U.PORTO
FEUP FACULDADE DE ENGENHARIA
UNIVERSIDADE DO PORTO

LCOM Project Report

Ano letivo 2015/2016

Mestrado Integrado em Engenharia Informática e Computação
Turma 6, Grupo 1
João Paulo Moreira Barbosa (up201406241@fe.up.pt)
José Luís Pacheco Martins (up201404189@fe.up.pt)

Índice

Introdução

Este projeto foi realizado no âmbito da unidade curricular Laboratório de Computadores, do 2º ano do Mestrado Integrado em Engenharia Informática e Computação da Faculdade de Engenharia da Universidade do Porto.

O objetivo deste projeto é implementar um jogo com base na linguagem C e em Assembly, que utilize o maior número de periféricos possíveis (timer, teclado, rato, placa gráfica...), numa versão do sistema operativo Minix, adaptada à unidade curricular LCOM.

Como tal decidimos implementar uma versão modificada do jogo Bomberman, de modo a cumprir os objetivos da unidade curricular.

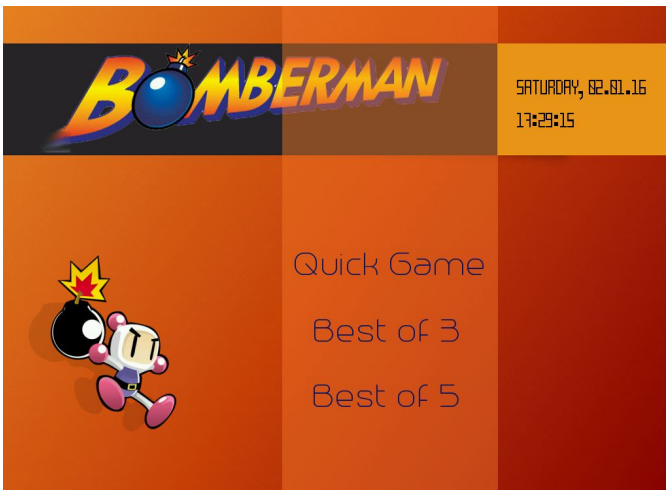
Ao longo deste relatório iremos explorar as principais funcionalidades do nosso jogo Bomberman, e também fornecer vários detalhes sobre a implementação do mesmo.

1.Instruções para o utilizador

1.1.Menu principal

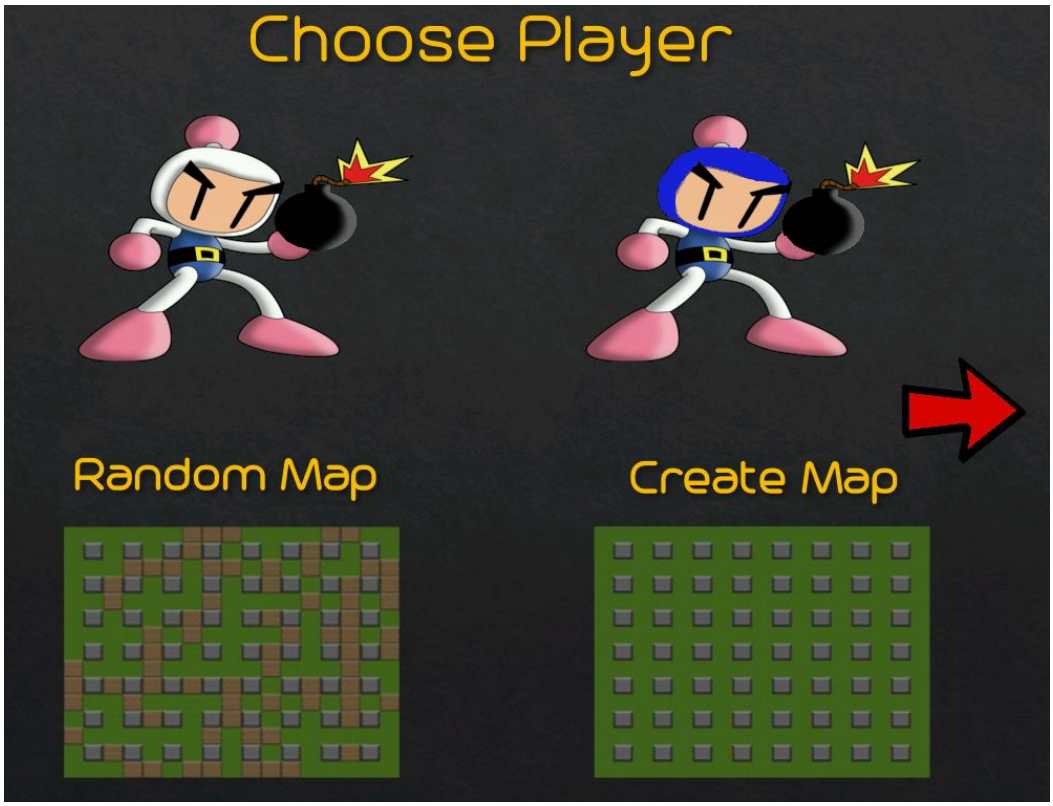
Este é o aspeto do menu principal do jogo. A partir daqui, o utilizador pode escolher entre as opções 'Singleplayer', 'Multiplayer' e 'Exit'. O controlo destas opções em todos os menus é feita via subscrição do rato. É também possível voltar atrás nos menus ou mesmo sair do jogo usando a tecla ESC.

1.2. Menu de Seleção do modo Multiplayer



Se o utilizador escolher o modo 'Multiplayer', então é-lhe apresentado o seguinte menu. Aqui, os jogadores podem decidir se querem jogo único, à melhor de 3, ou à melhor de 5. Para escolher, basta clicar com o rato em cima de uma das opções.

