Faculdade de Engenharia da Universidade do Porto

SNAKE

LCOM – Projeto – Turma 3 Grupo 3

Realizado por:

Inês Cardoso – up202005435
Joana Santos – up202006279
Mariana Carvalho – up202007620
Matilde Sequeira – up202007623

Índice

Índice 1

Introdução 2

1.Instruções de utilização 3
- 1.1. Menu Inicial 3
- 1.2 Menu de Instruções 4
- 1.3 Mapa 5
- 1.4 Modo single-player 6
- 1.5 Modo multiplayer 6
- 1.6 Game over 7

2. Estado do projeto 9
- 2.1 Timer 9
- 2.2 Keyboard 10
- 2.3 Rato 10
- 2.4 Graphics Card 11
- 2.5 RTC 11

3. Organização e estrutura do código 12
- objects.h e objects.c 12
- snake.c e snake.h 12
- apples.c e apples.h 12
- game.c e game.h 12
- gameOver.c 12
- instructions.c 13
- timer.c 13
- keyboard.c 13
- graphics.c 13
- mouse.c 13
- rtc.c 13

4. Grafo de Dependências de Funções 15

5. Detalhes de implementação 16

Conclusão 20

Introdução

Para o projeto da cadeira LCOM 21/22 implementamos uma versão do jogo Snake.

Esta versão inclui modo *single-player* e modo *multi-player*. Em ambos os modos o jogador utiliza o teclado para movimentar a cobra e no último caso, o segundo jogador joga com o rato.

Ao começar o jogo, aparece um menu com as opções de ler as regras do jogo, escolher jogar em modo *single-player* ou modo *multi-player* e sair do jogo.

O jogo em si tem 4 objetos: a cobra (que tem 5 vidas representadas por 5 corações); a maçã vermelha (ao comer 3 maçãs, a cobra ganha 1 vida de volta), e 2 tipos de maçãs envenenadas que diferem no dano que dão à cobra: as maçãs pretas tiram 1 coração e as castanhas tiram metade de um coração.

Durante o jogo, o jogador move a cobra pelo ecrã, tentando comer as maçãs vermelhas (sendo que estas aparecem aleatoriamente no mapa), enquanto evita colidir com as paredes e com o seu próprio corpo. Se o jogador tocar com a cobra nas paredes ou no corpo da mesma, automaticamente perde o jogo.

Em modo *single-player*, as maçãs envenenadas aparecem aleatoriamente no mapa: a maçã castanha aparece de 10 em 10 segundos e a maçã preta aparece a cada 17 segundos. Contudo, no modo *multi-player*, como dito anteriormente, é o segundo jogador que decide onde é que as maçãs envenenadas aparecem.

Este segundo jogador, pode estar constantemente a colocar maçãs envenenadas no mapa, carregando nos botões do rato. Com o esquerdo posiciona uma maçã castanha e com o direito posiciona uma maçã preta, não sendo possível colocar uma maçã em cima de outra que já se encontre em jogo. Decidimos não restringir nem o número de maçãs, nem o tempo durante o qual o jogador as pode colocar, uma vez que implementamos a regra de que à terceira maçã vermelha que a cobra come, é-lhe regenerada uma vida.

Instruções de utilização

1.1 Menu Inicial

Quando o jogo inicia, aparece o menu principal com as opções de jogo: *single-player* (1xPC) e *multiplayer* (1x1); a opção de visualizar as instruções e a opção de abandonar o jogo e sair.

Para selecionar as opções, o utilizador terá de movimentar o rato e carregar com o botão esquerdo do mesmo, em cima de uma opção, se a quiser escolher. Para efeitos visuais, quando o jogador passa com o rato por cima de uma opção, o fundo desta, muda de cor, salientado a escolha.

