**Faculdade de Engenharia da Universidade do Porto**



Bomberman

Relatório do Projeto Final de LCOM

**Turma 4 - Grupo 2**

Diogo Sarmento, up202109663

Rodrigo Póvoa, up202108890

Tomás Câmara, up202108665

Tomás Sarmento, up202108778

28/05/2023

Índice

[**Introdução 3**](#_d7ttyhpk8cke)

[**1. Instruções de utilização 4**](#_30wczumcdm31)

[1.1. Menu inicial 4](#_z4ko6ngq703k)

[1.2. Introdução de usernames 5](#_5oatlyffohyn)

[1.3. Arena 7](#_dmikyr3yzwgi)

[1.4. Game over 11](#_l01uyici5sjo)

[**2. Estado do projeto 12**](#_sia00tgm5s96)

[2.1. Graphic card 12](#_tp0mv4orpwbl)

[2.2. Timer 12](#_wslljiosq7p3)

[2.3. Keyboard 13](#_obwh1315fmha)

[2.4. Mouse 13](#_pxa38bptusno)

[2.5. RTC 13](#_7haz1vqzm09d)

[**3. Organização e estrutura do código 14**](#_jiiw8bwlu25s)

[3.1. Timer Module(5%) 14](#_ve7vrd21tjeh)

[3.2. Keyboard Module(5%) 14](#_31relqme4vnf)

[3.3. Mouse Module(5%) 14](#_ik1pcpoacfc)

[3.4. Video\_gr module(5%) 14](#_54h1pbvdv6mg)

[3.5. RTC Module(5%) 14](#_5gibyxiye5a0)

[3.6. models.h header(10%) 14](#_t6ntgd3b5v2q)

[3.7. utils module(0.1%) 15](#_azsix4aso3u)

[3.8. Draw module(15%) 15](#_qgitdatjopeo)

[3.9. Controller module(15%) 15](#_c4mh4697vcle)

[3.10. kbc\_contoller module(1%) 15](#_9dthq6p3jfye)

[3.11. Main module(10%) 15](#_oitohnuiupa4)

[3.12. Game module(10%) 15](#_ykxhoz61jlro)

[3.13. SelectNames module(2%) 16](#_fiyl1l9gd35h)

[3.14. Load module(1%) 16](#_net616qrvqbc)

[3.15. Menu module(5%) 16](#_ybqx4tden4zy)

[3.16. Message module(5%) 16](#_4p7oeg97v69n)

[Function call graph 17](#_so95x4t4gmau)

[**4. Detalhes de implementação 18**](#_fixp8erjvyr3)

[**5. Conclusões 19**](#_p0w0ch8wih7b)

# Introdução

Para o nosso projeto final, decidimos adaptar o jogo Bomberman para MINIX. Bomberman é um jogo multiplayer top-down em que os vários jogadores podem largar bombas que causam dano em linhas verticais e horizontais a partir da bomba. O objetivo do jogo é ser o último jogador vivo, esquivando as bombas dos outros jogadores e tentando matá-los primeiro.

# Instruções de utilização

## Menu inicial

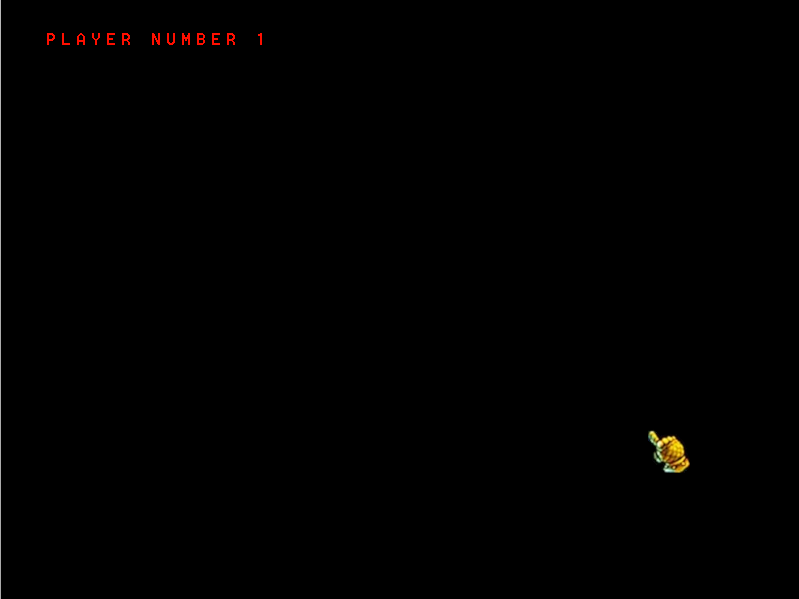
A primeira janela que aparece quando se abre o jogo é o menu inicial, que tem dois botões, de começar e sair do jogo, o nome do jogo no centro e a hora do computador no canto superior esquerdo.

