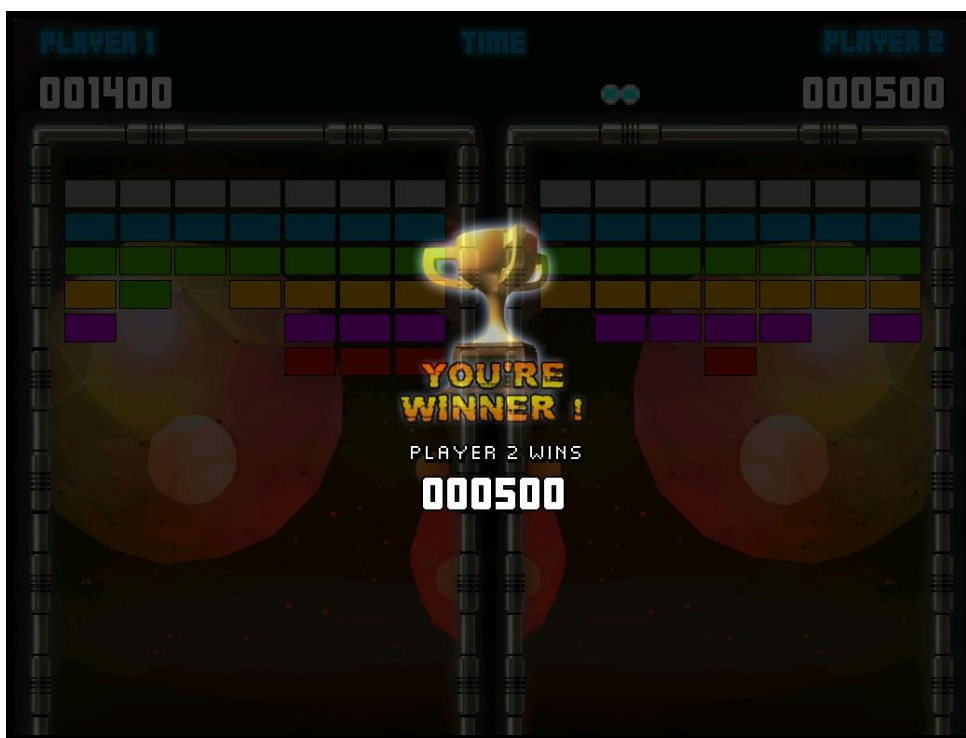


Figura 22 - ecrã do jogador vencedor

Aqui pode ver-se o jogador vencedor e a pontuação obtida por este no decorrer da partida.

Se o temporizador chegar a zero sem que haja vencedor, a partida termina e é mostrado o ecrã de empate (figura em baixo). Ainda em caso de empate, é mostrada a pontuação obtida por ambos os jogadores.

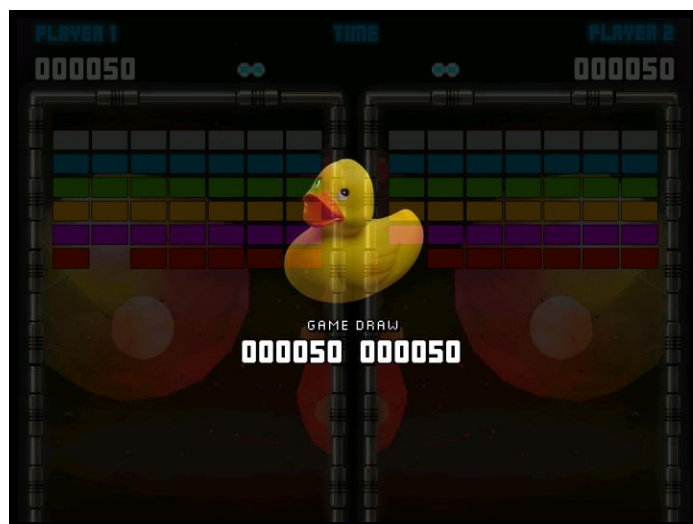


Figura 23 - ecrã mostrado numa situação de empate

Um jogador pode ainda desistir, utilizando os mesmos atalhos definidos na secção anterior. Nesse caso, o adversário sagra-se vencedor e é mostrado um ecrã a avisar do sucedido.

