Faculdade de Engenharia da Universidade do Porto

Projeto final

Laboratório de Computadores (LCOM)

Turma 4 - Grupo 4

Realizado por:

Beatriz Bernardo	up202206097
Diana Nunes	up202208247
Sofia Gonçalves	up202205020
Teresa Mascarenhas	up202206828

Índice

1. Introdução

Lab-rinth é um jogo 2D onde o objetivo principal é escapar de um labirinto dentro de um tempo estipulado. O jogador utiliza uma lanterna para visualizar o mapa, navegando pelos corredores e tentando descobrir botões onde pode atravessar, de modo a desbloquear determinadas portas espalhadas pelo caminho, enquanto procura a saída.

O jogo oferece dois modos de jogo: singleplayer e cooperativo.

No modo single-player, o jogador deve explorar o labirinto sozinho, usando a lanterna para iluminar o caminho e encontrar a saída antes que o tempo se esgote.

No modo cooperativo, dois jogadores jogam simultaneamente no mesmo labirinto. Eles devem cooperar para encontrar a saída, mas com a tensão adicional de competir para ver quem chega primeiro. Ambos os jogadores partilham o mesmo objetivo, mas devem equilibrar a cooperação com a competição para garantir que conseguem sair do labirinto antes do adversário.

2. Instruções de utilização do programa

2.1 Menu

Menu inicial

Quando o jogo começa, é exibido o menu inicial. Este possui 2 botões: um para sair do jogo (“Quit”) e outro para ir até ao menu que permite a escolha do nível pretendido (“Start”). Estes botões devem ser pressionados com o rato.

Fig 1. Menu inicial

Menu dos níveis

Ao carregarmos em “Start” no menu inicial, é exibido o menu para a escolha do nível pretendido. Este menu possui 3 botões: um para selecionar o nível 1 (“Level 1”), outro para o nível 2 (“Level 2”) e, por último, para o nível multiplayer (“Co-op”) . Estes botões devem ser pressionados com o rato, à semelhança dos do menu inicial, e direcionam o utilizador ao nível escolhido.

Fig 2. Menu dos níveis

2.2 Níveis

Neste jogo, é possível selecionar qual dos 3 níveis existentes o utilizador pretende experimentar. Cada um destes níveis possui um labirinto diferente com soluções distintas.

Além disso, as cores do labirinto mudam conforme a hora do dia. O labirinto de cada nível possui 3 paletas de cores: a primeira dura das 6h às 14h, a segunda das 14h às 20h e a terceira das 20h às 6h.

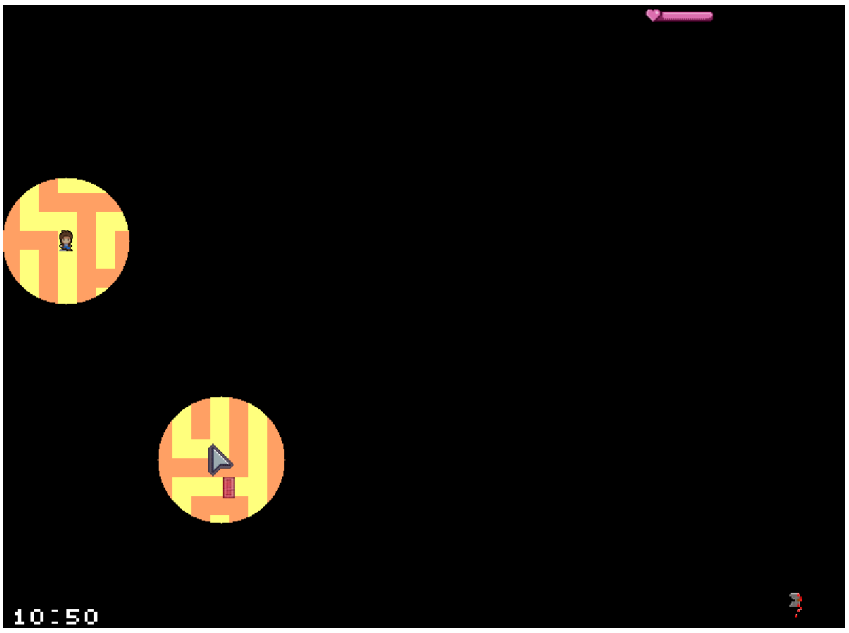

Fig 3. Nível 1 (Primeira paleta)

