U.PORTO
FEUP FACULDADE DE ENGENHARIA
UNIVERSIDADE DO PORTO

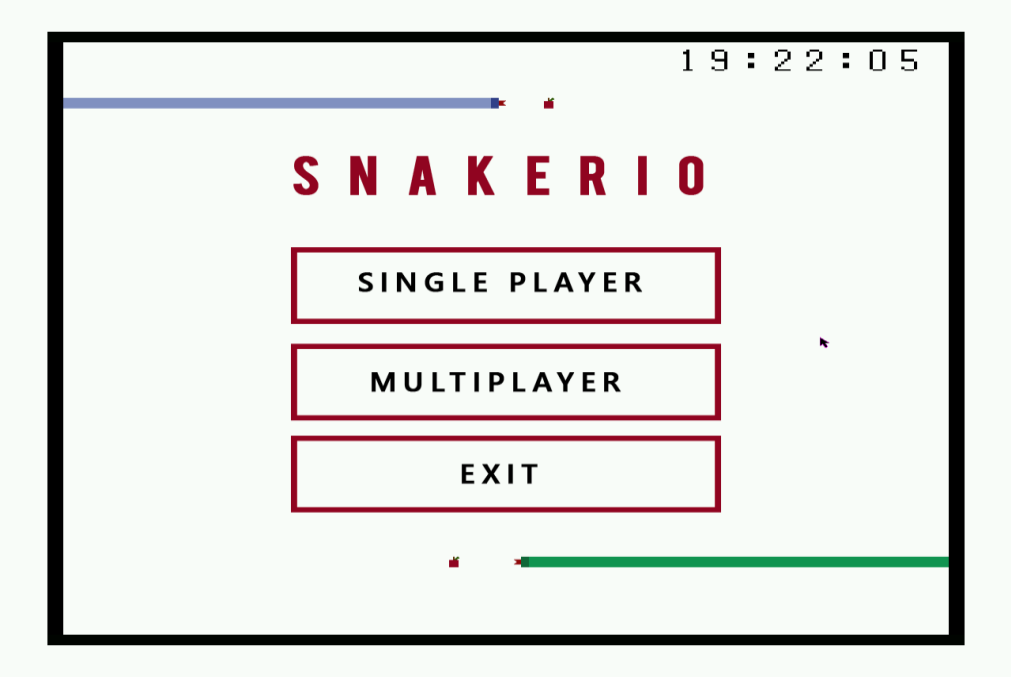
Laboratório de Computadores 2016/2017
Turma 4 - Grupo 14

Índice

I. Instruções de Utilização

Menu Inicial

Quando o jogo inicia, é mostrado um menu, no qual o utilizador pode selecionar uma das seguintes opções:



- *SinglePlayer*: o utilizador é redirecionado para um menu onde pode escolher a cabeça e o corpo da cobra (snake);

- *Multiplayer:* o utilizador é redirecionado para um submenu no qual pode escolher um de 2 tipos de jogo:
  - *Snake Gladiator*: cada um dos 2 jogadores joga com uma cobra;

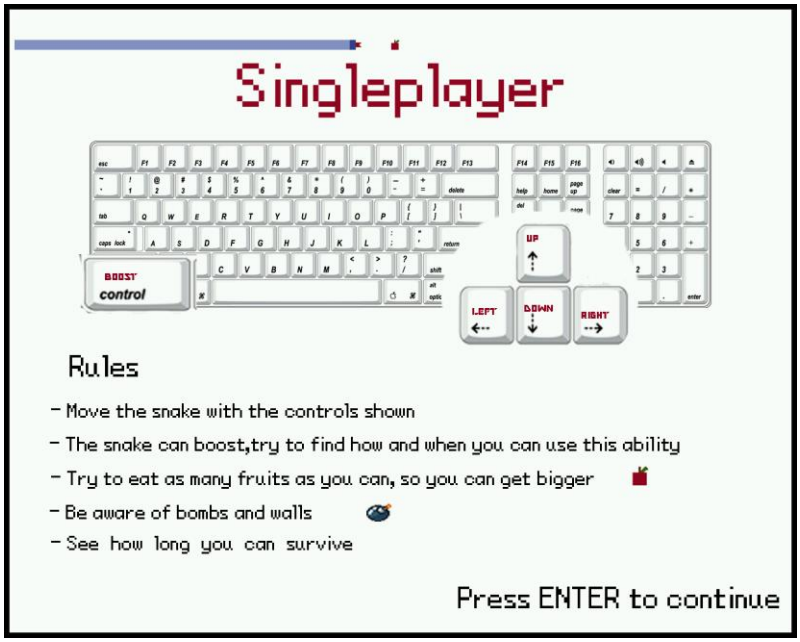
  - *Snake and Mouse*: um jogador joga com uma cobra e o outro jogador utiliza o rato para colocar maçãs ou bombas para o seu adversário.

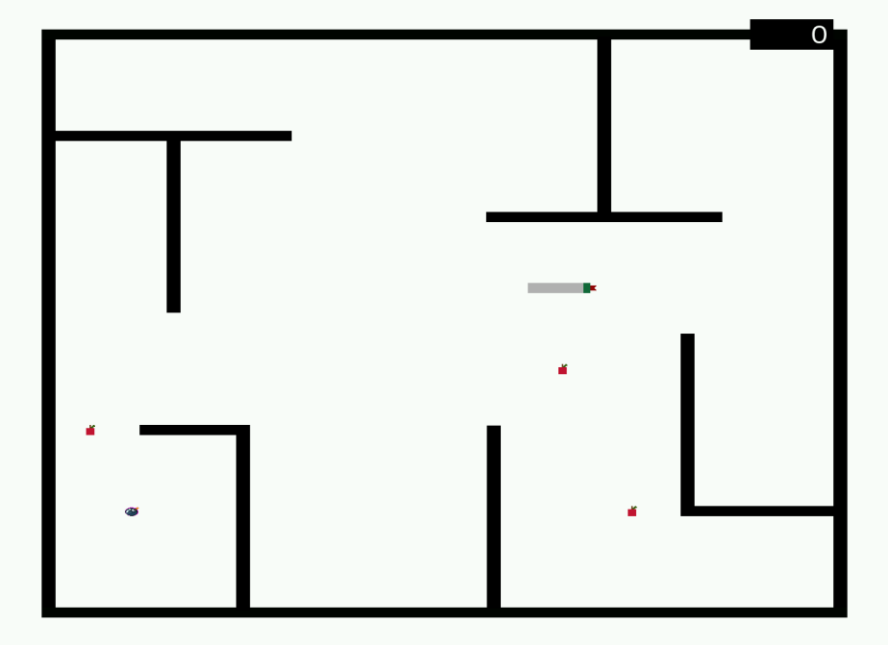
- *Exit*: sai do programa.