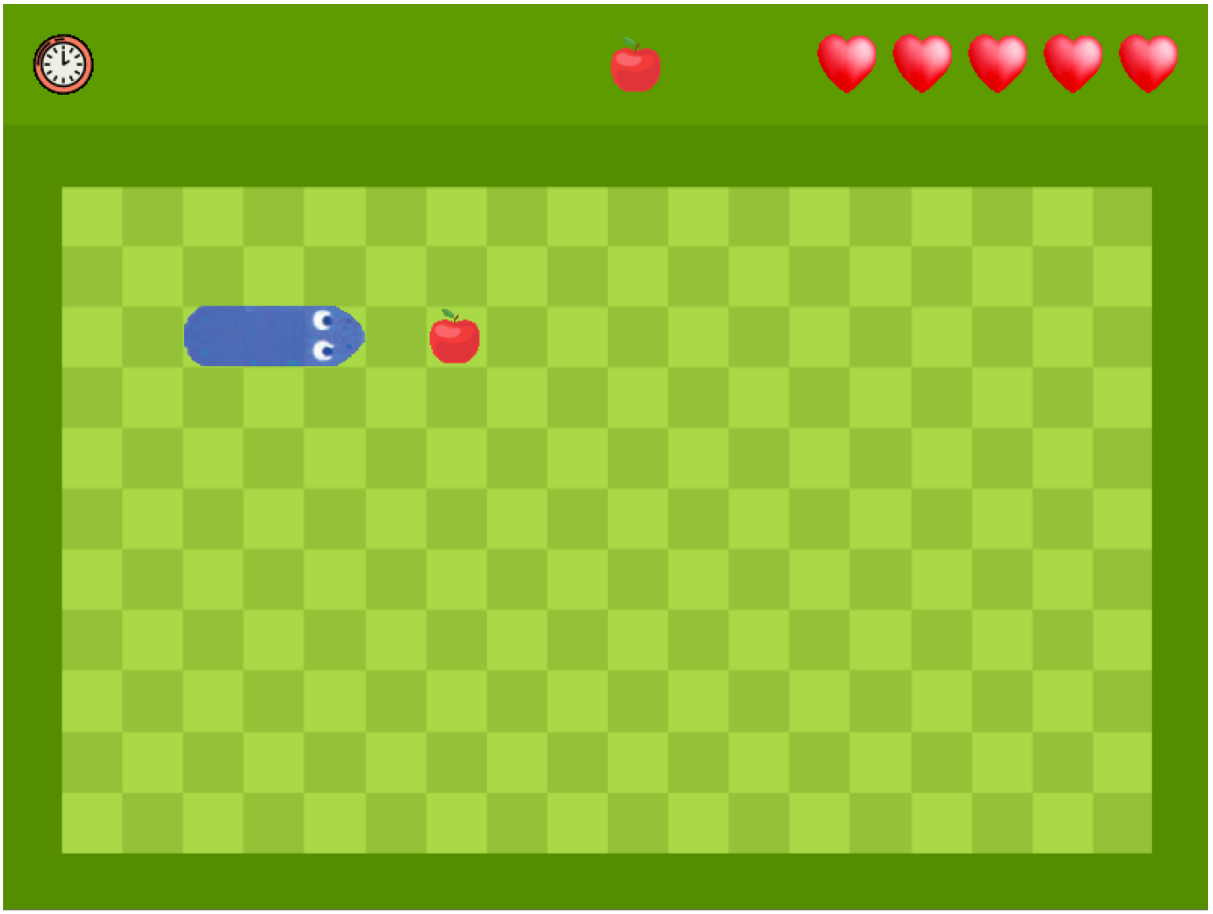
1.2 Menu de Instruções

Assim que o jogador selecione no menu principal, ver as instruções, aparece então a seguinte imagem.

Novamente, para selecionar a opção de regressar ao menu principal, o utilizador terá de movimentar o rato e carregar com o botão esquerdo do mesmo. Também, para efeitos visuais, quando o jogador passa com o rato por cima da opção, o fundo desta, muda de cor, salientado a escolha.

1.3 Mapa

Iniciando o jogo em modo *single-player* ou modo *multiplayer,* o mapa é o mesmo.



Em ambos os modos, o jogador movimenta a cobra usando as teclas a-w-s-d ou as setas do teclado e deve orientar-se pela direção da cabeça da cobra, uma vez que, com o aumento do corpo da cobra, pode haver situações em que o corpo se encontra em direções diferentes da cabeça.

Em relação ao movimento da cobra, este é contínuo, ou seja, o jogador não precisa de estar sempre a clicar nas teclas para que a cobra avance as quadrículas. A cobra continua o movimento na direção inicial até ser notificada de uma tecla que corresponda a uma direção diferente.