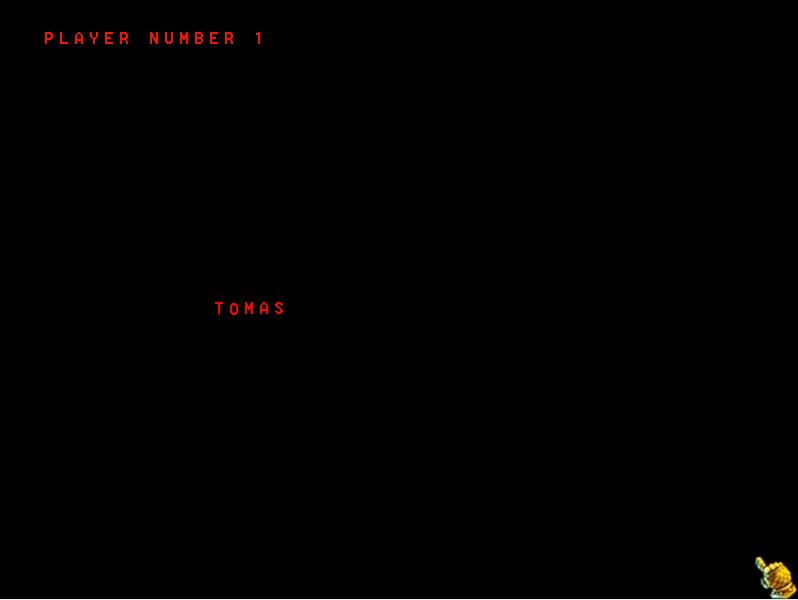
Quando se clica no “Start”, o jogo passa para a janela de introdução de nomes e, quando se clica no “Exit”, o jogo fecha e voltamos para a consola.



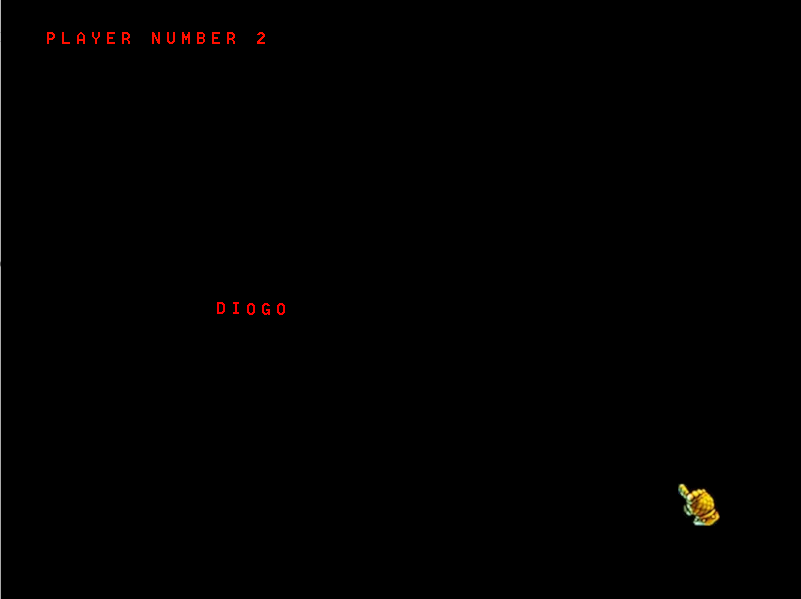
## Introdução de usernames

A janela de introdução de usernames é uma janela com apenas um background preto e o número do jogador em que cada jogador escreve o seu nome e depois prime no enter.





