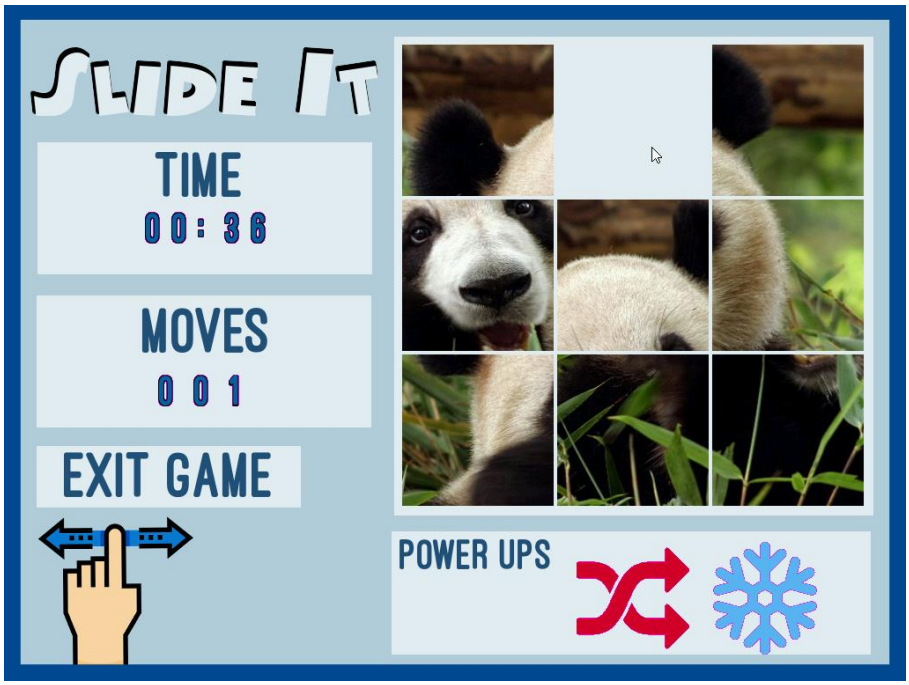
U.PORTO

FEUP FACULDADE DE ENGENHARIA
UNIVERSIDADE DO PORTO

SLIDE IT!



Trabalho Realizado por:

Pedro Fernandes - **up201603846**@fe.up.pt

Francisco Filipe - **up201604601**@fe.up.pt

Índice

1. Introdução 3
2. Manual do Utilizador 4
2.1 Menu Inicial 4
2.2 HighScores 5
2.3 Menu "Play Game" 6
2.4 Menu "Options" 7
2.5 Ecrãs de Jogo 8
2.5.1. Single Player (Regular / Time Trial / Arcade) 8
2.5.2. Multi Player 9
3. Estado do Projeto 10
3.1. Timer 10
3.2. Teclado 11
3.3. Rato 11
3.4. Placa Gráfica 11
3.5. RTC (Real Time Clock) 12
3.6. Porta de Série 12
4. Organização e Estrutura do Código 13
4.1. Bitmap 13
4.2. Game 13
4.3. Highscores 14
4.4. i8042 14
4.5. i8254 14
4.6. Init Game 15
4.7. Keyboard 14
4.8. Menu 15
4.9. Proj 16
4.10. RTC 16
4.11. Timer 16
4.12. UART 16
4.13. VBE 17
4.14. Video_GR 17
4.15. Peso de cada módulo no Projeto 17
4.16. Call Graph 18
5. Detalhes da Implementação 20
5.1. Lógica de Jogo 20
5.2. Highscores 20
5.3. Multiplayer 21
5.4. State Machine 21
5.5. Rato (cursor) 21
6. Conclusões 22
7. Apêndice 22

Introdução

No âmbito da unidade curricular Laboratório de Computadores decidimos implementar como projeto um *slide puzzle*. Este tipo de jogos baseia-se em tentar solucionar uma imagem previamente dividida em vários quadrados e baralhada. Esta imagem tem sempre uma peça que lhe é retirada para permitir que as restantes peças se possam mover ao longo do tabuleiro, visto que desta forma há sempre um local no tabuleiro que se encontra vazio. O jogo termina quando o jogador solucionar a imagem que lhe é pedida.

De modo a possibilitar a utilização de todos os periféricos (tendo sido realizados nas aulas práticas ou não) decidimos fazer algumas variações deste jogo e implementar diversos modos de jogo. Isto resultou num trabalho mais completo e uma maior aprendizagem da nossa parte tanto dos periféricos abordados nas aulas práticas como aqueles que não puderam ser alvo destas.

Manual do Utilizador

Menu Inicial

No ínicio do nosso programa, quando este é aberto pelo utilizador, é apresentado o Menu Inicial. Para percorrer este menu pode utilizar tanto o rato como o teclado (teclas 'W' e 'S' para mover para cima e para baixo, respetivamente) . Este menu contém as seguintes opções:

