Uma imagem com texto

Descrição gerada automaticamente

**MARIO-MAN**

***POKEBALL EDITION***

Alberto José Ribeiro da Cunha(up201906325)

Martim Raúl da Rocha Henriques(up202004421)

Rúben Costa Viana(up202005108)

Tiago Filipe Magalhães Barbosa(up202004926)

Licenciatura em Engenharia Informática e Computação

Laboratório de Computadores

Turma 7 - Grupo 1

10 de junho de 2022

# User instructions

## Ecrã Principal

Iniciando o projeto com o comando lcom\_run proj é nos levado par ao menu inicial onde encontramos duas opções : "PLAY" e "EXIT".



O utilizador pode com recurso ao rato mover o cursor e selecionar uma dessas opções.

* **PLAY** - Ao selecionar esta opção o utilizador é levado para o ecrã de jogo e pode começar a jogar.
* Uma imagem com texto, interior, captura de ecrã

  Descrição gerada automaticamente
* **EXIT** - Ao selecionar esta opção o utilizador pode sair do jogo e terminar o programa.

Uma imagem com texto, interior, captura de ecrã

Descrição gerada automaticamente

Ao invés de clicar no botão "EXIT" também é possível pressionar a tecla ESC para terminar o programa.

## Ecrã de jogo

Após selecionada a opção de "PLAY" o jogo inicia com o jogador no centro do ecrã e com os adversários ("goombas") a nasceram em vários locais .

Uma imagem com texto

Descrição gerada automaticamente

O objetivo do jogo é capturar o máximo número de goombas com pokebolas antes do alarme de 1 minuto ou antes que o jogador seja apanhado por um goomba. O jogador para se movimentar deverá utilizar as teclas AWSD:

* A - mover-se para a esquerda
* W - mover-se para cima
* S - mover-se para baixo
* D - mover-se para a direita

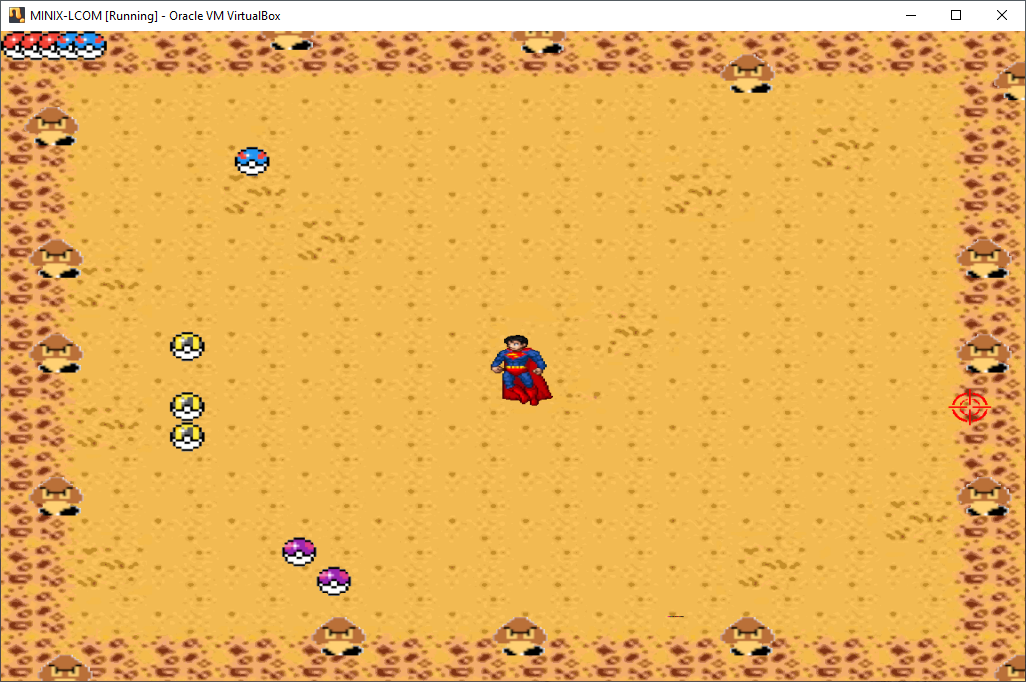
Uma imagem com texto, eletrónica

Descrição gerada automaticamenteSe estiverem pressionadas duas teclas ao mesmo tempo o jogador irá se mover nas duas direções , por exemplo , se tiver pressionado a A e o W. o jogador irá se mover na diagonal para cima e para a esquerda. O jogador pode utilizar o rato para mover a mira e utilizar o botão principal do rato (mouse1) para atirar as pokebolas na direção da mira.

Uma imagem com texto

Descrição gerada automaticamente

Cada pokebola é capaz de capturar um goomba e é possível disparar várias pokebolas segurando o botão principal do rato e movendo-o em várias direções.



Os goombas conseguem farejar o jogador e estão sempre a ir atrás deste a partir do momento que nascem. Quando um goomba é capturado o jogador ganha 1 ponto e o goomba renasce.

## Ecrã de fim de jogo

Após passar 1 minuto desde o início de jogo, quando um goomba consegue apanhar o jogador ou quando o jogador pressiona a tecla ESC durante o jogo, este termina e é apresentado um ecrã de "GAME OVER" apresentando os pontos que o jogador fez e duas opções : "PLAY AGAIN" e "EXIT"

Uma imagem com texto, eletrónica

Descrição gerada automaticamente

1. **PLAY AGAIN** - Permite ao jogador começar um novo jogo.