Existe uma para cada jogador.



## Arena

Na secção da arena, ambos os jogadores conseguem controlar as suas personagens: o player 1 usa o WASD para se mover e o espaço para largar bombas enquanto o player 2 utiliza as setas e o 0 do numpad, respetivamente.

Os tijolos são destrutíveis e algumas vezes têm moedas escondidas dentro que o jogador pode apanhar depois de os destruir, as paredes (os quadrados cinzentos) são indestrutíveis.



Aqui o player 1 largou uma bomba perto de si e depois moveu-se para não morrer. A explosão destruiu os blocos de tijolo e soltou a moeda.



Quando um jogador mata outro, a janela final abre anunciando o vencedor.



## Game over

Esta janela abre quando um dos jogadores morre. O botão continue volta o jogo ao menu inicial.



# Estado do projeto

| Dispositivo | Funcionalidade | Interrupções/polling |
| --- | --- | --- |
| Graphic card | -Exibindo o menu, o jogo e o menu | N/A |
| Timer | - Controlo de frame rate e através disso controlamos as animações dos sprites, a velocidade dos players, a explosão das bombas e o desaparecimento das chamas . | I |
| Keyboard | -escrever o nome dos jogadores  -controlar ações dos jogadores  -permitir sair do jogo a qualquer momento premindo esc  -controlar a seleção de opções no menu | I |
| Mouse | -controlar a seleção de opções pelo jogo | I |
| RTC | -exibir o tempo atual no menu  -mudar o fundo do jogo dependendo do tempo real | I |

## Graphic card

Foi utilizado o modo de vídeo 0x115, que possui coloração direta utilizando 24 bits por pixel e resolução de 800x600.

A inversão de página foi usada, também usamos o gerador de fonte de bitmap do codehead para criar xpms a partir da fonte para que possamos escrever o que quisermos sem nos preocuparmos em obter o xpm para cada caractere individual

## Timer

O timer é usado para controlar o frame rate fazendo atualizações 20 vezes por segundo.

No Game ele é utilizado para controlar a velocidade dos players, o tempo de espera entre animação do player, da coin, da bomba e da chama,também é utilizado para controlar o tempo para a bomba explodir e o tempo das chamas desaparecerem, por último utilizamos o timer para um timeout de quando um jogador morre exibindo em baixo que o outro jogador morreu e assim “pausando” o jogo temporariamente .

## Keyboard

O keyboard é utilizado para movimentar os players para player1 são as teclas A/W/S/D para o player 2 são as teclas 2/4/8/6 do numpad ou setas do numpad e permite colocar bomba para o player1 espaço e para o player2 0 do numpad durante o jogo a função responsável pelo respetivo controlo é PlayerControllers.

Durante o Jogo se premida a tecla esc o estado do jogo é alterado para exit que resulta em sair do jogo.

No menu também é possível selecionar opções com o teclado usando as teclas enter para selecionar e A/W/S/D.

## Mouse

O rato é utilizado pelo jogo todo possibilitando seleção de opções no menu e mesmo o botão de exit durante o jogo que direciona para o menu.

## RTC

O rtc (real-time clock) é usado no menu e durante o jogo. Primeiramente, no menu, é feito o acesso ao rtc através da funcão is\_updating\_rtc() e depois atualizado usando update\_time\_display(). Depois é desenhado no menu usando draw\_TimeNumbers() e atualizado perante interrupções do timer (diferente counter). No jogo, faz-se uma condição para verificar se as horas atuais forem superior a 8 e inferior a 20 , é desenhado um sol com uma relva mais viva. Caso contrário, é desenhada uma lua com uma relva mais “sombria”.

# Organização e estrutura do código

## Timer Module(5%)

Neste módulo encontram-se todas as funções que desenvolvemos durante o lab2 e que são necessárias para a utilização do timer. Todos os elementos contribuíram para a conclusão deste módulo.