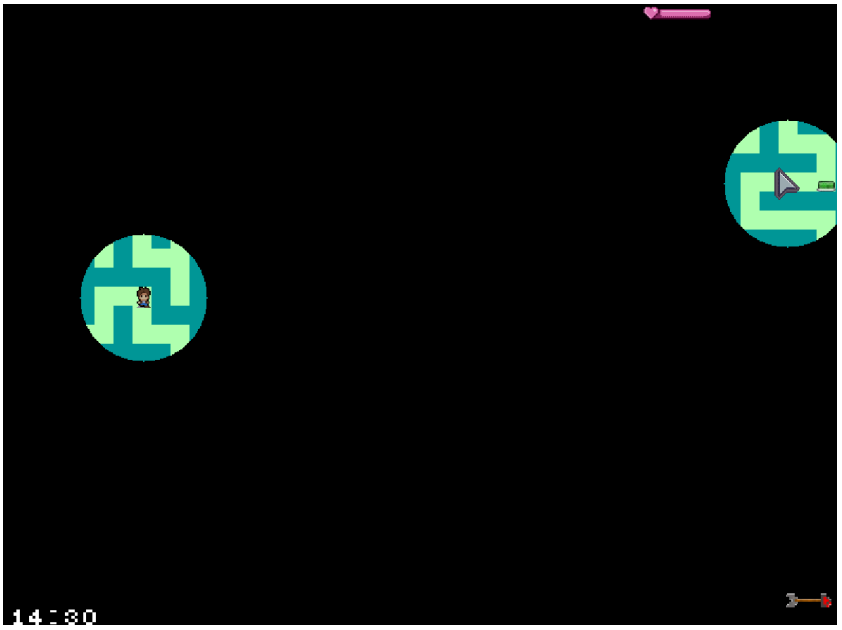

Fig 4. Nível 1 (Segunda paleta)

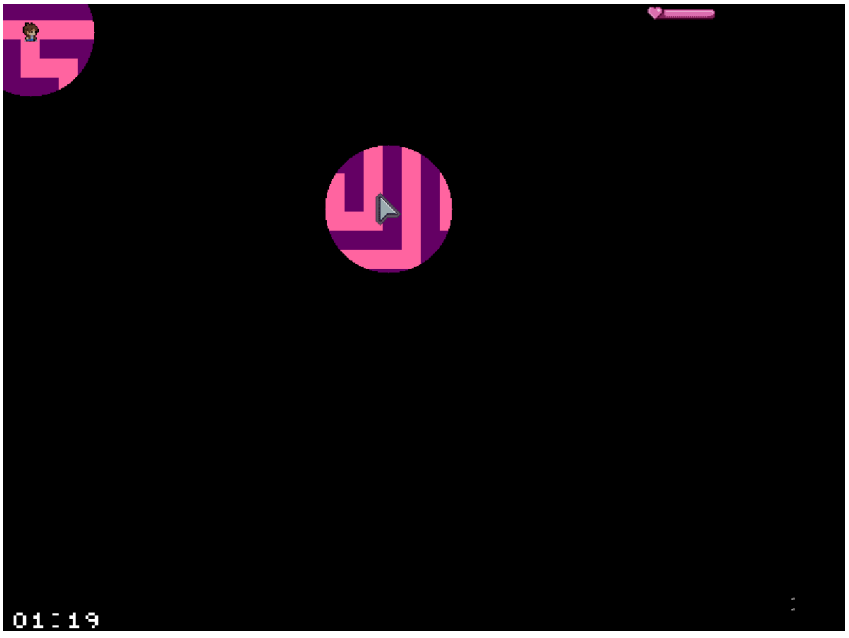

Fig 5. Nível 1 (Terceira paleta)

2.3 Labirinto

No labirinto, a sprite do player começa situada no canto superior esquerdo e o utilizador deve movê-la com o teclado para tentar chegar à saída que se encontra no canto inferior direito. Para mover para cima a tecla utilizada é 'W', para baixo 'S', para a esquerda 'A' e para a direita 'D'.

O utilizador só consegue ver uma pequena parte do labirinto, portanto, para conseguir visualizar mais pode mover o rato que possui uma lanterna. Além disso, deve descobrir botões onde pode atravessar, para desbloquear certas portas que se encontram no caminho.

Por fim, se não conseguir chegar à saída antes de o tempo terminar, é direcionado para a tela de game over mas se conseguir vai para a tela de vitória.