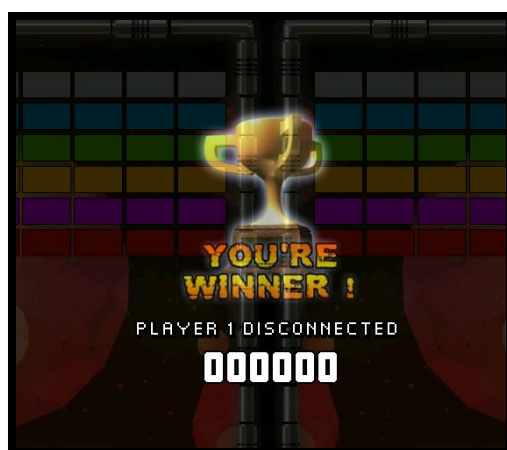


Figura 24 - ecrã mostrado em caso de desistência de um jogador

No modo multiplayer não existe tabela de pontuações pelo que no fim de cada partida os jogadores serão redirecionados para o menu inicial, após 5 segundos, sem que as melhores pontuações obtidas por eles sejam guardadas.

2. Informação do Projeto

Na implementação dos menus (inicial e modo *multiplayer*) utilizaram-se os seguintes dispositivos:

Dispositivo	Finalidade	Interrupção (S/N)
<i>Rato</i>	Navegar nos menus e interagir com os botões.	S
<i>Placa de Vídeo</i>	Desenhar no ecrã o menu, os botões e as animações.	N
<i>Timer</i>	Atualizar o ecrã.	S

Na implementação do modo singleplayer utilizaram-se os seguintes dispositivos:

Dispositivo	Finalidade	Interrupção (S/N)
<i>Rato</i>	Controlar a barra e abandonar modo, por opção do jogador.	S
<i>Teclado</i>	Controlar a barra e abandonar modo, por opção do jogador.	S
<i>Placa de Vídeo</i>	Desenhar no ecrã o nível, blocos, barra, bola, pontuação, vidas e tabela de pontuações.	N
<i>Timer</i>	Atualizar o ecrã.	S
<i>RTC</i>	Obter a data atual para a tabela de pontuações.	N

Na implementação do modo multiplayer utilizaram-se os seguintes dispositivos:

Dispositivo	Finalidade	Interrupção (S/N)
<i>Rato</i>	Controlar a barra e abandonar modo.	S
<i>Teclado</i>	Controlar a barra e abandonar modo.	S
<i>Placa de Vídeo</i>	Desenhar no ecrã o nível, blocos, barra, bola, pontuação, vidas e tabela de pontuações.	N
<i>Timer</i>	Atualizar o ecrã.	S
<i>RTC</i>	Controlar o temporizador e definir um alarme para <i>timeout</i> das abas de final de jogo.	S
<i>Porta Série</i>	Enviar/receber informação do teclado (em partidas <i>multiplayer</i> através da porta série apenas).	S

3. Organização/Estrutura do código

O código criado para o projeto está organizado nos módulos a seguir apresentados.

3.1 arkanix.c

Principal módulo do projeto.

Criou-se a *struct* **ArkanixState** para guardar informação sobre o estado atual do jogo e dos scancodes do teclado, os *enums* **GameState** e **ControlMode** que representam o estado atual e a preferência do utilizador para o botão controlo do menu inicial, e ainda a *struct* **hsscreen_t** que regista o nome do jogador atual e gera um ecrã de introdução de nome quando o jogador consegue ultrapassar uma das melhores pontuações.

- Peso relativo deste módulo: 12%
- Membro responsável: Diogo Marques

3.2 bitmap.c

Contém as funções que permitem ler e desenhar informação no ecrã contida em ficheiros do tipo *bitmap* (extensão BMP). Para que tal fosse possível, criou-se a uma *struct* denominada **bitmap_t** que guarda o cabeçalho do ficheiro e a informação de cor nele contida.

- Peso relativo deste módulo: 5%
- Membro responsável: Pedro Melo

3.3 generate.c

Contém quatro funções que geram todos os níveis do Arkanix (para ambos os modos *singleplayer* e *multiplayer*), preenchendo as áreas de jogo dos diferentes níveis com os diferentes blocos nas suas devidas posições.

- Peso relativo deste módulo: 8%
- Membro responsável: Diogo Marques, Pedro Melo

3.4 input.c

Contém as funções que permitem trabalhar com o rato e o teclado (identificar teclas e botões pressionados bem como tratamento de interrupções associadas a estes periféricos).

Para lidar com os menus criou-se uma *struct* **mouse_t** que guarda o estado do rato e dos *packets*, e os *enums* **mmode_t** e **keycode_t** para identificar o modo de operação do rato (cursor ou barra) e a tecla pressionada, respetivamente.

