

Faculdade de Engenharia da Universidade do Porto



BATTLESHIPS

LCOM_2022/2023_T16_G5



Membros:

João Miguel da Silva Lourenço (up202108863)

Tiago Ribeiro de Sá Cruz (up202108810)

Tomás Filipe Fernandes Xavier (up202108759)

Resumo

Este projeto foi desenvolvido no âmbito da cadeira de Laboratório de Computadores.

Com este projeto visamos utilizar vários drivers no sistema de MINIX para desenvolver uma versão do jogo “*Batalha Naval*”, ou, em inglês, “*Battleships*”.

Índice

1. User Instructions	3
1.1. Ecrã Inicial	3
1.2. Dentro da partida	3
1.2.1. Colocação de navios na board	4
1.2.2. Game loop	5
1.2.2.1. Fase de Ataque	5
1.2.2.2. Fase de Defesa	6
1.3. Fim de partida	7
2. Project Status	9
2.1. Timer	9
2.2. Keyboard	10
2.3. Mouse	10
2.4. Video Card	10
2.5. RTC	10
3. Code organization/structure	11
3.1. I/O Devices	11
3.1.1. controller/timer	11
3.1.2. controller/keyboard	11
3.1.3. controller/mouse	11
3.1.4. controller/graphic	11
3.1.5. controller/rtc	11
3.2. Main	12
3.3. Draw	12
3.4. Model	12
3.4.1. model/model.c	12
3.4.2. model/sprite.c	12
3.5. Utils	12
4. Implementation details	13
4.1. Drawing buffers	13
4.2. Round timer	13
5. Conclusions	14

1. User Instructions

1.1. Ecrã Inicial

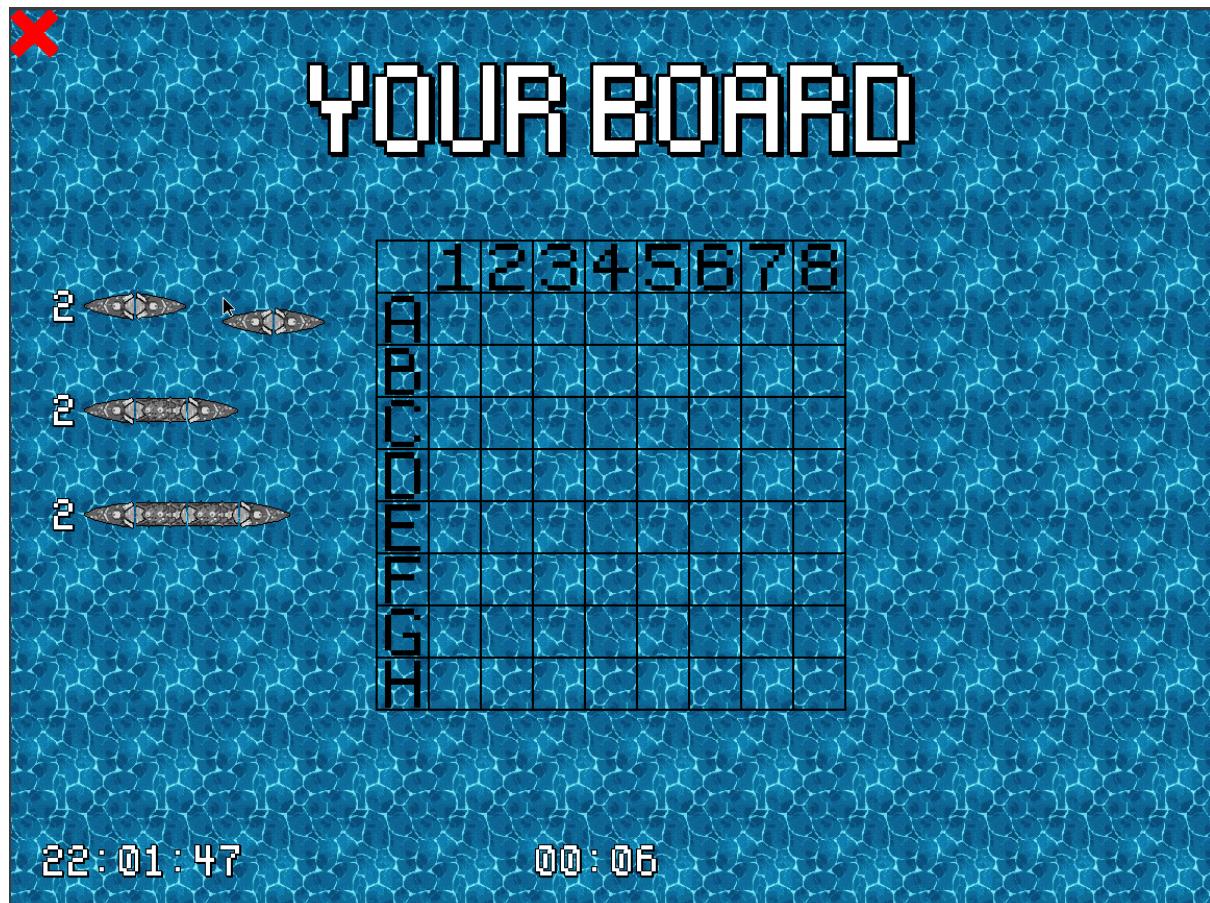


Ao iniciar o programa, o utilizador é apresentado com um menu simples onde pode ver o título do jogo (“**Battleships**”), o tempo atual (canto inferior esquerdo), sair do jogo e fechar o programa (centro da tela) e começar uma partida (centro da tela, acima do “EXIT”). Tanto a escolha de começar uma partida como de sair e fechar o programa podem ser realizadas através de clicar no local designado com o rato ou através do teclado.

1.2. Dentro da partida

Inicialmente, o utilizador será apresentado com uma visão da sua board de 8x8, ou seja 8 linhas por 8 colunas, onde as linhas são identificadas pelas letras do A ao H e as colunas são identificadas pelos números do 1 ao 8.

1.2.1. Colocação de navios na board



O utilizador tem a seu dispor 6 navios, 3 tipos de navios de dimensões diferentes (2 navios de cada tipo). Navios esses que terão que ser estrategicamente colocados na board com o intuito de fazer o oponente falhar ao máximo as tiles que possuem navios.