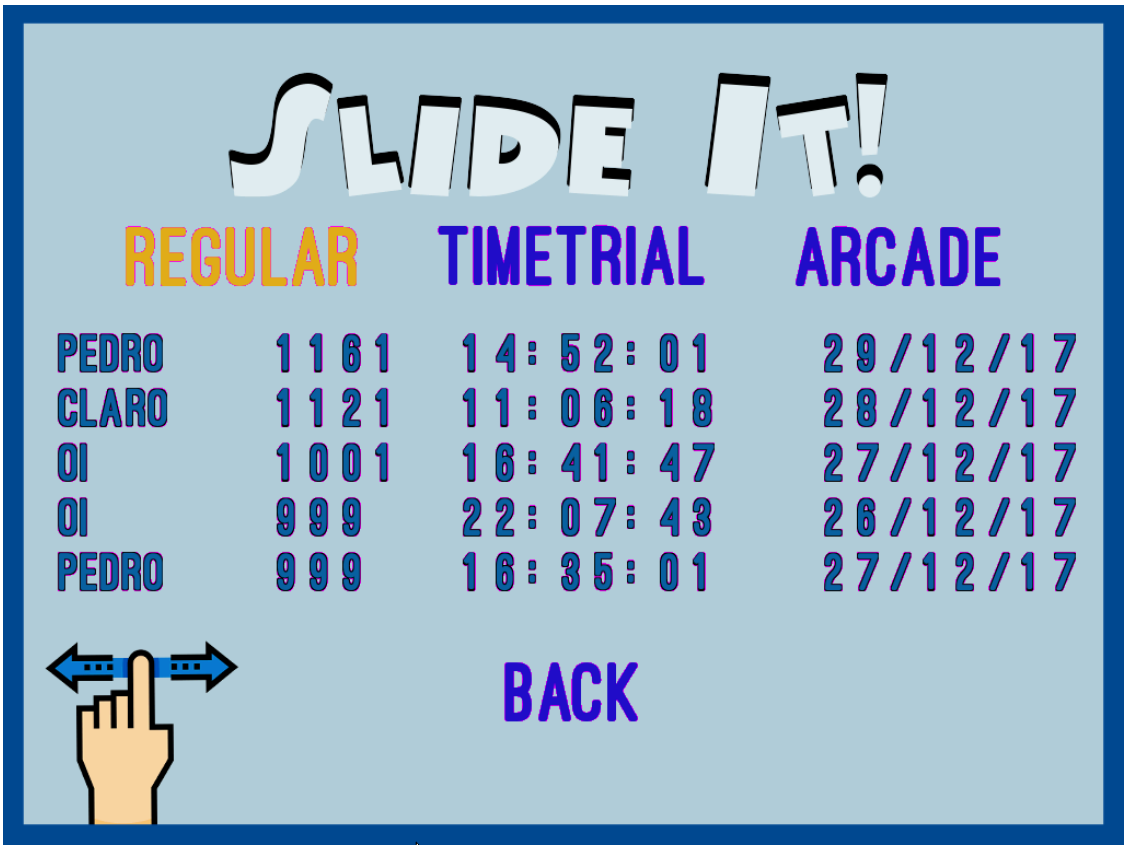
- Play Game : Ao carregar nesta opção o utilizador vai diretamente para outro ecrã onde poderá escolher o modo de jogo que deseja jogar e também alterar a sua dificuldade.
- Highscores: Ao carregar nesta opção é apresentado ao utilizador o ecrã dos highscores.
- Exit Game: Ao carregar nesta opção o jogo termina .

O jogo indica que uma opção está selecionada pelo jogador, quando essa opção tem uma cor diferente das restantes. Quando uma opção está selecionada, o jogador pode escolhê-la, ao clicar com o botão esquerdo do rato, ou então clicando na tecla ENTER do teclado.

HighScores

Se carregar na opção HighScores no Menu Inicial irá deparar-se com este ecrã. Aqui estão apresentados os Highscores dos diferentes modos (Regular Mode,Time Trial Mode e Arcade Mode). Ao passar o rato por cima de cada um dos modos de jogo são apresentados os 5 melhores resultados (caso existam) do modo correspondente. É também possível percorrer os diferentes Highscores com o teclado (teclas 'W' e 'S' para mover para a esquerda e para a direita, respetivamente). Tal como no Menu Inicial cada opção muda de cor quando selecionada.

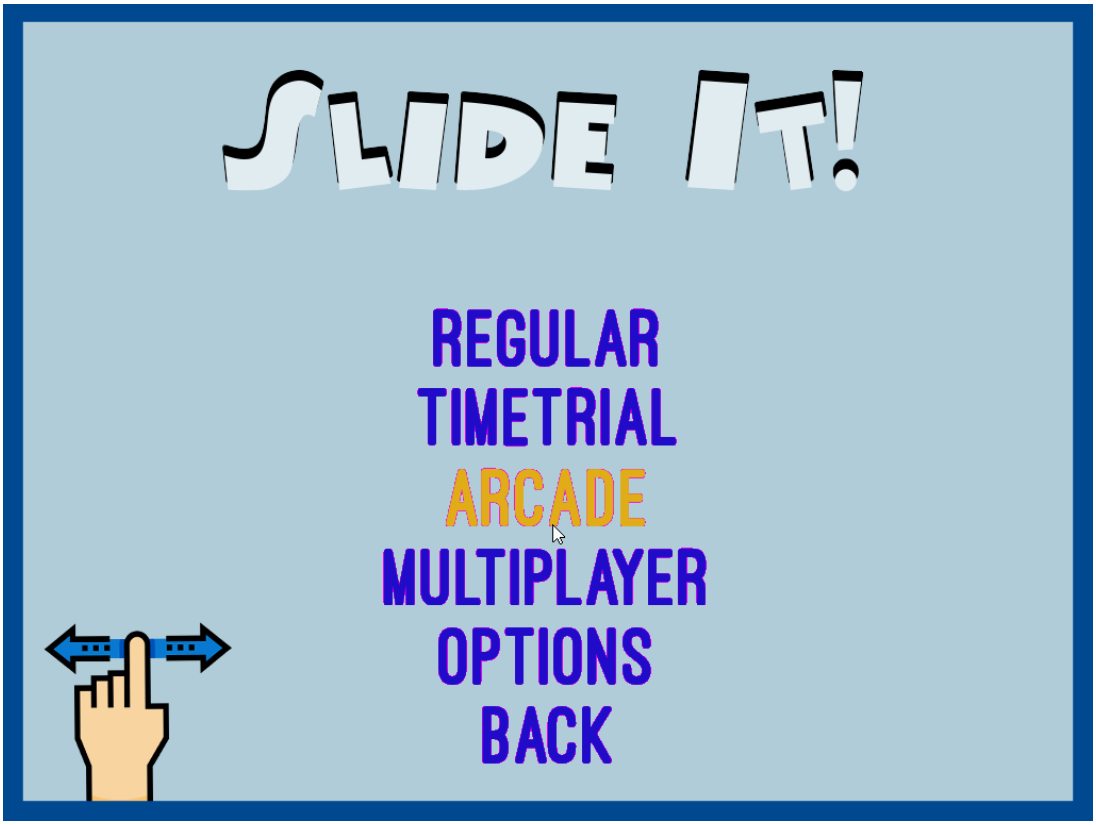
Este ecrã apresenta ainda a opção BACK que permite ao utilizador retroceder e voltar para o ecrã anterior que, neste caso, é o Menu Inicial. Pode fazê-lo carregando no botão esquerdo do rato, ou então clicando na tecla ENTER do teclado.



