Elemental Escape

Projeto final

Licenciatura em Engenharia Informática e Computação

Laboratório de computadores

Turma 10 | Grupo 1:

Francisca Guimarães – up202004229

Inês Oliveira - up202103343

João Oliveira - up202108737

Pedro Magalhães - up202108756

2º Semestre

Ano Letivo 2022/2023

Índice

Introdução

Este relatório descreve o projeto desenvolvido no âmbito da Unidade Curricular de Laboratório de Computadores, da Licenciatura em Engenharia Informática e Computação. O projeto consistiu no desenvolvimento de um jogo de nome "Elemental Escape", com foco no modo *single-player*.

A temática central do jogo é baseada nos elementos da natureza, com destaque para os elementos Água e Fogo, baseado no jogo original “Fireboy & Watergirl”.

O objetivo do jogo é chegar às portas finais, escapando de todos os obstáculos, tendo de existir entreajuda entre a Água e o Fogo, para que um deles possa abrir as barreiras para o outro passar. Se os jogadores não conseguirem terminar o jogo num tempo limite, perdem.

1. Instruções de utilização do programa

1.1. Menu inicial

Figura 1 Menu principal do jogo

Ao iniciar o jogo, o menu inicial é apresentado, fornecendo diversas opções para o usuário. No menu, é possível selecionar entre jogar com duas pessoas no mesmo computador (*single-player*) ou uma em cada virtual box usando serial port (*multi-player*), aceder o menu de instruções ou sair do aplicativo na opção "QUIT". A escolha das opções é realizada através do movimento do cursor do mouse, sendo necessário clicar com o botão esquerdo para selecionar a opção desejada.

1.2. Single Player

Apesar de ser um modo single-player, o jogo obriga a participação de dois jogadores, ainda que joguem alternadamente. No início do jogo, ambos os elementos começam na mesma posição inicial do labirinto, sendo o fogo controlado pelas quatro setas do teclado e a água controlada pelas teclas A, W, S e D. Embora os jogadores joguem em sequência, eles vão ter de jogar em parceria, sendo que um não pode concluir o jogo sem o outro.

Figura 2 Início do jogo

Durante o jogo, quando um dos elementos, seja a Água ou o Fogo, se posiciona próximo às alavancas, aspeto necessário para as abrir, o jogador precisa interagir com o jogo clicando com o rato nas mesmas para ativá-las. Essas alavancas giram, abrindo as barreiras que bloqueavam o caminho, permitindo que o outro elemento avance. No entanto, após a primeira ativação, é necessário que o jogador que avançou também acione outra alavanca para manter a barreira aberta, permitindo que o elemento que a ativou inicialmente também possa passar.

Figura 4 Alavanca fechada

Figura 3 Alavanca aberta

Contudo, existem barreiras de água e fogo, sendo que nessas os respetivos elementos conseguem passar sem que as barreiras abertas.

Figura 5 Abertura das alavancas

Por fim, para que concluam o jogo com sucesso os jogadores devem posicionar ambos os elementos nas suas respetivas portas dentro do tempo limite, que é exibido na parte superior da tela do jogo.