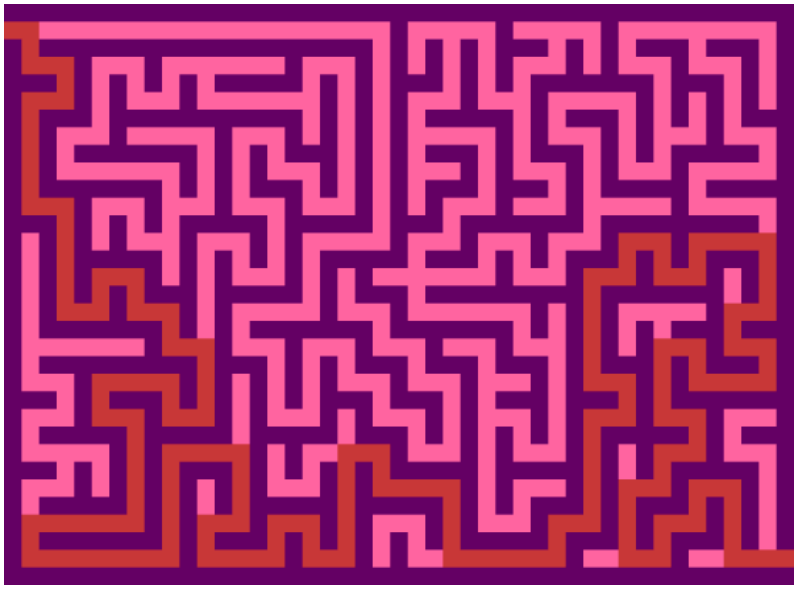

Fig 6. Solução do labirinto 1

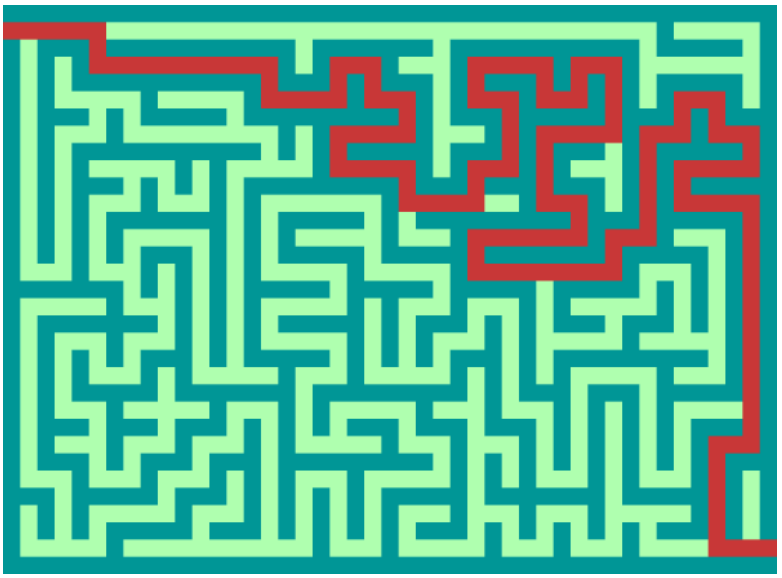

Fig 7. Solução do labirinto 2

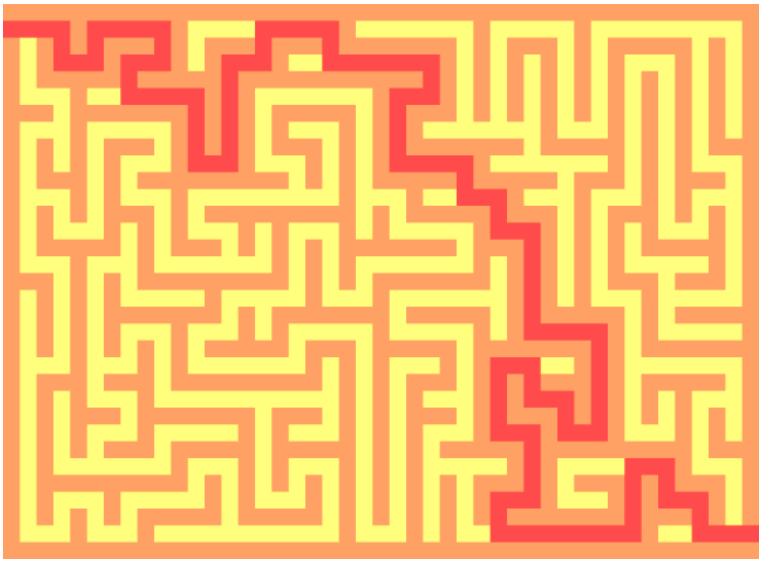

Fig 8. Solução do labirinto cooperativo

2.4 Tela de vitória

Esta tela aparece caso o utilizador consiga chegar à saída antes que o tempo de jogo acabe, mostrando o tempo que levou para finalizar o labirinto. Deve pressionar a tecla “Esc” para sair do jogo.

Fig 9. Tela de vitória

2.5 Tela de Game Over

Esta tela aparece caso o tempo de jogo acabe e o utilizador ainda não tenha conseguido chegar à saída. Deve pressionar a tecla “Esc” para sair do jogo.

Fig 10. Tela de Game Over

3. Estado do projeto

3.1 Funcionalidades

- Exploração dos menus e seleção de botões - mouse e video card
- Movimentação de uma personagem no jogo - keyboard e video card
- Descoberta do labirinto - keyboard, mouse e video card
- Ativação de botões e abertura de portas no labirinto, assim como a sua representação gráfica - keyboard e video card
- Seta a indicar o local de saída do labirinto - timer e video card
- Representação gráfica do tempo que falta para terminar o jogo - timer e video card
- Alteração das horas e das cores do labirinto - rtc e video card
- Comunicação entre dois jogadores - serial port
- Exibição das horas e dos minutos na tela do jogo - rtc e video card
- Exibição do tempo que o utilizador demorou a completar o labirinto - timer e video card.

3.2 Tabela de dispositivos

Dispositivo	Para que serve?	Interrupção / Polling
Timer	Determinar a duração do jogo e para regular a alteração de determinadas sprites ao longo do tempo	Interrupção
Keyboard	Mover as personagens que se encontram no jogo com as teclas A, W, S e D	Interrupção
Mouse	Selecionar opções do menu principal e do menu de níveis	Interrupção
Video Card	Exibir toda a interface do jogo e dos menus	Nenhum
RTC	Alterar o tema do jogo de acordo com as horas atuais	Interrupção
Serial Port	Comunicação do jogo entre dois jogadores (multiplayer)	Interrupção

3.2.1 Timer

O timer é utilizado para determinar o tempo de cada jogo.