Menu "Play Game"

Quando o utilizador, a partir do Menu Inicial, carregar na opção "Play Game" irá ser redirecionado para este menu. Tal como no Menu Inicial tanto o rato como o teclado podem ser utilizados para percorrer as diferentes opções e quando uma opção se encontra selecionada, esta difere na cor em relação às restantes. As opções presentes neste menu são:

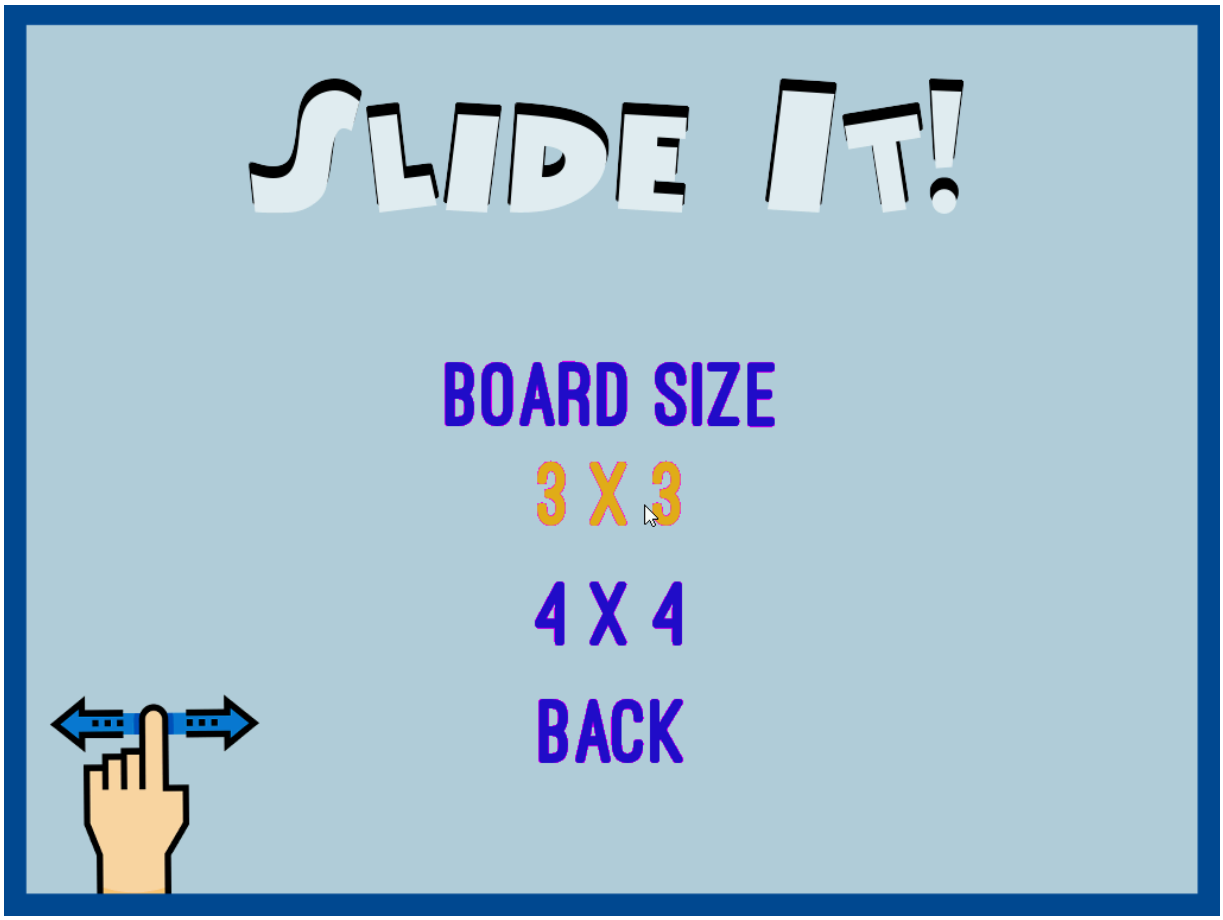
- Regular: Se o utilizador carregar nesta opção o jogo irá iniciar no modo Normal.
- Time Trial: Se o utilizador carregar nesta opção o jogo irá iniciar no modo Time Trial.
- Arcade: Se o utilizador carregar nesta opção o jogo irá iniciar no modo Arcade.
- Multiplayer: Se o utilizador carregar nesta opção o jogo irá iniciar no modo Multiplayer
- Options: Se o utilizador carregar nesta opção será direcionado para um Menu de Opções onde poderá mudar a dificuldade do puzzle.
- Back: Ao carregar nesta opção o utilizador irá retroceder e voltar ao ecrã anterior que, neste caso, é o Menu Inicial.



Menu "Options"

Caso o utilizador, a partir do Menu "Play Game", carregue na opção "Options" irá ser redirecionado para este ecrã. Neste ecrã apenas se pode alterar a dificuldade do Puzzle, mudando o o tamanho do tabuleiro. Ao carregar na dificuldade desejada, aparentemente, nada acontece, apenas é mudada a dificuldade que não tem qualquer influência na parte gráfica do programa.

Este ecrã possui um botão BACK que permite retroceder e voltar para o menu anterior.