Para selecionar uma opção, basta mover o rato, clicando no botão esquerdo em cima do retângulo correspondente à opção que deseja.

Modo *SinglePlayer*

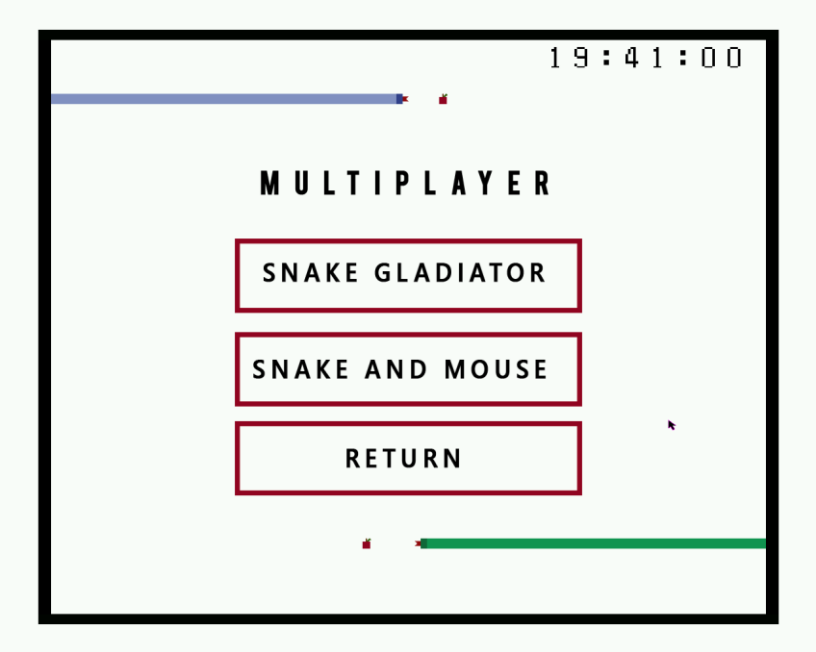
Neste modo de jogo, o objetivo é o jogador, controlando a sua cobra, comer o máximo de maçãs possível sem colidir com nenhuma parede nem bomba. Por cada maçã comida, a cobra cresce o equivalente a uma porção do seu corpo (correspondente ao tamanho da cabeça) e, ao fim de 3 maçãs comidas, aumenta a sua velocidade de deslocação e ganha 1 segundo de boost, que pode ser usado mantendo pressionada a tecla CRTL situada no canto inferior esquerdo do teclado, durante o tempo que pretender. O jogador pode gastar o boost todo de uma só vez ou de forma faseada. Para manter alguma curiosidade e interesse de descoberta do jogo, por parte do jogador, apenas é dito ao jogador qual é a tecla para fazer boost, mas não quando é que o tem nem durante quanto tempo.





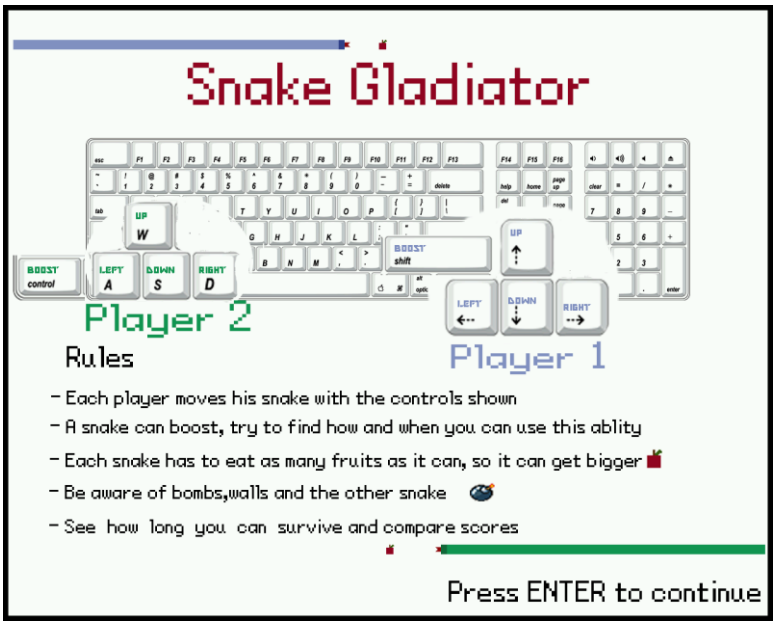
Modo *MultiPlayer*

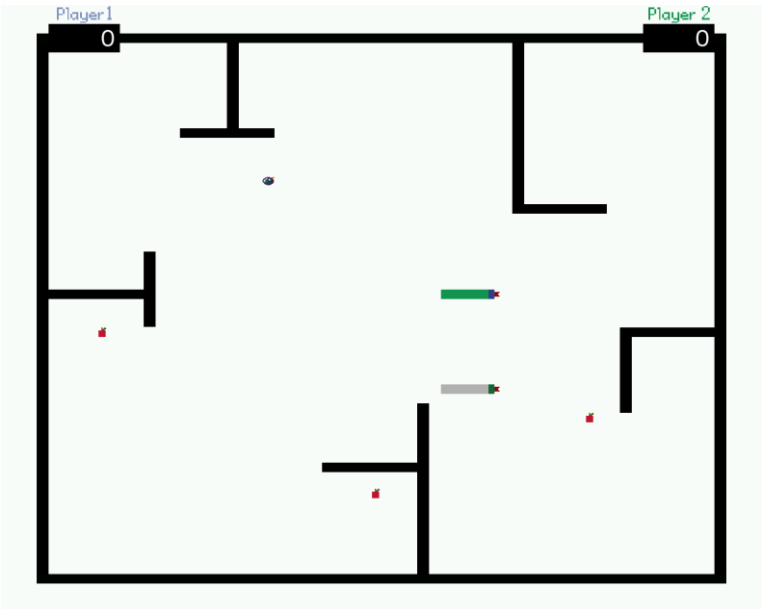
Neste submenu, os jogadores podem retornar ao menu principal ou escolher uma das seguintes opções:



Snake Gladiator

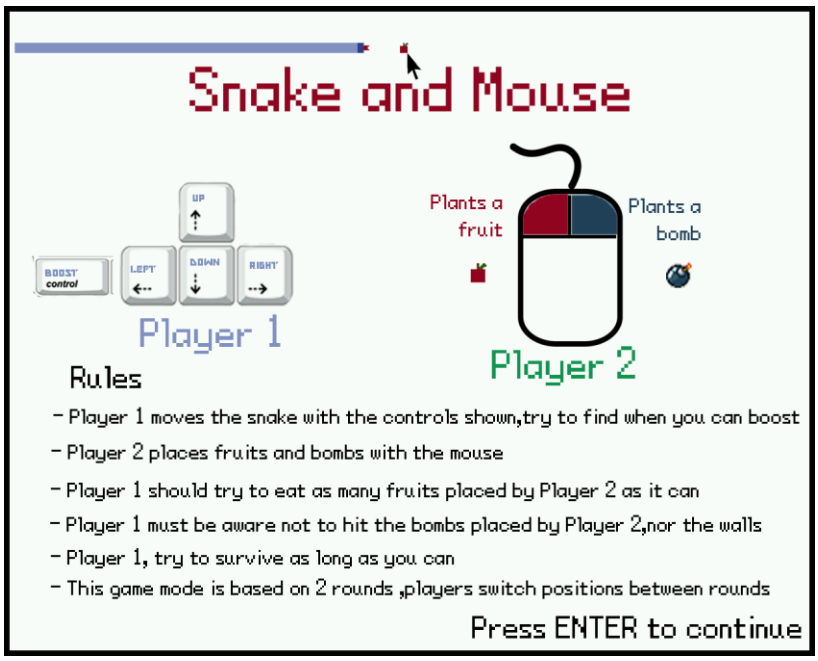
Neste modo, ambos os jogadores selecionam o corpo da sua cobra (como já foi referido anteriormente). Após isso, o jogo começa. O objetivo é idêntico ao do modo *SinglePlayer*, sendo que neste caso as cobras competem uma contra a outra, diretamente. Neste modo, cada uma das cobras tem as suas teclas para controlo da deslocação e para o respetivo boost, que é também desconhecido por parte do utilizador em relação ao momento em que está disponível e à sua duração. Para além das colisões existentes no modo *SinglePlayer*, é considerada também a colisão entre cobras e, quando tal acontece, perde aquela que bateu com a cabeça no corpo da outra.

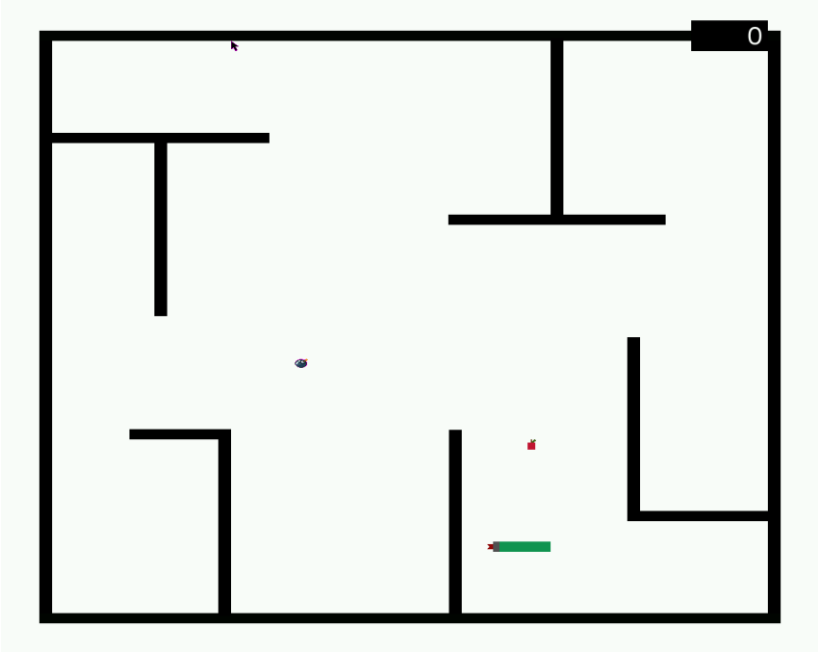




Snake and Mouse

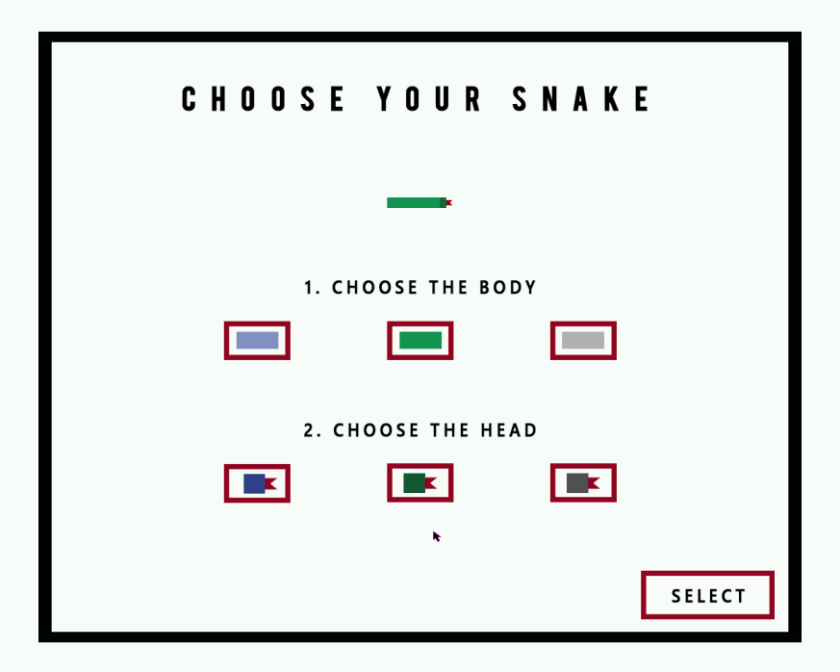
Neste segundo modo de *Multiplayer*, os jogadores jogam alternadamente, primeiro o jogador 1 no teclado e o jogador 2 no rato, e depois nas posições inversas. Neste modo, cada um dos jogadores pode escolher tanto o corpo como a cabeça da sua cobra. De seguida, após as regras acerca do modo de jogo, aparecerá o mapa com os elementos, como indicado na imagem. O jogador que está a controlar a cobra utiliza as setas para a fazer mudar de direção. O outro jogador utiliza o rato para colocar objetos no mapa, na posição onde se encontra o cursor: clicando no botão esquerdo, coloca uma maçã; e clicando no botão direito, coloca uma bomba. O objetivo é o jogador da cobra aguentar o máximo de tempo. Após os 2 jogos, ganha o jogador que obtiver mais pontos.