As funções implementadas para o Timer encontram-se no ficheiro timer.c.

3.2.2 Keyboard

O keyboard é utilizado para a movimentação dos personagens no jogo. Após a obtenção de um determinado **makecode**, se este tiver relevância no jogo (neste caso, 'A', 'W', 'S' ou 'D'), será efetuado um movimento na direção correspondente ('A' para a esquerda, 'D' para a direita, 'W' para cima e 'S' para baixo) na sprite controlada pelo jogador. A sprite é alterada sempre que ocorre uma mudança de direção. Quando o **breakcode** é acionado, o movimento é interrompido e a sprite é trocada. Além disso, podemos utilizar o teclado para sair do jogo ao pressionar a tecla 'ESC'.

As funções implementadas para o Keyboard encontram-se nos ficheiros keyboard.c e KBC.c.

3.2.3 Mouse

A cada interrupção do rato, o cursor do rato no jogo é atualizado de acordo com o movimento aplicado pelo jogador, permitindo selecionar ou apontar para uma determinada posição a qualquer momento. Durante o jogo, quando a opção selecionada do menu de níveis é "Level1" ou "Level2", o rato é usado como se fosse uma lanterna para facilitar a visualização do mapa que se encontra escondido. Também utilizamos o cursor do rato para selecionar qualquer uma das opções do menu principal e do menu de níveis.

As funções implementadas para o Mouse encontram-se no ficheiro mouse.c.

3.2.4 Video Card

O Video Card é utilizado para exibir todos os fundos existentes no jogo, os botões necessários para a navegação dos menus, as personagens controladas pelos jogadores, a barra de tempo (que indica quanto tempo falta para o jogo

terminar), uma seta que indica a saída, as portas e botões que permitem avançar pelo labirinto.

A resolução utilizada no nosso projeto é de 800x600 pixels, no modo 0x115. O modelo de cores é direto e os bits per pixel (R:G:B) são 24 (8:8:8), logo com 2^{24} (cerca de 17 milhões) cores possíveis.

Para a renderização dos sprites, aplicamos a técnica de double buffering. Temos dois buffers e um apontador draw buffer que aponta para o buffer que não está sendo exibido no momento, ou seja, onde vamos desenhar o que será mostrado a seguir. Isso garantiu que o processamento das teclas para mover a personagem e do cursor do mouse na tela fosse exibido de forma fluida e jogável.

Assim, nos menus copiamos diretamente o background para o draw buffer usando memcpy na nossa função draw_background e para todos os outros sprites executamos a pintura no draw buffer do mapa de cores obtido pelo load_xpm do sprite na função drawing_sprite. O page flipping é feito através da update_flip_frames, que faz a troca dos buffers com uma VBE function 0x07 (Set Start of Display).

Foi implementado um sistema que verifica a cor do pixel para onde a personagem se quer mover para detectar as colisões da mesma com as paredes do labirinto e com as portas existentes pela função check_collision(). A personagem utiliza diversos sprites animados que mudam conforme a direção do movimento, criando a ilusão de animação.

Também foi criada uma função para desenhar as horas no canto inferior esquerdo do ecrã, atualizando-as dígito por dígito, de acordo com a hora atual. Por último, a paleta de cores, associada a cada labirinto, é alterada a determinadas horas do dia.

As funções implementadas para o Video Card encontram-se no ficheiro graphics.c.

3.2.5 RTC: Real Time Clock

O RTC é usado principalmente para determinar o tema do jogo/labirinto de acordo com as horas. Entre as 6h e as 14h, o ambiente do jogo é diurno, dispondo de uma paleta em tons claros. Entre as 20h e as 6h, o ambiente é noturno, com uma paleta em tons de rosa e roxo. Nas restantes horas, o labirinto mantém uma paleta de cores em tons de verde, simulando o período da tarde.

Os registos de data e hora do RTC são exibidos de forma intuitiva no ecrã do jogo, atualizando as variáveis do projeto que contêm esses valores a cada minuto através da interrupção de atualização (rtc_update_time_info()).

As funções implementadas para o RTC encontram-se no ficheiro rtc.c.

3.2.6 Serial Port

No modo cooperativo de jogo, a comunicação entre máquinas virtuais (VMs) é assegurada pelo serial port. Consiste num protocolo de comunicação que foi implementado para transmitir dados, que neste caso se resumem aos scancodes que indicam a direção do movimento de cada personagem.

Para garantir a integridade e a ordem correta dos dados recebidos, foi desenvolvida uma fila de espera. Essa fila impede a perda de dados e assegura que as informações do teclado sejam processadas sequencialmente, sem interrupções por dados intermediários.

Para além disso, criamos uma rotina de sincronização para conectar as VMs corretamente no início do jogo (sp_handle_start_multi()).