1.3. Opções de jogo

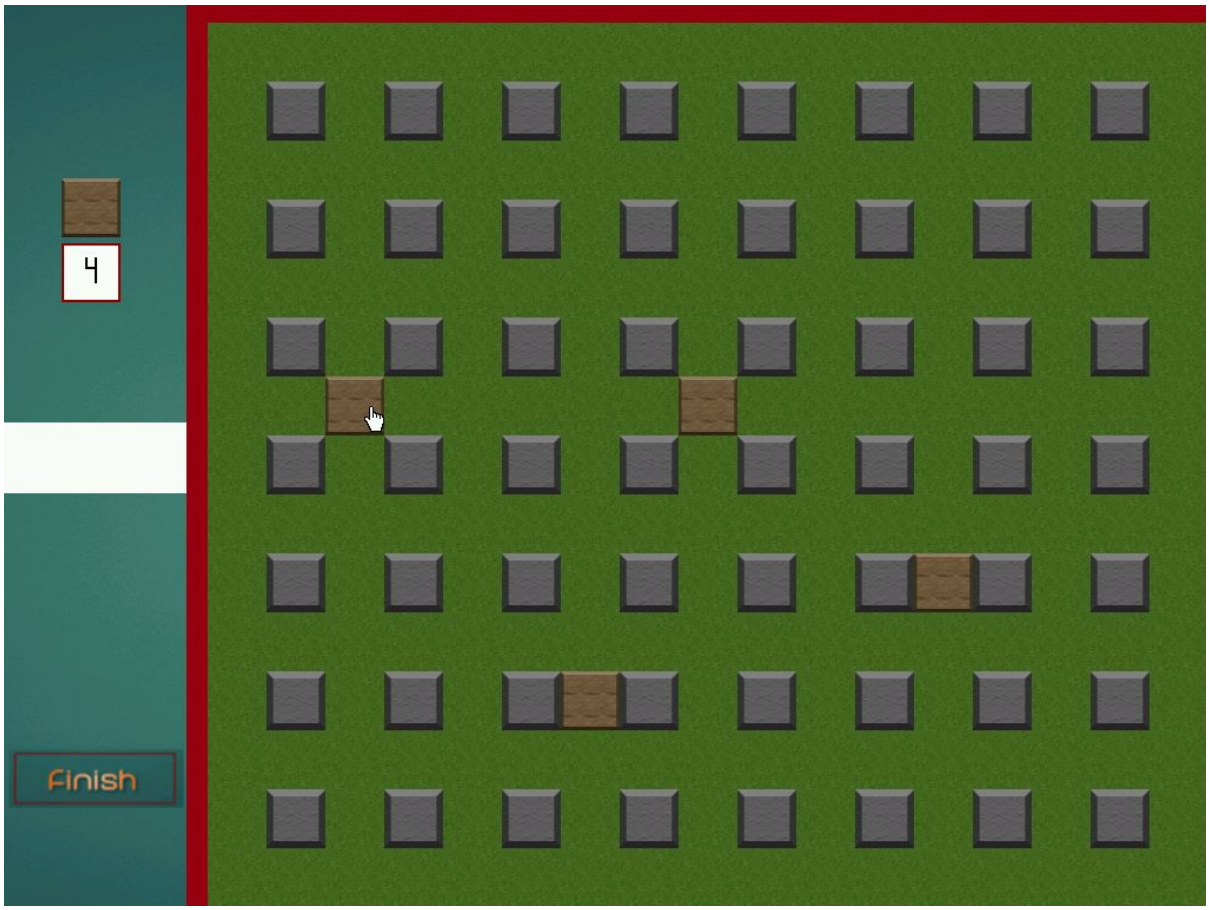


A partir do menu Multiplayer, ou diretamente da opção Singleplayer do menu principal, o utilizador pode proceder à escolha das opções do jogo.
Neste menu, é permitido escolher o avatar que prefere (branco ou azul), e se pretende criar um mapa, ou se quer que este seja feito de forma aleatória e automática.

Para selecionar as opções desejadas, basta utilizar o rato: sempre que se aproxima das imagens, um retângulo amarelo aparecerá à volta, significando que poderá bloquear a sua escolha. Ao bloquear (com um clique), o retângulo ficará vermelho, podendo, obviamente, cancelar esta mesma escolha clicando novamente na imagem.
Quando tiver uma opção de avatar e uma opção de mapa bloqueada, o utilizador pode clicar na seta e proceder para o jogo.

Note-se, no entanto, que no modo Multiplayer, só é permitido escolher o avatar de um jogador (o jogador que inicia o jogo no canto superior esquerdo), ficando o segundo obrigatoriamente restrito ao que falta.

1.4. Criar Mapa

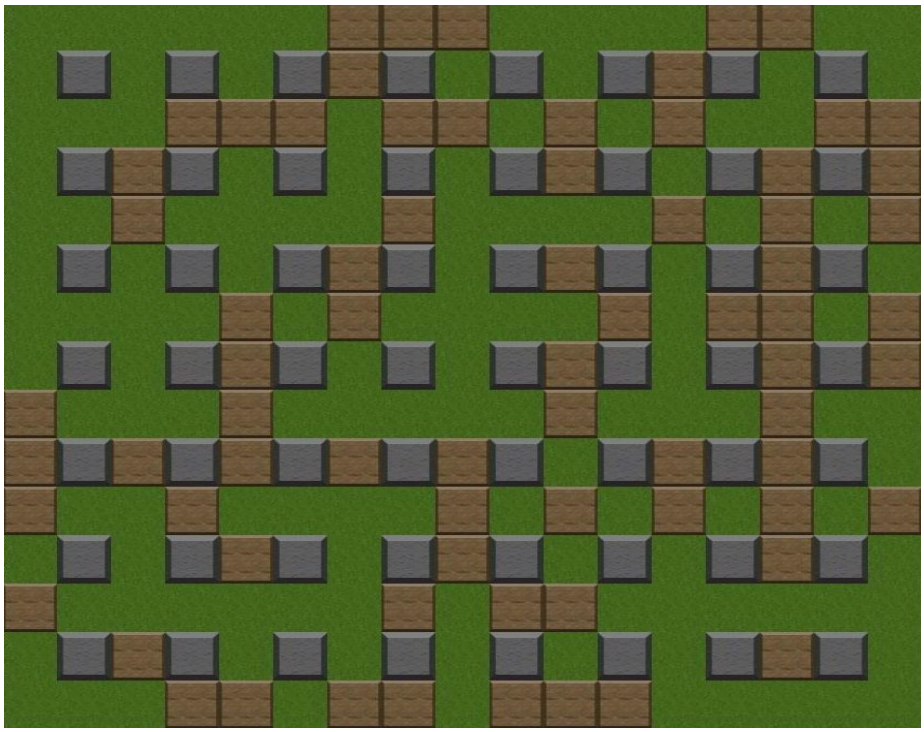


Esta é a interface apresentada ao jogador quando este seleciona a opção de criar um mapa no menu das opções de jogo.