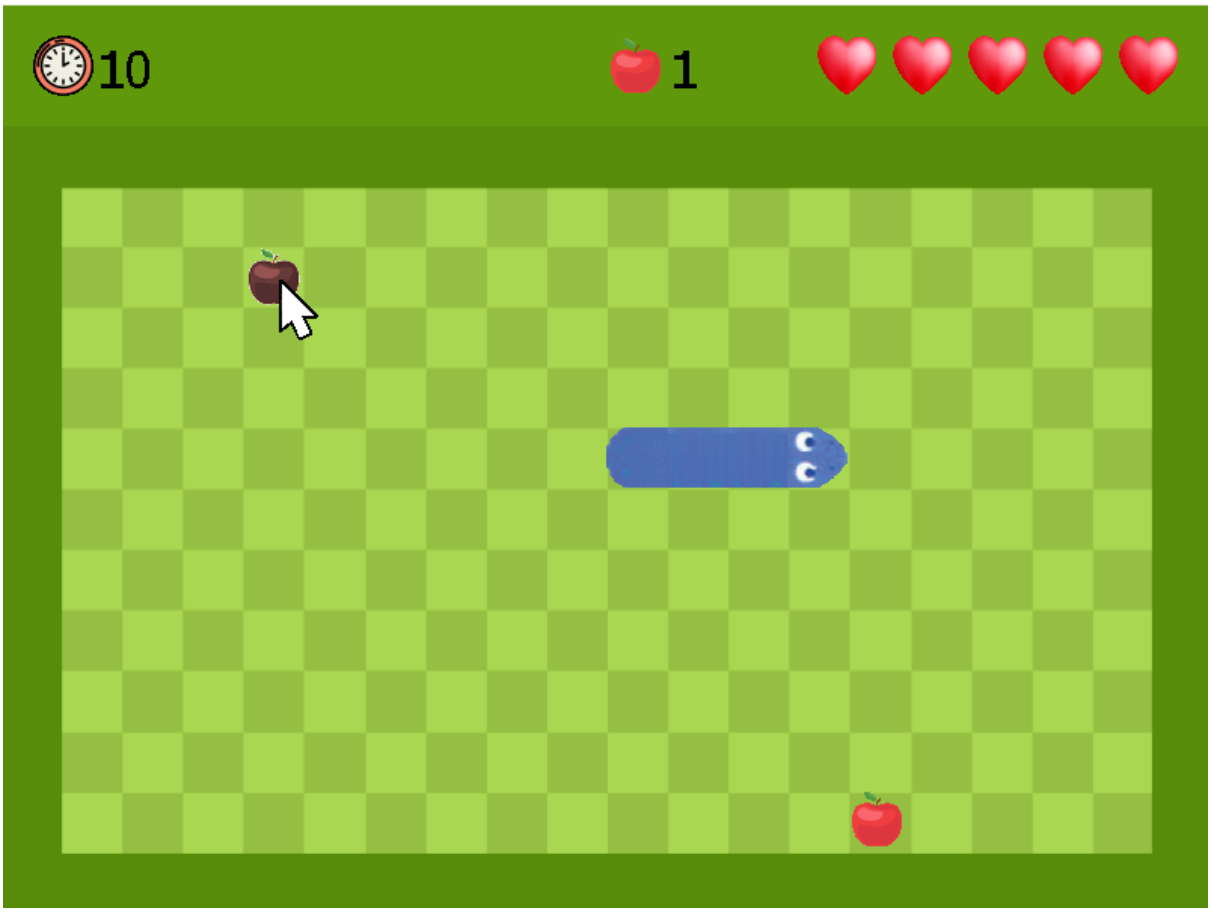
As maçãs vermelhas aparecem aleatoriamente no mapa, e só surge uma nova quando a cobra come a maçã que lá estava.

1.4 Modo *single-player*

Em modo *single-player*, as maçãs envenenadas aparecem aleatoriamente no mapa: a maçã castanha aparece de 10 em 10 segundos e a maçã preta aparece a cada 17 segundos.

1.5 Modo *multiplayer*

No modo *multiplayer,* é o segundo jogador que decide onde é que as maçãs envenenadas aparecem.