- Peso relativo deste módulo: 7%
- Membro responsável: Diogo Marques

3.5 level.c

Contém as funções que permitem criar, atualizar e desenhar os diferentes níveis e objetos do nível para o modo *singleplayer* do Arkanix, as respetivas áreas de jogo, a informação do jogador e os blocos.

Contém ainda funções usadas em ambos os modos de jogo que detetam as colisões da bola com as paredes da área de jogo, com a barra e com os blocos, bem como as funções que gerem o comportamento destes quando são atingidos pela bola.

Criaram-se as *structs* **Block**, **playfield_t** e **level_t** para guardar essa informação e os *enums* **game_s** (*game state*) e **pfield_t** (*playfield type*) que indicam o estado de uma partida e o tipo de área de jogo (*singleplayer*, ecrã partilhado à esquerda, ecrã partilhado à direita), respetivamente.

- Peso relativo deste módulo: 12%
- Membro responsável: Pedro Melo

3.6 menu.c

Contém as funções que permitem criar, atualizar e desenhar no ecrã o menu inicial e o menu do modo *multiplayer*, bem como os botões neles contidos e respetivas animações.

Para lidar com os menus criou-se a *struct* **menu_t** e o *enum* **menu_s** útil para registar o estado em que o menu se encontra.

- Peso relativo deste módulo: 8%
- Membro responsável: Diogo Marques, Pedro Melo

3.7 rtc.c

Contém as funções que permitem controlar o dispositivo; a escrita e leitura de pacotes e estados, tratamento de interrupções e a leitura da data atual do computador.

Criou-se a *struct* **rtc_time_t** que guarda informação sobre a data atual do computador (dia, mês, ano, horas, minutos e segundo).

- Peso relativo deste módulo: 6%
- Membro responsável: Diogo Marques

3.8 score.c

Contém as funções que permitem criar uma tabela de pontuações, atualizar e desenhar no ecrã a pontuação dos jogadores, assim como as funções utilizadas para ler e escrever de um ficheiro de pontuações e inserir, ordenar as pontuações na tabela de pontuações no modo singleplayer.

Criaram-se as *structs* **score_t** e **table_t** que guardam, respetivamente, uma entrada da tabela de pontuações e a tabela de pontuações em si.

- Peso relativo deste módulo: 8%
- Membro responsável: Diogo Marques

3.9 serial.c

Contém as funções que permitem a leitura e envio de pacotes de dados por *polling* e o tratamento de interrupções relativas à porta série.

Não foram criadas *structs* neste módulo.

- Peso relativo deste módulo: 4%
- Membro responsável: Diogo Marques

3.10 sprite.c

Contém as funções que permitem criar (*sprite_create*), eliminar (*sprite_destroy*), posicionar (*sprite_position*), animar (*sprite_update*) e desenhar uma *sprite* no ecrã (*sprite_draw*).

Criou-se a *struct* **sprite_t** que guarda informação sobre a posição de uma *sprite* (em ambos os eixos), o seu tamanho (comprimento x largura) e a sua velocidade (em ambos os eixos).

- Peso relativo deste módulo: 5%
- Membro responsável: Pedro Melo

3.11 ss.c

Contém as funções que permitem criar, eliminar, atualizar e desenhar um ecrã partilhado (*splitscreen*) no modo *multiplayer*, assim como desenhar no ecrã a informação relativa aos jogadores (vidas, pontuação, tempo restante da partida, etc...)

Contém ainda as funções que determinam o resultado de uma partida *multiplayer*, que geram os ecrãs de término da partida (vencedor, empate, desistência) e as funções necessárias ao envio de informação pela porta série (para o modo serial).

Criou-se a *struct* **ss_t** que guarda informação de todos os elementos contidos no modo ecrã partilhado.

- Peso relativo deste módulo: 10%
- Membro responsável: Diogo Marques

3.12 timer.c

Contém as funções que permitem controlar o *timer* (criação e eliminação de instâncias, tratamento de interrupções). Para guardar esta informação criou-se a *struct Timer*.

- Peso relativo deste módulo: 6%
- Membro responsável: Pedro Melo

3.13 vbe.c

Contém a função que permite obter informações sobre o modo de vídeo utilizado.