Para selecionar qual navio colocar, o utilizador pode clicar na imagem do navio que deseja, situada no lado esquerdo da tela, e depois clicar na board. Para facilitar a colocação de cada navio, depois de selecionado um navio ele é imprimido na tela em conjunto com o cursor do mouse, para o utilizador ter uma melhor ideia de quais tiles exatamente estão a ser preenchidas com um navio.

Alternativamente, o utilizador pode fazer as mesmas ações com o teclado. Para selecionar um navio com o teclado o utilizador deve premir a tecla referente ao tamanho de cada navio, por exemplo, se o utilizador quiser selecionar o navio de tamanho 2 então deve premir a tecla **2**. Para o colocar na board é só preciso escolher a tile através do **WASD**, **W** para ir para cima, **S** para baixo, **A** para a esquerda e **D** para a direita.

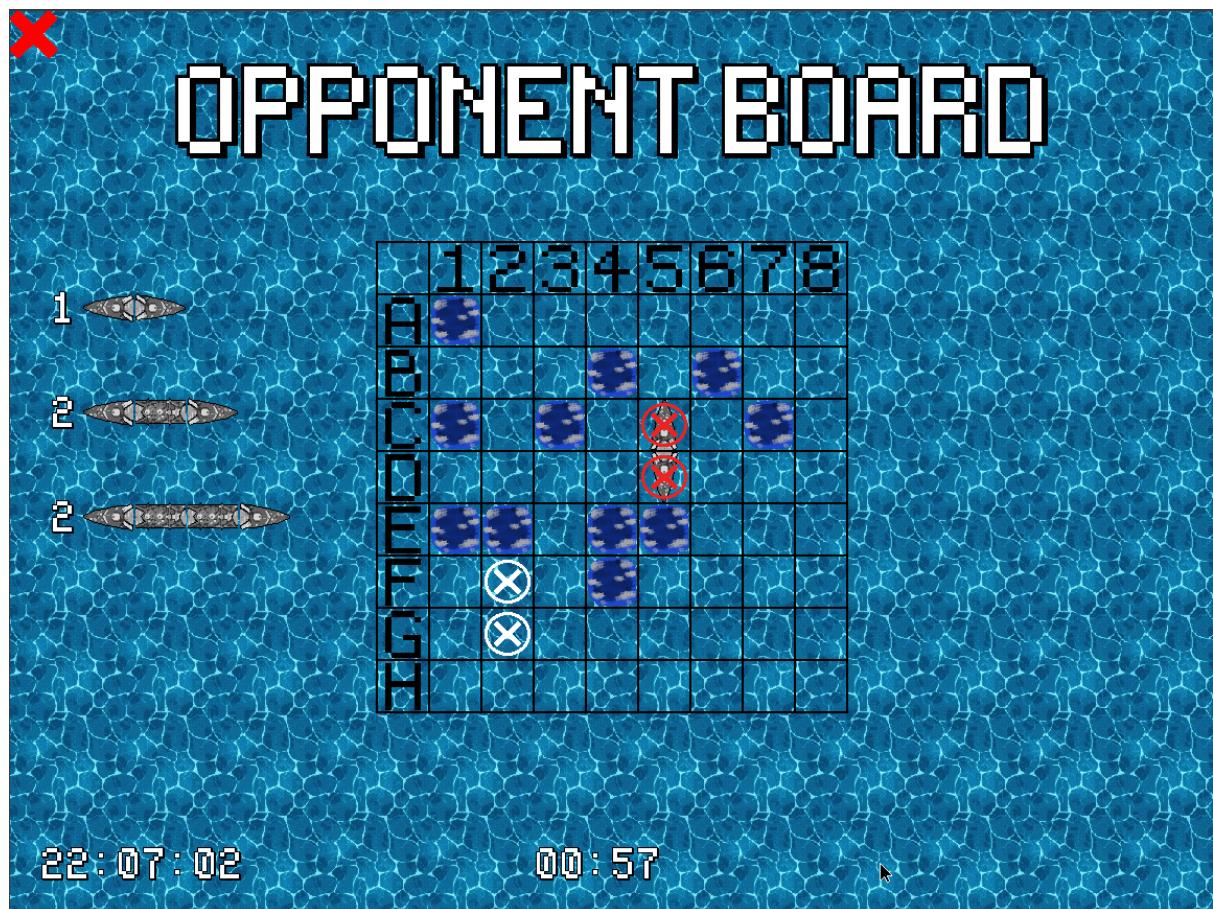
Além do que foi já referido, o utilizador pode também mudar a orientação de cada navio, seja horizontalmente ou verticalmente. Para o fazer deve utilizar ou a tecla **R** no teclado, ou o botão direito do rato.

1.2.2. Game loop

Uma partida de **Battleships** consiste em duas fases alternadas:

- Fase de ataque: Fase em que o utilizador vê a board do adversário, restringida pela informação que o utilizador tem sobre ela (os navios do adversário estão ocultos ao utilizador até serem destruídos).
- Fase de defesa: Fase em que o utilizador vê a sua própria board enquanto o oponente decide qual **tile** escolher para atacar.