De seguida, o processamento das informações recebidas e a gestão da fila de espera são manipulados pela função sp_handle_received_info(), que decodifica os dados de entrada e executa as ações correspondentes, assegurando uma comunicação eficiente e sem falhas entre as VMs durante o jogo.

As funções implementadas para o Serial Port encontram-se no ficheiro

serialPort.c.

4. Organização e estrutura do código

4.1 Módulo do Timer

Este módulo inclui as funções desenvolvidas no Lab2 para o dispositivo Timer, que abrangem funcionalidades relacionadas à subscrição de interrupções e à incrementação do tempo de jogo.

Percentagem: 8 %

Contribuidores:

- Beatriz Bernardo
- Diana Nunes
- Sofia Gonçalves
- Teresa Mascarenhas

4.2 Módulo do Keyboard

Este módulo inclui as funções desenvolvidas no Lab3 para o dispositivo Keyboard, que abrangem funcionalidades relacionadas à subscrição de interrupções e leitura dos scancodes do keyboard.

Percentagem: 6 %

Contribuidores:

- Beatriz Bernardo
- Diana Nunes
- Sofia Gonçalves
- Teresa Mascarenhas

4.3 Módulo do Mouse

Este módulo inclui as funções desenvolvidas no Lab4 para o dispositivo

Mouse, que abrangem funcionalidades relacionadas à subscrição de interrupções e leitura dos pacotes enviados pelo mouse.

Percentagem: 4 %

Contribuidores:

- Beatriz Bernardo
- Diana Nunes
- Sofia Gonçalves
- Teresa Mascarenhas

4.4 Módulo do Graphics

Este módulo inclui as funções desenvolvidas no Lab5 para o dispositivo Graphic Card, que abrangem funcionalidades relacionadas à configuração do modo gráfico, à pintura de pixels e page flipping.

Percentagem: 7 %

Contribuidores:

- Beatriz Bernardo
- Diana Nunes
- Sofia Gonçalves
- Teresa Mascarenhas

4.5 Módulo do RTC

Este módulo inclui funções desenvolvidas para o dispositivo RTC, que abrangem funcionalidades relacionadas à subscrição de interrupções e à atualização da hora

Percentagem: 5 %

Contribuidores:

- Teresa Mascarenhas

4.6 Módulo da Serial Port

Este módulo trata tanto de todas as interrupções relacionadas com o serial port, como das funcionalidades relacionadas com a transmissão de dados entre duas VM's.

Percentagem: 13 %

Contribuidores:

- Teresa Mascarenhas
- Sofia Gonçalves

4.7 Módulo do Sprite

Neste módulo podemos encontrar funções responsáveis pela construção de sprites e carregamento de ficheiros xpm. Ademais, nesta classe ainda são implementadas funções responsáveis pelo desenho da lanterna (que nos permitirá visualizar, parcialmente, o labirinto). A estrutura desta classe foi inspirada nos slides do professor Pedro Souto.

Percentagem: 10 %

Contribuidores:

- Beatriz Bernardo
- Diana Nunes
- Sofia Gonçalves
- Teresa Mascarenhas

4.8 Módulo do Logic

Neste módulo encontram-se todas as funções auxiliares que lidam com os diferentes tipos de interrupções.

Percentagem: 8 %

Contribuidores:

- Diana Nunes
- Sofia Gonçalves
- Teresa Mascarenhas

4.9 Módulo do Game

Este módulo, além de gerenciar toda a interface do jogo, trata do movimento da personagem, controla as entradas do teclado e/ou rato, e detecta colisões com as paredes.

Para além disso, é responsável pela animação da personagem, atualizando os sprites animados conforme a personagem se move, alternando para o próximo sprite após um determinado número de chamadas de função.

Percentagem: 26 %

Contribuidores:

- Beatriz Bernardo
- Diana Nunes
- Sofia Gonçalves
- Teresa Mascarenhas

4.10 Módulo do KBC

Neste módulo os *"controllers"* do KBC são tratados, encapsulando o keyboard e o mouse.

Percentagem: 4 %

Contribuidores:

- Beatriz Bernardo
- Diana Nunes
- Sofia Gonçalves
- Teresa Mascarenhas

4.11 Módulo da Queue

Este módulo implementa uma queue (fila - estrutura de dados) em C. Para a sua implementação, seguimos os passos deste tutorial: Creating a Queue in C | DigitalOcean

Percentagem: 6 %

Contribuidores:

- Teresa Mascarenhas

4.12 Módulo do Proj

Este módulo é responsável pela inicialização, configuração e controle do loop principal do jogo. Ele gerencia o hardware através de interrupções, integrando também a lógica do jogo.