- Peso relativo deste módulo: 2%
- Membro responsável: Diogo Marques

3.14 video.c

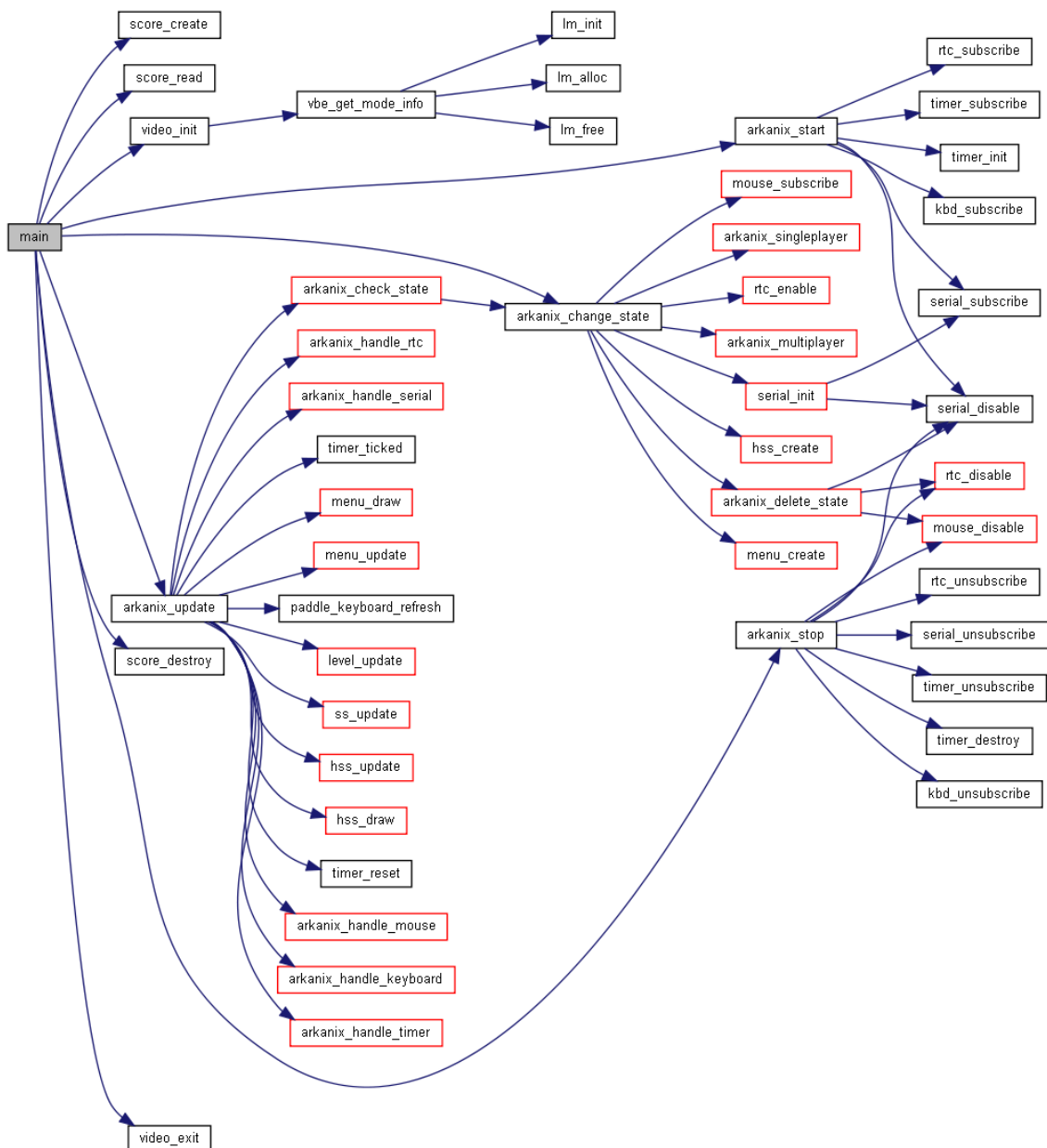
Contém as funções que permitem inicializar a placa de vídeo num dado modo gráfico e executar as diversas operações de manipulação de *pixels*, *buffers* de vídeo, sobreposição de camadas, transparência, texto e desenho.

Criaram-se as *structs* **glyph_t** e **font_t** que guardam informação sobre caracteres e fontes, respetivamente, bem como funções auxiliares para desenhar caracteres, imprimir cadeias de caracteres e números no ecrã numa dada fonte.