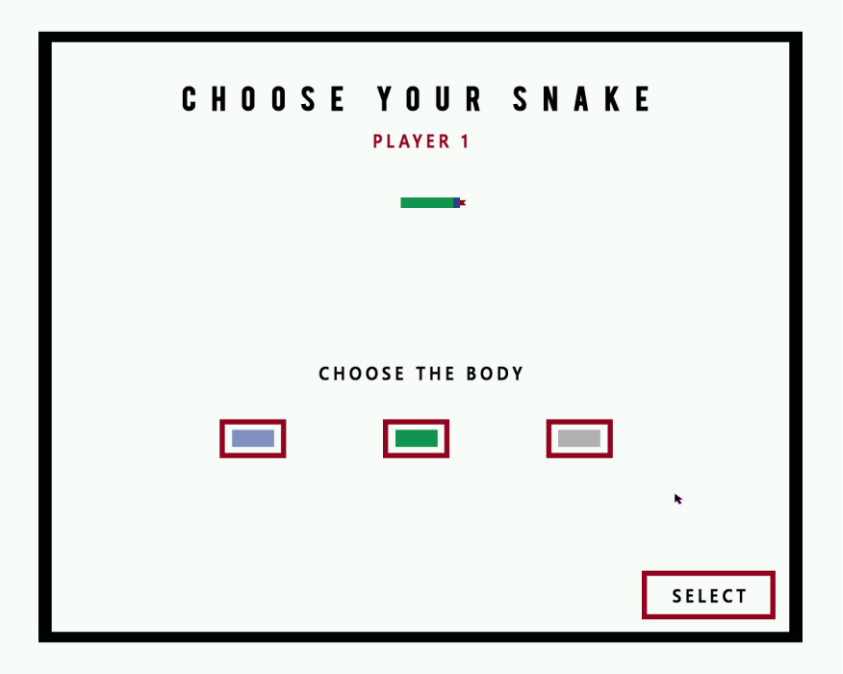


Menus de escolha de elementos da cobra

Para a escolha dos elementos que compõem cada cobra existem 2 menus, sendo que o primeiro permite ao utilizador escolher a cabeça e o corpo a utilizar (nos modos de *SinglePlayer* e *Multiplayer - Snake and Mouse*).



O segundo menu apenas permite a cada um dos jogadores escolher o corpo que quer para a sua cobra, sendo que os corpos das 2 cobras têm que ser distintos (no modo de *Multiplayer - Snake Gladiator*).



Em ambos os menus, de cada vez que o jogador faz uma seleção de elementos (cabeça e corpo, no 1º caso, ou só corpo, no 2º caso), visualiza a figura da cobra correspondente à escolha efetuada, podendo fazer alterações sucessivas, até encontrar uma combinação ao seu gosto.

II. Estado do Projeto

Dispositivos utilizados e Descrição da Utilização de cada Periférico

Periférico	*Finalidade*	*Interrupções?*
Timer	Atualizar o estado do jogo	Sim
Teclado	Interface	Sim
Rato	Interface com o utilizador nos vários menus e num dos modos de jogo	Sim
Placa Gráfica	Geração de imagens	Não
RTC	Obtenção da hora atual mostrada no menu principal	Não

Timer

Este dispositivo é utilizado tanto para fazer o controlo da velocidade e update do movimento da cobra, como para controlar a chamada às funções de geração de frutos e obstáculos, e possíveis colisões. Este periférico serve também para invocar as funções de display de todas as imagens existentes no jogo (as interrupções do mesmo são geridas no ficheiro man_events).

Teclado

Este periférico é utilizado para avançar a visualização das regras do jogo e controlo do movimento da(s) cobra(s) nos 3 modos de jogo, bem como o boost para as mesmas. O teclado é também utilizado para retornar ao menu após o fim do jogo e para colocar/retirar o jogo de pausa. As interrupções deste periférico são também geridas no ficheiro man_events.

Rato

Este periférico permite ao utilizador controlar e selecionar as várias alternativas existentes nos diversos menus, bem como avançar entre os mesmos. Para além disso, o rato é também utilizado no modo *Snake and Mouse* para, primeiro o jogador 2 e depois o jogador 1, colocarem maçãs e/ou obstáculos para o seu adversário comer ou se desviar.

Placa Gráfica

A placa gráfica é utilizada para a geração e desenho das imagens no ecrã, utilizando as cores no modo RGB565 e a configuração no modo gráfico A11 (1280*1024 pixeis).

Para uma melhor fluência na variação das imagens, foi utilizada a técnica de double buffering, na qual as imagens são geradas por camadas num buffer auxiliar e depois apenas se faz uma cópia integral deste buffer para o buffer principal (video_mem). Por forma a evitar que mesmo assim pudessem ocorrer problemas de fluência, optou-se por ter o buffer principal uma frame atrasada em relação ao estado real do jogo. Assim, as imagens são mostradas e geradas da seguinte forma:

1 - Cópia integral do que está no double_buffer para a video_mem;
2 - O double_buffer é limpo (é impressa a imagem de fundo);
3 - Atualização do double_buffer com o estado atual do jogo.

No entanto, este desfasamento não é relevante, já que esta atualização é feita a cada interrupção do timer.

Real Time Clock

Este periférico é utilizado para obter a hora atual a ser mostrada no menu principal.

III. Organização e Estrutura do Código

Main (7%)

Módulo inicial do jogo, onde se inicia o modo de video e a estrutura de bitmpas presente tanto no módulo menu como no módulo graphics. É também neste módulo que são subscritas e recebidas as interrupções de todos os periféricos dentro do ciclo while existente. Após receber as interrupções, este módulo chama as funções de leitura dos valores das interrupções (presentes no módulo referente a cada periférico) e posteriormente as funções de processamento desses mesmos valores presentes no módulo man_events.

No fim do jogo, é terminado o ciclo while, e dá-se unsubscribe as interrupções dos vários periféricos subscritos anteriormente.