1.2.2.1. Fase de Ataque



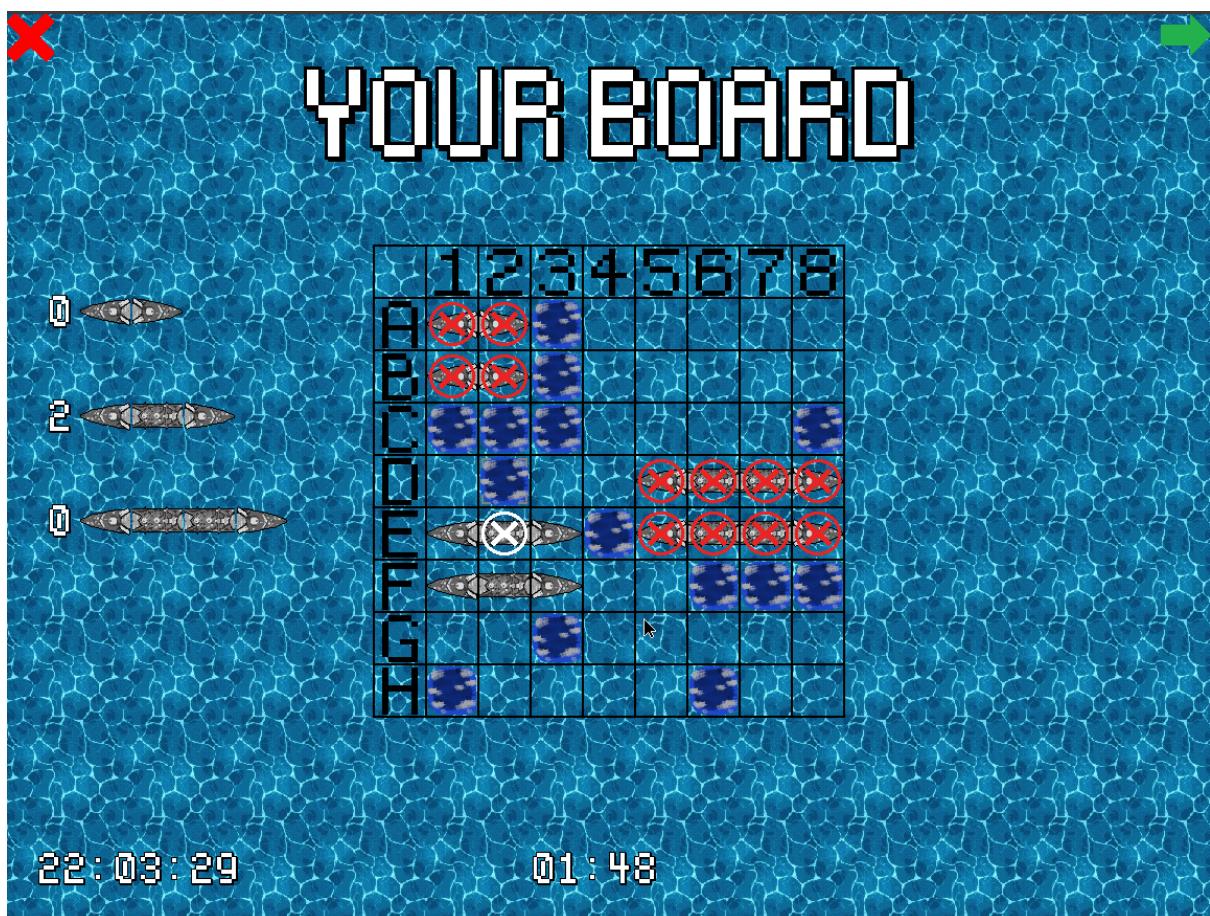
Durante a fase de ataque, o utilizador será apresentado com a board do oponente, podendo, no entanto, alternar entre as boards, para ter uma melhor noção do estado do jogo e saber onde estão os próprios navios.

O objetivo é tentar atingir as tiles que possuem navios até todos os navios terem sido destruídos.

Durante a fase de ataque, depois de escolhida qual tile atacar, há 3 resultados diferentes:

- O utilizador não acertou nenhum navio. Neste caso a tile terá uma mudança visual para representar a falta de navio e não pode ser mais selecionada para atacar.
- O utilizador acertou um navio, mas o navio ainda não foi completamente destruído. Neste caso a tile escolhida apresenta uma cruz branca.
- O utilizador acertou um navio e ele foi completamente destruído. Neste caso o navio é posto completamente à vista do utilizador, com uma cruz vermelha em cada tile que está situado.

1.2.2.2. Fase de Defesa



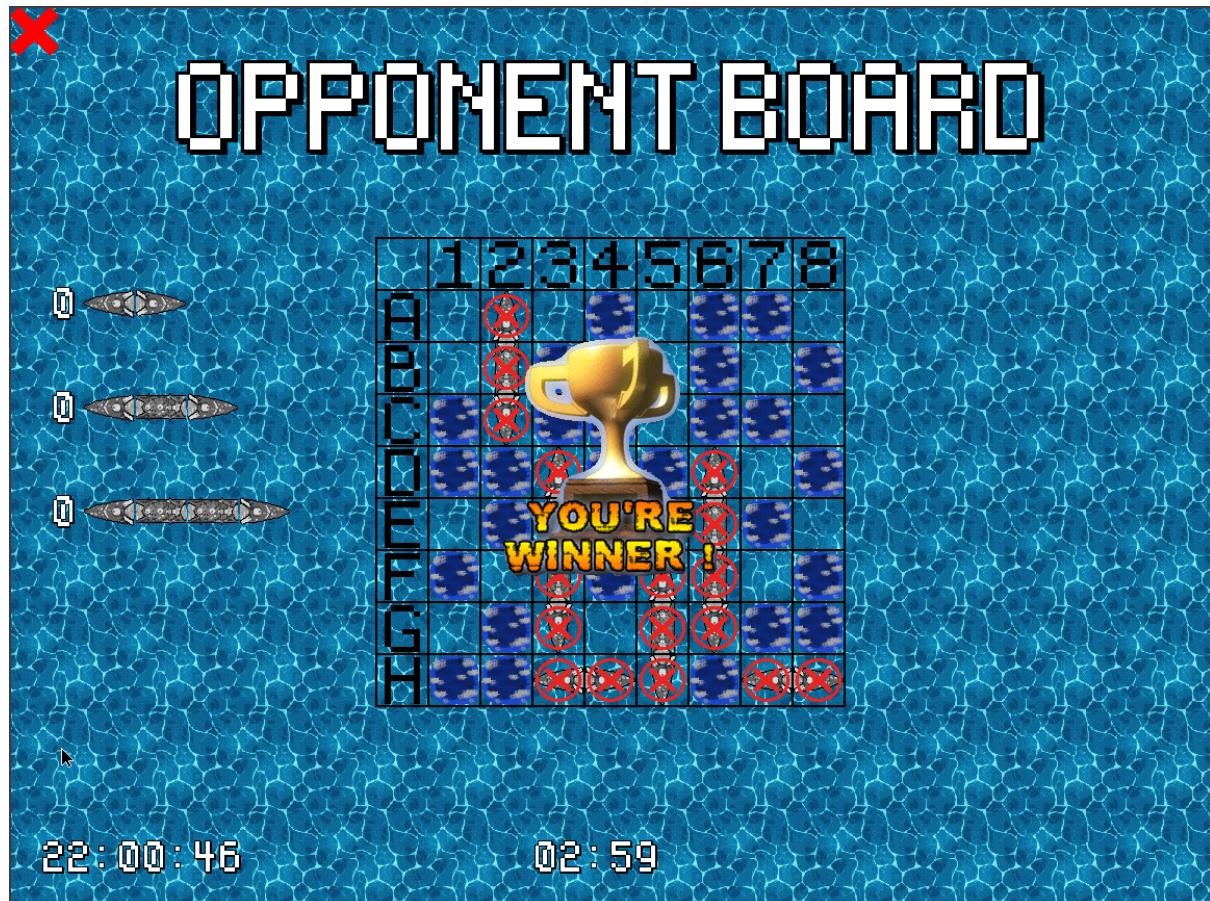
Durante a fase de defesa, o utilizador é apresentado com a sua própria board enquanto o oponente decide qual tile atacar.