## Keyboard Module(5%)

Neste módulo encontram-se todas as funções que desenvolvemos durante o lab3 que são necessários para a utilização do keyboard.Todos os elementos do grupo colaboraram para a conclusão deste módulo.

## Mouse Module(5%)

Neste módulo encontram-se todas as funções que desenvolvemos durante o lab4 que são necessárias para a utilização do rato.Todos os elementos ajudaram para a conclusão deste módulo.

## Video\_gr module(5%)

Neste módulo encontram-se todas as funções que desenvolvemos durante o lab5 que são necessárias para a utilização da placa gráfica.Todos os elementos cooperaram para a conclusão deste módulo.

## RTC Module(5%)

Neste módulo encontram-se todas as funções necessárias para a utilização do RTC, ler registos necessários e atualizar as horas, minutos e segundos. Este módulo foi desenvolvido pelo aluno Rodrigo Póvoa.

## models.h header(10%)

Neste header encontram-se declaradas todas as structs e enums que criamos relacionadas diretamente com o jogo que criamos como por exemplo Struct Player e Enum GameState.Todos os elementos ajudaram para o desenvolvimento deste header.

## utils module(0.1%)

Neste módulo encontram-se funções desenvolvidas nos labs que são utilizadas pelos models um exemplo a util\_get\_LSB.Todos os elementos ajudaram para o desenvolvimento deste module.

## Draw module(15%)

Neste mòdulo encontram-se todas as funções que dão draw desde draw\_game a draw\_bomb, também é neste ficheiro que se encontram as funções que dão load dos xpm para xpm\_image\_t para depois darmos draw pm8 8\_8 que é proveniente da placa\_gr module assim damos draw de qualquer xpm através deste module.

## Controller module(15%)

Neste módulo encontra-se grande parte de todas as funções necessárias para controlar a lógica do jogo, por exemplo PlayerController que é o controlador dos players e é responsável por verificar se o player pode deslocar-se para a direita dar deploy a bomb ,etc.

Neste módulo encontram-se também os PlayersSpriteControllers que muda o sprite atual variando da ação do jogador.

## kbc\_contoller module(1%)

Este módulo é utilizado somente no SelectNames module e é utilizado para receber e ler os inputs do keyboard.

Este módulo foi desenvolvido por Tomás Câmara.

## Main module(10%)

Este módulo é responsável pelo mainloop onde dependendo do state enum GameState pode iniciar o Menu() ou Game().

Este módulo foi desenvolvido com a colaboração de todos os elementos.

## Game module(10%)

Este módulo é responsável pelo “jogo” onde se encontra o ciclo de interrupções que possibilitam a utilização do mouse, keyboard, mouse , placa gráfica e assim a execução do jogo.

Durante o jogo é também possível

Durante o jogo é possível sair e ir diretamente para o menu se premir esc key ou o botão com o mouse

Neste ficheiro também se encontra alguma lógica relacionada com o mouse mouse\_api\_game.

. Este módulo foi desenvolvido com a colaboração de todos os elementos.

## SelectNames module(2%)

Este módulo é responsável pelo ciclo de interrupções que possibilitam a utilização do mouse, keyboard, mouse e placa gráfica para a introdução dos nomes dos jogadores mudando de state quando foram introduzidos 2 nomes de 2 jogadores.

Este módulo possibilita a introdução dos nomes dos jogadores.

Este módulo foi desenvolvido principalmente por Tomás Sarmento.

## Load module(1%)

Este módulo é responsável por carregar a ArenaModel através de uma array.

Este módulo foi desenvolvido por Diogo Sarmento.

## Menu module(5%)

Este módulo é responsável pelo ciclo de interrupções que possibilitam a utilização do mouse, keyboard, mouse e placa gráfica para a seleção de opções.

Neste módulo é onde é carregada a arena e após a seleção de start quer pelo rato ou teclado o state é alterado para game.