Módulo desenvolvido por ambos os alunos (50/50)

Bitmap (6%)

Este módulo é responsável por carregar e desenhar imagens bitmap, através das funções LoadBitmap e DrawBitmap/drawbackground, respetivamente.

Desta forma, utilizando o código disponibilizado pelo ex-aluno Henrique Ferrolho para a geração das imagens, tentamos tirar partido da possibilidade de verificação da cor de cada pixel, optando assim por adaptar a função DrawBitmap para realizar transparência considerando como cor transparente 0xffffff8 considerando rgb 5:6:5.

Deste modo, a função drawbackground faz algo semelhante à função drawBitmap antes de ter sido editada, fazendo apenas cópia integral do Bitmap para o buffer.

Módulo adaptado por ambos os alunos (50/50)

Man_Events (9%)

Neste módulo, são processadas e geridas as interrupções provenientes dos vários periféricos. Este processamento tem em conta uma máquina de estados, também incluída neste módulo, e a partir da qual são feitas todas as mudanças de estados do jogo. Assim, esta máquina de estados encontra-se organizada da seguinte forma:

Estados:

- MENU_T - Estado do menu inicial
- MPMENU_T - Estado do menu de escolha dos modos de *Multiplayer*
- WAIT_T - Estado onde são mostradas as regras do modo de jogo escolhido e se espera o enter por parte do utilizador para avançar
- SP_T - Estado de modo de jogo *Singleplayer*
- MOKB_T - Estado do modo de jogo *Multiplayer Snake and Mouse*
- KBC_T - Estado do modo de jogo *Multiplayer Snake Gladiator*
- EXIT_T - Estado final do jogo
- END_T - Estado de gameover, que ocorre no final de cada jogo de *SinglePlayer* ou *Multiplayer*.
- PAUSE_T - Estado de pausa do jogo
- CHOOSE_SN_T - Estado da apresentação do menu de escolha do corpo da cobra
- START_DELAY_T - Estado de contagem decrescente no início do jogo

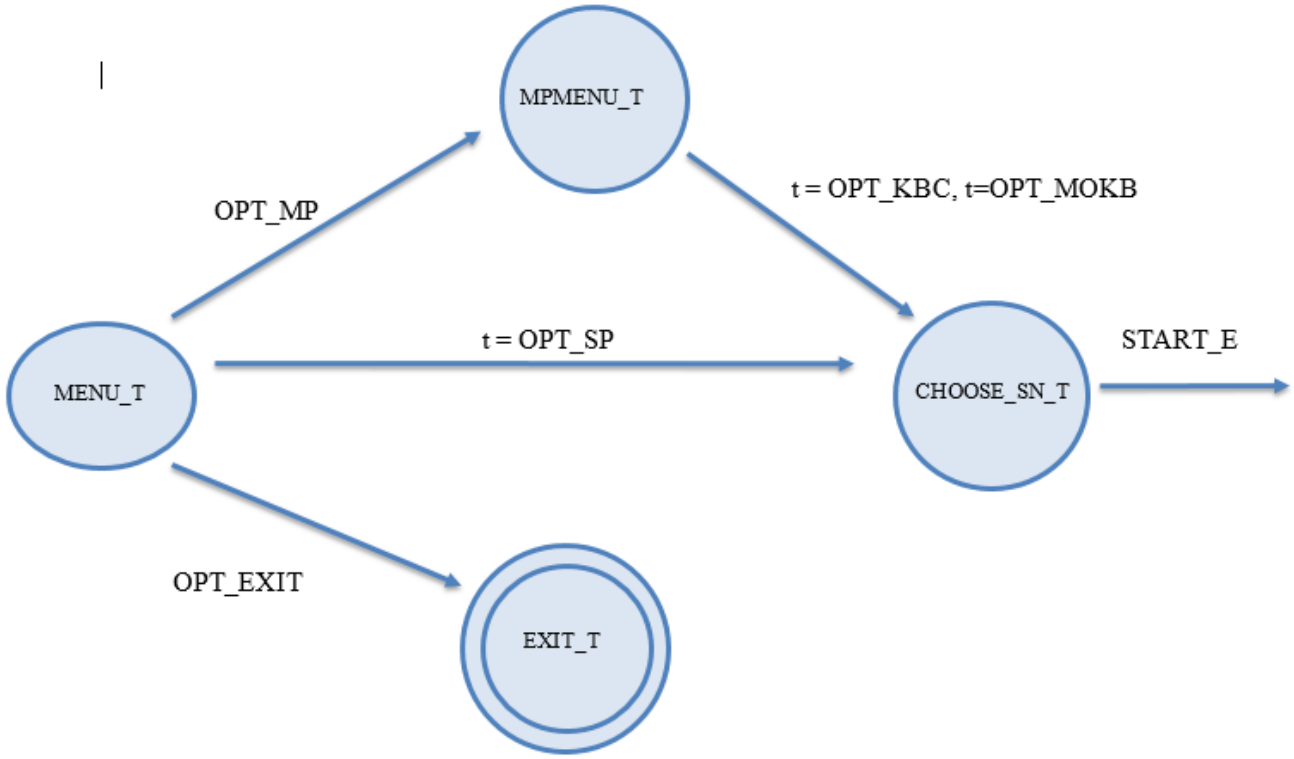
Eventos:

- OPT_SP - Evento de transição para o modo de *SinglePlayer*
- OPT_MP - Evento de transição para o menu de *Multiplayer*
- OPT_KBC- Evento de transição para o modo de jogo *Multiplayer Snake Gladiator*
- OPTB_MOKB - Evento de transição para o modo de jogo *Multiplayer Snake and Mouse*
- OPT_EXIT – Evento de transição para o estado final do jogo (EXIT_T)
- COLISION - Evento de transição para o estado END_T devido a colisão
- START_E - Evento de início do jogo, representando a transição para o estado WAIT_T e nas transições entre o menu de escolha dos elementos do jogo e o menu de regras (estado wait_t). Este evento é também utilizado para a transição entre este ultimo estado e o estado countdown de início de cada modo de jogo (START:DELAY_T)
- ESC_PRESSED - Evento de transição para o estado de pausa do jogo (PAUSE_T)