Esta fase é bastante simples, o utilizador não tem que fazer nada, só espera que o oponente ataque uma tile. Depois de atacada, os resultados são semelhantes aos 3 diferentes resultados da fase de ataque

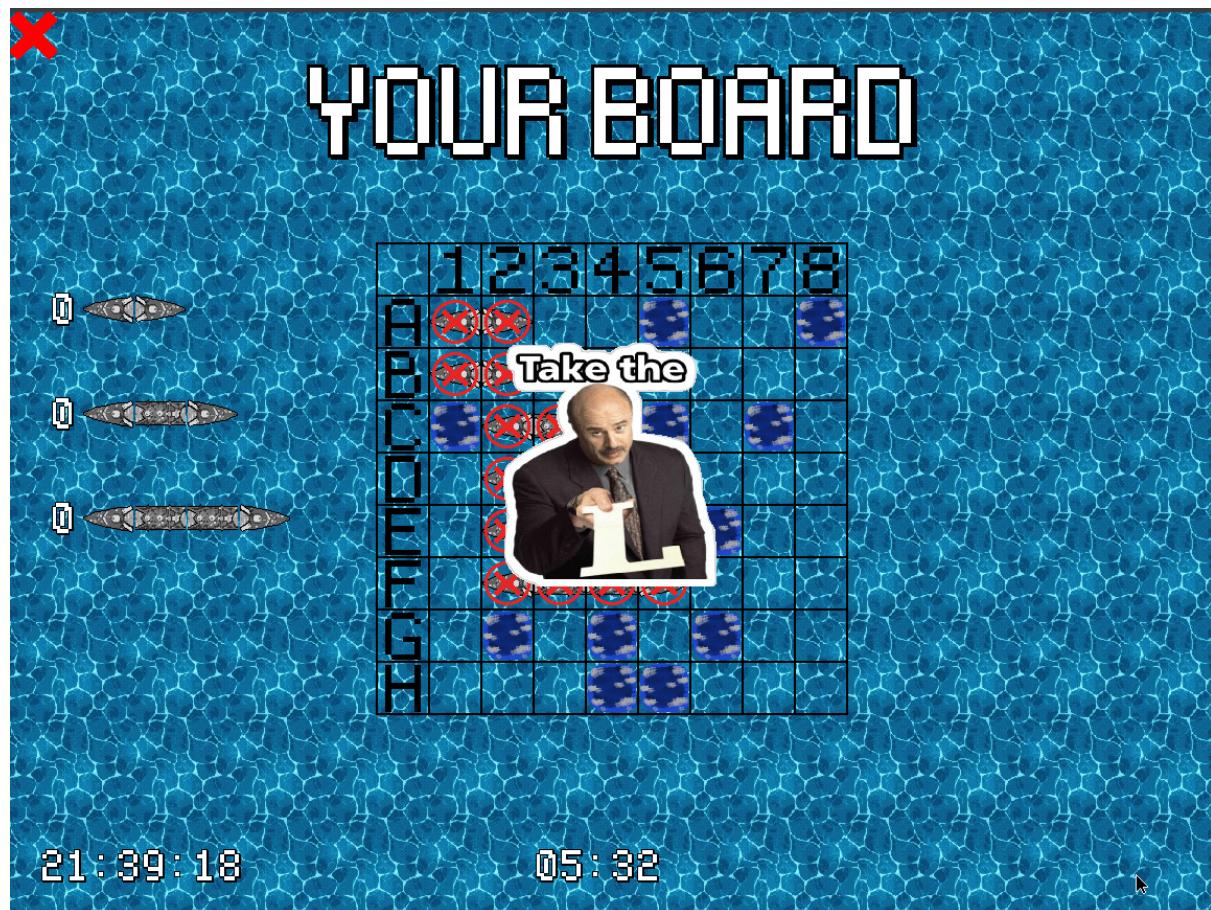
1.3. Fim de partida

Cada partida tem 2 finais diferentes, ou o utilizador ganha, ou perde. Em cada caso, é mostrado uma imagem diferente na tela:

Caso o utilizador ganhe:



Caso o utilizador perca:



2. Project Status

De seguida está apresentada uma tabela representativa dos diferentes I/O devices usados para realizar este projeto.

DEVICE	USO	INT VS POLLING
Timer	Controlar a frame rate e o delay dos ataques do AI	Interrupts
Keyboard	Selecionar as opções do menu/board	Interrupts
Mouse	Selecionar as opções do menu/board	Interrupts
Video card	Display do conteúdo visual	N/A
RTC	Mostrar a hora, minuto e segundos atuais	Interrupts

2.1. Timer

O timer foi utilizado para:

- Controlar a frame rate, 60FPS neste caso;
- Controlar o delay que o AI tem entre cada escolha de ataque.

A implementação das funcionalidades do timer estão no `timer.c` na pasta `controller/timer`.

2.2. Keyboard

O keyboard foi utilizado para:

- Selecionar as opções do menu;
- Selecionar o navio para colocar;
- Mudar a orientação do navio;
- Selecionar uma **tile** na board;
- Alternar entre a board do utilizador e do oponente.