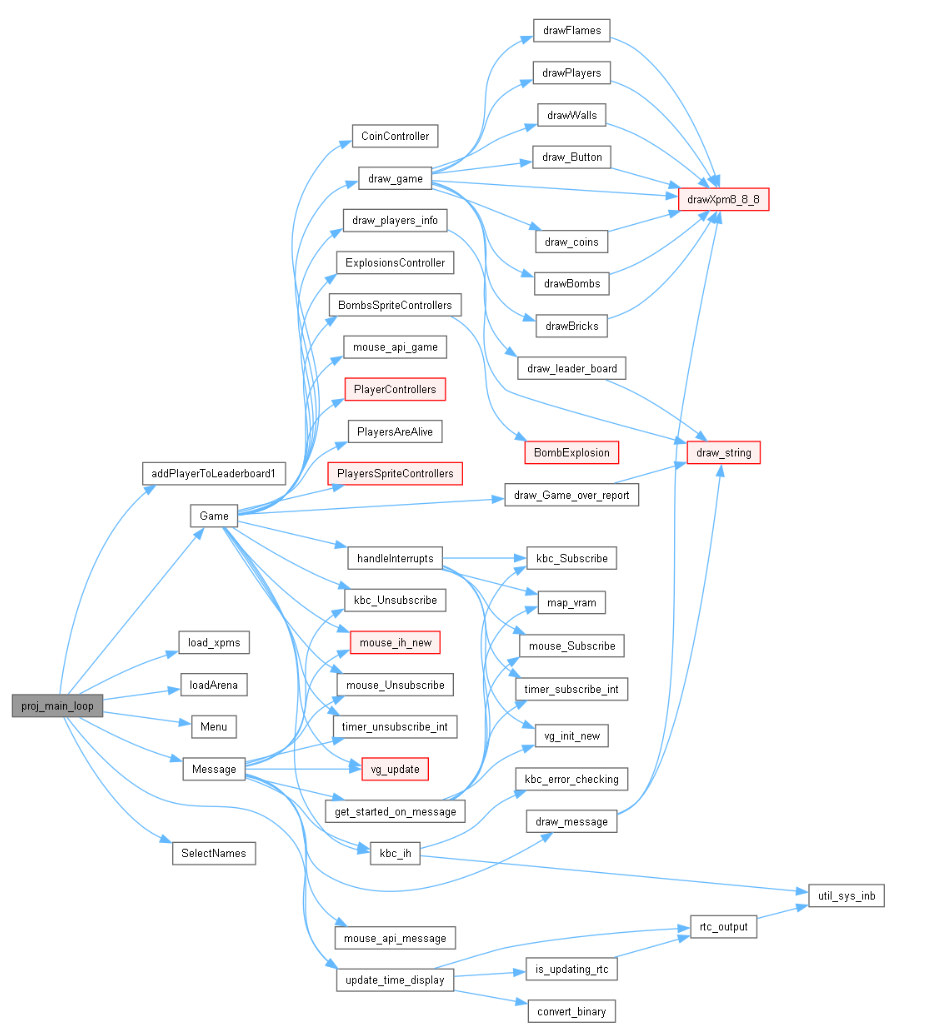
Este módulo foi desenvolvido com a colaboração de todos os elementos.

## Message module(5%)

Este módulo é responsável pelo ciclo de interrupções que possibilitam a utilização do mouse, keyboard, mouse e placa gráfica para exibir o ecrã de game over apresentando o vencedor ou no caso de empate TIE guardando o resultado a leaderboard que é exibida durante o Game .

Este módulo foi desenvolvido com a colaboração de todos os elementos.

## Function call graph



# Detalhes de implementação

Implementamos uma ***state machine*** para saber o estado em que o jogo está e dessa forma certas funções comportam-se de maneiras diferentes.

O nosso menu consiste em dois botões que podem ser controlados tanto com o teclado (W,S) ou pelo rato usado o objeto (MenuModel) para ajudar na implementação dos botões do menu a interagir com o rato e teclado, tendo também o tempo atual.

O nosso menu também tem a capacidade de facilmente implementarmos diferentes níveis mas por vários contratempos não conseguimos implementar.