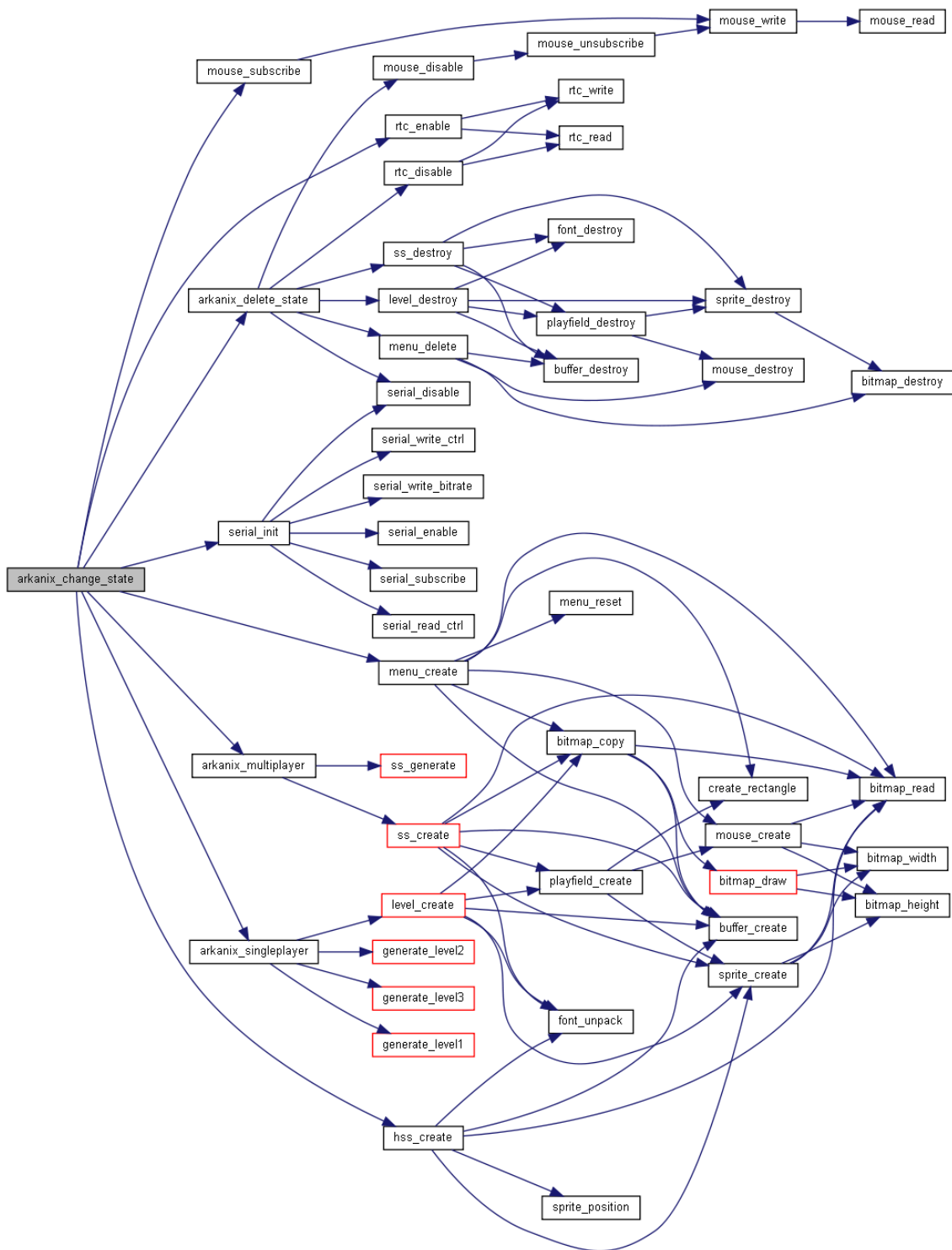
Existe ainda a *struct* **rectangle_t**, que permite a criação de retângulos dados dois pontos no ecrã (canto superior esquerdo e canto inferior direito) que são passados como argumento da função *rectangle_create*. Foram criadas duas funções de desenho para retângulos, *draw_rectangle* e *draw_rectangle_fill* que desenharam num *buffer* de vídeo um retângulo a traço e um retângulo a cheio, respetivamente.