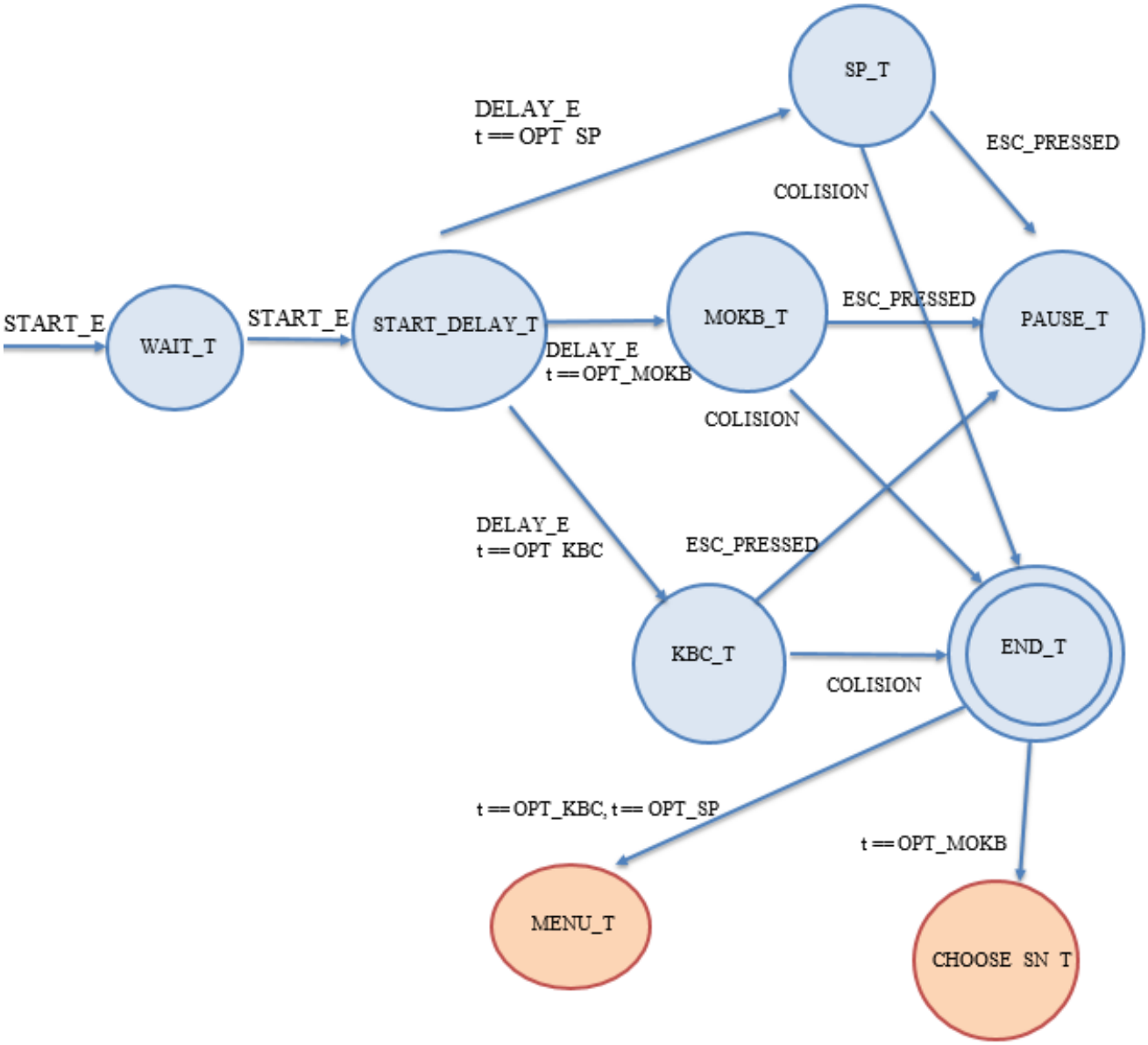
- DELAY_E - Evento de transição para o estado de contagem decrescente no início do jogo (START_DELAY_T)

A máquina de estados pode ser dividida em 2 máquinas de estados, por forma a simplificar o raciocinio por detrás desta.

Esta primeira parte da máquina de estados é referente às possíveis escolhas que o utilizador pode realizar nos vários menus. Apos o estado CHOOSE_SN_T existe uma transição para a segunda parte desta máquina.



Esta segunda parte da máquina de estados refere-se à parte da realização dos vários modos de jogo existentes.



Nota: os estados a laranja são estados já existentes na primeira parte da máquina de estados.

Para além da máquina de estados, este módulo contém também as funções de processamento do conteúdo das interrupções provenientes de vários periféricos. Assim, existem as seguintes funções:

- keyboard_event_handler - função responsável por processar o que é lido do buffer do teclado após uma interrupção proveniente do mesmo. É, portanto, nesta função que são chamadas as funções de update da matriz devido a alteração na trajetória da cobra por parte do utilizador, sem esquecer o processamento do click na tecla enter/esc para dar seguimento e pausa ao jogo.

- timer_event_handler - função responsável por processar as interrupções do timer e atualizar o display do jogo constantemente.

- mouse_event_handler - função que analisa o valor dos pacotes recebidos do rato, verificando o valor de x, y e do click dos botões. É também analisado o facto de o movimento ter dado overflow em algum dos eixos.

Módulo adaptado por ambos os alunos - Carlos 60% , Luís 40%

Graphics (9%)

Este módulo é um dos mais importantes em todo o jogo, pois é ele o responsável pelo update da Matrix que representa o jogo (tanto em relação a movimentos como colisões ou até mesmo verificar se a cobra comeu alguma maçã). É também aqui que são chamadas algumas funções de geração de objetos como maçãs e bombas e verificação do número de objetos existentes de cada um destes tipos na Matrix do jogo.

Deste modo, independentemente de haver alguma alteração ou não do estado de jogo, é sempre chamada a função draw_screen() responsável pelo display tanto do background como do estado atual da matriz.

Neste módulo existe a estrutura de dados Bitmaps_struct onde são guardados apontadores para todos os bitmaps utilizados no jogo. Existe também uma função que faz o load de todas as imagens existentes nesta estrutura.

Módulo adaptado por ambos os alunos - Carlos 40% , Luís 60%

Snake (9%)

Este é também um módulo bastante importante para o jogo, pois contem duas das structs principais do jogo.

No início, a cobra tem um tamanho de 5 elementos (incluindo a cabeça), sendo que, a cada 6 ms, move-se um segmento.

Assim, para a organização deste elemento importante do nosso jogo, utilizamos algo semelhante a uma lista duplamente ligada pois permite para cada segmento da cobra (elemento do tipo da struct segment_snake) saber onde está o segmento exatamente a seguir e o segmento exatamente anterior.