A implementação das funcionalidades do keyboard estão no **KBC_Keyboard.c** e **keyboard.c** ambos situados na pasta [controller/keyboard](#).

2.3. Mouse

O mouse foi utilizado para:

- Selecionar as opções do menu;
- Selecionar o navio para colocar;
- Mudar a orientação do navio;
- Selecionar uma **tile** na board;
- Alternar entre a board do utilizador e do oponente.

A implementação das funcionalidades do mouse estão no **KBC_Mouse.c** e **mouse.c** ambos situados na pasta [controller/mouse](#).

2.4. Video Card

A Video card foi utilizada para:

- Desenhar na tela as xpms previamente carregadas;

Neste projeto, usamos o modo **0x14C**, com resolução 1152x864, e **Direct Color Mode** com **32 bits por pixel**, ou seja (8):8:8:8.

A implementação das funcionalidade da video card estão no **graphic.c**, situada na pasta [controller/graphic](#).

2.5. RTC

O RTC foi utilizado para:

- Mostrar ao utilizador a hora atual, atualizada ao segundo.

Para tal, foi necessário gerar interrupções periódicas, uma por segundo.

A implementação das funcionalidades do RTC estão no **rtc.c** na pasta [controller rtc](#).

A função que desenha o relógio na tela é a função **draw_rtc()** situada no **draw.c**.

3. Code organization/structure

3.1. I/O Devices

3.1.1. controller/timer

- **Descrição:** Este módulo contém as funções desenvolvidas no Lab2. As funções permitem ativar as interrupções e configurar o dispositivo.
- **Peso relativo:** 5%

3.1.2. controller/keyboard

- **Descrição:** Este módulo contém as funções desenvolvidas no Lab3, com algumas alterações. As funções permitem ativar as interrupções e processar o input do utilizador para depois realizar uma certa ação.
- **Peso relativo:** 7%

3.1.3. controller/mouse

- **Descrição:** Este módulo contém as funções desenvolvidas no Lab4, com algumas alterações. As funções permitem ativar as interrupções, sincronizar os byte de um packet do mouse e, depois de completo, atualizar as informações do mouse.
- **Peso relativo:** 7%

3.1.4. controller/graphic

- **Descrição:** Este módulo contém as funções desenvolvidas no Lab5. As funções permitem dar set ao modo gráfico, inicializar os frame buffers e desenhar xpm's na tela.
- **Peso relativo:** 10%

3.1.5. controller/rtc

- **Descrição:** Este módulo contém as funções necessárias para ser lida e interpretada a informação guardada no Real Time Clock. Funções essas que permitem ativar as interrupções do dispositivo, ler a informação do RTC, esperar caso o RTC esteja a atualizar os dados e conversão de BCD para decimal.
- **Peso relativo:** 2%

3.2. Main

- **Descrição:** Este módulo consiste na função `proj_main_loop()`, que chama as outras funções desenvolvidas encarregadas de dar setup a diversas opções necessárias para o bom corrimento do jogo. Além disso, possui também a função de terminar o programa.
- **Peso relativo:** 5%

3.3. Draw

- **Descrição:** Este módulo é o responsável por desenhar tudo que aparece na tela, desde o relógio, aos navios, à board. Além disso, também é responsável de dar sets a todos os frame buffets utilizados e também alternar entre os buffers.
- **Peso relativo:** 10%

3.4. Model

3.4.1. model/model.c

- **Descrição:** Este módulo representa a maior parte do código escrito. É responsável pela game logic toda. Tem não só a "state machine ", que alterna entre os diferentes estágios do programa, como também as funções que decidem que tiles serão atacadas pelo AI.
- **Peso relativo:** 50%