Percentagem: 3 %

Contribuidores:

- Diana Nunes
- Sofia Gonçalves

4.13 Function call graph

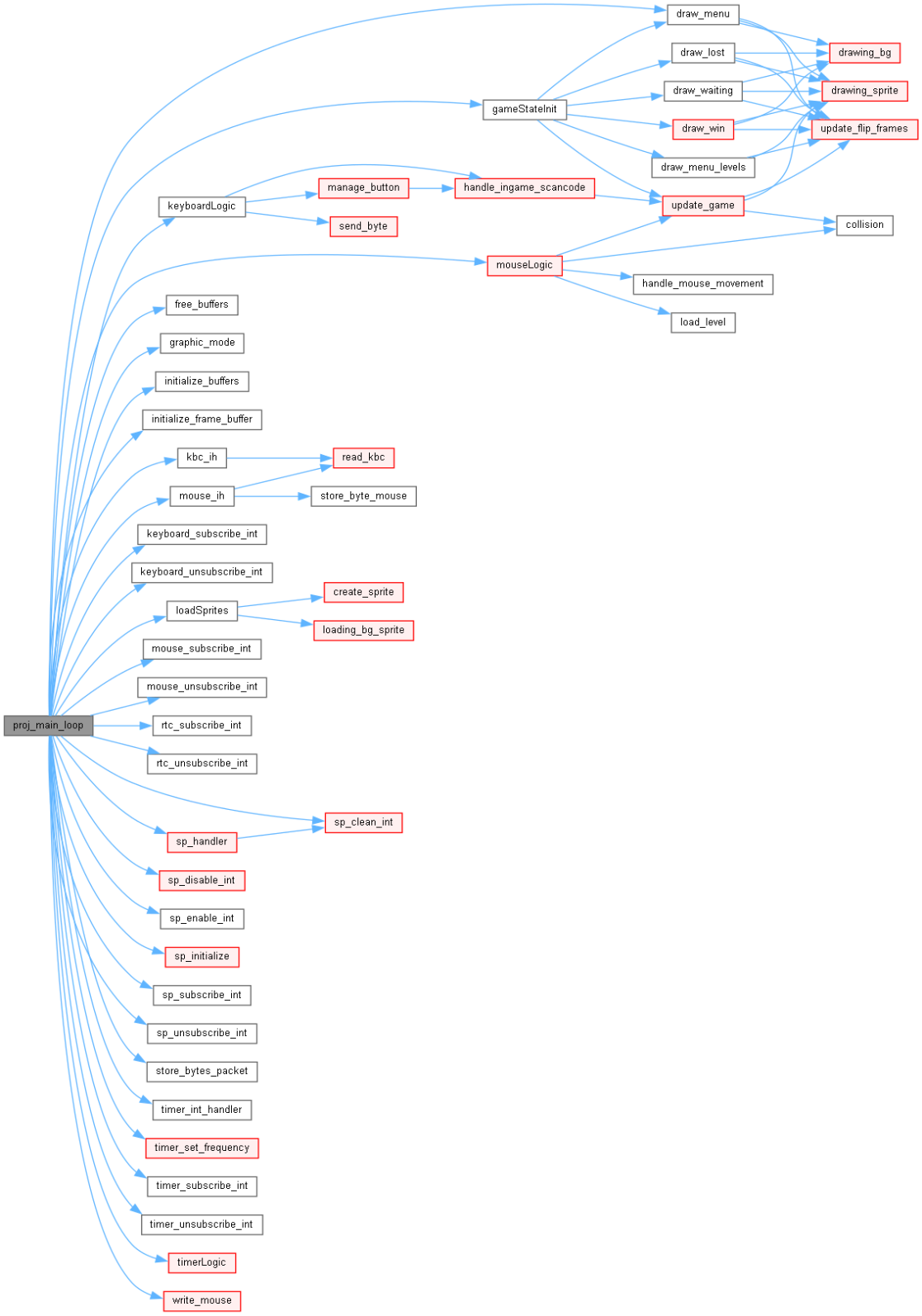

Fig 11. Function call graph

5. Detalhes da implementação

5.1 Máquina de estados

O jogo é composto por vários estados: MENU, LEVELS, GAME, MULTI, WIN, LOSE, EXIT. Estes estados estão definidos numa enumeração chamada "GameState". O estado inicial do jogo é o MENU definido no main loop que se encontra no proj.c.

As transições de estados ocorrem no ficheiro "logic.c" e são gerenciadas por uma variável booleana chamada "gameState_change". Quando ocorre uma mudança de estado, essa variável é definida como verdadeira. Caso "gameState_change" seja verdadeira, o novo estado é inicializado.

- No estado MENU, o utilizador pode escolher entre as opções Start e Exit. Selecionar Start levará o utilizador ao estado LEVELS, enquanto selecionar Exit o levará ao estado EXIT, que irá terminar o programa.
- No estado LEVELS, o utilizador pode escolher o nível que deseja jogar. Selecionar Level1 ou Level2 mudará o estado para GAME, enquanto selecionar co-op mudará o estado para MULTI.
- Os estados WIN e LOSE são determinados com base no sucesso