1. **EXIT** - Permite ao jogador regressar ao menu.

O jogador ao invés de clicar no botão "EXIT" também pode pressionar a tecla ESC para regressar ao menu.

# Project Status

|  |  |  |
| --- | --- | --- |
| **Dispositivo** | **Para quê** | **Int.** |
| Timer | Controlar a frame rate | Sim |
| Teclado | Mover o jogador  Terminar jogo mais cedo  Termina programa no menu  Regressar ao menu no fim de jogo | Sim |
| Rato | Mover o cursor do menu  Selecionar opções no menu  Mover a mira do jogo  Lançar pokebolas no jogo | Sim |
| Video Card | Iniciar o modo de vídeo do programa  Desenhar sprites no ecrã | N/A |
| Real Time Clock(RTC) | Obter a data atual de início de jogo  Programar um alarme para 1 minuto após o início do jogo |  |

Para além destes dispositivos, tínhamos como ideia inicial implementar a porta de série que permitia ligar duas máquinas virtuais possibilitando assim que um dos jogadores controlasse o super homem enquanto outro controlava os goombas que o tentam apanhar. No entanto, devido à falta de tempo não foi possível implementar esta funcionalidade.

## Timer

Foram utlizados os interrupts do timer para atualizar o ecrã de uma forma constante, desenhando todo o estado do jogo a cada interrupt. Assim permitia-nos mover objetos que não se moviam através de rato ou teclado e apenas se moviam constantemente com uma velocidade definida pela frequência destas interrupções. Também a cada interrupção do timer , como existiam objetos que se moviam, verificamos sempre se existiam colisões e atualizávamos o estado de cada objeto e o estado do jogo de acordo com isso.

Assim a cada interrupção do timer é chamada a função UpdateScreen() que utiliza um case para verificar o estado do jogo : "PLAY","MENU", " GAME\_OVER" , etc.. e desenhava todos os sprites necessários recorrendo à função draw\_sprite\_proj() e realizava posteriormente a cópia desses sprites do buffer para a memória de vídeo usando a função double\_buffer().

Quando o programa está no jogo em si, as interrupções do timer movem todos os objetos necessários utilizando as funções movePlayer(), moveBullets e moveGoombas() e verifica as colisões entre estes usando as funções checkPlayerColisions() e checkGoombaColisions().

É utilizada a função time\_int\_handler() para lidar com as interrupções aumentando um contador, no entanto, neste momento esse contador não é utilizado pois é utilizado o Real Time Clock (RTC).

## Teclado

As interrupções do teclado são usadas para verificar se alguma tecla foi pressionada e alterar a posição do jogador ou alterar o estado do programa (tecla ESC).

Assim a cada interrupção utilizamos a função kbc\_ih() para , no caso de existir uma tecla pressionada, obter o scancode da mesma. Depois com recurso à função UpdateStateKbd() alteramos o estado do programa no caso da tecla ESC ou alteramos a direção do jogador no caso das teclas A,W,S ou D tendo em consideração os make e breakcodes para saber se a tecla foi pressionada ou largada. Todos estes updates são realizados dependendo do estado do programa sendo usada uma implementação de uma máquina de estados.

## Rato

As interrupções do rato são usadas para construir os packets de um movimento do rato e quando este packet estiver completo atualizar o estado do cursor ou mira dependendo se estamos no menu ou no jogo e verificar se o botão principal do rato foi pressionado para selecionar uma opção no menu ou para atirar pokebolas.

É utilizada a cada interrupção a função kbc\_ih() que permite ler do rato os bytes necessários para a construção de um packet de movimento do rato e a função UpdateStateMouse() que verifica se o packet está construído e utiliza a função organize\_packets() para colocar a informação do packet numa struct que facilite a utilização destes dados . Depois de ter o packet totalmente construído e a informação de movimento ou de botões pressionados, a função UpdateStateMouse(), dependendo do estado do programa(máquina de estados) irá atualizar a posição do cursor no menu ou da mira no jogo e irá ,tendo em conta se o botão principal tiver sido pressionado, mudar o estado do programa se uma opção tiver sido selecionada ou lançar uma pokebola na direção da mira durante o jogo. Esta função, quando selecionada a opção de play no menu , utiliza a função InitializeGame() para "spawnar" os goombas, as pokebolas e o jogador nas posições corretas e para ler a data atual e programar um alarme para 1 minuto depois acabar o jogo.

## Video Card

Foi utilizado modo 0x14C com resolução de 1152x864 com "Direct color model" e com 32 bits por pixel((8):8:8:8). Para iniciar o modo de vídeo é usado a função vg\_init() e para voltar ao modo de texto é usado a função vg\_exit(). Foi utilizado um método de "double buffering" onde para desenhar todos os sprites do estado atual do programa, estes são primeiros desenhados num buffer utilizando a função draw\_sprite\_proj() que utiliza a função vg\_draw\_pixel() para colocar cada pixel do sprite no buffer. Depois usa-se a função double\_buffer() que copia o que estiver no buffer para a memória de vídeo(VRAM).

## Real Time Clock (RTC)

O real time clock é utilizado tanto para obter a data atual como para programar um alarme para 1 minuto após o início do jogo.

A cada interrupção do RTC é chamada a função rtc\_ih() que lê o registo C do RTC usando a função read\_from\_rtc() para limpar as flags e dependendo da origem da interrupção(update ou alarm) alerta o programa para o disparo do alarme. O programa usa a função rtc\_get\_date() para obter a data atual e depois colocar os registos de alarme em conformidade com o pretendido.