Neste menu o jogador pode colocar blocos livremente no mapa, utilizando para isso o rato. Com um clique, o jogador coloca um bloco no mapa, e com outro clique em cima de um bloco já colocado, o jogador retira esse mesmo bloco.

O jogador deve colocar um minimo de 60 blocos no mapa para poder avançar. Na barra apresentada ao lado do mapa existe um contador do numero de blocos colocados até ao momento,um botão para quando o jogador quiser terminar a criação do mapa e prosseguir para o jogo e um retângulo branco utilizado para mostrar mensagens ao jogador: por exemplo, “60 BLOCKS MINIMUM”, quando o jogador tenta avançar para o jogo mas ainda não colocou o minimo de 60 blocos no mapa, e “PLAYER SPAWN”, quando o jogador tenta colocar um bloco no local onde o jogador nasce.

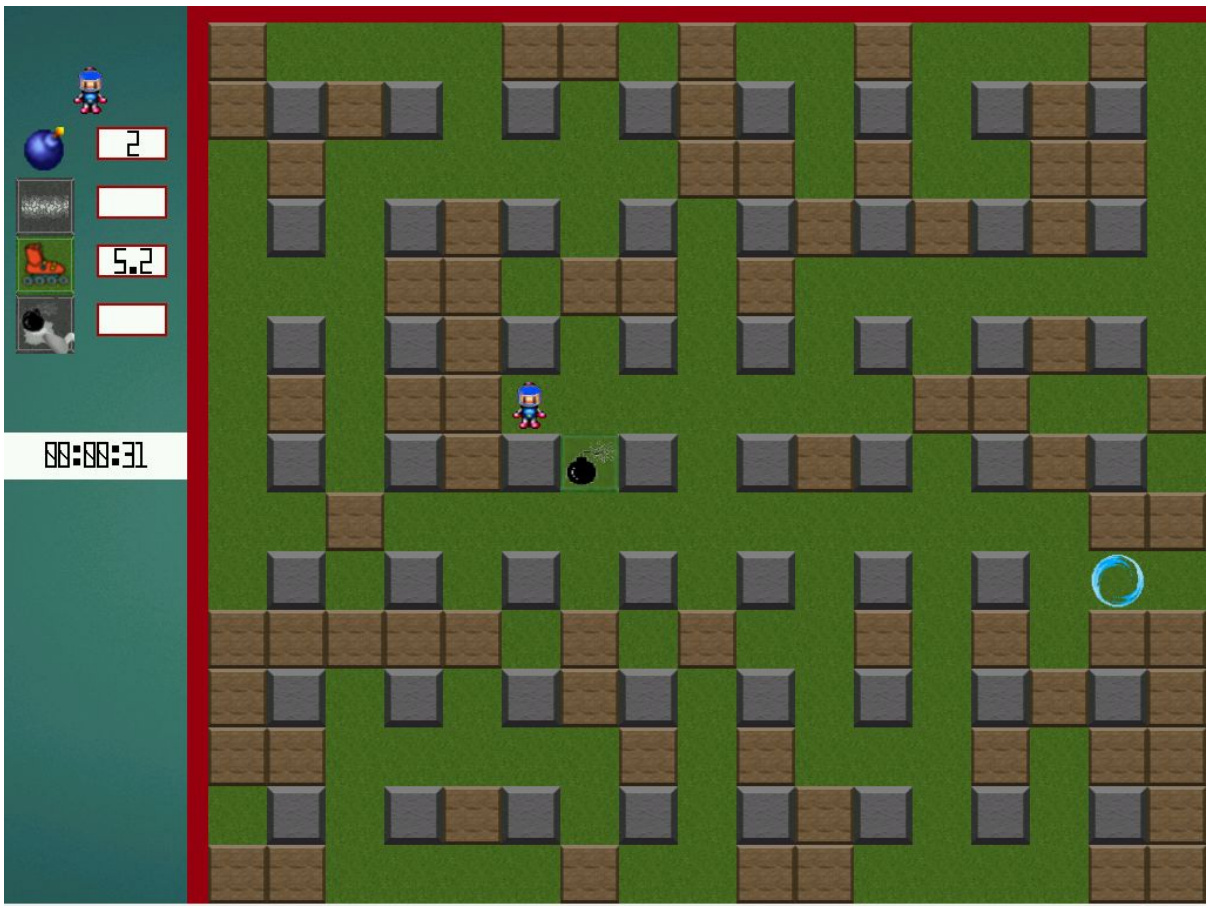
1.5. Mapa Aleatório



Se no menu das opções de jogo for selecionado mapa aleatório, o jogador iniciará de forma imediata o jogo e o mapa será criado de forma automática tendo já em consideração a posição inicial dos jogadores não colocando blocos nessas posições.

Deve-se referir que os blocos cinzentos mostrados na imagem em cima são fixos e não podem ser destruídos durante o jogo, o que varia de mapa para mapa é a disposição dos blocos castanhos.