Para além disto, para cada segmento podemos ainda saber qual a sua posição na matriz e qual a sua direção (RIGHT, LEFT, UP, DOWN) e orientação (HORIZONTAL, VERTICAL).

Deste modo, possuimos então uma struct mais global, cujo nome é Snake e que contém um apontador para o segmento inicial (head) e o segmento final (tail), não esquecendo os atributos size, velocity (velocidade de movimento da snake que é variável ao longo do jogo), boost (que indica se a cobra está nesse momento em boost ou não) e boost_time que indica quanto tempo de boost ainda lhe resta.

Assim, neste módulo, a snake é criada e são feitos todos os updates em relação à mesma, tanto de movimento como de incremento do seu tamanho. É aqui que são realizadas também as funções de update do boost da cobra.

Por fim, existe também a função set_snake que serve para, na cobra de demonstração existente nos menus de escolha da cabeça/corpo, colocá-la na posição inicial e desta forma permitir que o movimento seja sempre contínuo.

Módulo desenvolvido por Luís Martins

Objects (4%)

Este módulo contém a estrutura de dados GameObject que, para cada objeto, guarda a sua posição para posteriormente avaliar se é uma posição válida para ser colocada na Matriz do jogo. Esta avaliação é feita no módulo graphics e tem as seguintes restrições:

- não estar numa posição já ocupada por outro elemento
- estar a uma distância mínima de 2 colunas e/ou 2 linhas da cabeça da(s) cobra(s)
- não ser colocado numa posição tal que, quando for comida, possa haver ambiguidade em relação ao local para onde se deve incrementar a cobra.

Este módulo e a respetiva struct permitem que o jogo tenha novos desenvolvimentos, através da colocação de novos objetos e atribuindo cotações diferentes para cada objeto, podendo assim dar continuidade ao jogo.

Módulo desenvolvido por Carlos Freitas

Menus (7%)

Este módulo contém todas as funções relativas ao display dos menus existentes no jogo e update da posição e desenho do rato - update_pos_mouse. Neste módulo estão também incluídas as funções responsáveis pela alteração de Bitmap relativo às componentes das cobras, com base nas escolhas do utilizador - change_body e change_head, sem esquecer também as funções para display da pontuação no final de cada jogo.

Neste módulo está também a função relativa ao display das horas no menu principal.

Módulo desenvolvido por Carlos Freitas

Constants (3%)

Este módulo é utilizado para guardar as constantes relativas ao RTC, à placa gráfica e às posições dos botões nos menus para verificar as posições do rato.

Módulo desenvolvido por ambos os alunos (50/50)

i8254

Este módulo foi importado do código do laboratório 2.

Módulo disponibilizado pelo professor para a realização do laboratório 2

Date (3%)

Neste módulo temos a estrutura de dados *date_rtc* onde são guardados os dados relativos à hora atual, informação esta que é utilizada para ser mostrada no menu principal.

Este módulo contém também a função udpate_date que é responsável por ler os valores da hora atual através do real time clock.

Módulo desenvolvido por Luís Martins

Rtc (4%)

Neste módulo são obtidos e convertidos para binário os valores da hora atual, para que se possa posteriormente mostrar nos menus.

Módulo desenvolvido por Luís Martins

Timer (7%)

Este módulo foi importado do código do laboratório 2.

Módulo desenvolvido por ambos os alunos (50/50)

Keyboard (7%)

Este módulo foi importado do código do laboratório 3.

Módulo desenvolvido por ambos os alunos (50/50)

ASSEM_SCAN(módulo em assembly – 4%)

Este módulo contém código em Assembly e é utilizado para verificar a possibilidade de fazer a leitura do que está disponível no buffer do teclado, para que posteriormente esse valor seja processado pela função keyboard_event_handler existente no módulo man_events. Baseado no código do laboratório 3.

Módulo desenvolvido por Carlos Freitas

Mouse (7%)

Este módulo foi importado do código do laboratório 4.

Módulo desenvolvido por ambos os alunos (50/50)

Vídeo_gr (5%)

Este módulo foi importado do código do laboratório 5, apenas com a adição das funções getHorResolution, getVerResolution e getDoubleBuffer, necessárias para o módulo Bitmap e da função printf_buffer necessária a cópia do conteudo do double_buffer para a video_mem

Módulo desenvolvido por ambos os alunos (50/50)

VBE (3 %)

Este módulo foi importado do código do laboratório 5.

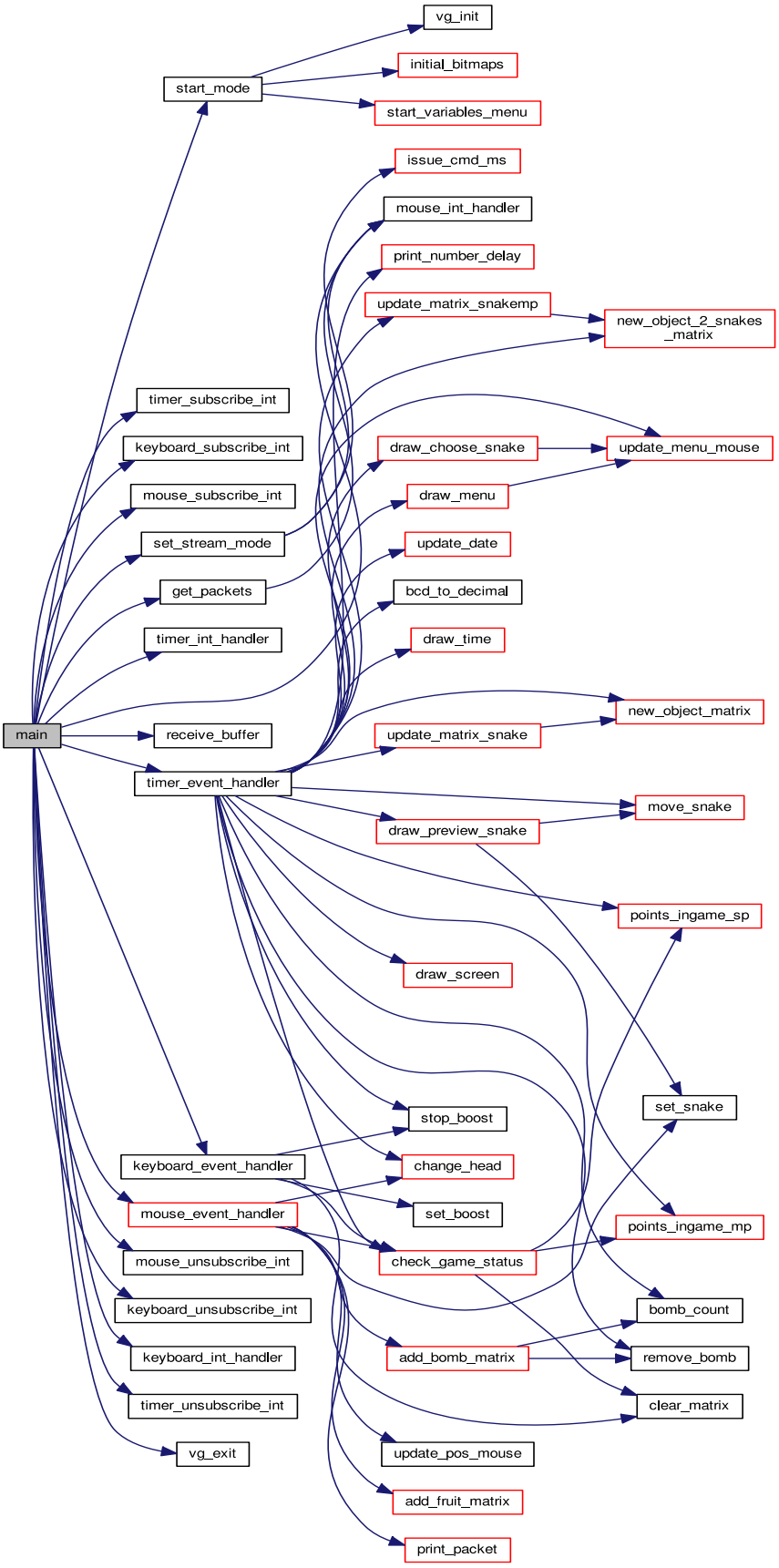
Módulo disponibilizado pelo professor para a realização do laboratório 2