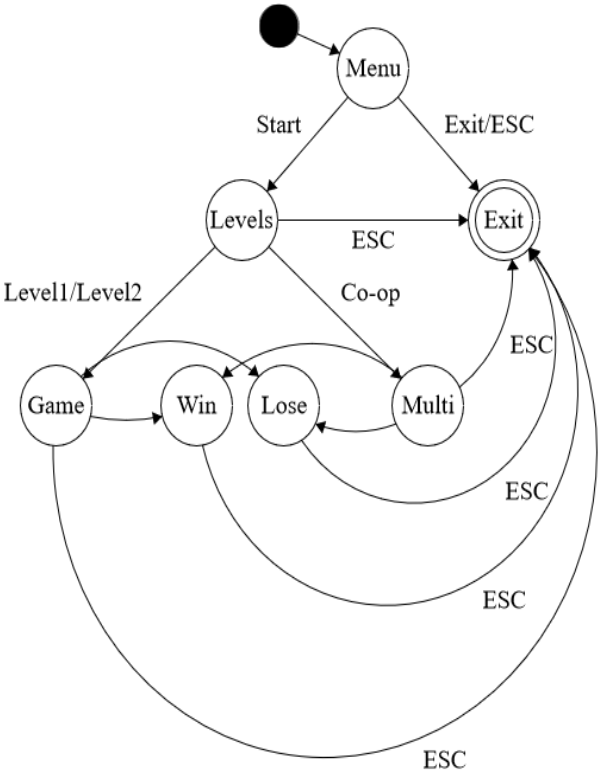

Fig 12. Máquina de estados

ou fracasso do utilizador em completar o labirinto dentro do tempo estipulado.

5.2 Layering

Neste projeto, foi aplicado o conceito de Layering (camadas) para organizar o código de forma mais estruturada e modular. O código foi dividido em três camadas principais: Apresentação, Lógica e Dados.

Camada de Apresentação (proj.c) :

- Gere o loop principal do jogo;
- Gere as interrupções do hardware;
- Chama a função correspondente na camada de lógica.

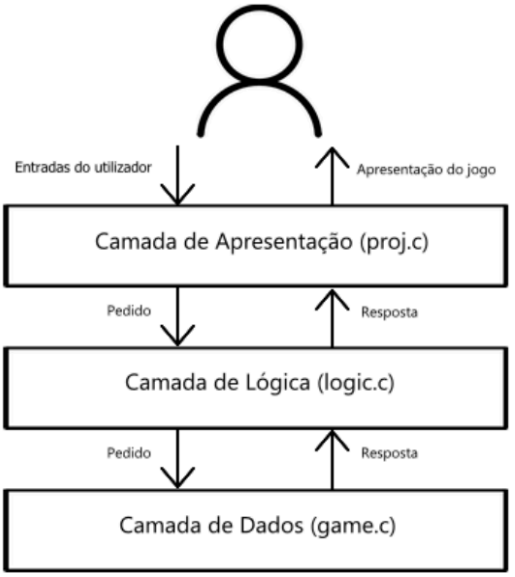

Fig 13. Camadas do código

Camada de Lógica (logic.c) :

- Processa as chamadas recebidas da camada de apresentação, aplicando as regras do jogo ao chamar as funções correspondentes da camada de dados.
- Atualizar o estado do jogo (“GameState”).

Camada de Dados (game.c) :

- Fornece funções para acessar e modificar os elementos do jogo.
- É chamada pela camada de lógica para obter dados necessários e atualizar o estado do jogo conforme as regras aplicadas.

5.3 Event-driven Code

O código é também "Event-Driven", ou seja, ele responde a eventos que ocorrem no sistema, como por exemplo:

- O loop principal continua a ser executado até que o jogador pressione a tecla ESC para sair.
- Dentro do loop principal, o programa aguarda a receção de interrupções de hardware. Quando uma interrupção é recebida, o programa determina qual dispositivo gerou a interrupção e executa o tratamento e a lógica do jogo correspondentes.
- Quando o jogador decide sair do jogo, o código desativa todas as assinaturas de interrupção e realiza outras tarefas de limpeza necessárias antes de encerrar a execução.

Desta forma, fica claro que o código segue o paradigma "Event-Driven", uma vez que sua execução é orientada pelos eventos ocorridos, como entradas do utilizador, sinais de temporizador, entre outros, ao invés de seguir uma sequência linear de instruções.

5.4 Sprite

Os objetos do tipo Sprite armazenam todas as informações necessárias sobre as imagens que queremos desenhar: a sua posição na tela com coordenadas x e y, a altura e largura da imagem, a velocidade (caso se mova) e o mapa de cores obtido pelo load_xpm.

No início do jogo, carregamos o xpm e criamos todos os sprites utilizados, exceto a do mapa do labirinto, que só é carregado quando o nível é escolhido. Posteriormente para a pintura dos sprites só é necessário aceder ao mapa da sprite específica, o que torna este processo mais eficiente.

5.5 Object orientation

Neste projeto foi também utilizada a orientação a objetos. Como podemos ver no tópico 5.4, a estrutura Sprite funciona como uma classe, visto que encapsula dados relacionados a um sprite (análogo aos atributos de uma classe), havendo várias instâncias desta estrutura (análogo a objetos).

1. **Encapsulamento**: Os dados e as funções que operam sobre esses dados estão agrupados logicamente.
2. **Abstração**: As funções fornecem uma interface clara para criar, destruir e manipular sprites sem expor os detalhes de implementação.
3. **Modularidade**: Cada sprite e as suas operações são tratadas como uma unidade separada, facilitando a manutenção e o desenvolvimento do código.