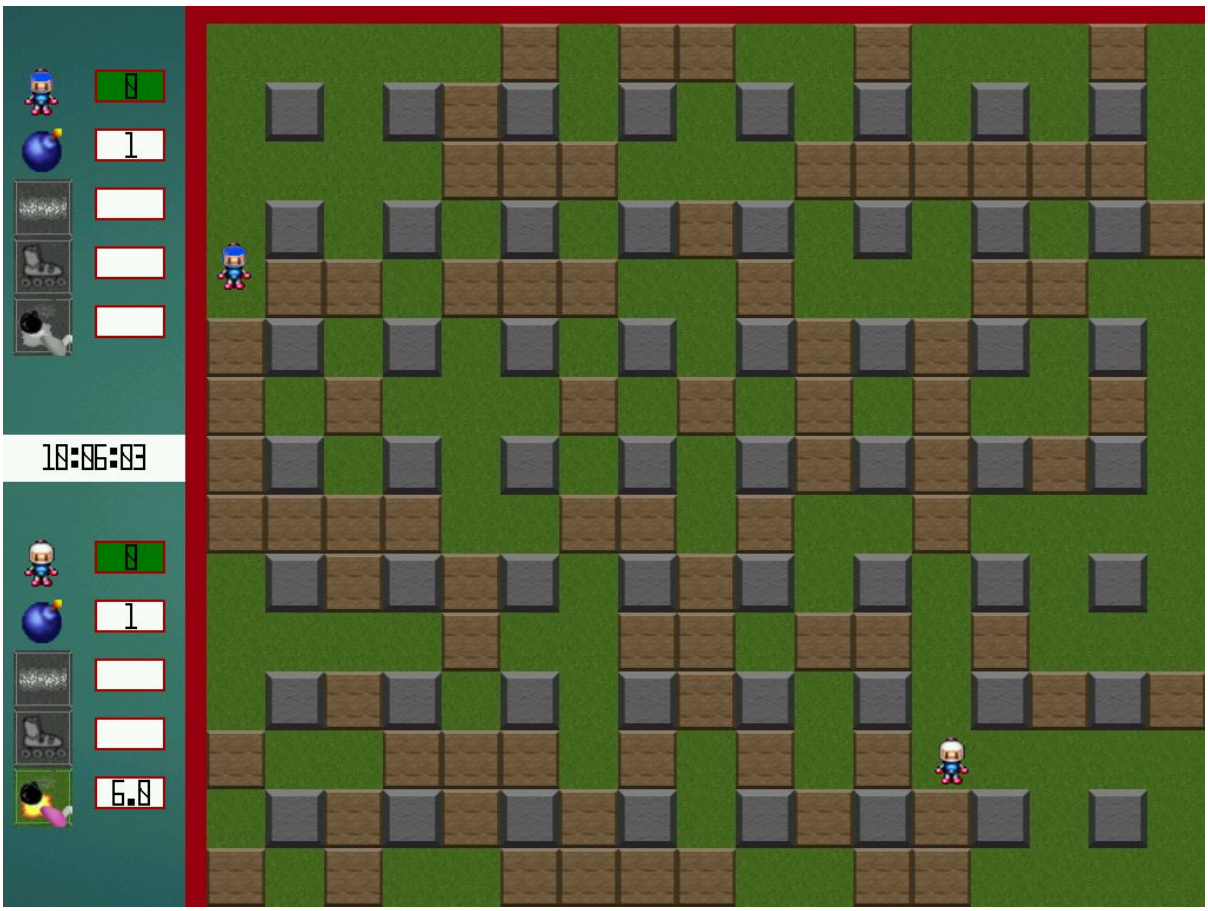
1.6. Modo Singleplayer



No modo Singleplayer, o jogador começa no meio do mapa, e o objetivo deste modo é conseguir chegar ao portal (representado pelo anel azul).
O movimento do avatar é feito pelo teclado, nas setas, e a colocação de bombas no ENTER.

Do lado esquerdo é visível as informações relativas ao jogador, nomeadamente, o número de bombas disponíveis, os power-ups ativos o tempo restante destes, e o tempo que está a demorar a passar o nível.

1.7. Modo Multiplayer



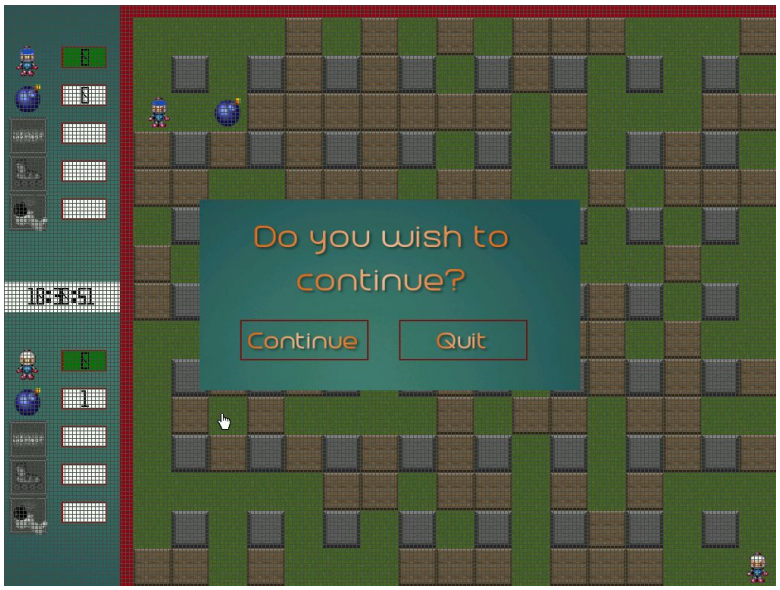
No modo Multiplayer, existem dois jogadores sendo que um inicia o jogo no canto superior esquerdo do mapa e o outro no canto inferior direito.

O objetivo deste modo é conseguir matar o adversário e ao mesmo tempo não ser morto pelo adversário. Para tal, deve garantir que o jogador está no alcance da bomba quando esta explode.

O movimento do avatar que inicia o jogo no canto superior esquerdo é feito pelo teclado, com o uso das teclas W (cima), S (baixo), A (esquerda), D (direita), e a colocação de bombas no SPACE, já o movimento do avatar que inicia o jogo no canto inferior direito é feito também usando o teclado, mas através das setas, e a colocação das bombas faz-se com o uso da tecla ENTER.