Ecrãs de Jogo

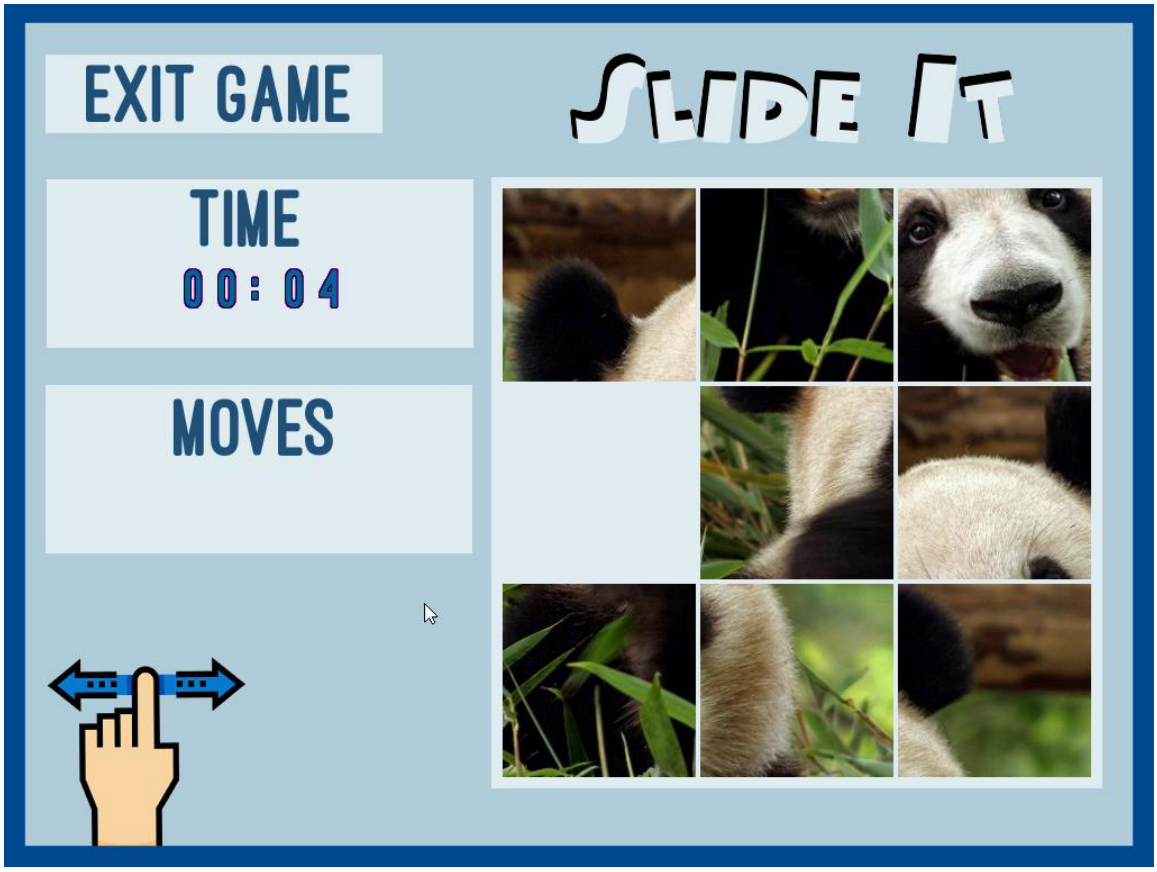
Single Player (Regular / Time Trial / Arcade)

Em todos os modos os modos de jogo single player o ecrã apresentado é o mesmo (abaixo apresentado). Podemos dividir o ecrã em duas secções. Uma mais à esquerda e outra mais à direita.

A secção mais à esquerda do ecrã é onde se encontra toda a informação durante o jogo. Nesta secção é apresentado ao utilizador:

- Tempo: Dependendo do modo em que o utilizador se encontra o tempo irá contar de forma ascendente (Regular e Arcade) ou descendente (Time trial).
- Movimentos: Apenas indica o número de movimentos que o utilizador vai realizando.
- Botão "Exit Game": Este botão apenas sai do modo de jogo atual e retorna ao Menu Inicial.

A secção mais à direita contém o jogo em si, onde se situa o tabuleiro. Aqui o utilizador pode utilizar o rato ou então o teclado para executar o movimento das peças. Utilizando o rato, basta apenas clicar na peça que deseja mover e caso esta possua um movimento válido, esse movimento será executado. No caso do teclado são utilizadas as teclas A, W,S e D para mover uma peça para a esquerda, cima, baixo e direita respetivamente.



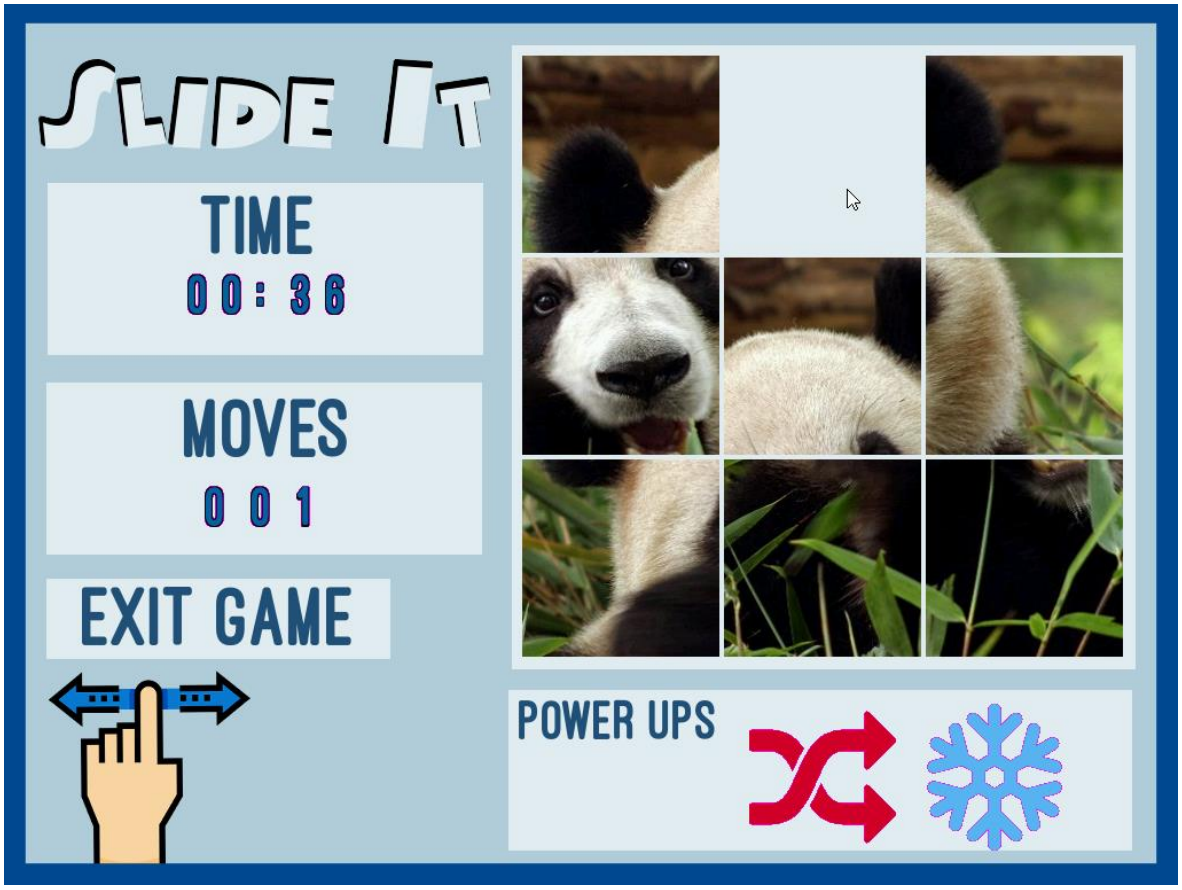
Multi Player

O ecrã de jogo em modo MultiPlayer é muito semelhante ao ecrã de jogo ao modo Single Player. Apenas são adicionados power ups na secção mais à direita do ecrã.