Figura 6 Posição para vencer o jogo

1.3. Menu de instruções

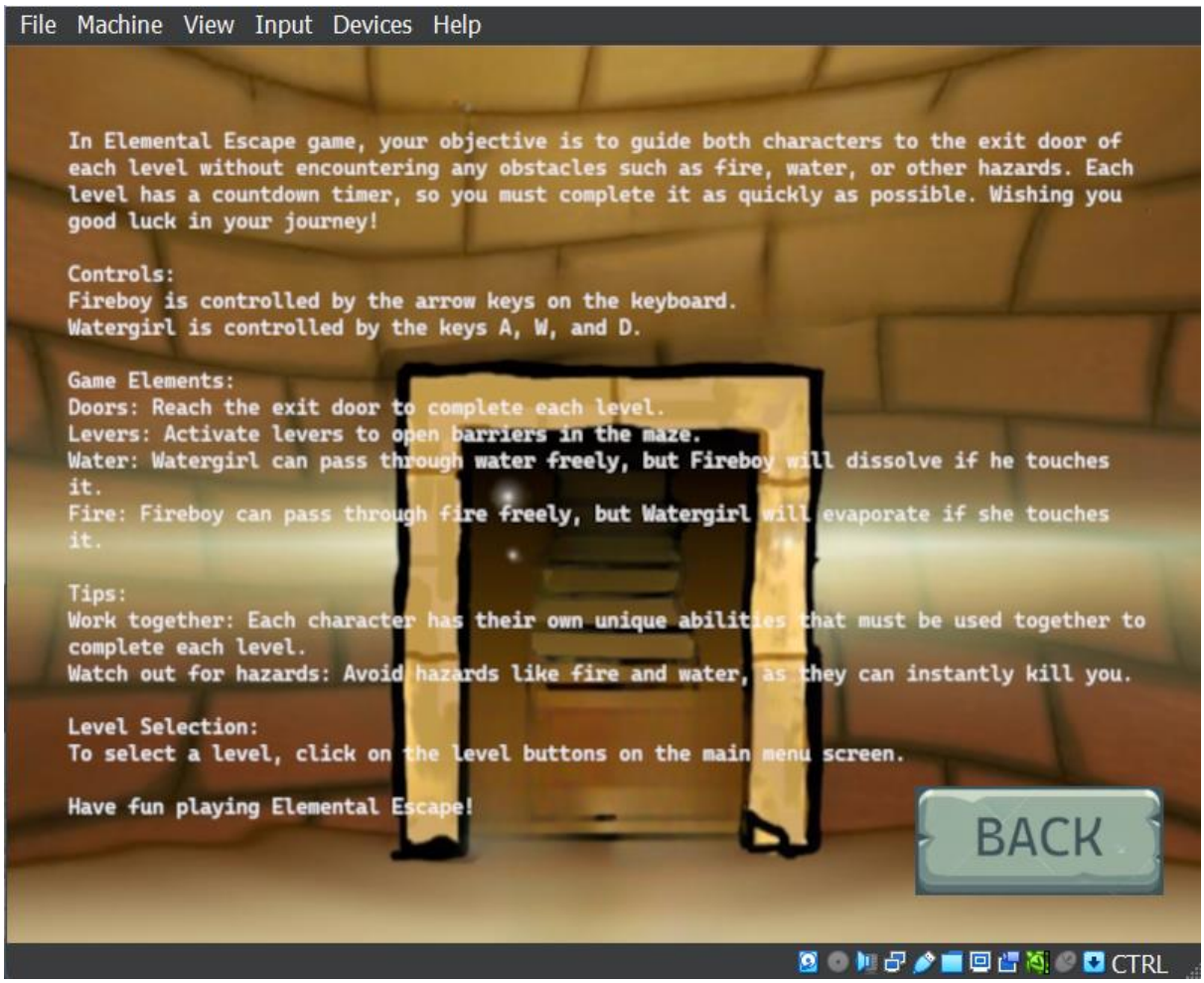


Figura 7 Menu de instruções

Nesta interface, é exibida uma apresentação textual que fornece informações relevantes para a conclusão bem-sucedida do jogo. Para retornar ao menu inicial, basta selecionar o botão "BACK" utilizando o cursor do mouse.

1.4. Menu de vitória/derrota

Quando os jogadores vencem o jogo é carregado menu de vitoria e quando perde aparece também o menu de derrota. Em ambos existe um botão, que quando se clica com o rato, volta ao menu principal.

Ao vencer o jogo, os jogadores serão direcionados para o menu de vitória, onde terão a opção de voltar ao menu principal ao clicar no botão “Main Menu”, utilizando o rato. Da mesma forma, ao perder, será exibido o menu de derrota, com o mesmo botão do menu anterior. Desta forma, os jogadores terão a oportunidade de jogar novamente.

Figura 8 Menu de vitória

Figura 9 Menu de derrota

2. Estado do Projeto

Dispositivo	Funcionalidade	Interrupções
Timer	Limite de tempo para o jogo e	Sim
Keyboard	Movimentar os jogadores.	Sim
Mouse	Carregar nos botões para navegar entre os diversos menus e movimentar as alavancas.	Sim
Graphics Card	Interface do jogo.	Não
Real Time Clock	Apresentar a hora no menu.	Sim
Serial Port	Não foi implementado.	Não

2.1. Timer

O *timer* é usado para controlar a frequência de update do ecrã. Para o nosso projeto escolhemos uma frequência de 60 Hz. O *timer* foi usado para vários aspetos do nosso jogo, nomeadamente, para o uso de double buffer e a troca do conteúdo entre eles, controlo do tempo limite para jogar o nível, controlo do tempo de abertura das barreiras, chamada de funções para desenhar o tempo do mapa e desenho do novo ecrã após o tempo de jogo terminar e as alavancas fecharem.