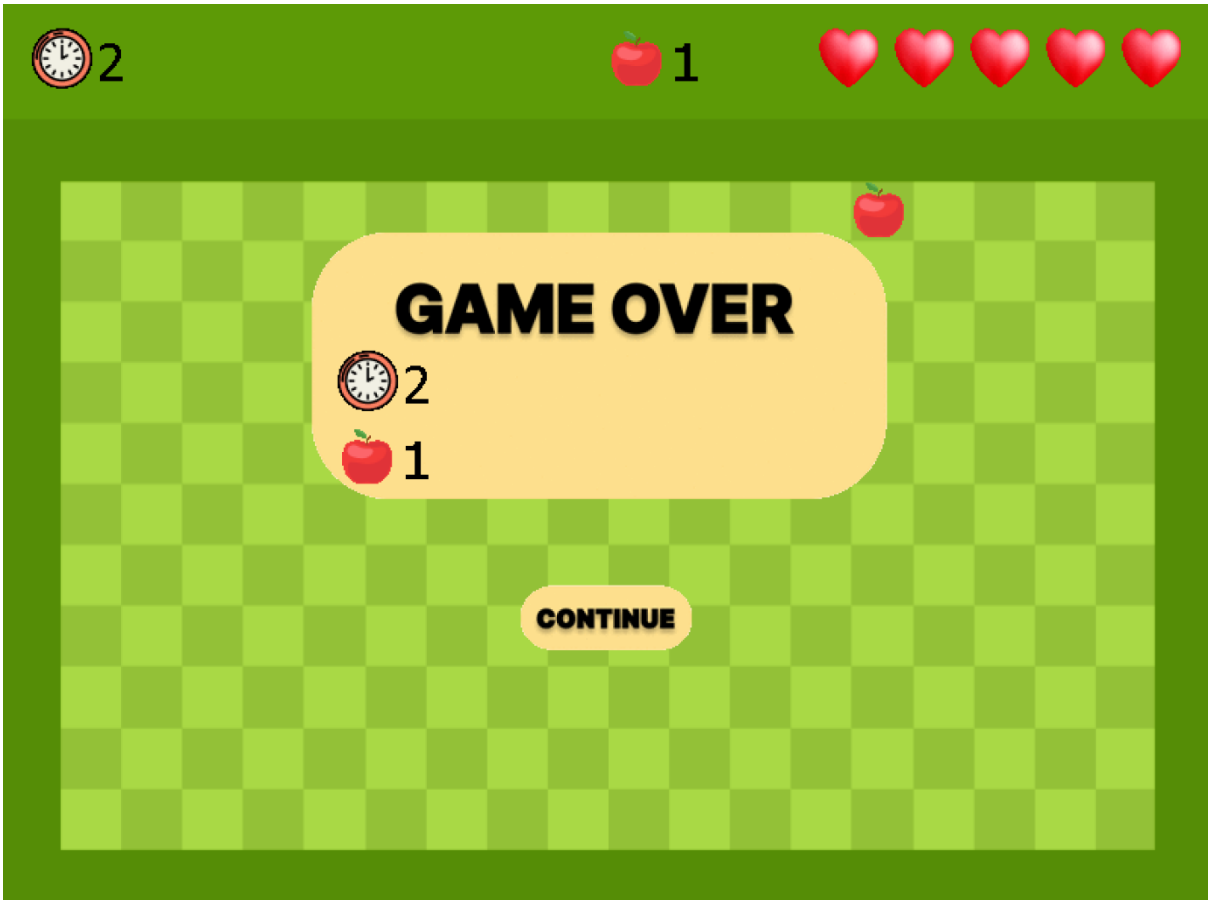


Este segundo jogador, pode estar constantemente a colocar maçãs envenenadas no mapa, carregando nos botões do rato. Com o esquerdo posiciona uma maçã castanha e com o direito posiciona uma maçã preta, não sendo possível colocar uma maçã em cima de outra que já se encontre em jogo. Decidimos não restringir nem número de maçãs, nem o tempo durante o qual o jogador as pode colocar,

uma vez que implementamos a regra de que à terceira maçã vermelha que a cobra come, é-lhe regenerada uma vida.

1.6 Game over

Quando o jogador perde, ou seja, se a cobra foi contra uma parede, se colidiu consigo própria ou se já não lhe restam vidas, aparece a imagem de *game over.*



Aqui são apresentados os resultados da partida: o tempo até o jogador perder e o número de maçãs vermelhas que a cobra comeu.

Aparece também a opção de continuar, que redireciona o jogador de volta para o menu principal.

Como nos outros botões, para selecionar a opção de continuar, o utilizador terá de movimentar o rato e carregar com o botão esquerdo do mesmo. Para efeitos

visuais, quando o jogador passa com o rato por cima da opção, o fundo desta, muda de cor, salientado a escolha.

2. Estado do projeto

DISPOSITIVO	FUNCIONALIDADES	INTERRUPÇÕES / POLLING
Timer	Contar tempo de jogo. Controlar o frame-rate. Animações.	Interrupções
Keyboard	Permitir que utilizador movimente a snake com as teclas “A”, “W”, “S”, “D” ou com as setas.	Interrupções
Mouse	Permite que o utilizador possa efetuar as suas escolhas no menu. Em modo single player, permite ao segundo utilizador a colocação de maçãs envenenadas com os botões do rato.	Interrupções
Graphics card	Apresentar a interface do jogo ao utilizador.	Nenhuma
RTC (Real Time Clock)	Gerar interrupções periódicas que permitem então a colocação das maças.	Interrupções

2.1 Timer

O timer é utilizado para controlar o frame-rate, - frequência de 60 frames por segundo – criar eventos e realizar animações.

No modo single-player (função **singlePlayerMode**, no ficheiro **game.c**) e multiplayer (função **multiplayer**, no ficheiro **game.c**), o timer é usado para permitir que a cobra se esteja sempre a movimentar. Além disso, é ainda utilizado para contar o tempo de jogo, que é apresentado na tela ao jogador no decorrer do jogo, e aparece como score na tela de GameOver.

As funções relacionadas com o timer encontram-se no ficheiro **timer.c**.