Lmlib (3%)

Este módulo foi importado do código do laboratório 5.

Módulo disponibilizado pelo professor para a realização do laboratório 2

IV. Gráfico de Chamada de Funções



V. Detalhes de Implementação

Para a implementação do nosso projeto, decidimos utilizar código em camadas, já que permite manter toda a estrutura do projeto mais organizada e concisa, evitando a repetição de código, e de forma a torná-lo eficiente.

Para uma melhor estruturação dos vários estados entre os quais o jogo pode variar, decidimos criar uma máquina de estados (já referida no módulo man_events) fundamental para que as funções de handler dos vários periféricos possam gerir as suas interrupções e atuar conforme o estado atual do jogo.

Para uma fácil gestão da posição dos vários elementos durante cada jogo, é utilizada uma Matriz (já referida no módulo graphics), que permite em qualquer altura verificar possíveis colisões entre a cobra e qualquer obstáculo ou maçã. Assim, esta matriz é um elemento fundamental para o jogo, pois é a partir dela e das suas verificações que são feitas todas as alterações necessárias ao jogo, tanto em relação ao tamanho, velocidade ou boost como também em relação à possível existência de objetos em certos pontos. Assim, toda a dinâmica dentro dos modos do jogo é simplificada e são avaliados todos os parâmetros importantes de uma forma bastante eficiente, que permite uma melhor fluência e organização do jogo.

Por fim, é de referir (como já foi dito no módulo snake), que a cobra existente no jogo é baseada numa lista ligada. Esta estruturação é bastante importante, pois permite que em qualquer momento se saiba qual a posição de cada segmento da cobra e, deste modo, se possa fazer um movimento sempre coerente com o corpo da cobra, ainda que esta mude de direção. Para além disso, os outros atributos da estrutura snake permitem facilmente avaliar a sua velocidade, tamanho (essencial para saber qual o número de pontos alcançados por cada cobra) e até mesmo a possibilidade de boost ou não.

Em suma, toda a estrutura já montada e organizada permite que o jogo possa ser facilmente desenvolvido, acrescentando novos recursos e funcionalidades.

VI. Conclusões

A elaboração deste projeto permitiu-nos aprofundar e melhorar as nossas capacidades para o desenvolvimento e estrutura do código, bem como para a articulação dos vários elementos que constituem o projeto.

Durante o desenvolvimento do trabalho, o grupo funcionou em equipa, com cooperação de ambos os elementos, conseguindo ultrapassar as dificuldades e resolver os problemas que foram surgindo.

Queremos deixar um agradecimento ao aluno Henrique Ferrolho, pelo código disponibilizado para a leitura e desenho de Bitmaps e pelas informações que tem disponibilizadas no seu blog (difusal.blogspot.pt) acerca do sistema operativo minix e funcionamento dos periféricos.

Para a realização do projeto foi fundamental todo o apoio do professor e do monitor, tanto durante a realização dos laboratórios como nas três primeiras semanas dedicadas ao projeto. As aulas teóricas foram também importantes, pois forneceram as bases necessárias à compreensão e desenvolvimento, quer dos laboratórios, quer do projeto.

O único ponto a referir menos positivo, é a organização dos guiões dos laboratórios, já que estes poderiam ser mais simplificados e estruturados por pontos.

Em suma, esta cadeira permitiu-nos desenvolver as nossas capacidades em relação à linguagem C e permitiu-nos perceber como funcionam periféricos e como podem ser programados.

Apêndice

Instruções de Instalação

Este jogo foi desenvolvido para correr no sistema operativo minix.

Para a instalação, é necessário apenas obter os ficheiros do projeto, fazendo download (através do comando svn checkout) dos ficheiros contidos no seguinte link: https://svn.fe.up.pt/repos/lcom1617-t4g14/proj

Nota: para o caso de este link não estar acessível, bastará substituir “svn.fe.up.pt" pelo seguinte ip: 193.136.28.211.

Após isto, será necessário entrar na pasta com nome “scripts” que se encontra dentro do repositório transferido.

Dentro desta pasta, o utilizador deverá garantir que tem permissões de root. Para o caso de ter iniciado sessão com o utilizador lcom basta escrever su e pressionar enter para ter permissões de root.

Após isto, deverá escrever o comando: “sh install.sh” para copiar os ficheiros de imagens para o diretório /home/proj/res e, caso este não exista, cria-o.

De seguida deverá correr o comando “sh compile.sh” para compilar os ficheiros de código do projeto.

Por fim, basta correr o comando “sh run.sh” para correr o projeto, e divertir-se a jogá-lo.