Para tornar o jogo mais interessante e equilibrado no modo MultiPlayer e também de modo a possibilitar a implementação do Serial Port foi decidida a utilização de power ups. Quando um jogador se encontra prestes a terminar um puzzle (dois terços das peças já se encontram nas posições corretas) é dada a possibilidade ao outro jogador de utiliar os power ups para prejudicar o jogador que estiver quase a terminar. Os power ups implementados neste programa foram:

- SHUFFLE: Quando é ativado volta a baralhar o tabuleiro do adversário de quem ativou.

- FREEZE: Quando é ativado congela o jogo do adversário durante 10 segundos impossibilitando a realização de qualquer jogada.



Estado do Projeto

Na implementação do nosso projeto estão presentes todos os periféricos no âmbito desta unidade curricular. Não só aqueles abordados nas aulas práticas mas também os dois periféricos extra estudados apenas nas aulas teóricas .

Periférico	Função	Interrupts
Timer	Contar o tempo, controlar frame rate em caso de uso de animações	SIM
Teclado	Navegar os menus, mover as peças de puzzle e para entrada de texto	SIM
Rato	Navegação nos menus e movimento de peças de puzzle no jogo	SIM
Placa Gráfica	Exibição dos menus e do jogo	NÃO
RTC	Guardar data dos 5 melhores jogos de cada modo de jogo	NÃO
Serial Port	Envio/receção de carateres para possibilitar modo multijogador	SIM

Timer

Neste projeto o timer é utilizado para cotrolar o tempo em diversas ocasiões, sendo elas:

- Contabilizar o tempo que o utilizador demora a solucionar um puzzle.
- Contabilizar o tempo do power up FREEZE a partir do momento que é ativado até ao seu limite de 10 segundos.
- Contabilizar o tempo em que, a seguir à ativação de um power up, é impossível proceder à ativação de outro power up (O nosso intervalo é de 1 minuto, assim um utilizador só pode ativar power ups de 1 em 1 minuto e caso o outro utilizador se encontre na iminência de terminar o seu puzzle).

Também está presente a utilização deste periférico no momento de baralhar um tabuleiro, mais precisamente na parte gráfica de baralhar em que é usada uma animação. O timer está presente para controlar o frame rate das peças que se movem durante esta função.

Ficheiros onde está presente: timer.c timer.h i8254.h

Teclado

O teclado é utilizado ao longo de todo o programa. Este periférico está presente em:

- Menus: Para facilitar a navegação dentro de todos os menus do programa.
- Jogo: Na realização de jogadas para a resolução de um puzzle.
- Highscores: Na escrita do nome de utilizador caso este mereça entrar na tabela de Highscores.

Ficheiros onde está presente: keyboard.c keyboard.h i8042.h

Rato

O rato é utilizado tanto na navegação de menus como durante o jogo em si.

Nos menus este periférico apenas serve para navegar por entre as diferentes opções de cada menu e, clicando com o botão esquerdo do rato, selecionar a opção desejada.

Já durante o jogo o rato pode servir para mover as peças desejadas carregando com o botão esquerdo deste na peça que que se pretende mover (caso esta tenha um movimento válido) ou então para selecionar power ups caso esteja em modo multiplayer ou sair para o menu inicial.

Ficheiros onde está presente: mouse.c mouse.h i8042.h

Placa Gráfica

A placa gráfica, como era de esperar, está presente ao longo de todo o programa.

Esta foi usada no modo 0x117, com uma resolução de 1024x768 num formato RGB 5-6-5 (16 bits) com um total de 64k cores disponíveis.

Também foi implementado double buffer. No entanto este apenas se encontra implementado nos menus e não durante o decorrer do jogo em si. Esta decisão foi tomada devido ao caráter estático do nosso jogo . Visto que não é um jogo onde ocorram muitos eventos ao mesmo tempo, não nos pareceu vantajosa a utilização de double buffer durante o jogo.

Ficheiros onde está presente: video_gr.c video_gr.h vbe.c vbe.h bitmap.c bitmap.h pixmap.h read_xpm.c read_xpm.h sprite.h

RTC (Real Time Clock)

Este periférico apenas é utilizado para guardar a data dos 5 melhores resultados de cada modo de jogo que estarão presentes nos highscores.

Apesar de não serem usadas interrupções foi implementado um interrupt handler em assembly para este periférico que vai buscar os valores do RTC aos registos respetivos.

Ficheiros onde está presente: rtc.c rtc.h rtc_ih.S rtc_macros.h

Porta de Série

A porta de série é utilizada para relatar eventos como:

- Ativar um power up.
- Desistir do jogo.
- O adversário estar perto de terminar.
- Verificar se os dois jogadores estão preparados para jogar.

A cada interupção é verificado se um destes eventos aconteceu e caso isso seja verdade são executadas as ações caraterísticas de cada evento.

Ficheiros onde está presente: uart.c uart.h

Organização e Estrutura do Código

Bitmap

O crédito deste módulo vai maioritariamente para o Henrique Ferrolho, uma vez que o código original foi retirado do seu blog (http://difusal.blogspot.pt/2014/09/minixtutorial-8-loading-bmp-images.html). As únicas alterações feitas a este módulo foram:

- Double buffering na função drawBitmap: Visto que nem sempre é usado double buffering no nosso programa, decidimos criar um novo parâmetro para esta função que identifica a necessidade de usar double buffering ou não.

- Nova struct no header file deste módulo (Puzzle Bitmap): Esta struct foi implementada de maneira a facilitar a verificação da solução cada vez que o jogador executa uma jogada. Esta possui um bitmap associado e um inteiro indicando a posição correta desta no tabuleiro. Assim facilmente se pode verificar se o jogador já conseguiu solucionar o puzzle comparando a posição presente nessa struct com a posição em que ele se encontra atualmente.