- Peso relativo deste módulo: 7%
- Membro responsável: Diogo Marques

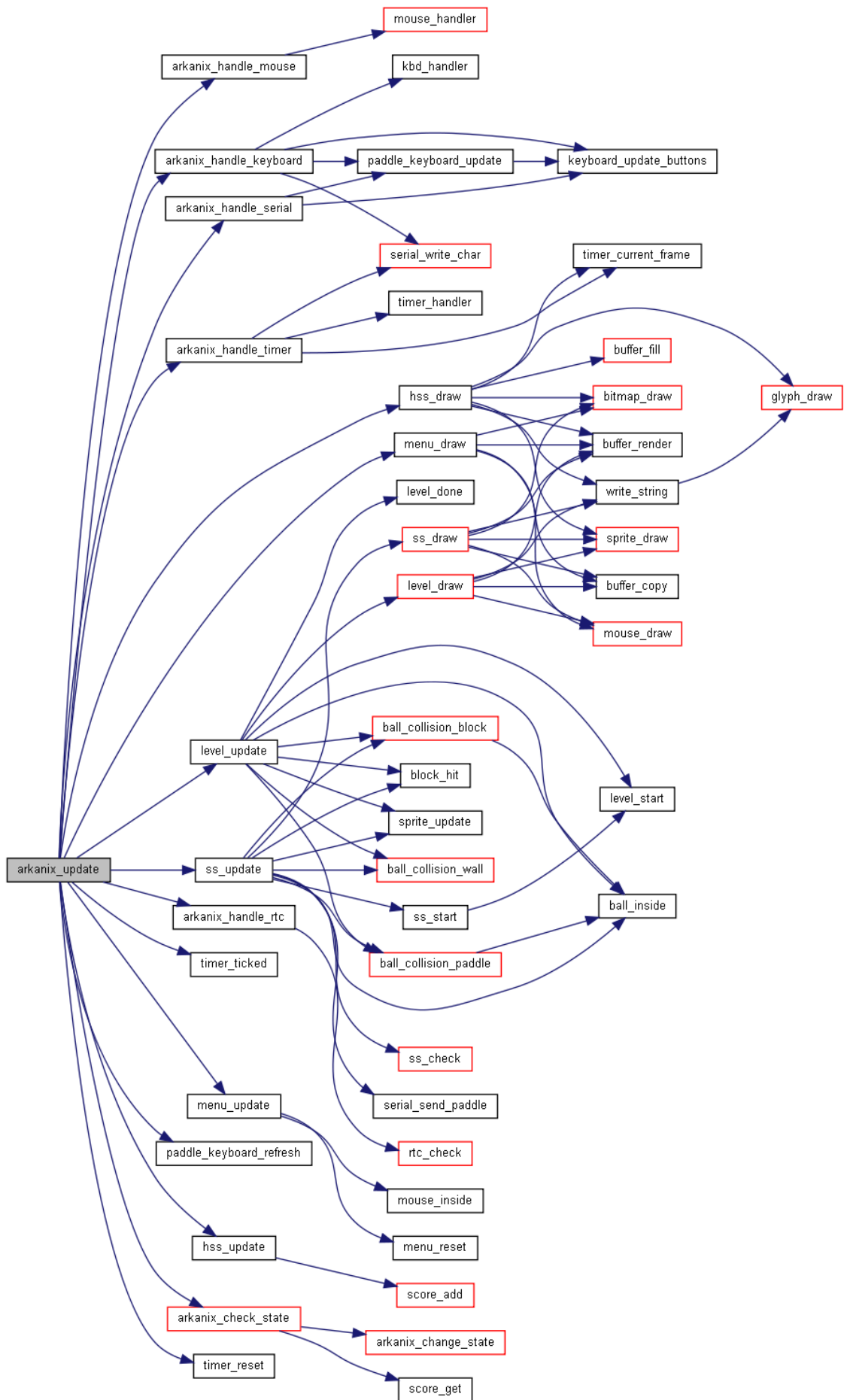
4. Grafos das chamadas de funções



A função *main* começa por chamar as funções *sef_startup()* e *sys_iopenable()*, necessárias para garantir o acesso exclusivo aos periféricos do sistema e executar rotinas em *assembly* para o controlo dos mesmos, respetivamente. Primeiro inicializa a tabela de melhores pontuações, depois a função *arkanix_update()* é chamada dentro de um ciclo que termina assim que o utilizador pressionar a tecla *Esc* ou premir o botão *Quit* no menu. Assim que abandonar o ciclo, segue-se o processo inverso ao de inicialização, a libertação de recursos: destruir a tabela de pontuações, desactivar e fazer *unsubscribe* das interrupções, destruir as *structs* criadas durante o decorrer do jogo, libertar as regiões de memória alocadas pela função *malloc*, e por fim regressar ao modo texto para retomar o estado inicial do sistema.