2.2. Keyboard

O teclado é usado para controlar o movimento dos dois jogadores e para a opção de sair do jogo. Quando pressionamos uma tecla, é enviado um *makecode* que vai ser interpretado e depois acionada uma ação correspondente a essa tecla. As teclas W, A, S e D ativam o movimento do jogador "água" e as teclas das setas para cima, baixo, esquerda e direita ativam o movimento do jogador "fogo".

Por fim, quando o *breakcode* da tecla é interpretado, o movimento do boneco é parado. Para além disso, usando a tecla "q" é possível sair do jogo. É de realçar que todo o movimento dos jogadores é tratado pela função **move_action(GameState state, int (*check_move)(), void (*move_player)())**.

2.3. Mouse

A cada movimento do rato, as interrupções correspondentes são tratadas, verificando se o movimento é válido e atualizando a posição do cursor. O mouse é utilizado para ativar os botões nos diversos menus, utilizando o botão esquerdo para realizar essa ação. Além disso, ele é responsável pela ativação das alavancas quando um jogador estiver a uma distância igual ou inferior a 40 pixeis.

Todas as operações relacionadas à construção dos pacotes do *mouse*, validação de movimento e ações correlacionadas são tratadas pela função *handle_mouse_interrupt()*. Vale ressaltar que a atualização dos cliques do mouse é tratada por meio de um *array* de ponteiros de função, permitindo a verificação adequada de cada clique de acordo com o estado atual do jogo.

2.4. Graphics Card

Para a configuração da placa gráfica, utilizamos o modo 0x115, o qual possui uma resolução de 800x600 pixels e utiliza um formato de 32 bits por pixel. Implementamos a técnica de *double buffering* por meio da função update_buffers(), permitindo uma transição suave entre os quadros exibidos na tela.

Todas as componentes do jogo, como menus, botões e elementos interativos, foram desenhadas utilizando essa abordagem. Um destaque importante é o uso de um *array* de ponteiros de função, que permitiu o desenho adequado de cada componente de acordo com o estado atual do jogo.

2.5. Real Time Clock

O *Real Time Clock* é utilizado para mostrar a hora atual no menu inicial. Quando fazemos um *polling* do *real time clock* do *minix3* armazenamos num *array* o ano, o mês, o dia, a hora, os minutos e os segundos, porém apenas mostramos as horas, os minutos e os segundos.

É de realçar que o display do tempo é tratado pela função **draw_rtc()** e a atualização do tempo é tratada pela função **rtc_update_time().**

3. Organização do Código/Estrutura

O nosso código está organizado em dois módulos: "*game*" e "*devices*". O primeiro contém toda a estrutura lógica do código, enquanto o segundo inclui os labs realizados ao longo do semestre, com alterações necessárias. Essa divisão em módulos proporciona uma organização clara e facilita a compreensão e alteração do código.

3.1. Modulo devices (36%)

Dispositivo	Desenvolvido por	Peso no projeto
Timer	*	8%
Keyboard	*	5%
Mouse	*	8%
Graphics Card	*	10%
Real Time Clock	Pedro Magalhães	5%

(*) Uma vez que os labs foram realizados de forma contínua ao longo do semestre, acreditamos que a sua avaliação não deve ser considerada de forma individual, mas sim como parte integrante do processo de aprendizagem e desenvolvimento.

3.2. Modulo game

3.2.1. Game (4%)

Na classe game é controlada a inicialização e encerramento do jogo, através das classes start_game() e finish_game(), que permite configurar os elementos do jogo e libertação dos recursos alocados durante a execução, de forma adequada.

Desenvolvido por: Francisca Guimarães (25%), Inês Oliveira (25%), João Oliveira (25%), Pedro Magalhães (25%)

3.2.2. Draw (10%)

No modulo de desenho são utilizados *buffers* duplos para o desenho dos elementos visuais. Assim, podemos encontrar funções para alocar os buffers e atualizá-los, bem como funções específicas para desenhar diferentes elementos do jogo, como sprites móveis, sprites estáticos, elementos de tempo, mapas e menus.

Desenvolvido por: Francisca Guimarães (25%), Inês Oliveira (25%), João Oliveira (25%), Pedro Magalhães (25%)

3.2.3. Elements (10%)

O código que encontramos neste modulo gere a alocação e liberação de recursos do jogo, bem como a leitura do mapa do jogo a partir de um ficheiro .txt. A função read_map() realiza a leitura do arquivo de mapa, identificando diferentes caracteres para definir as posições das paredes, alavancas, barreiras e armadilhas no jogo.