2.2 Keyboard

O keyboard é utilizado no modo singleplayer e multiplayer para permitir o movimento da cobra utilizando as teclas A, W, S D ou as setas. Foi criada uma struct, no ficheiro **snake.c**, à qual estão associadas variáveis para cada uma das direções estáticas, bem como as imagens da cabeça da cobra a elas associadas, que são atualizadas na função de **singlePlayer** e **multiPlayer** ao ser comparado o scanCode com o make-code.

O keyboard é ainda utilizado na tela de gameOver, permitindo o regresso ao menu principal quando se pressiona a tecla “ENTER”, e no menu principal, que permite o fecho da janela de jogo ao ser pressionada a tecla “ESC”.

As funções relacionadas com o keyboard encontram-se no **keyboard.c**.

2.3 Rato

O rato é utilizado para navegar pelos menus e para colocar maçãs envenenadas no modo multi-player, através de um cursor - uma struct que contém a posição deste e a imagem a ele associada.

O evento do rato, que permite depois saber se foi pressionado algum botão é obtido a partir da função **mouse_get_event**, no ficheiro **mouse.c**. No modo multi-player, ao ser detetado o evento associado à pressão sobre o botão esquerdo é colocada uma maçã castanha na posição em que o cursor se encontra, e o mesmo se aplica às maças pretas em relação ao evento relativo à pressão do botão direito do rato.
De notar que a posição do cursor é atualizada em cada interrupção, com base no packet obtido na função **parse_packet**.

No menu, e nas telas de gameOver e instruções que contêm botões, verifica-se ainda a colisão do cursor com estes, que permite então que a cor dos botões seja alterada em caso de colisão, e posteriormente, se detetado o evento de pressão sobre o botão esquerdo, permite com que se mude o estado do jogo.

As funções relacionadas com o mouse encontram-se no **mouse.c**.

2.4. Graphics Card

Para o projeto foi utilizado o modo de vídeo 0x115, com resolução de ecrã 800x600, e 24 bits por pixel.

Recorreu-se à técnica de double buffering, sendo a cada interrupção desenhados elementos (como o cursor do rato) para uma memória auxiliar. Nas interrupções do timer, por forma a controlar o frame-rate, é copiado o conteúdo do buffer para a memória principal, usando a função **copy_buffer_to_mem()**.

Foram utilizados ficheiros XPM's para todas as imagens presentes no jogo, sendo estes desenhados com as funções **draw_xpm** e **draw_xpm_video_mem**, mediante se queira desenhar no buffer, ou para a memória principal.

As funções relacionadas com a placa de vídeo encontram-se no ficheiro **graphics.c**.

2.5 RTC

O RTC é utilizado para que, no modo single-player, apareçam periodicamente maçãs envenenadas (castanhas e pretas) em posições aleatórias do ecrã, como é observável na função **periodicApples** do ficheiro **snake.c**, e como será especificado nos detalhes de implementação.

As funções relacionadas com a implementação do RTC encontram-se no ficheiro **rtc.c**

3. Organização e estrutura do código

objects.h e objects.c (7%)
Neste ficheiro encontra-se uma struct **objects** utilizada para se poder associar a cada objeto um espaço de VRAM e associar-lhe um xpm.
Todas participaram na elaboração deste ficheiro igualmente.

snake.c e snake.h (12%)
Nestes ficheiros encontram-se as funções relativas ao desenho e funcionamento da cobra no jogo, incluindo inicialização e colisões com ela mesma.
É de realçar a existência de uma struct no snake.h denominada **snake_part**, que trata da posição, direção e imagens de uma parte do corpo da cobra.
Apesar de todas terem participado na elaboração deste ficheiro, estas funções tiveram a principal intervenção de: Joana Santos e Matilde Sequeira.

apples.c e apples.h (10%)
Nestes ficheiros encontram-se as funções relativas ao desenho das maçãs, inicialização e colocação destas no mapa. De notar ainda a existência de uma struct chamada **Apple**, que contém a posição, tipo e (objeto) xpm a ela associada.
Todas trabalharam igualmente na elaboração deste ficheiro.

game.c e game.h (9%)
Neste ficheiro encontram-se as funções que inicializam as variáveis e desenham o background. Encontram-se ainda funções relativas ao modo singlePlayer e multiplayer, estando presentes nestas duas funções loops que interagem com os diferentes dispositivos, através de interrupções que geram os diversos eventos do jogo.
Todas participaram na elaboração deste ficheiro igualmente.

gameOver.c (5%)
Neste ficheiro encontram-se as funções responsáveis por mostrar a tela de Game Over ao utilizador com o seu score, bem como tratar de verificar as colisões do rato com o botão de “Back to Menu” por forma a realizar a animação de seleção do botão, e mudança de tela caso o botão seja pressionado.