5.6 Colisões

Para lidar com as colisões da personagem com as paredes do labirinto, comparamos a cor de pontos específicos no labirinto para onde a personagem pretende se mover com a cor do chão. Se for diferente, significa que há uma parede ou que está fora da tela, e então, não se desloca.

Os pontos que são verificados dependem da direção para a qual a personagem se quer movimentar:

- Para cima, o A e o B.
- Para baixo, o C e o D.
- Para a direita, o B e o D.
- Para a esquerda, o A e o C.

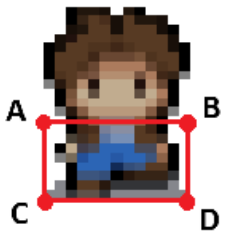

Fig 14. Hitbox do player

5.7 Frame Generation

Neste projeto, a frame generation (geração de imagens) é realizada maioritariamente pelo módulo do Graphics (graphics.c) e Sprite (sprite.c).

- **Inicialização**:
  - Configura os buffers de vídeo e define o modo gráfico.
  - Aloca memória para o buffer de fundo.
- **Loop de Renderização**:
  - No loop principal, (proj_main_loop) os eventos de hardware são processados para atualizar o estado, sendo que os sprites que precisam de ser desenhados no quadro atual são determinados na lógica do jogo consoante esse estado.
- **Desenho do Quadro**:
  - Copia o fundo para o buffer de desenho.
  - Desenha os sprites no buffer de desenho.
- **Page Flipping**:
  - Alterna os buffers, trocando o buffer de desenho e o buffer de exibição.
  - Limpa o novo buffer de desenho para prepará-lo para o próximo quadro.

Esse processo garante que os quadros sejam gerados e exibidos de maneira eficiente, proporcionando uma experiência de jogo contínua e sem interrupções perceptíveis.

5.8 RTC

A implementação do RTC é feita principalmente através da função rtc_update_time_info(), que atualiza a estrutura rtc_info time_info com o tempo atual lido do RTC. Essa estrutura é então usada para alterar as cores do labirinto e exibir as horas durante o jogo.

A função rtc_update_time_info() lê os registros do RTC que contêm informações sobre segundos, minutos, horas, dia, mês e ano. Ele interpreta esses valores de acordo com o modo de contagem binária ou BCD (Binary Coded Decimal) do RTC e armazena-os na estrutura time_info. Em seguida, os valores dessa estrutura são usados conforme necessário para exibir o tempo do jogo ou para fazer alterações nas cores do labirinto.

5.9 Serial port

A implementação da comunicação via porta de série (serial port) é feita principalmente através da função sp_initialize(), que configura a porta de série e inicializa as filas de recepção e envio. Essa configuração permite a comunicação entre dispositivos durante o jogo.

A comunicação é gerenciada principalmente através das funções receive_byte() e send_byte(). A função receive_byte() lê dados do registrador de buffer de recepção (RBR) e armazena-os na fila de recepção, enquanto a função send_byte() coloca dados na fila de envio e inicia a transmissão se a fila estiver vazia.

Durante o jogo, a função manage_button() é usada para enviar códigos de teclas pressionadas através da porta de série. Ela converte os códigos de tecla em códigos de transmissão e envia-os usando send_byte(). Isso permite que os eventos de tecla sejam compartilhados entre dispositivos.

As interrupções da porta de série são gerenciadas pela função sp_ih(), que lida tanto com a recepção quanto com a transmissão de dados, chamando as funções apropriadas com base no tipo de interrupção detectada.

Além disso, a função sp_handle_start_multi() gerencia o início do jogo em modo multijogador, sincronizando os dispositivos através da troca de códigos específicos.

6. Conclusões

Através da realização deste projeto conseguimos colocar em prática os conhecimentos que nos foram transmitidos, relativamente aos periféricos de um computador, quer seja no que diz respeito ao seu funcionamento, como à sua gestão.

Como o projeto desenvolvido é um jogo, tivemos que estruturar o nosso programa em torno da noção de estados do jogo. Assim, tal como abordado nas aulas teóricas, decidimos implementar uma máquina de estados.

Nesse sentido, fomos obrigados a desenvolver uma nova forma de pensamento crítico e de programação, bastante diferente do que estávamos habituados. Além disso, o facto de o nosso tema de jogo ser pouco convencional para este tipo de projetos exigiu que pensássemos mais profundamente sobre como concretizar ideias que inicialmente eram apenas abstrações.

Enfrentamos algumas dificuldades ao longo do projeto, sendo as principais a implementação do page flipping e do serial port. Sendo assim, no que diz respeito ao page flipping, consideramos que foi algo que surgiu da necessidade de tornar o nosso jogo mais eficiente, e de modo a evitar uma renderização excessiva de toda a interface.

Para concluir, apesar das adversidades enfrentadas, sentimos que o projeto foi bem-sucedido, pois conseguimos implementar as funcionalidades a que nos propusemos.