A função *arkanix_change_state()* é responsável pela transição entre estados (*menu*, *singleplayer*, *multiplayer*). Esta função, por sua vez, chama as funções *arkanix_singleplayer()* e *arkanix_multiplayer()* que inicializam e geram os níveis para o modo *singeplayer* e *multiplayer* respetivamente. É também responsável por ativar as interrupções para os restantes periféricos *on-demand*, isto é, quando forem necessários para o modo de jogo em questão.



5. Detalhes da implementação

5.1 Placa de vídeo

O modo gráfico VESA utilizado neste projeto foi o 0x117 (1024x768, 16 bits por pixel). As imagens utilizadas encontram-se no formato bitmap e para tal foram criadas funções para leitura e desenho de imagens no mesmo formato. A nossa implementação suporta também imagens *bitmap* com largura ímpar, uma limitação que surgiu logo no início do projeto por acidente, quando nos deparamos com erros gráficos no desenho das imagens. Dada a escassez de informação objetiva na *internet* para o nosso problema, não sabíamos se este era causado por erros no nosso código ou por limitações do formato, o que mais tarde viemos a descobrir.

Foi criada uma macro para gerar cores no formato de bits empacotados 5:6:5 (*make_rgb565*) que tem como argumentos três números inteiros de 8 bits (de 0 a 255) que representam cada componente de cor do sistema *RGB* (*vermelho*, *verde* e *azul*). Além desta, existem outra três macros (*BLUE*, *RED*, *GREEN*) que fazem o processo inverso, isto é, a partir de uma cor representada neste formato (16 bits) extraem as suas componentes de 8 bits.

As macros foram essenciais na construção dos blocos e na construção do algoritmo de *alpha blending* para conseguir efeitos de transparência entre dois *buffers* nos ecrãs, que pode ser observado nos ecrãs de final de partida do modo *multiplayer*.

Para desenhar no ecrã recorreu-se a técnicas de *double buffering* e sobreposição de camadas. Tanto o modo *singleplayer* como o módulo *multiplayer* apresentam três buffers:

- *background* - permanente, não pode ser alterado por nenhuma função do jogo, guarda apenas a imagem de fundo do nível, que é convertida do formato *bitmap* para *buffer* aquando a inicialização do nível.
- *blocks_layer* – atualizado sempre que o jogador destrói um bloco ou perde uma vida; é para este *buffer* que o *background buffer* é copiado; além do fundo, são desenhados nele os blocos existentes na área de jogo do(s) jogador(es), a pontuação e as vidas respetivas.
- *active_buffer* - é o buffer “ativo”, aquele que é atualizado com maior frequência durante o decorrer do jogo, e onde é desenhada diretamente a camada ativa do jogo (a barra e a bola de cada um dos jogadores). É o único *buffer* que é copiado para a memória gráfica no final de cada interrupção do *timer* (período de refrescamento do ecrã) - pode conter apenas o *blocks_buffer*, que por sua vez já contém o *background*, se a camada ativa não for atualizada. Numa situação típica, este *buffer* é a sobreposição dos três referidos anteriormente.

5.2 Input (rato e teclado)

A *struct mouse_t* foi implementada e utilizada com vista a guardar os *packets* de informação enviados pelo rato, calcular a posição absoluta do rato (em cada eixo) com base nos *packets* recebidos, a velocidade instantânea (em cada eixo), o estado dos botões (dois estados possíveis para cada um: *pressed* e *released*), e uma *flag* “updated” que indica se a última interrupção recebida alterou algum desses valores.

O teclado não possui uma estrutura de dados própria - foi implementado um “emulador” do rato de forma a ser possível utilizar os *scancodes* do teclado na *struct mouse_t*. Os três botões do rato foram mapeados a três teclas do teclado (esquerdo - *Enter*, meio - barra de

espaços, direito - *Esc*) e as direcções às teclas *WASD* (cima, esquerda, baixo, direita) para o caso do modo *multiplayer*, ou às teclas de setas, no caso do ecrã de introdução do nome.

5.3 RTC

No arranque do modo *multiplayer* são ativados dois tipos de interrupções do *real-time clock (RTC)*: *Alarm Interrupts (AI)* e *Update Interrupts (UI)*. Os *UI* geram uma interrupção a cada segundo e portanto fez sentido serem utilizados na contagem decrescente do temporizador do modo *multiplayer*, libertando assim recursos do *Timer 0*, cuja única tarefa é o refrescamento do ecrã.

