Faculdade de Engenharia da Universidade do Porto

MAGE SHOWDOWN

Projeto Final - LCOM - Turma 9 - Grupo 03

Realizado por:

Manuel Silva, up202108874

Maria Laranjeira, up202004453

Pedro Fonseca, up202108653

Índice

Introdução

Para o nosso projeto final, decidimos criar um jogo chamado "Mage Showdown", onde os jogadores assumem o papel de magos que lançam feitiços para derrotar os monstros e alcançar o final do jogo. O jogo será apresentado em uma perspectiva 2D, com uma visão Top-Down (vista de cima para baixo).

O jogo teria dois modos: Singleplayer e Multiplayer. No modo Singleplayer, o principal objetivo será passar por todos os níveis. Dentro de cada nível, o jogador deveria alcançar a zona final, eliminando os monstros.

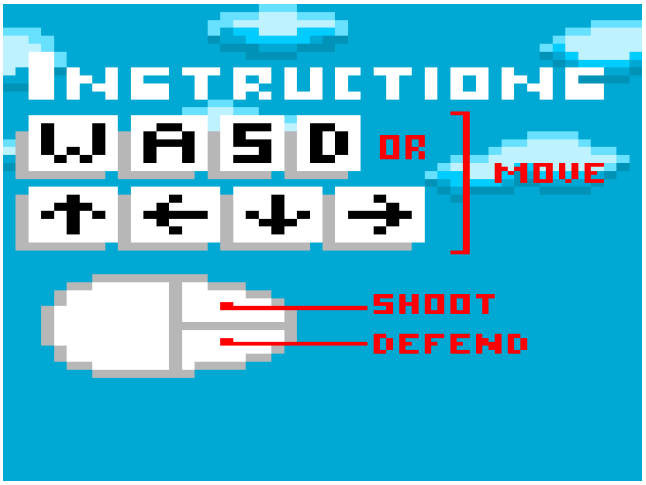
No modo Multiplayer, os jogadores participariam numa arena onde o seu objetivo seria eliminarem-se um ao outro. Cada jogador teria três vidas, e o primeiro jogador a perder todas as suas vidas seria derrotado. A batalha entre os magos é intensa e estratégica, com cada jogador utilizando seus poderes mágicos para superar seus oponentes.

Instruções de Utilização

1.1 Menu Inicial

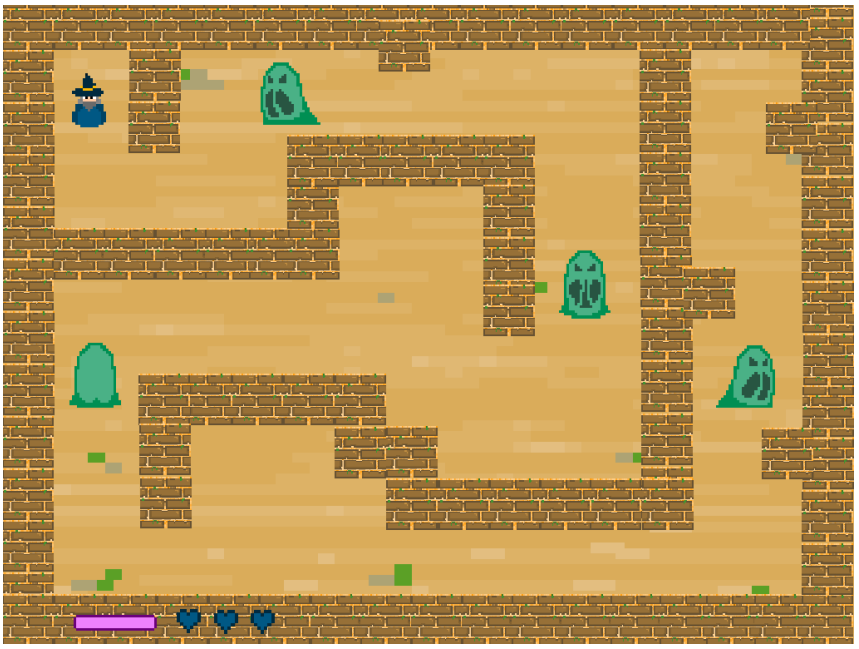
Ao iniciar o jogo, é mostrado inicialmente o menu inicial, onde é possível escolher o modo de jogo desejado (Single-Player ou Multi-Player), ir para a secção Help (para se informar sobre os controlos) ou sair do jogo (clicando em Exit). Para navegar neste menu deve ser utilizado os movimentos do rato para mover o cursor, e o clique no botão esquerdo para selecionar a opção desejada.