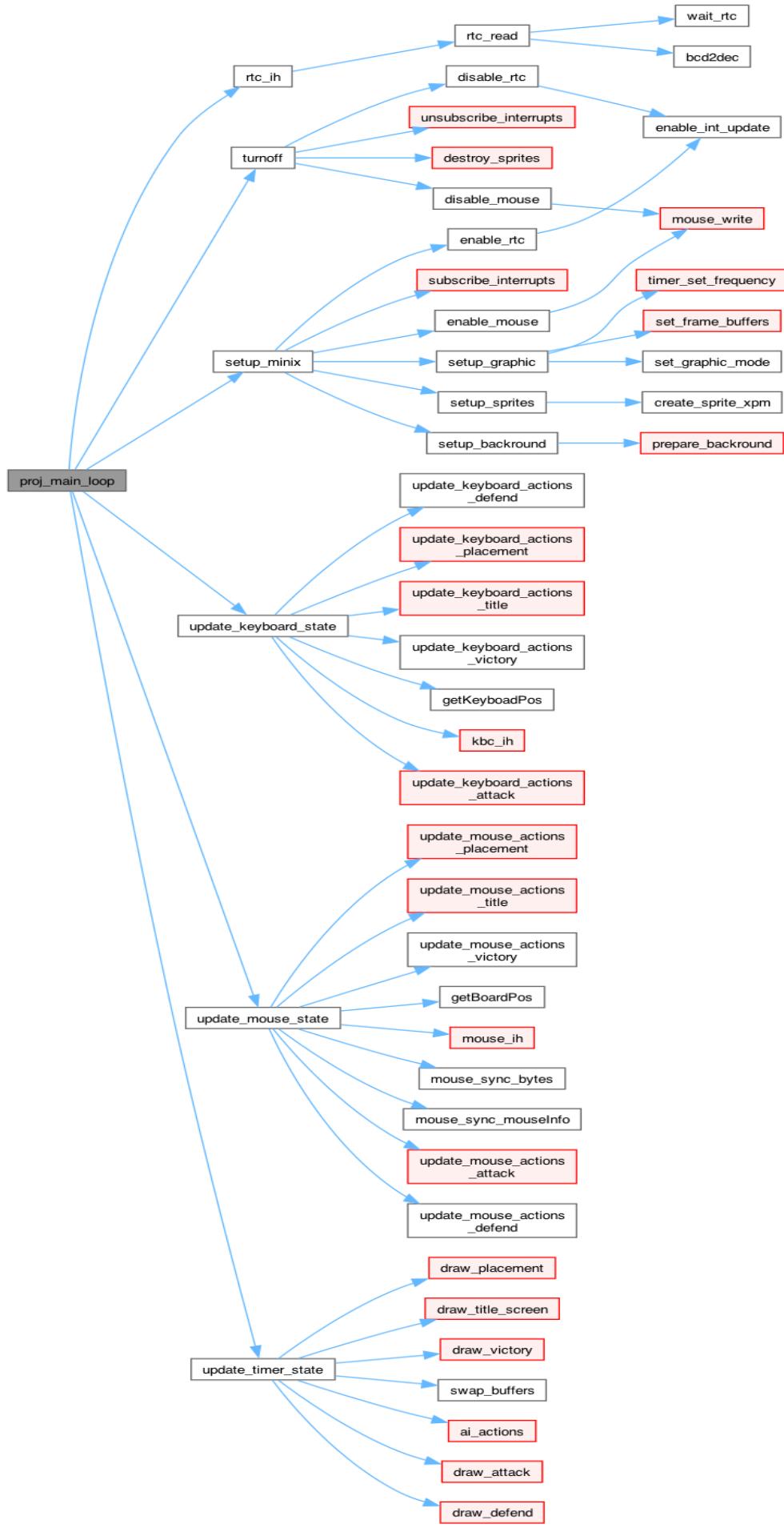
3.4.2. model/sprite.c

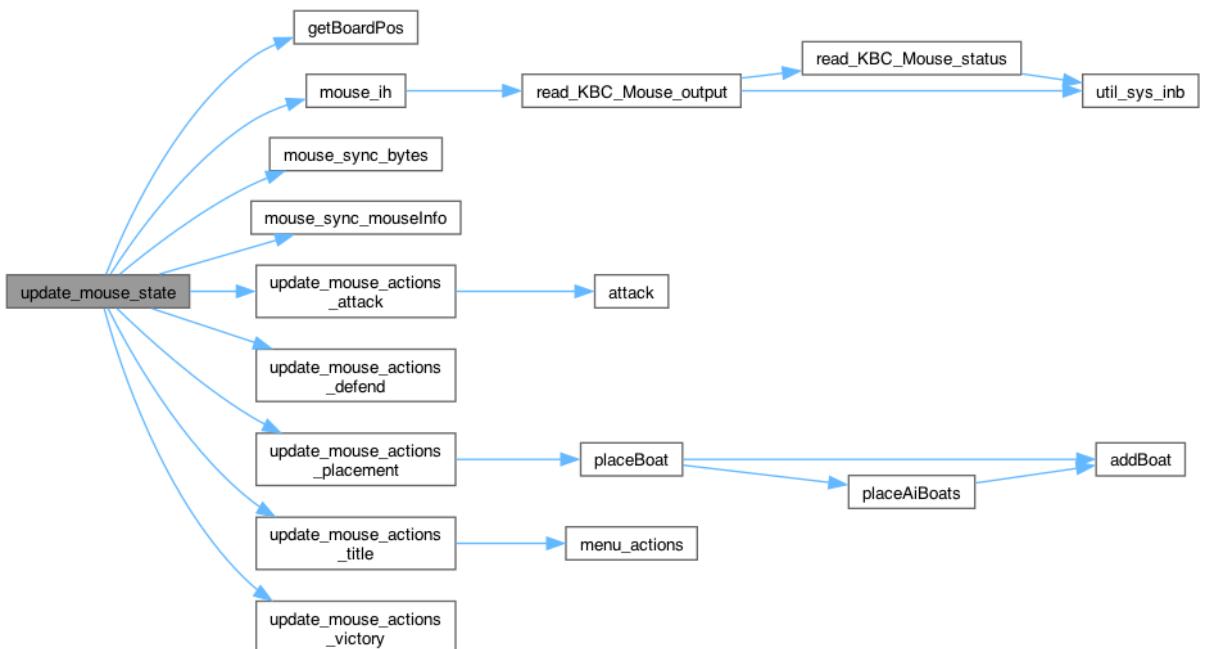
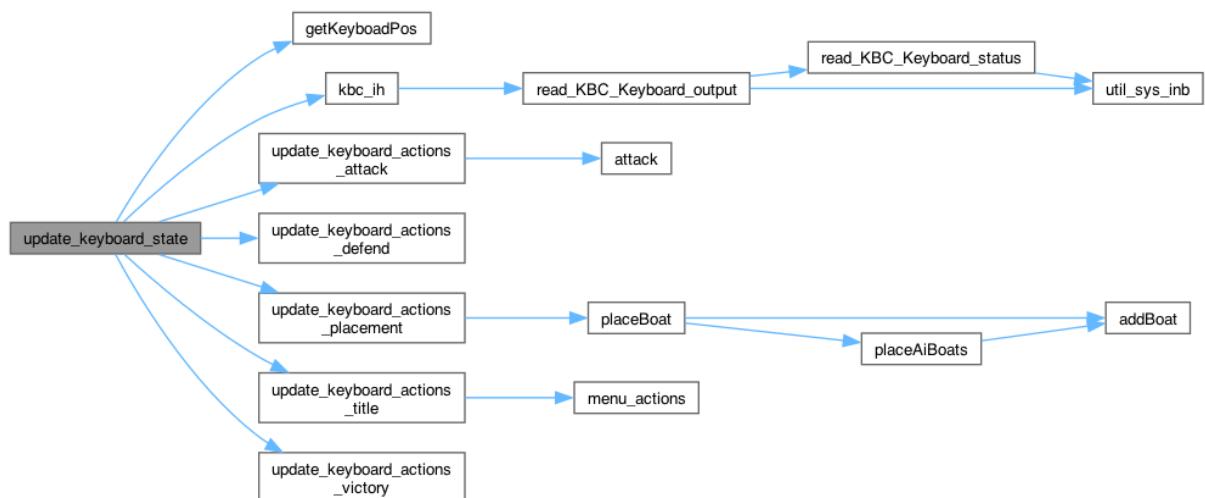
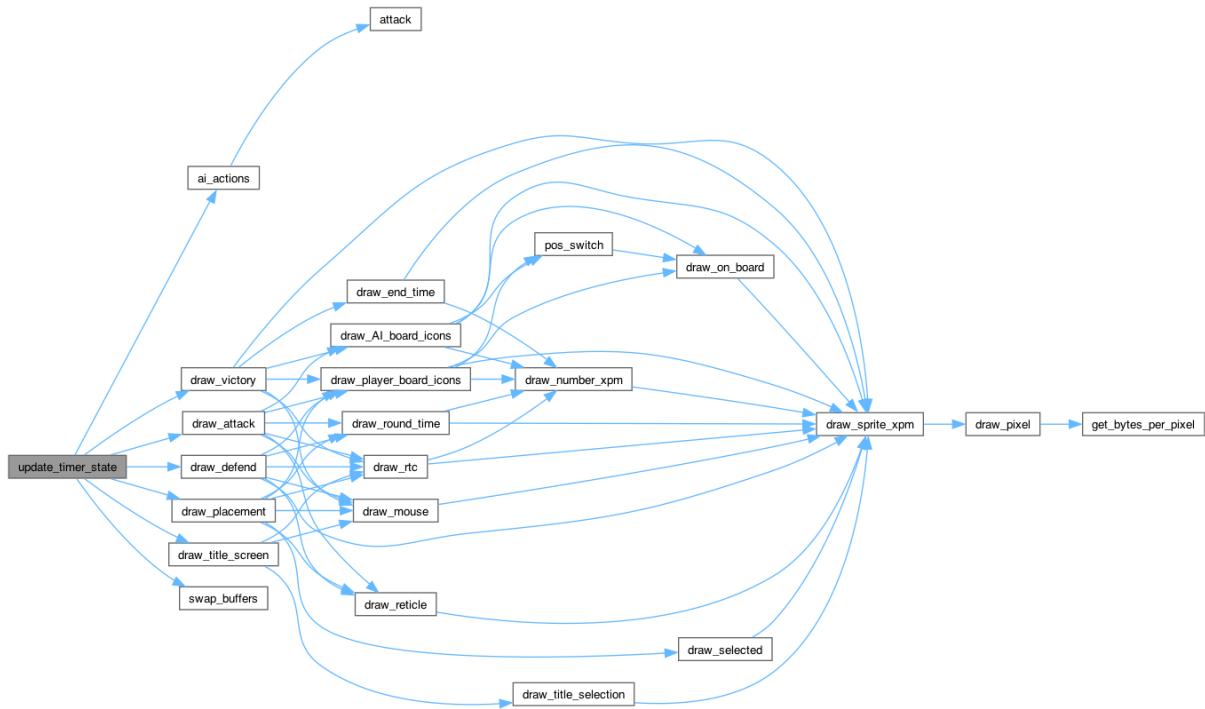
- **Descrição:** Este módulo é responsável por criar(dar load a xpm e converter para sprite) e destruir cada sprite (libertar o espaço dedicado a essa sprite).
- **Peso relativo:** 3%