Módulo Alterado por: Pedro Fernandes

Game

Este módulo está relacionado com toda a lógica de jogo e a execução desta. Aqui estão presentes as funções responsáveis pelo jogo em si e pelos seus diferentes modos. Torna-se assim m dos módulos fundamentais deste projeto.

Aqui são implementados os ciclos de interrupções necessários para cada modo de jogo , tratando cada um deles as interrupções necessárias fazendo chamadas aos handlers correspondentes e também a funções auxiliares dentro deste módulo. É neste módulo que se encontram funções como:

- Baralhar o tabuleiro (tanto o algoritmo como a parte gráfica que mostra no ecrã as peças a serem baralhadas).
- Trocar duas peças após uma jogada,
- Verificar se um movimento é válido (quer através do teclado ou do rato).
- Verificar a solução.
- Implementação dos diferentes modos de jogo (Regular, Time Trial, Arcade e Multiplayer).
- Funções para desenhar no ecrã mensagens apropriadas a certos eventos (ganhar, perder, desistir, etc).

Módulo desenvolvido por: Francisco Filipe e Pedro Fernandes (50% / 50%)

Highscores

Este módulo é responsável por tudo aquilo que se relaciona com o ecrã Highscores. É tratada a leitura e escrita dos highscores em ficheiros .txt de modo a poder guardar a informação de uma sessão de jogo para mais tarde poder ser utilizada noutra sessão. As funções "draw" de todas as compontentes dos highscores também estão aqui presentes.

Para facilitar na organização da informação de cada highscore foram criadas algumas structs:

- Time: contém os componentes necessários para caraterizar o tempo em que um highscore foi feito (dia, mês, ano, hora, minutos e segundos).

- Highscore: guarda a informação de um Highscore (a data em que foi estabelecido – através da struct Time, o nome de utilizador e o seu tamanho, a pontuação eo modo de jogo).

Para guardar todos os highscores criámos outra struct chamada Highcores que contém 3 arrays de Highscore, cada um referente a um modo de jogo diferente. Também nesta struct está declarado o tamanho de cada array podendo ser no máximo 5 visto que cada modo de jogo só pode ter no máximo 5 highscores na tabela,

Módulo desenvolvido por: Pedro Fernandes.

i8042

Módulo que contém as macros necessárias para a correta utilização do controller do teclado e rato. Importado das aulas práticas e adptado ao projeto.

Módulo desenvolvido por: Francisco Filipe

i8254

Módulo que contém as macros necessárias para a correta utilização do timer. Importado das aulas práticas e adptado ao projeto.

Módulo desenvolvido por: Pedro Fernandes

Keyboard

Código importado das aulas práticas. Apenas se retiraram as funções que achámos desnecessárias.

Init Game

Aqui estão implementadas todas as funções de load de bitmaps assim como as funções de inicialização da maioria das nossas estruturas de dados. Neste módulo estão declaradas algumas structs como:

- GameBitmaps: struct de bitmaps com os bitmaps necessários ao jogo
- PuzzlePiece: struct que contém informação sobre uma peça do tabuleiro (Bitmap associado a essa peça, coordenadas e largura da peça).
- Board: struct que contém informações sobre o tabuleiro de jogo (cordenadas iniciais, tamanho, número de peças e um array de peças de puzzle).

Módulo desenvolvido por: Francisco Filipe e Pedro Fernandes (50% / 50%).

Mouse

Código maioritariamente importado das aulas práticas. Foram retiradas as funções que achámos desnecessárias e adicionadas outras mais importantes para o projeto como por exemplo verificar se o rato se encontra dentro das coordenadas limites da janela gráfica e funções que retornam cada uma das coordenadas atuais do rato. Também foi implementado o interrrupt handler deste periférico em assembly.

Módulo desenvolvido por: Francisco Filipe e Pedro Fernandes (50% / 50%)

Menu

Este módulo é reponsável por toda a navegação no menu, Foi implementada uma máquina de estados para facilitar esta navegação. Este módulo é responsável por verificar eventos provenientes do rato ou teclado e atualizar a máquina de estados devidamente, Também são implementadas todas as funções "draw" relacionadas com o menu. Para facilitar a organização da informação foram criadas as seguintes structs:

- MenuItem: struct com a informação sobre uma opção de um menu (dois bitmaps – um para quando a opção não se encontra selecionada e outra para quando se encontra, as coordenadas x e y desta opção, e o seu tamanho em altura e largura).
- Menu: struct com a informação sobre um determinado menu (ID do menu, inteiro indicando qual a opção do menu que se encontra atualmente selecionada,número de opções desse menu, e um array de MenuItem com as opções desse menu).

Módulo desenvolvido por: Francisco Filipe e Pedro Fernandes (50% / 50%)

Proj

Este módulo é onde se encontra o main. A sua função é apenas a de inicializar o jogo. Inicializa o modo de janelas gráfica e faz a chamada à função que inicializa o jogo em si. Também é este módulo que no final do programa vai sair do modo janela gráfica e retornar ao modo de texto.

Módulo desenvolvido por: Francisco Filipe

RTC

É onde se faz a leitura da informação proveniente dos registos relativos ao RTC e se colocam nas respetivas variáveis. Não foram usadas interrupts neste periférico mas foi implementado em assembly um handler. É ele que lê dos respetivos registos e coloca no devido lugar a informação, atuando assim como um interrupt handler.

Módulo desenvolvido por: Pedro Fernandes

Timer

Código importado das aulas práticas. Apenas se retiraram as funções que achámos desnecessárias.

UART

Este módulo é responsável pela pela comunicação entre dois computadores. Neste módulo estão presentes funções responsáveis pela configuração do serial port. O nosso serial port foi configurado da seguinte maneira:

- 8 bits per char.
- 2 stop bits.
- Even parity

Possui também funções para manipular a porta de série, Leitura da informação recebida, enviar informação através da porta de série, verificar o estado do buffer de entrada e de saída e as funções subscribe e unsubscribe. O handler deste módulo foi implementado em assembly.