1.2 Instruções



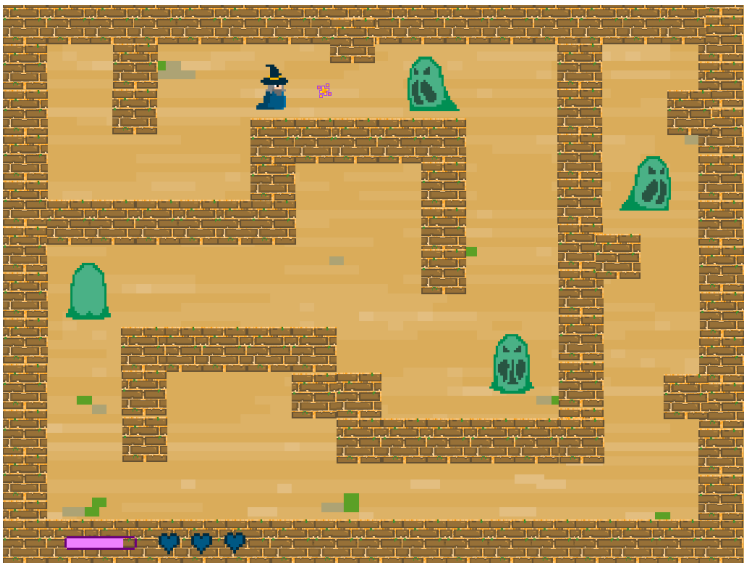
1.3 Single Player

1.3.1 Map



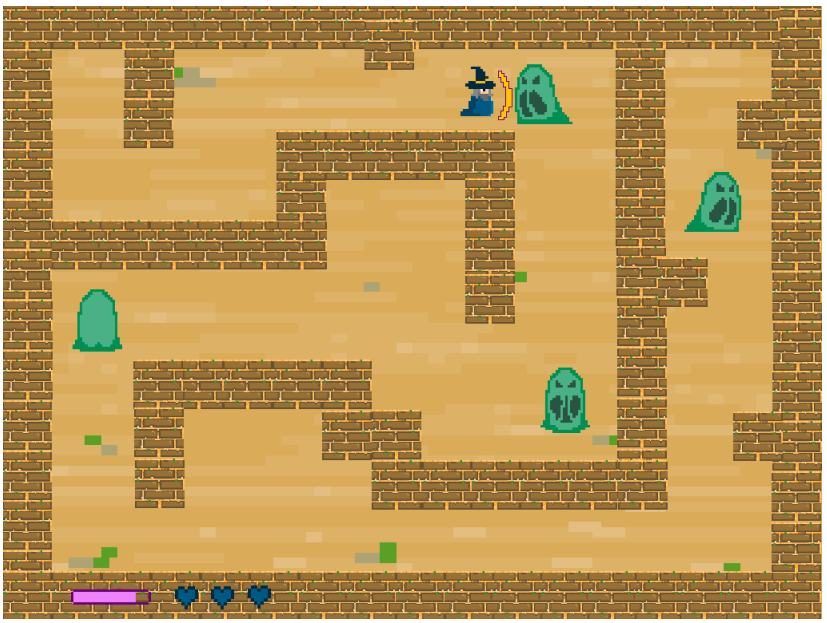
Quando se inicia o jogo no modo Single-Player, é carregado o primeiro nível que consiste numa espécie de labirinto infestado por monstros.

O jogador move-se com o teclado usando as teclas WASD ou com as setas. De modo a completar o nível, o jogador deve limpar o labirinto, matando todos os monstros. Para tal, o jogador deve lançar os seus poderes mirando no inimigo e pressionando o botão esquerdo do rato.