3.5. Utils

- **Descrição:** Este módulo contém as funções desenvolvidas no Lab2. São funções pouco utilizadas, mas úteis de qualquer forma.
- **Peso relativo:** 1%

3.6. Function Call Graphs





4. Implementation details

4.1. Drawing buffers

Uma ideia que tivemos e da qual estamos particularmente orgulhosos é ter 2 buffers que representam o background da title screen e o background do jogo. No início do jogo, desenhamos nesses buffers os sprites, que são estáticos, e não precisam realmente ser alterados, como o próprio plano de fundo, o título do jogo e as opções de iniciar/sair, ou o tabuleiro do jogo e os ícones dos barcos ao lado do tabuleiro. Dessa forma, economizamos um pouco de desempenho ao evitar percorrer vários pixels nos sprites (alguns deles tão grandes quanto a tela) e, em vez disso, apenas copiamos esses buffers de background para o buffer de desenho. Outra vantagem disso é que também não precisamos limpar o buffer de desenho, pois o ato de copiar a imagem de plano de fundo sobre ele garante que os pixels remanescentes sejam limpos.

4.2. Round timer

Uma das últimas features que adicionamos foi o temporizador de cada ronda. Ao iniciar uma nova partida, um novo temporizador aparece na parte de baixo da tela, à direita do relógio do RTC. A ideia inicial foi simplesmente calcular o tempo decorrido através de interrupções do timer 0, mas acabamos por implementar de forma diferente e mais eficiente em termos de performance. Tendo em conta que o nosso programa já tem um RTC funcional, que, a cada segundo, aumenta um segundo à hora atual, decidimos utilizar isso para determinar o temporizador. Quando o utilizador clica em start, guardamos numa variável global chamada `initialTime` os minutos e segundos do momento em que a partida começou. Através disso, a cada segundo, é chamada a função `draw_round_time()` que calcula a diferença do tempo atual e do tempo inicial e imprime na tela.

PS: Achamos desnecessário a adição de horas ao temporizador, pois em termos de implementação é exatamente igual aos minutos e segundos e, não só achamos que fica feio, como também achamos que ninguém fica mais de 1 hora a jogar uma partida de batalha naval. No caso em que o tempo passa de uma hora, os minutos e segundos voltam a contar do 0.