Desenvolvido por: Francisca Guimarães (25%), Inês Oliveira (25%), João Oliveira (25%), Pedro Magalhães (25%)

3.2.4. Interrupts (10%)

Esta classe é responsável pelas interrupções de todos os dispositivos: timer, teclado e rato. Assim, a cada interrupção de um destes dispositivos, uma ação é tomada, seja esta o fim do jogo, a contagem regressiva para o fecho das barreiras, o movimento do jogador ou os cliques do rato.

Desenvolvido por: Francisca Guimarães (25%), Inês Oliveira (25%), João Oliveira (25%), Pedro Magalhães (25%)

3.2.5. Moves (10%)

A classe moves.c apresenta várias funções relacionadas ao movimento dos jogadores, à verificação das colisões e ao tratamento de cliques do mouse.

Desenvolvido por: Francisca Guimarães (25%), Inês Oliveira (25%), João Oliveira (25%), Pedro Magalhães (25%)

3.2.6. Allocations_manager (10%)

Neste modulo encontramos o código responsável pela gestão das alocações dos recuso do jogo, da sua exclusão e da configuração dos estados iniciais.

Desenvolvido por: Francisca Guimarães (33%), Inês Oliveira (33%), João Oliveira (33%), Pedro Magalhães (0%)

3.2.7. Sprite (5%)

Cada sprite é armazenado em uma estrutura chamada sprite_t, a qual possui os campos de posição x e y, além de um atributo que indica se está sendo pressionado pelo mouse.

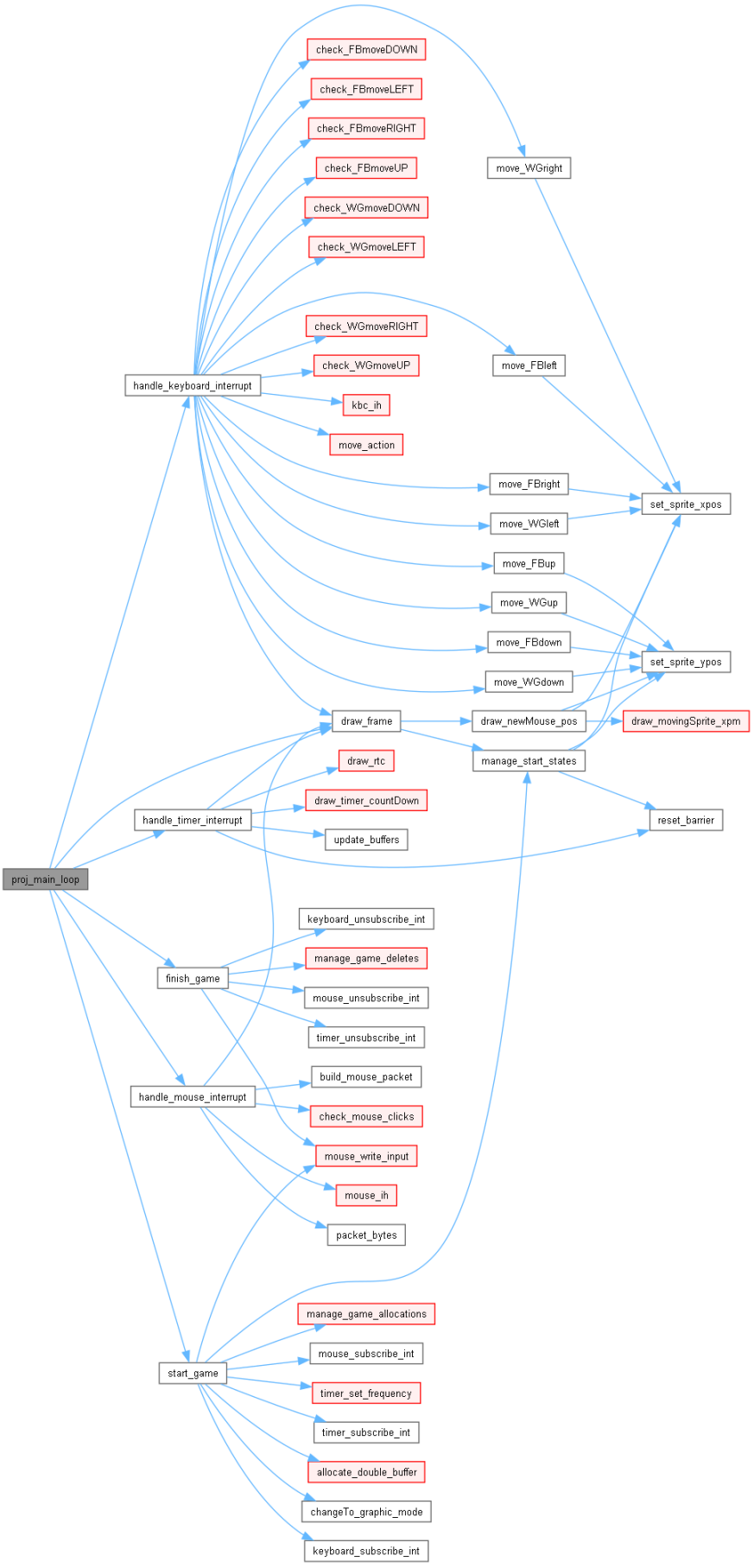
Desenvolvido por: Francisca Guimarães (25%), Inês Oliveira (25%), João Oliveira (25%), Pedro Magalhães (25%)