O jogador também tem a possibilidade de se defender do dano dos monstros invocando um

escudo mágico:



Contudo todas estas ações gastam Mana, que é representada pela barra roxa no canto inferior esquerdo. A Mana regenera com o tempo mas é gasta muito mais rapidamente do que é regenerada, por isso há que a gerir estrategicamente.

1.4 Multiplayer

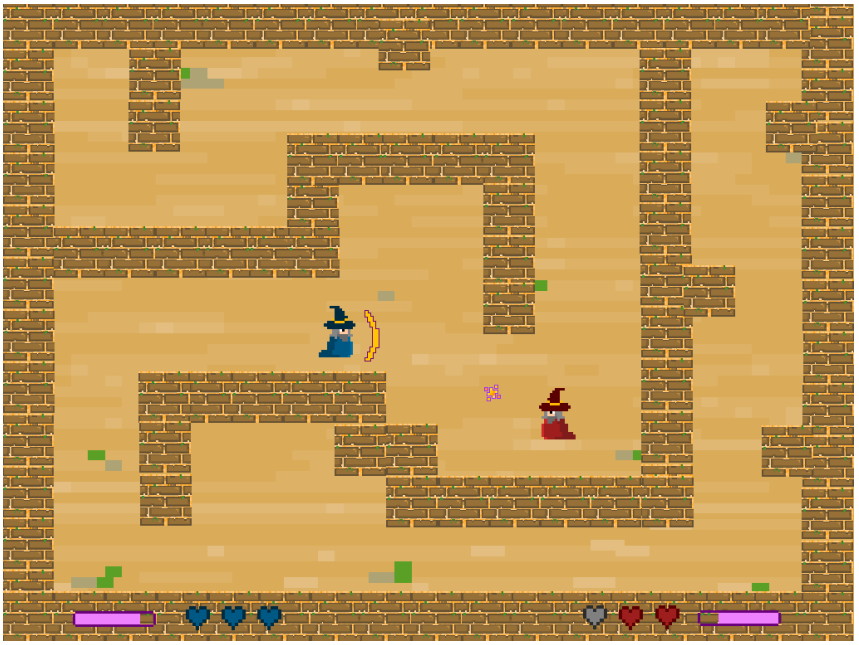
1.4.1 Waiting

Quando é carregado no botão Multi-Player, o utilizador entra numa sala de espera, onde se espera que haja outro utilizador que se junte para, nesse momento, ser iniciado o jogo.

O utilizador pode a qualquer momento retornar ao menu inicial carregando na tecla ESC.

1.4.2 Game

Quando estão os dois jogadores conectados, o jogo é carregado, sendo o jogador 1 (o primeiro a aceder o modo Multi-Player) o jogador azul, e o jogador 2 (segundo jogador a aceder o modo) o jogador vermelho.

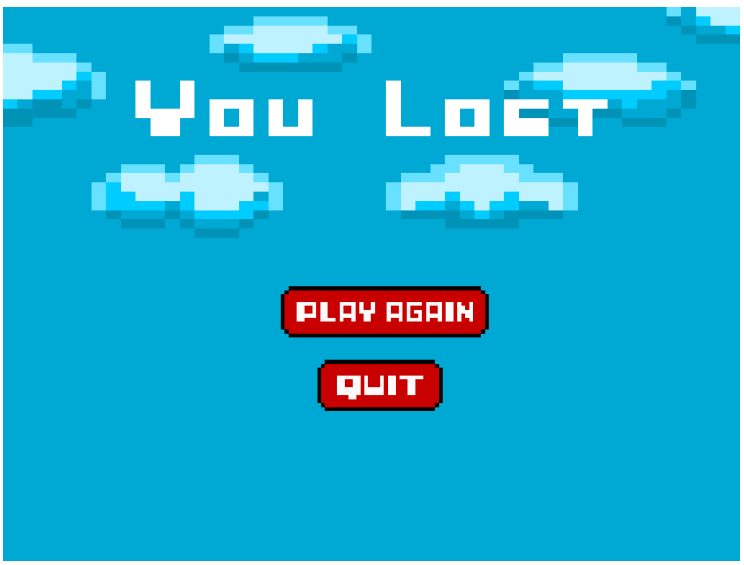


O objetivo é simples: Tentar Matar o mago adversário atingindo-o 3 vezes, de modo a esgotar os seus 3 corações. O mago que prevalece, ganha o jogo indo para o menu GameWin (secção 1.4). O jogador que perder vai para o menu GameOver (secção 1.4). Se a qualquer momento o utilizador quiser voltar ao menu inicial apenas precisa de premir a tecla ESC.