No estado de seleção de usernames utilizam-se algumas funções tipo draw\_string() que itera uma string caracter a caracter e vai imprimindo no ecrã,

esta função dá call a uma outra que se chama vg\_draw\_Char() recebendo uma cor a posição do canto superior esquerdo ele vai aceder a xmp\_font que tem bastantes caracteres ordenados pela aci table desde “!” ao “=” esta parte do código pode ser alterada facilmente para alterar a fonte mas não chegou a ser implementado por falta de tempo , também no estado de seleção de nomes se encontra o .kbc\_controller que interpreta um scancode criado pelo controlador do teclado e implementa apenas letras de “a” a “z” dando return a sua char e se não for sucedido dá return a “=”.

O nosso jogo em si é uma array em que “H” são paredes indestrutíveis “h” bricks destrutivos “C” são coins “b” um dos bombermans e “B” é um dos outros bomberman é chamada uma função loadArena(array) que recebendo uma array da build aos dependendo do mapa de tanto dos bricks dos bombermans.

O jogo quando está no State game ele chama a funcao draw\_game()

que da draw a tudo que pertence ao jogo tipo walls bricks bombermans.

A leaderboard mostra o player que ganhou o jogo com mais coins com ranking de 3 indo eliminado e gerir todas as possibilidades foi um bocado desafiante do ponto de vista da lógica.

As sprintes que foram implementadas neste jogo seguem o modelo que o professor mostrou nas aulas uma array com as xpm ,um double coldown ,um double current\_time , um int current\_xpm que lida com as sprites e temos uma função

PlayersSpriteControllers() que lida com todas as sprites tipo coins players.

# Conclusões

▶ Problemas ou “pausas” inesperadas que levaram a algumas funcionalidades a não serem implementadas:

1. Implementação de uma fonte modular foi estupidamente complicada mesmo com a aplicação que o professor recomendou codeheads acabou por dar mais trabalho do que devia pois nesta cadeira não foi abordado com lidar com bitmaps o que acabou por causar problemas quando se utilizou um conversor para xpm lidar com uma fonte numa xpm foi mais complicado do que fazer os números de 1 a 10 e o resultado ficou a desejar com certas letras que aleatoriamente ficavam ligeiramente mais altas ou baixas.
2. A implementação do pagefliping deu bastante trabalho e depois de bastantes erros foi feita mas houve bastante confusão a lidar com isso.
3. O mouse ficava lagged em certas ocasiões em dispositivos de desempenho inferior antes de ter sido revelado o comando que optimiza o MINIX o que levou a uma perda de tempo enorme a optimizar o mouse que já estava otimizado.
4. No início de desenvolvimento começamos a desenvolver os states de forma independente o que causa ter de iniciar as interrupts e a vbe em cada estado e unsub de interrupts quando sai do estado nessa parte se tivéssemos mais tempo metemos domain loop das interrupts com os controllers no main module

▶ Funcionalidades que gostaríamos de implementar no futuro, que não foram implementadas:

1. Diferentes mapas (níveis) para adicionar mais variabilidade ao jogo.
2. Power-ups: poderes especiais que dão habilidades permanentes/temporárias de forma a dar vantagem aos mais corajosos.

▶ Principais feitos do projeto

1. Aprendemos a trabalhar com VBE functions
2. Aprendemos a trabalhar diretamente com controladores

▶ Lições aprendidas

Chegamos à conclusão de que esta cadeira é uma das cadeiras mais difíceis e interessantes do nosso curso, face à pouca informação relativa aos conteúdos. No início do curso e do projeto, deparamos com muitas dificuldades, nomeadamente na implementação dos dispositivos pela primeira vez, mas com um maior entendimento dos conteúdos, foi-se diminuindo a dificuldade… Exceto nas partes em que não existe labs para o rtc e para o serial port ( embora não tenhamos implementado isso). Relativamente ao projeto, como já tínhamos experiência em criar jogos ( em Laboratório de Desenho e Teste de Software , Fundamentos de Programação), já tínhamos uma ideia decente para a lógica do jogo, mas o mesmo foi dificultado pelo baixo nível de programação. Estamos orgulhosos do resultado final, embora tenhamos conhecimento que somos capazes de muito mais, mas devido a fatores externos (outros projetos), foi difícil distribuir as horas para cada cadeira (e horas de sono também). O facto de não termos um monitor atribuído à nossa turma prejudicou-nos relativamente a outras turmas durante o semestre.