4.3. AI

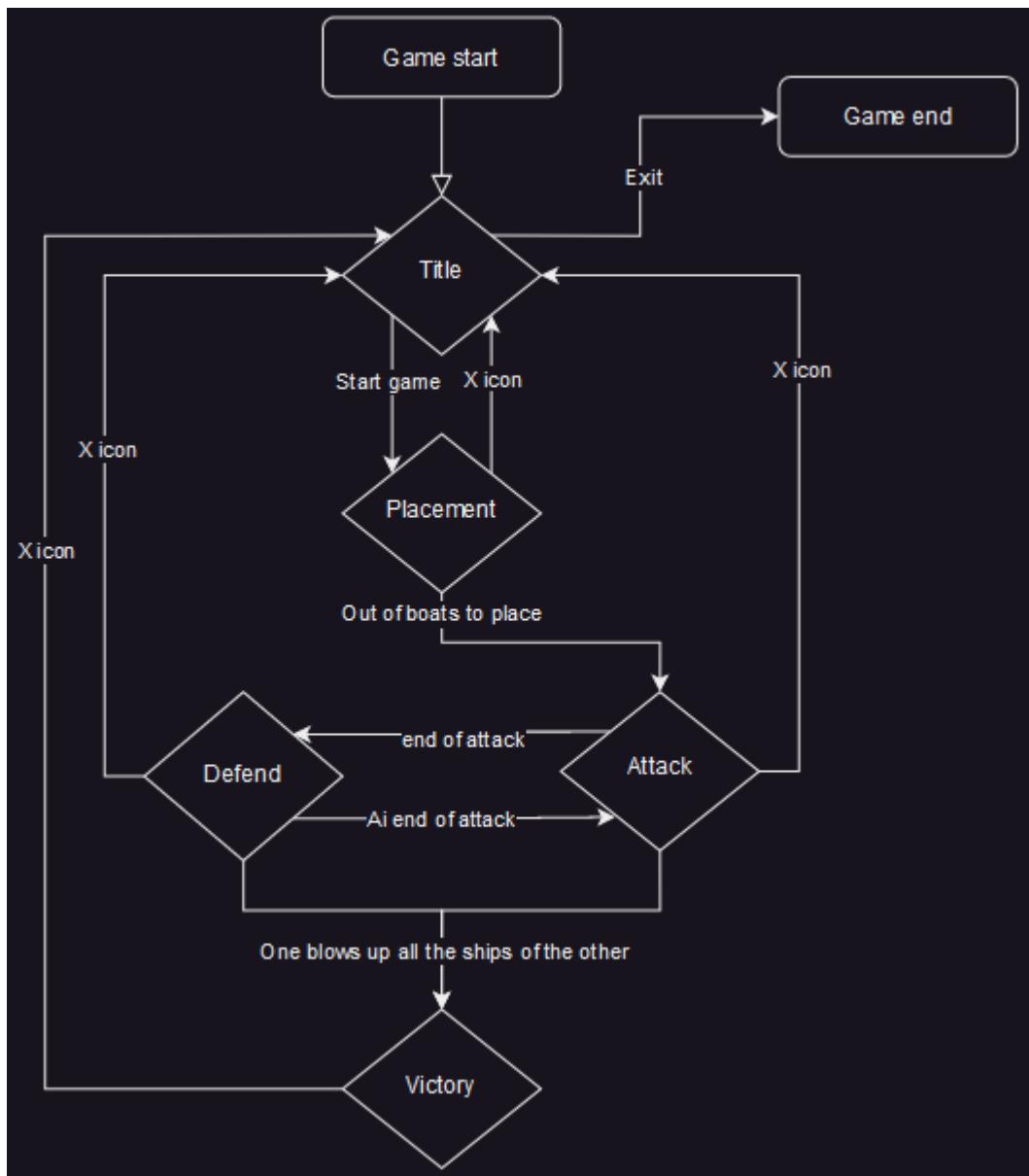
Inicialmente, o AI atacava de forma bastante simples, era gerado um número aleatório de 1 a 3, para determinar o número de vezes que o AI “muda de ideia” sobre qual tile atacar, depois era escolhido um sítio aleatório que ainda não tivesse sido atacado. No entanto, depois de muitos testes, chegamos à conclusão que era quase impossível perder para o AI, portanto decidimos refazer isso. Agora, o AI ataca aleatoriamente até acertar numa tile com navio. Após acertar num navio, a tile é guardada numa variável e depois é escolhido aleatoriamente um número de 1 a 4 que determinará que tile atacar depois, 1 para atacar a de cima, 2 para a de baixo, 3 para a da esquerda e 4 para a da direita. Depois, ele verifica se essa tile já foi atacada, se sim, então ataca aleatoriamente outra tile, senão então ataca a tile escolhida. Desta forma, o jogo torna-se um pouco mais difícil.

4.4. State machine

Uma das partes mais essenciais do flow do nosso código é a utilização de uma state machine. O nosso programa tem 5 estados: Title, Placement, Defend, Attack, Victory.

Title representa quando vemos o titleScreen. Placement é quando o jogo se encontra na fase de colocar os barcos na board. Defend é quando o AI ataca o player. Attack é quando o utilizador ataca o AI. Victory é quando um dos lados vence.

A transição de Title para Placement ocorre quando o player seleciona a opção start. De Placement para attack o utilizador simplesmente necessita de posicionar todas os barcos que tem disponivel. De Attack para Defend o Utilizador tem de tentar atacar uma zona do board do AI que não tem um barco. De Defend para Attack é o mesmo mas com utilizador e AI invertido. De Attack e Defend para Victory um dos lados tem de ganhar. De qualquer um deles para o Title tem de clicar no icon x ou clicar no esc no teclado. Para fechar o jogo faz-se o mesmo mas do estado Title.



5. Conclusions

Em conclusão, embora o tempo tenha sido pouco para desenvolver este projeto, achamos que conseguimos alcançar um patamar aceitável e consideramos o jogo completo.

Infelizmente, devido a problemas com o tempo que tivemos para realizar o projeto, devido a outros projetos de igual importância com data de entrega próxima deste, não conseguimos implementar tudo que desejávamos.

Uma das razões para termos escolhido **Battleships** como tema do projeto foi porque era um jogo que facilmente englobava todos os I/O devices listados como parte da cadeira, mas, no entanto, por razões já referidas, não tivemos tempo para implementar o serial port, ou seja, um jogo que, no início, tínhamos como ideia criar um modo para 2 jogadores acabou por ser single-player contra um AI.

Não só o serial port, como também tivemos que desistir de adicionar animações, nomeadamente para quando fosse atacado uma tile e fosse destruído um navio. De novo, problemas com o tempo que tivemos para realizar o projeto.

Consideramos que, com estas adições que ficaram por fazer, o nosso projeto poderia ter atingido um patamar não de aceitabilidade, mas de excelência.

Além disso, achamos que, para o peso que têm em conjunto (15%), o RTC e o serial port deveriam ter sido dados mais importância. Foram falados nas aulas teóricas, mas não foram dados exercícios práticos tal como os outros devices. Devido a isto, muita gente optou por não incluir estes devices no seu projeto. No nosso caso, conseguimos implementar o RTC puramente graças aos monitores.