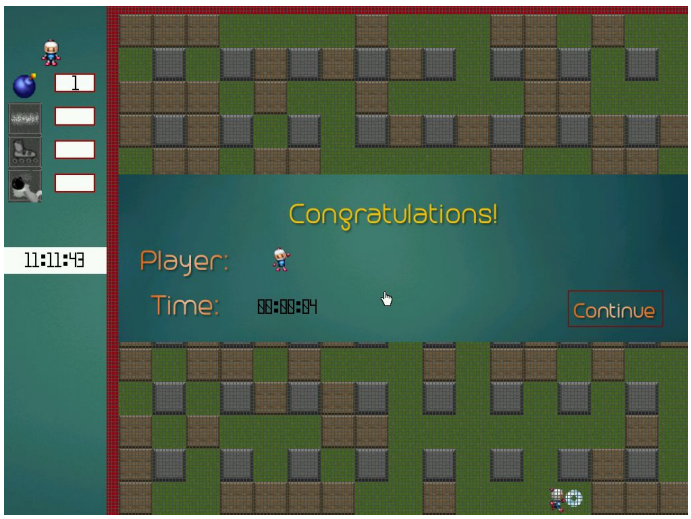
Do lado esquerdo são visíveis informações relativas aos dois jogadores, nomeadamente, o número de bombas disponíveis, os power-ups ativos o tempo restante destes, e as horas do dia no retângulo branco central que divide a informação dos dois jogadores.

1.8. Menu de Pausa



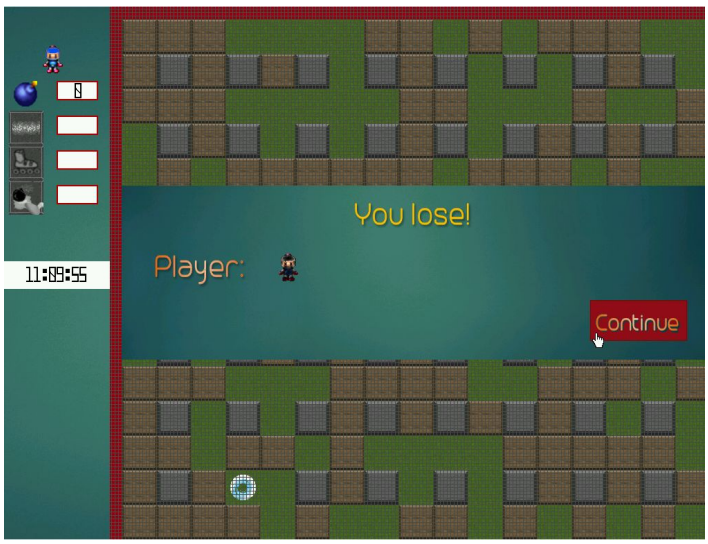
Em qualquer dos modos de jogo (Singleplayer ou Multiplayer) o jogador pode pausar o jogo a qualquer momento, usando para isso a tecla ESC e deparar-se-á com o menu apresentado ao lado. Neste menu o jogador pode utilizar o rato para continuar o jogo ou sair, mas também utilizar as teclas ESC e ENTER para continuar a jogar.

1.9. Menu final do jogo



Este menu é apresentado no modo Singleplayer, quando o jogador consegue alcançar o portal, mostrando o avatar do jogador e o tempo que demorou a chegar ao portal.

Menu apresentado no modo Singleplayer, quando o jogador morre com a própria bomba.



Menu apresentado no modo Multiplayer, mostrando o avatar do jogador vencedor e o jogador derrotado, ligeiramente queimado pela bomba.

1.10. Power-Ups

Existem um total de 4 power-ups implementados no jogo:

Bomba

Este power-up, quando apanhado, incrementa em 1 o número de bombas que o jogador pode colocar no mapa ao mesmo tempo. O número de bombas nunca diminiu, podendo o jogador ter até um máximo de 20 bombas.

Aumento da Velocidade

Este power-up, quando apanhado, aumenta a velocidade do jogador para o dobro, durante 10 segundos. Se o jogador conseguir apanhar novamente este power-up durante estes 10 segundos, a velocidade mantém-se o dobro da normal, mas são somados 10 segundos ao tempo que restava para o power-up terminar.

Aumento do alcance da explosão

Este power-up, quando apanhado, aumenta o alcance das bombas do jogador em um bloco durante 10 segundos. Se o jogador conseguir apanhar novamente este power-up durante estes 10 segundos, o alcance das bombas é aumentado novamente, e a duração do power-up também é incrementado em 10 segundos.

Chutar a Bomba

Este power-up, quando apanhado, permite ao jogador chutar as suas bombas. As bombas chutadas deslizam pelo mapa até encontrarem um obstáculo, parando aí, ou até explodirem. Este tem uma duração de 10 segundos. Se o jogador conseguir apanhar novamente este power-up durante estes 10 segundos, são somados 10 segundos ao tempo que restava para acabar o power-up, podendo, assim, tê-lo ativo durante mais tempo.

2. Estado do projeto

Todas as funcionalidades referidas anteriormente foram implementadas e estão a funcionar devidamente.

Era pretendido também implementar a porta série no jogo, como foi referido na proposta do projeto, mas devido á falta de tempo, tomámos a decisão de apresentar um jogo mais estável e agradável graficamente, ao invés de apresentar um jogo mais rudimentar e uma tentativa de implementação da porta série.

Como observação, o uso da porta série destinava-se a permitir que o modo Multiplayer funcionasse em computadores separados.

Dispositivo	Funcionalidade	Com Interrupções ?
Timer	Controlar a frame rate, animações, duração dos power-ups e o tempo de jogo no modo Singleplayer.	Sim
Teclado	Usado para controlar o jogador, e também para navegação nos menus.	Sim
Rato	Usado no controlo dos menus ao longo do programa, e na criação do mapa.	Sim
Video Card	Apresentação visual dos aspetos gráficos do jogo (menu, avatares, rato...)	Não
RTC	Manutenção das informações referentes às horas e ao dia	Não (por polling)

2.1. Timer

As interrupções do timer permitem controlar a frame rate do jogo, gerir as animações de vários objetos (como o avatar do jogador, bomba, portal, explosões, etc.), contar o tempo que o jogador demora a alcançar o portal no modo Singleplayer, e atualizar a duração dos power-ups. As funções referentes ao uso do timer encontram-se implementadas nos ficheiros **timer.h** e **timer.c**, e estas são usadas nos ficheiros **Bomberman.c** e **menu.c**.

2.2. Teclado

O Teclado permite controlar os avatares de cada jogador e os seus power-ups, retroceder para o menu anterior ainda na navegação nos menus, através da tecla ESC, e pausar o jogo, através da mesma tecla.