3.3. Documentação (5%)

- XPMS: Francisca Guimarães e Inês Oliveira

- Documentação Doxygen: João Oliveira e Pedro Magalhães

3.4. Gráfico de chamada de função



4. Detalhes de Implementação

No nosso jogo, adotamos uma abordagem orientada a eventos, em que todo o programa é baseado em interrupções geradas pelo utilizador. Essas interrupções incluem pressionar teclas, clicar e mover o mouse, atualizações e alarmes do RTC (Relógio em Tempo Real), além de interrupções periódicas do timer e do RTC. Cada evento produz uma resposta específica por parte do programa, tornando-o reativo.

Para gerir o fluxo do jogo, utilizamos a criação de estados de jogo, implementada por meio do enum "GameState". Definimos diferentes estados, como "START_MENU" (menu inicial), "GAMEWIN_MENU" (menu de vitória), "GAMEOVER_MENU" (menu de game over), "INSTRUCTIONS_MENU" (menu de instruções) e "PLAYING" (jogando). Cada estado modifica o comportamento de várias funções do jogo, o que nos permite implementar uma máquina de estados eficiente. Alternância de estados faz se usando o rato e o teclado.

A frame generation é feita usando o *driver* da gráfica para desenhar pixéis com base em xpms. A frame gerada é definida pela máquina de estados, por exemplo, o state START_MENU faz desenhar o background do menu inicial e os botões, e se estivermos no state PLAYING, é desenhado mapa, os personagens, e os elementos do mapa.

Dado que o jogo é baseado num labirinto, desenvolvemos algoritmos de deteção de colisões que levam em consideração a posição dos elementos e das barreiras presentes no jogo. Esses algoritmos permitem uma deteção precisa de colisões, proporcionando uma experiência mais realista e desafiadora.

O RTC foi implementado no menu inicial para mostrar a hora atual do *minix*. Mostra a hora, os minutos e os segundos.

Para além disto, adicionamos uma estrutura *context*, na qual colocamos todos as informações relativas ao jogo, como por exemplo, o estado, os *sprites* (organizados em *arrays*) os elementos relativos ao jogo, que são por sua vez estruturas que têm a informação dos elementos, como por exemplo, se a porta está aberta ou se a armadilha é de fogo ou água. Com esta estrutura tornamos fácil o acesso aos elementos do jogo e centralizamos os elementos de forma a tornar o jogo mais modular e de fácil entendimento.

5. Conclusão

Durante o desenvolvimento do projeto, exploramos com facilidade cada um dos dispositivos abordados na aula durante o semestre, resultando na implementação bem-sucedida da maioria das funcionalidades previstas. Assim, podemos concluir que a realização do trabalho foi benéfica para todos os elementos do grupo.

No entanto, o aspeto mais desafiante consiste no facto de não existirem Labs para o RTC e para a porta de série. A falta de informação dificultou a sua implementação considerando o intervalo de tempo que nos foi dado para desenvolver o projeto, impedindo-nos de atingir a nota máxima.

Por fim, o trabalho foi realizado em equipa, contando com um esforço equilibrado de todos os elementos, permitindo finalizar o trabalho nas datas pretendidas.