1.5 Won / Lost

Quando o jogador completa o nível ou ganha no modo Multi-Player é carregado o menu de GameWon e quando o jogador perde em qualquer um dos modos é carregado o menu do GameOver.

Em single player, se no menu de GameWon, o jogador deve então carregar no botão PLAY AGAIN para jogar de novo ou QUIT para retornar ao menu inicial. Se no menu GameOver, o jogador deve pressionar o botão PLAY AGAIN para tentar novamente o nível ou QUIT para retornar ao menu inicial.



Estado do Projeto

DISPOSITIVO	FUNCIONALIDADE	INTERRUPÇÕES
Timer	Controlo do frame-rate. Gerir eventos no jogo. Animações.	Sim
Keyboard	Movimentos do jogador. Navegar nos menus.	Sim
Mouse	Navegar nos menus. Disparo de poderes.	Sim
Graphics Card	Apresentar a interface do jogo.	Não
Real Time Clock	Apresentar o dia e a hora no fim para contar tempo de jogo.	Sim
Serial Port	Ainda em produção.	Sim

1.6 Timer

O timer é usado para controlar o frame-rate fazendo atualizações no ecrã 60 vezes por segundo.

No modo play, o timer é usado para controlar a mana da qual os poderes do mago dependem. O jogador tem disponível uma barra cheia de mana no início do jogo que se gasta por cada poder lançado e por cada escudo invocado.

No modo multiplayer, o timer funciona da mesma maneira, sendo que os dois jogadores devem saber gerir a mana estrategicamente.

O timer é usado para controlar a regeneração dos níveis de mana, sendo esta regenera um certo valor a cada segundo, mas a uma velocidade muito mais lenta que a velocidade a que se gasta.

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

1.7 Keyboard

O keyboard é usado para movimentar o jogador usando as teclas WASD ou as setas. Foram criadas variáveis bool para cada direção estática, para controlar o movimento do jogador, de acordo com o make-code ou o break-code do teclado e o jogador em questão. As funções relacionadas com o keyboard encontram-se no **kbd.c**.

1.8 Mouse

O rato é usado para navegar nos menus usando um cursor. Através da função `mouse_events` é obtido o evento do rato, sendo possível saber se foi premido algum botão e desta forma facilitar a navegação.

As funções relacionadas com o mouse encontram-se no **mouse.c** que é também o ficheiro onde guardamos informação sobre o nosso cursor e os seus movimentos.

1.9 Graphics Card

Para o projeto foi utilizado o modo de vídeo 0x118, 1024x768 com 16.8M de cores (8:8:8).

Foi usada a técnica de double buffering, em que a cada interrupção desenhamos Para uma memória auxiliar e nas interrupções do timer (de modo a controlar o frame-rate) copiamos o conteúdo do nosso buffer para a memória principal usando a função `load_double_buffer`.

Foram usados XPMs para todas as imagens presentes no jogo. Todas estas XPMs foram de autoria nossa. No modo multiplayer, o sprite dos dois jogadores é o mesmo mas com cores diferentes. Para fazer estes XPMs foi utilizado a aplicação PISKEL. Usamos também XPMs dos números para demonstrar o dia e a hora obtidos através do RTC.

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

1.10 RTC

O RTC é usado nos menus para obter o dia e a hora atual. São utilizadas as funções `calculate_digit_xpm` e `draw_time` para escolher e desenhar cada digito da hora atual. Estas funções são chamadas no início e fim de gameplay para calcular o tempo total de jogo.