Módulo desenvolvido por: Francisco Filipe e Pedro Fernandes (50% / 50%)

VBE

Código importado das aulas práticas. Apenas se retiraram as funções que achámos desnecessárias.

Video_GR

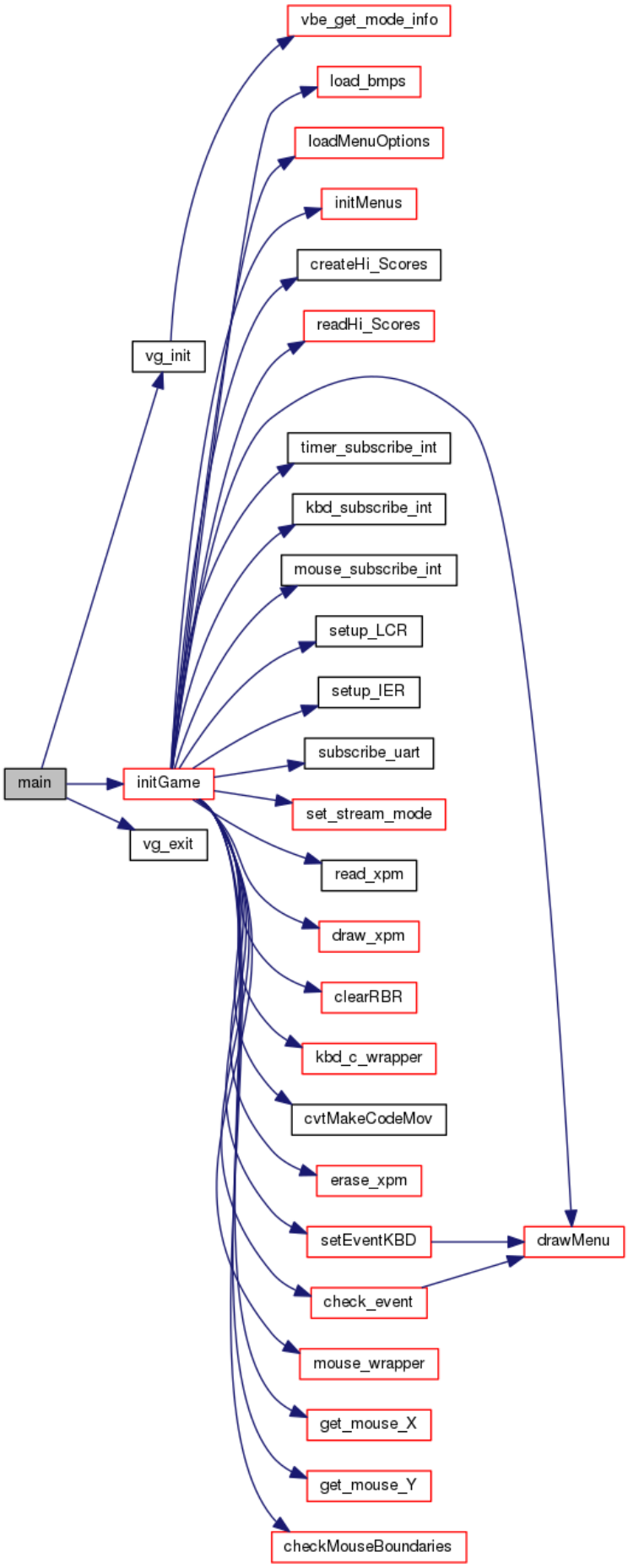
Código maioritariamente importado das aulas práticas. Neste módulo apenas foi implementado o double buffering para obter uma animação mais fluida e sem flickering.

Módulo desenvolvido por: Francisco Filipe e Pedro Fernandes (50% / 50%)

Peso de cada módulo no Projeto

MÓDULO	PESO NO PROJETO
BITMAP	6%
GAME	12%
HIGHSCORES	4%
I8042	3%
I8254	3%
INIT_GAME	6%
KEYBOARD	8%
MOUSE	8%
MENU	12%
PROJ	6%
RTC	5%
TIMER	8%
UART	7%
VBE	4%
VIDEO_GR	8%

Call Graph


vbe_get_mode_info
load_bmps
loadMenuOptions
initMenus
createHi_Scores
readHi_Scores
vg_init
timer_subscribe_int
kbd_subscribe_int
mouse_subscribe_int
setup_LCR
setup_IER
subscribe_uart
main
initGame
set_stream_mode
vg_exit
read_xpm
draw_xpm
clearRBR
kbd_c_wrapper
cvtMakeCodeMov
erase_xpm
setEventKBD
drawMenu
check_event
mouse_wrapper
get_mouse_X
get_mouse_Y
checkMouseBoundaries

Devido ao elevado número de funções, não foi possível colocar um gráfico adequado, enquanto mostrando todas as funções do programa.

Desta forma, é possível visualizar na documentação gerada pelo doxygen os gráficos de chamada e do chamador para todas as funções (caso existam). De seguida, encontra-se uma breve descrição das principais funções do programa, incluindo as que chamam a função driver_receive().

init_game: Inicializa a parte principal do jogo.

check_highscore: Recebe input do utilizador e adiciona novos objetos de highscore ao array correspondente

regular_timetrial: Começa um novo jogo quer no modo normal ou no modo time trial dependendo dos parâmetros.

arcade_mode: Começa um novo jogo no modo arcade.

multi_player_game: Começa um novo jogo no modo multiplayer.

players_ready: Espera que os dois jogadoes estejam preparados para jogar.

animate_pieces: Troca uma dada peça com a peça vazia, movendo os seus bitmaps a uma velocidade adequada.

show_full_image: Mostra a imagem completa do puzzle que vai serjogadodurante 3 segundos.

init_board: Initializes Inicializa o tabuleiro de acordo com o número de peças e com as peças.

Shuffle: Baralha a imagem presente no tabuleiro.

check_solution: Verifica se o utilizador completou o puzzle.

check_movement: Verifica se é possível trocar duas peças e nesse caso troca tanto em memória como no ecrã,com animação ou não.

check_mouse_movement: Verifica se é possível trocar duas peças e nesse caso troca tanto em memória como no ecrã,com animação ou não.

check_event: Recebe um evento e atua de acordo com este.

set_event_mouse: Recebe um evento do rato que vai ser usado na função check_event,

set_event_kbd: Recebe um evento do teclado que vai ser usado na função check_event.