As funções referentes ao uso do teclado encontram-se implementadas nos ficheiros **keyboard.h** e **keyboard.c**, e estas funções são usadas nos ficheiros **Bomberman.c** e **menu.c**.

2.3. Rato

O Rato permite selecionar as opções apresentadas nos menus e permite também ao utilizador criar o seu próprio mapa.

As funções referentes ao uso do rato encontram-se implementadas nos ficheiros **mouse.h** e **mouse.c**, e estas funções do rato são usadas nos ficheiros **Bomberman.c** e **menu.c**.

2.4. Video Card

A placa de vídeo é utilizada para mostrar toda a interface gráfica do jogo.
Esta é utilizada em modo gráfico no modo 0x117 com a resolução de 1024x768, permitindo-nos usar um total de 64K cores.

Por forma a garantir que o rato era atualizado mais vezes no menu e que o mapa não precisava de estar sempre a ser redesenhado, foram usados 2 buffers, para além da video_mem (buffer principal).

O terceiro buffer é utilizado para desenhar o rato (quando o jogador se encontra a navegar nos menus) ou o mapa (em situação de jogo). O jogador é depois desenhado no segundo buffer (em situação de jogo).

No desenvolvimento da interface gráfica do nosso jogo foram apenas utilizados bitmaps.
São utilizadas sprites animadas, detecção de colisões, e uma fonte para escrever texto no ecrã.

As funções referentes ao uso da placa gráfica encontram-se implementadas nos ficheiros **video_gr.h**, **video_gr.c**, **vbe.h**, **vbe.c**, **Bitmap.h** e **Bitmap.c**.
Estas funções são usadas nos ficheiros **Bomberman.c** e **menu.c** em várias ocasiões, assim como em "métodos" de desenho, nos módulos map, player, bomb e mouse.

2.5. RTC

O RTC (Real Time Clock) é utilizado para mostrar a data, hora e dia da semana no menu inicial, e para mostrar a hora no modo Multiplayer.
As funções referentes ao uso do RTC encontram-se implementadas nos ficheiros **rtc.h** e **rtc.c**, que são utilizadas nos ficheiros **Bomberman.c** e **menu.c**.

3. Organização/Estrutura do código

Contribuição - (Responsável/Outro elemento)

3.1. Bomberman

Este é o principal módulo do projeto, responsável por relacionar todos os outros módulos. Neste módulo estão as funções responsáveis por implementar o jogo propriamente dito, nos seus diferentes modos.

Responsável: Ambos
Contribuição: 50/50
Importância para o projeto: 15%

3.2. Map

Este modulo é responsável pela implementação do mapa. Está implementado de forma a assemelhar-se à programação orientada a objetos, possuindo uma struct com os seus "membros dados" e vários "métodos" que permitem a utilização da "classe".

Existem métodos para desenhar o mapa, para colocar os power-ups no mapa de forma aleatória, para detetar colisões, entre outros métodos.

O mapa é representado por um array bidimensional de inteiros que indicam o que vai ser desenhado naquela posição do mapa. Todos os bitmaps que formam o mapa são quadrados com dimensões 50x50 pixeis e são desenhados com base nesse array.

Responsável: José Martins
Contribuição: 60/40
Importância para o projeto: 13%

3.3. Player

Este modulo é responsável pela implementação do jogador propriamente dito. Está implementado de forma a assemelhar-se à programação orientada a objetos, possuindo uma struct com os seus "membros dados" e vários "métodos" que permitem a utilização da "classe", tal como no mapa.

Existem métodos para desenhar o jogador, para atualizar as informações dos power-ups das suas bombas e da sua posição, tanto no array do mapa como a posição em pixels, etc. A função que atualiza a posição do jogador realiza detecção de colisões, a par da função definida no mapa.

Responsável: João Barbosa
Contribuição: 60/40
Peso para o projeto: 13%

3.4. Bomb

Este modulo é responsável pela implementação da bomba, e está implementado de forma a assemelhar-se à programação orientada a objetos, possuindo uma struct com os seus "membros dados" e vários "métodos" que permitem a utilização da "classe", tal como no mapa e no player.
A bomba tem vários estados, e os métodos desenvolvidos permitem atualizar o estado da bomba, tal como desenhá-la no mapa.

Responsável: Ambos
Contribuição: 50/50
Peso para o projeto: 8%

3.5. Bitmap

Este modulo é responsável por todo o tratamento de bitmaps, necessário para a execução de toda a parte gráfica do nosso projeto.
Existem várias funções implementadas neste modulo, onde se destacam as funções responsáveis pela leitura, desenho e destruição dos bitmaps, assim como as funções que nos permitem desenhar texto e números no ecrã mediante a utilização de uma fonte.

Responsável: Ambos
Contribuição: 50/50
Peso para o projeto: 8%

3.6. Menu

Este módulo é responsável por todos os menus implementados no nosso jogo. Foi desenvolvido no final do projeto e possui várias funções, cada uma responsável pela apresentação de um menu. Possui também a função responsável pela desenho da data, hora e dia no menu.

Responsável: Ambos
Contribuição: 50/50
Peso para o projeto: 13%

3.7. Main

Este é talvez um dos módulos mais simples do projeto, visto que apenas inicia o modo gráfico da placa de video, faz a leitura de todas as bitmaps do jogo e faz a chamada de funções definidas nos módulos menu e Bomberman, conforme o atual estado do jogo.

Responsável: Ambos
Contribuição: 50/50
Peso para o projeto: 1%

3.8. Keyboard

Este módulo contém as funções relativas à interação do utilizador com o teclado. As principais funcionalidades implementadas no módulo são a subscrição e cancelamento das interrupções do teclado, assim como a leitura dos códigos de cada tecla.
Este módulo for inteiramente desenvolvido nas aulas laboratoriais.

Responsável: Ambos
Contribuição: 50/50
Peso para o projeto: 5%

3.9. Mouse

Este módulo implementa as funcionalidades oferecidas pelo rato usadas no nosso projeto, sendo responsável pela leitura das informação e consequente atualização do rato.
Parte deste módulo foi desenvolvido nas aulas laboratoriais, mas, para o desenvolvimento do projeto, decidimos implementar uma estrutura de forma a assemelhar-se à programação orientada a objetos, possuindo uma struct com os seus "membros dados" e vários "métodos" que permitem a utilização da "classe", tal como no mapa, player e bomba.