Apesar de todas terem participado na elaboração deste ficheiro, estas funções tiveram a principal intervenção de: Inês Cardoso e Mariana Carvalho.

instructions.c (5%)

Neste ficheiro encontram-se as funções responsáveis por mostrar na tela a imagem das instruções e regras, bem como tratar de verificar as colisões do rato com o botão de “Back to Menu” por forma a realizar a animação de seleção do botão, e mudança de tela caso o botão seja pressionado.
Apesar de todas terem participado na elaboração deste ficheiro, estas funções tiveram a principal intervenção de Inês Cardoso e Mariana Carvalho.

timer.c (15%)

Neste ficheiro estão presentes as funções desenvolvidas no Lab2 relacionadas com as interrupções do Timer, que fossem necessárias para o projeto. Estas funções foram construídas aquando da realização do Lab2, pelo que todas contribuíram igualmente para a sua realização.

keyboard.c (6%)

Neste ficheiro estão presentes as funções desenvolvidas no Lab3 relacionadas com as interrupções do keyboard, que fossem necessárias para o projeto. Estas funções foram construídas aquando da realização do Lab2, pelo que todas contribuíram igualmente para a sua realização.

graphics.c (15%)

Neste ficheiro estão presentes as funções desenvolvidas no Lab5 relacionadas com as funções que permitem a realização de double-buffering e do desenho das imagens xpm utilizadas. Estas funções foram construídas aquando da realização do Lab2, pelo que todas contribuíram igualmente para a sua realização.

mouse.c (11%)

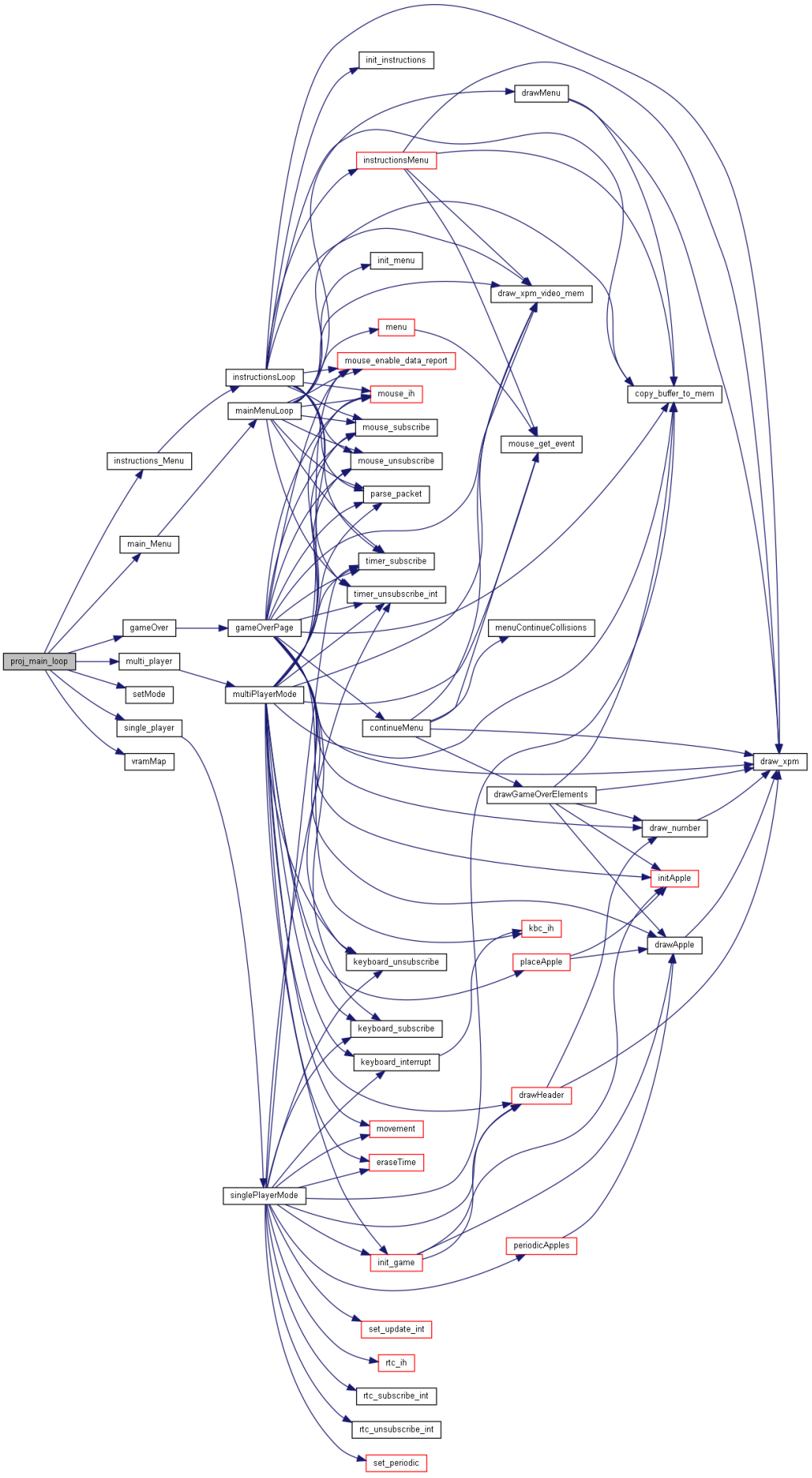
Neste ficheiro estão presentes as funções desenvolvidas no Lab4 relacionadas com as interrupções do mouse, que fossem necessárias para o projeto. Estas funções foram construídas aquando da realização do Lab2, pelo que todas contribuíram igualmente para a sua realização.