Detalhes da Implementação

Lógica de Jogo

Na implementação deste projeto, à medida que íamos avançando no seu desenvolvimento íamo-nos apercebendo de alguns pormenores em relação a este jogo. Um dos quais foi a forma de baralhar(**shuffle(Board *board)**).

Inicialmente pensávamos que as peças podiam ser colocadas todas de forma aleatória no tabuleiro e que este seria sempre solucionável. No entanto verificámos que isso não era verdade. Deste modo encarámos a função de baralhar da seguinte forma: partindo de um tabuleiro em que as peças estão todas inicialmente na sua posição correta, íamos permutando a peça vazia com as suas peças vizinhas de forma aleatória. Assim, se partirmos do tabuleiro inicial garantimos que qualquer que seja a sequência de peças no final de baralhar o tabuleiro, este é sempre solucionável.

No entanto existiam ainda outros problemas relacionados com a forma de baralhar. Outro dos quais era a garantia de que no final de executar a função de baralhar o tabuleiro, este não se encontrasse resolvido ou quase resolvido. Para isso apenas foi feita uma verificação de quantas peças se encontravam na posição correta no final de baralhar. Esta verificação era dada pela função **check_solution(Board *board)** que retorna o número de peças na posição correta. Se o número de peças que se encontrasse na posição correta fosse maior que o número de peças máximo estabelecido, então executava-se a função de baralhar as vezes necessárias até que este critério fosse cumprido.

Highscores

Os Highscores são calculados tendo em conta a dificuldade em que o utilizador jogou o puzzle, o tempo que demorou a resolver e o número de movimentos que necessitou(**unsigned int calc_score(unsigned int game_mode, unsigned int time, unsigned int difficulty, unsigned int nMovements)**).

A fórmula é a seguinte:

Pontos por dificuldade: Dificuldade fácil (tabuleiro 3x3) – 500 pontos.

Dificuldade difícil (tabuleiro 4x4) – 1000 pontos.

Pontos por tempo: Modo Regular/Arcade :pontos = $\frac{1}{tempo}$

Modo Time Trial : pontos = tempo.

Pontos por movimentos: pontos = $\frac{1}{tempo}$

Pontos totais: Pontos por dificuldade + Pontos por tempo + Pontos por movimentos.

Multiplayer

Para implementar o modo multiplayer (onde é implementada a porta de série) decidimos usar a implementação presente nos slides - **state-machine replication .** Esta tem por base a seguinte ideia: o jogo é executado em dois computadores diferentes. Ou seja, cada computador vai executar os mesmos pedaços de código, logo é necessário que cada um destes terminais processe a informação dos dois computadores conetados entre si.

Esta forma pareceu-nos ser a mais indicada pois uma vez que em modo multiplayer o ecrã de jogo será o mesmo para cada jogador, a porta de série apenas necessita de tratar dos eventos em si e transferir os eventos de um terminal para o outro. Cada terminal trata dos eventos localmente.

State Machine

Foi decidido também realizar a implementação de uma máquina de estados para facilitar a navegação no menu. Esta máquina encontra-se implementada no módulo Menu (menu.c e menu.h). Tal como em qualquer máquina de estados, a ideia por detrás desta máquina era: tendo um estado e um determinado evento, transitar para um outro estado. Isto facilita a navegação nos menus de jogo pois a cada clique do rato ou a cada pressionar da tecla ENTER apenas é necessário verificar o estado, o evento pretendido e gerar os comportamentos necessários relativos ao novo estado ou à transição para o novo estado (ex. desenhar um novo menu, mudar de opção dentro do mesmo menu,etc).

Funções mais importantes: **check_event, set_event_kbd e set_event_mouse.**

Rato (cursor)

Para a implementação do cursor do rato no nosso jogo, a ideia inicial era a utilização de xpm e assim o foi feito. Visto que foi um dos primeiros periféricos a implementar e na altura desconhecíamos da possibilidade de utilização de bitmaps, a parte gráfica deste periférico foi implementada através de pixmaps. Como a ideia já estava pensada para pixmaps acabámos por deixar esta implementação.

A cada interrupt do rato este é apagado do ecrã e redesenhado noutras coordenadas. No entanto o significado de apagar não é o de eliminar o que se encontrava naquelas coordenadas, mas sim repôr o que havia lá estado antes. Para isso, e visto que apenas precisámos de pixmaps para o cursor do rato, adaptou-se a função de desenhar xpms para guardar os pixeis que estavam presentes antes de desenhar o rato num array, e só depois desenhar o cursor em si. Para repôr estes pixeis implementou-se a função erase_xpm que volta a substituir o cursor pelos valores dos pixeis antes presentes.

Conclusões

Em relação ao desenvolvimento deste projeto, achámos ser um projeto trabalhoso, mas enriquecedor e uma boa forma de colocar em prática os conhecimentos aprendidos ao longo das aulas práticas.

Ainda em relação ao projeto gostaríamos de salientar que ambos os elementos do grupo se empenharam igualmente para que o trabalho fosse concluído com sucesso e a carga de trabalho atribuída a cada um foi o mais uniforme possível.

Em relação à unidade curricular, achamos que a ideia desta é algo importante. O contacto com periféricos nesta altura pode trazer grandes vantagens no futuro uma vez que nos ajuda a perceber de uma forma mais aprofundada e de mais baixo nível o contacto destes com o computador e as trocas de informação realizadas entre estes. Outro ponto positivo foi o facto de desenvolvermos as nossas capacidades de programação em C, uma linguagem de relativamente baixo nível e que no nosso ponto de vista foi mais difícil de interiorizar que C++. No entanto, o que nos apercebemos ao longo da unidade curricular foi que, por se tratar de um tema com o qual nunca antes tínhamos estado em contacto o choque inicial foi grande e mesmo no decorrer desta unidade curricular sentimos algumas dificuldades na compreensão de certos temas. A existência de algumas bases do funcionmento dos periféricos numa outra unidade curricular antes estudada poderia ter sido vantajosa e tornado esta unidade curricular menos “agressiva”.

Apêndice

Além dos procedimentos normais, para correr o nosso programa é necessário:

- Navegar até à pasta **proj/scripts**.
- Executar o comando **./install.sh.**

Pode ser também necessário dar permissões ao script com: **chmod u+x install.sh**.