1.11 Serial Port

No modo multiplayer, os dados são transferidos entre os dois computadores usando a serial port. Esses dados consistem em códigos previamente estabelecidos para que o outro computador saiba se o jogador está online ou não, que correspondem ao movimento do jogador (WASD ou setas) e tiro (clique com o botão esquerdo do mouse). A função interpret_data é executada assim que os dados são recebidos e altera as variáveis necessárias com os dados.

As interrupções subscritas são as do Received Data Transmitter Holding Empty Register e Line Status. O polling e as interrupções são usados em conjunto para diminuir a perda de informações e a produção de erros. O UART está configurado para enviar 8 bits por caractere com paridade ímpar em uma taxa de bits de 115200.

O ficheiro serial_port.c contém funções para leitura e escrita de e para a serial port. Juntamente com UART FIFOs, as filas (arquivo queue.c) foram construídas com o intuito de minimizar as perdas de informação.

3. Organização e Estrutura do Código

1.12 Timer Module (5%)

Neste ficheiro estão presentes as funções desenvolvidas no Lab2 relacionado com as interrupções do timer. Todos contribuíram igualmente para a realização deste módulo.

1.13 Keyboard Module (5%)

Neste ficheiro estão presentes as funções desenvolvidas no Lab3 relacionado com as interrupções do keyboard. Todos contribuíram igualmente para a realização deste módulo.

1.14 Mouse Module (4%)

Neste ficheiro estão presentes as funções desenvolvidas no Lab4 relacionado com as interrupções do mouse. Para além disto, foi adicionado o código relativo ao Cursor (estrutura de dados criada para guardar informação sobre o cursor nos menus), os seu movimentos, colisões e funções para o desenhar e limpar. Todos contribuíram igualmente para a realização deste módulo.

1.15 Graphics Card Module (6%)

Este módulo contém as funções desenvolvidas no Lab5 relativo à placa de vídeo. Para além disso tem também as funções encarregadas com a implementação do double buffer. Todos contribuíram igualmente para a realização deste módulo.

1.16 RTC Module (7%)

Módulo relacionado com o Real-Time Clock. Neste módulo encontram-se as funções de comunicação com o RTC, quer para subscrever as interrupções, quer para ler os registos e atualizar a hora e dia. Todos contribuíram igualmente para a realização deste módulo.

1.17 Serial Port Module (12%)

Neste ficheiro encontram-se presentes as funções relacionadas com a serial port: subscrever, configurar, enviar informação e ler informação. Todos contribuíram igualmente para a realização deste módulo.

1.18 Power Module (6%)

Neste módulo encontra-se a declaração da estrutura de dados Power que guarda informação relativa a um poder. Ambos contribuíram igualmente para a realização deste módulo.

1.19 Button Module (4%)

Módulo relativo à estrutura de dados Button, que guarda a informação do tipo de botão, posição, imagens, e que permite criar botões e desenhá-los. Todos contribuíram igualmente para a realização deste módulo.

1.20 Monster Module (6%)

Neste módulo estão presentes as funções relativas à estrutura Monster, que é usada para guardar informação sobre um inimigo no jogo. Permite também a gestão de todos os eventos relacionados com os inimigos, desde movimentos a colisões. Todos contribuíram igualmente para a realização deste módulo.

1.21 Game Module (10%)

Neste ficheiro está presente o ciclo de interrupções que permite a execução do programa. Foi implementada uma state machine quer para o estado do programa. Consoante o estado do jogo, será chamado o Interrupt Handler correspondente, que lida com a interrupção da forma pretendida. Para além disso, é neste módulo que se encontra o código que permite o funcionamento do modo Singleplayer. Todos contribuíram igualmente para a realização deste módulo.

1.22 Level Module (5%)

O objetivo deste módulo é a gestão dos níveis do jogo. Foi implementada uma estrutura de dados Level que guarda todas as informações relevantes sobre um nível. Estas informações são as imagens de background e paredes, número de monstros, assim como uma lista deles, e uma posição considerada o fim do nível. Desta forma, tornou a criação de novos níveis consideravelmente mais intuitiva. Todos contribuíram igualmente para a realização deste módulo.

1.23 Time Module (5%)

Aqui está presente uma estrutura Time, onde guardamos informação sobre o dia e a hora, com a ajuda do RTC, e para mostrá-la no ecrã. Todos contribuíram igualmente para a realização deste módulo.