Responsável: Ambos
Contribuição: 50/50
Peso para o projeto: 5%

3.10. RTC

Este módulo é responsável pela implementação das funções necessárias para conseguirmos obter as informações sobre a data, horas e dia da semana, através do periférico RTC (Real Time Clock).

Responsável: João Barbosa
Contribuição: 60/40
Peso para o projeto: 5%

3.11. Sprite

Este módulo é exclusivamente utilizado para guardar as informações da sprite do jogador, contendo a sua posição e velocidade, e os bitmaps associados ao jogador. É importante referir que os bitmaps da sprite estão organizados em arrays para facilitar a criação de animações.

Responsável: José Martins
Contribuição: 60/40
Peso para o projeto: 5%

3.12. Timer

Este trata-se de um módulo bastante curto e simples, que contém as funções que foram necessárias para a implementação do timer no nosso jogo, sendo elas a subscrição e o cancelamento de interrupções.

As funções presentes neste módulo foram inteiramente desenvolvidas nas aulas laboratoriais.

Responsável: José Martins
Contribuição: 60/40
Peso para o projeto: 1%

3.13. Utilities

Este módulo foi utilizado apenas para definir algumas estruturas de dados que se mostravam importantes em vários outros módulos, foi a forma encontrada para evitar repetição de código.

Responsável: José Martins
Contribuição: 60/40
Peso para o projeto: 1%

3.14. Vbe

Este módulo foi inteiramente desenvolvido nas aulas laboratoriais e contem funções importantes para a inicialização da placa gráfica, em modo gráfico.

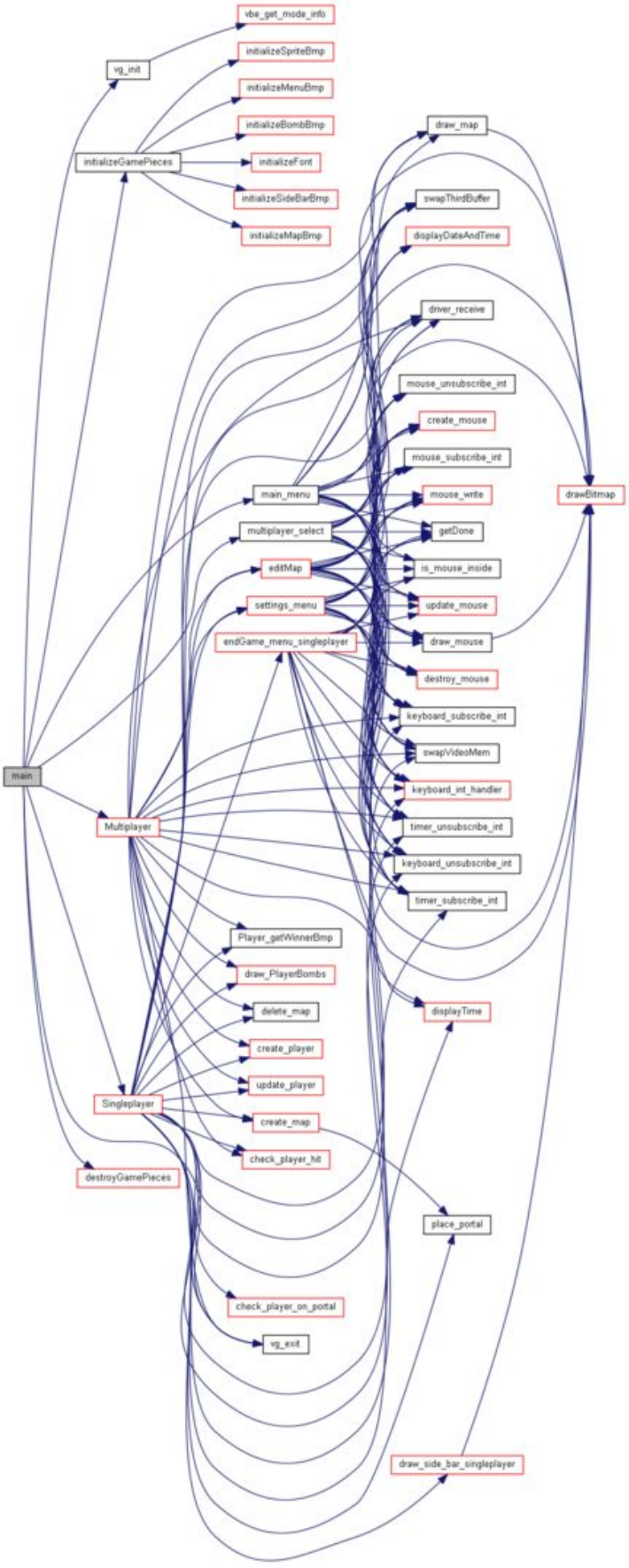
Responsável: José Martins
Contribuição: 60/40
Peso para o projeto: 1%

3.15. Video_gr

Este módulo contem as funções que permitem a inicialização da placa gráfica em modo gráfico, assim como voltar ao modo texto.
Contém também funções que permitem desenhar retângulos e linhas e copiar o conteúdo de um buffer para outro.

Responsável: José Martins
Contribuição: 60/40
Peso para o projeto: 6%

4. Diagrama de chamada de funções



5.Detalhes da implementação

5.1 Bitmaps

Para a utilização de bitmaps recorremos ao código fornecido no website do estudante Henrique Ferrolho, que, por sua vez, se baseou num post do FedoraForum.

Note-se que, apesar de o código ter sido extraído das fontes acima referidas, foi necessário alterá-lo, para se adequar á estrutura do nosso código. Concretamente, foi necessário modificar a função **drawBitmap**:
-De forma a ler a informação corretamente no modo de vídeo que foi utilizado;
-Permitir a utilização de uma cor como transparência (no nosso caso verde 0x07E0);

Foi também necessário criar uma nova função, **drawBitmapWithBoundaries**, para ser possível restringir a imagem a ser desenhada (exemplo: desenho de texto - fonte).