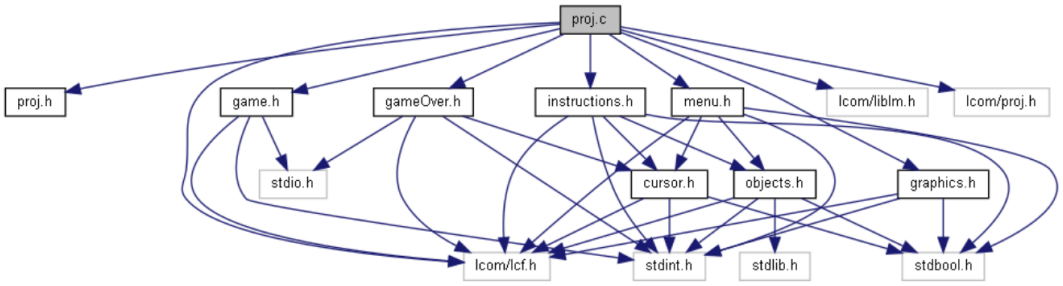
rtc.c (5%)

Neste ficheiro estão presentes as funções relativas às interrupções do RTC.

Apesar de todas terem participado na elaboração deste ficheiro, estas funções tiveram a principal intervenção de: Matilde Sequeira e Joana Santos.

4. Grafo de Dependências de Funções




proj.c
proj.h
game.h
gameOver.h
instructions.h
menu.h
lcom/liblm.h
lcom/proj.h
stdio.h
cursor.h
objects.h
graphics.h
lcom/lcf.h
stdint.h
stdlib.h
stdbool.h

5. Detalhes de implementação

O ciclo e estado do jogo é controlado pela **state machine** que foi implementada. Existe, portanto, um enum, **BaseState**, com todos os possíveis estados do jogo: mainMenu, singlePlayer, multiPlayer, instructions, gameover e leave.
Dependendo então do estado em que o jogo se encontra, são chamadas as respetivas funções que controlam o ciclo do jogo.

O **rato** foi implementado para ser utilizado no modo *multiplayer* (para o segundo jogador posicionar no mapa as maçãs envenenadas) e para que o utilizador possa navegar facilmente pelas páginas do menu e selecionar as suas opções.
Assim, a lógica por trás do menu, da página das instruções, do game over e do ciclo do modo *multiplayer*, centra-se na gestão de desenhar as imagens de fundo e a imagem do cursor.

Para implementar o rato foi criado um **double buffer**, de modo a tornar o movimento do cursor mais fluido. Ao introduzir o double buffer, há, então, um buffer que guarda a imagem que está a ser usada atualmente e outro que guarda a mesma imagem, mas onde vai ser desenhado por cima o cursor do rato, de modo a não intervir no que está por baixo.

A deteção, nos menus: menu principal, instruções e menu de game over; do movimento do rato, de modo a perceber se o utilizador está com o cursor em cima de algum botão, foi feita a partir das funções **menuOptionCollisions**(cursor *mouse_c), **menuContinueCollisions**(cursor *mouse_c) e **menuInstructionsCollisions**(cursor *mouse_c). Estas funções verificam a posição do cursor do rato e confirmam se esta se encontra dentro dos limites calculados em pixéis para os botões.

Para ambos os modos: *single-player* e *multiplayer*, a lógica do jogo gere-se à volta do movimento da cobra, assim, sempre que a cobra se move, são verificadas todas as suas colisões: cobra com ela própria, cobra com a maçã e cobra com limites do ecrã. As colisões são sempre feitas tendo em conta a posição da cabeça da cobra.