Sempre que um jogo do modo *multiplayer* termina, é ativado um alarme de 5 segundos, escrevendo nos registos de alarme do *RTC* e é apresentada ao utilizador o ecrã de fim da partida. Esse alarme funciona como um *timeout* e gera uma interrupção de alarme (*AI*) para sinalizar ao programa que este deve abandonar o modo *multiplayer* e regressar ao menu inicial.

O *handler* é chamado no ciclo principal de interrupções sempre que o *RTC* gera uma interrupção, ficando responsável por ler o valor do *register C*, verificar qual dos dois tipos de interrupções ocorreu e de processar a mesma.

5.4 Timer

É utilizada uma implementação minimalista, que apenas se serve das interrupções do *timer 0* para atualizar valores da struct *Timer*, e portanto não altera a configuração de nenhum dos *timers* do computador.

A struct é constituída por três valores: *count*, *current_frame* e *ticked*, sendo este último uma *flag* que indica à função responsável por processar todas as interrupções do programa se o *timer* foi atualizado desde a última interrupção que recebeu. Os restantes valores são incrementados a cada interrupção do *Timer 0*, pois ambos acumulam o número de interrupções processadas, no entanto o valor de *current_frame* é colocado a “0” após 60 interrupções.

5.5 Porta série

A porta série implementada transmite e recebe dados a uma taxa de bits fixa de 2400 bauds durante todo o seu tempo de funcionamento. A transmissão é feita por varrimento (*polling*), isto é, é verificado o estado do *Transmitter Holding Register*, e caso este esteja vazio (sem dados para enviar nem receber) e tenha ocorrido uma interrupção do teclado, envia o respetivo *scancode* para o computador de destino.

Quando o utilizador entra no modo *multiplayer* em série, o jogo fica em estado de espera (é apresentada a mensagem “*WAITING FOR PLAYERS...*” até receber um pacote específico do computador ao qual está ligado. O computador do adversário, por sua vez, também envia o mesmo pacote enquanto não receber uma resposta. Os pacotes são enviados a um período constante, a cada 0.5 segundos. Assim, os jogadores não precisam de entrar simultaneamente no modo *multiplayer* em série e um dos jogadores pode entrar no jogo a qualquer momento.

A receção dos dados enviados é feita no destino através de interrupções geradas pela *UART*. O *handler* recebe a interrupção, verifica o tipo de interrupção (foram ativadas apenas interrupções do tipo *Receiver Data Available* e *Line Status Register*), e processa o *scancode* recebido. É chamada a função *paddle_keyboard_update()* para atualizar a posição da barra do adversário e as teclas especiais que pressionou (*Enter*, barra de espaços, *Esc*).

6. Avaliação









Embora tenham sido explicados os vários excertos de código em linguagem C fornecidos para serem utilizados nos laboratórios e no projeto final; estes excertos, na sua maioria "uma orientação para o que deveria ser feito" e por vezes bastante genéricos, foram abordados de um ponto de vista pouco prático.

Em consequência, deparamo-nos com a dificuldade de passar da "teoria à prática", não por não se conhecer a matéria, mas porque mesmo acompanhados dos diapositivos das aulas teóricas, dos apontamentos aí tirados e dos guiões das aulas laboratoriais não se sabia por onde começar a escrita do código (por nunca ter havido um verdadeiro contacto com a parte prática).

Assim sendo, sugerimos que o professor construa com os alunos pequenos excertos de código na aula e que os execute, podendo até mostrar propositadamente formas incorretas de escrever o código e explorar as consequências que daí advêm. A oportunidade de contactar com esta componente mais prática seria uma grande ajuda na realização dos trabalhos práticos.

A. Anexos

A1 Tabela de blocos

Bloco	Nome/cor	Pontos	Vida
	Cinza	1000	3
	Azul	300	2
	Amarelo	300	2
	Verde	100	1
	Magenta	100	1
	Vermelho	50	1
	“FEUP”	500	2
	Branco	250	1

A2 Recomendações de instalação

Foi criado um *shell script* para compilar, configurar e instalar automaticamente o Arkanix, bem como copiar todos os seus recursos e dependências (imagens e fontes) para os diretórios correspondentes. A única interação necessária do utilizador será correr o script em modo privilegiado (*root*), após fazer *checkout* do diretório de raiz do projeto:

```
chmod 777 install.sh
su
./install.sh
```