5.2 Colisões

As colisões foram um tópico importante no desenvolvimento do projeto, que causou alguns sobressaltos no início do desenvolvimento do mesmo. No entanto, conseguimos encontrar uma forma simples e eficaz de detetar as colisões do jogador com o mapa, devido às estruturas de dados desenvolvidas.

De forma sucinta, visto que a "classe" player tinha duas posições, uma no array do mapa e outra na "classe" sprite (um dos "membros dado" do player, que corresponde á posição real e onde ele é desenhado), o que foi feito, então, foi verificar se o jogador estava alinhado com os blocos: se não estivesse, estaria a colidir.

Uma vez alinhado foi desenvolvida uma função na "classe" map que verificava se o bloco que estava adjacente ao jogador, de acordo com a direção em que ele se estava a mover, era um bloco com o qual ele devia colidir. Caso isto se verificasse, o jogador não se moveria, simulando assim o efeito da colisão.

5.3 Estados

Tal como se pedia num dos tópicos para este projeto, foi implementado o mecanismo de máquina de estados para alguns módulos, mais concretamente para o estado do jogo, o estado do jogador, e o estado da bomba.

Os estados de jogo são os seguintes:

```
typedef enum {MENU, BEST_OF_1, BEST_OF_3, BEST_OF_5, MULTIPLAYER,
    SINGLEPLAYER, END, MULTIPLAYERSELECT} GAME_STATE;
```

MENU, MULTIPLAYERSELECT - O jogo encontra-se num destes estados, quando o utilizador está a navegar pelos diferentes menus.

BEST_OF_1, BEST_OF_3, BEST_OF_5 - Estes estados permitem fazer a distinção das várias opções de jogo, para o modo Multiplayer.

SINGLEPLAYER - Utilizado quando o jogador está em modo Singleplayer. Serve também de informação, para saber como criar o mapa.

MULTIPLAYER - Serve de informação, para saber como criar o mapa.

Os estados do jogador são:

```
typedef enum { ALIVE, DEAD } PLAYER_STATE;
```

ALIVE, DEAD - Utilizados, essencialmente, para reconhecer o vencedor e perdedor do jogo. Sempre que um jogador é criado, começa no estado ALIVE, e muda quando é atingido por uma bomba.

Os estados da bomba são:

```
typedef enum { DEPLOYED, EXPLODING, DONE } BOMB_STATE;
```

DEPLOYED - Estado da bomba, quando o jogador deposita uma bomba no mapa.

EXPLODING - Passado 3 segundos da bomba estar no estado DEPLOYED, passa automaticamente a este estado. Neste estado, o conteúdo do mapa é atualizado.Significa que a bomba está a explodir.

DONE - Após a bomba ter estado 2 segundos no estado EXPLODING, passa ao estado referido, ocorre a atualização do mapa, e posterior remoção da bomba, tanto do mapa, como do jogador.

5.4 Assembly

Outro dos outros objetivos requisitados para o projeto, e devidamente cumprido, foi a criação e utilização de código em assembly, mais concretamente, extended assembly.

Este tipo de codificação foi utilizado para praticamente todas as funções "get" do Player.

Exemplo:

```
double Player_getVelocityCounter(player_t *player) {
    double out;
    asm ("movl %1, %%eax;  movl %%eax, %0;"
        :"=r"(out)         /* output */
        :"r"(player->velocity_counter)         /* input */
        :"%eax"          /* clobbered register */
        );
    return out;
}
```

Também foi utilizado numa das funções de atualização das bombas:

```
void move_bomb(bomb_t* b,DIRECTION dir,int x_end,int y_end){
    asm ("movl %1, %%eax;  movl %%eax, %0;"
        :"=r"(b->d)         /* output */
        :"r"(dir)          /* input */
        :"%eax"           /* clobbered register */
        );

    asm ("movl %1, %%eax;  movl %%eax, %0;"
        :"=r"(b->x_endpos)         /* output */
        :"r"(x_end)               /* input */
        :"%eax"                  /* clobbered register */
        );


    asm ("movl %1, %%eax;  movl %%eax, %0;"
        :"=r"(b->y_endpos)         /* output */
        :"r"(y_end)               /* input */
        :"%eax"                  /* clobbered register */
        );

    asm ("movl %1, %%eax;  movl %%eax, %0;"
        :"=r"(b->kicked_bomb)         /* output */
        :"r"(1)                      /* input */
        :"%eax"                     /* clobbered register */
        );
}
```

A aprendizagem para a criação e desenvolvimento de código em Inline Extended Assembly foi feito de forma autônoma e fora de aulas, com recurso ao site da Ibiblio - GCC Inline Assembly - HOWTO.

6. Avaliação da Unidade Curricular

De uma forma geral, a unidade curricular tornou-se uma grande fonte de aprendizagem e desenvolvimento técnico. Isto porque é necessário, num curto espaço de tempo, compreender e desenvolver o que nos é proposto, sem termos uma ideia muito completa de como é suposto realizar as funções pedidas nem do funcionamento do periférico a ser explorado.

No entanto, existiram algumas adversidades que, na nossa opinião, poderiam ser, eventualmente, superadas:

- O tempo de espera por vezes demasiado elevado para esclarecer dúvidas durante as aulas laboratoriais.
- A submissão das aulas laboratoriais poderia ser realizada numa data fixa, independente dos horários dos alunos.
- Todo o conteúdo necessário à concretização dos laboratórios deveria estar nos guiões laboratoriais, e não num misto entre estes e os slides das aulas teóricas e os guiões laboratoriais, pois às vezes é despendido muito tempo à procura da informação.

7. Autoavaliação

João Barbosa

Participação: 50%

Contribuição: 50%

José Martins

Participação: 50%

Contribuição: 50%

8. Instruções de instalação

Para instalar o jogo é necessário colocar o ficheiro contido na pasta **conf** do diretório **proj** na pasta **etc/system.conf.d** (necessário permissões root).

Depois é necessário fazer **make** dentro da pasta **src** do diretório **proj** e executar o seguinte comando na mesma pasta "**sh run.sh**".