Começando pela colisão com as paredes, sempre que a cobra avança, é verificada a posição da sua cabeça em relação a cada parede. Se a cabeça da cobra estiver para lá de uma parede é retornado 1, sinal de que o jogo acabou.

A cobra também não pode passar por cima de si mesma, pelo que sempre que esta avança, é verificada se a sua cabeça colide com alguma parte do corpo. Para isto, é chamada a função **colisionWithItSelf( )** que percorre o vetor de snakeBody e retorna 1 se colidir.

Ao longo do jogo, sempre que é criada uma maçã, esta é colocada num array de **Apples** (Apple applesArray[195]). Durante o movimento da cobra, é chamada a função **collisionWithApple**( ) que retorna 1 se o jogo chega ao final, ou seja se o jogador ficou sem vidas, ou 0 se o jogo continua.

A função **collisionWithApple( )** começa por verificar se na nova posição da cabeça da cobra existe alguma maçã. Para isso é chamada a função **isApple**(int x, int y) que percorre o vetor de maçãs e vê se existe alguma maçã na posição (x,y) passada como parâmetros. Esta função devolve a posição da maçã no array e em caso de não existir nenhuma maçã naquela posição, é retornado -1.

Feita esta verificação, no caso de a maçã comida ser do tipo **red**, o número de maçãs comidas aumenta em uma unidade, a cobra aumenta de tamanho, e é gerada uma nova maçã aleatória no mapa. No caso de ser do tipo **brown** ou **black**, a cobra aumenta na mesma de tamanho e a sua vida diminui.

No final, a maçã na última posição do array, passa para a posição i (posição da maçã comida) e o número de maçãs diminui em uma unidade.

Por último, é de salientar o uso de **RTC**. No projeto, este componente foi implementado com o objetivo de gerar interrupções periódicas, de modo a controlar o número de maçãs pretas e castanhas que iam aparecendo no mapa.

Assim, é para o modo *single-player* que o componente se destaca, visto que no modo *multiplayer* é o segundo jogador que coloca as maçãs envenenadas em jogo.

O RTC produz interrupções em períodos de 0.5 segundos, correspondente ao período máximo possível.

A cada 0.5 segundos, surge então a interrupção que leva ao incremento de uma variável que controla o número de maçãs. É sempre verificada se essa variável

atinge o valor limite, este que corresponde ao dobro do tempo que é pretendido a que cada maçã envenenada apareça.

Para a maçã preta, como referido na introdução, é pretendido que esta surja a cada 17 segundos. Da maneira que o RTC apenas provoca interrupções num período de 0.5 segundos, então define-se o limite da variável de controlo como sendo 34 (34*0.5 = 17s). Quando a variável atinge 34, é, portanto, desenhada a maçã preta no mapa. O processo é análogo para a maçã castanha, no entanto o limite da variável de controlo é 20 para que a cada 10 segundos surja uma nova maçã castanha.

Conclusão

Em relação ao que foi proposto no Project Proposal, conseguimos concluir e implementar tudo o tinha sido planeado menos a Leaderboard. Decidimos deixar este elemento de fora, pois considerámos que não teria o maior significado em termos de resultados. Concordamos que era mais importante dedicarmo-nos às outras partes do projeto e realizá-las a um melhor nível. Contudo, incluímos no final de cada partida, ou seja, quando o jogador perde, uma tela que mostra os resultados do jogo: a duração da partida e o número de maçãs vermelhas que o jogador conseguiu coletar.

Como melhorias que tornariam o projeto mais interessante, vemos a implementação do serial port para o modo *multiplayer*, que inicialmente consideramos não ser fácil de implementar por conta do tempo que tínhamos disponível, e por não existir um Lab que nos pudesse auxiliar.

Apesar dos Labs terem sido uma boa preparação para o projeto, sentimos que a ausência dos mesmos para o RTC e o para o serial port dificultaram bastante a implementação destes componentes no projeto.

O momento do projeto em que sentimos mais dificuldade foi a implementação do rato, de maneira que o cursor se mexesse fluidamente. Inicialmente não percebemos como implementar um bom movimento do cursor, porém, depois de pesquisar, concluímos que a utilização de double buffer seria a opção correta.

Ao dar por terminado o desenvolvimento deste projeto, podemos concluir que foi um trabalho desafiante. Exigiu que tivéssemos de perceber sozinhas como implementar certos tópicos. Pensamos que para realizar um projeto bom, neste espaço de tempo, e tentar incluir o máximo de dispositivos exigiu mais entrega da nossa parte, para realmente tentar perceber o funcionamento dos dispositivos e a sua implementação.

Por outro lado, concluímos que através da realização do trabalho, conseguimos aprender como usar interfaces de programação dos dispositivos de entrada/saída que vemos ser o grande objetivo da cadeira.