1.24 Menu Module (7%)

Aqui está o código relacionado com o menu principal: os Interrupt Handlers respectivos do menu e as funções que permitem dar load. Todos contribuíram igualmente para a realização deste módulo.

1.25 GameOver Module (2%)

Aqui está o código relacionado com o modo GameOver, no caso do jogador perder, está presente o Interrupt Handler e as respectivas funções que permitem dar load. Todos contribuíram igualmente para a realização deste módulo.

1.26 GameWon Module (2%)

Aqui está o código relacionado com o modo Game Win, se o jogador ganhar, aqui está presente o Interrupt Handler e as funções que permitem dar load a este state. Todos contribuíram igualmente para a realização deste módulo.

1.27 Instructions Module (1%)

Aqui está o código relacionado com o modo Instructions, aqui está presente o Interrupt Handler e as funções que permitem dar load a este state. Todos contribuíram igualmente para a realização deste módulo.

1.28 Player Module (11%)

Neste ficheiro estão todas as funções relacionadas com a estrutura de dados Player, desde o seu movimento, animações, colisões, e toda a gestão em geral de todos os eventos relacionados ao jogador. Todos contribuíram igualmente para a realização deste módulo.

1.29 Queue Module (2%)

Módulo com a implementação das queues, úteis na leitura e transmissão de informação de e para a serial port. Todos contribuíram igualmente para a realização deste módulo.

Detalhes da Implementação

Foi implementada uma **state machine** (GameState) para saber em que estado o jogo está. A state machine permitiu chamar diferentes Interrupt Handlers dependendo do estado, o que facilitou o uso de apenas alguns dispositivos em certos momentos, e outros em diferentes ocasiões.

Na implementação dos menus, o nosso método consistiu em desenhar apenas um **fundo default** (sem elementos), e, a este, adicionar os botões pretendidos. Com isto, e tendo os sprites de todos os elementos pré-carregados, ficou fácil de dar display a cada menu pois era só desenhar o seu fundo respetivo, seguido dos botões necessários.Este método facilitou também a animação dos botões quando o cursor colide com estes, mudando a sua imagem.

As **colisões** são detectadas pixel a pixel para testar se há colisão ou não. As colisões com os inimigos é verificada com a posição do player, e, caso algum pixel esteja na mesma posição que um inimigo, o jogador perde. Nas colisões do cursor nos menus é usada a posição relativa do cursor, e é verificado se esta posição está dentro ou fora de algum botão. Caso esteja dentro de algum botão, é atualizado uma variável “current_selection”, existente em cada menu, e com esse valor, é alterada a imagem do botão correspondente..

Conclusões

Com muita pena nossa não nos foi possível desenvolver este trabalho com o rigor que gostaríamos. Devido a uma elevada carga de trabalhos, muitas das nossas ideias não puderam ser traduzidas em features do nosso jogo.

Nunca antes tínhamos desenvolvido um projeto desta dimensão sem utilizar o paradigma de programação orientada a objetos, pelo que este tipo de programação em C foi bastante desafiante sobretudo tendo em conta que programamos não só o jogo como todos os devices. Consideramos o projeto bastante interessante, já que programar desta maneira nos permitiu entender a raiz do funcionamento do software como um jogo, bem como manipulá-la: programar o teclado para desempenhar as funcionalidades que desejamos, a placa gráfica para desenhar da maneira que pretendemos, controlar o movimento do rato, ditar acontecimentos de acordo com o timer… etc. Contudo, consideramos que o facto de não haver informação suficiente que nos possa orientar não só no sentido de o que desenvolver mas também de como desenvolver é um grande ponto negativo, que causa grande desmotivação, e dificulta desnecessariamente o projeto. O nosso maior desafio foi, contudo, o debugging. Visto que através do MINIX não temos maneira de fazer debugging como faríamos num IDE tradicional, o nosso troubleshooting resumiu-se à colocação de vários prints pelo código e a leituras extensivas do trace.txt, o que na grande maioria das vezes apenas nos permitia localizar o problema mas não a sua razão de ser.