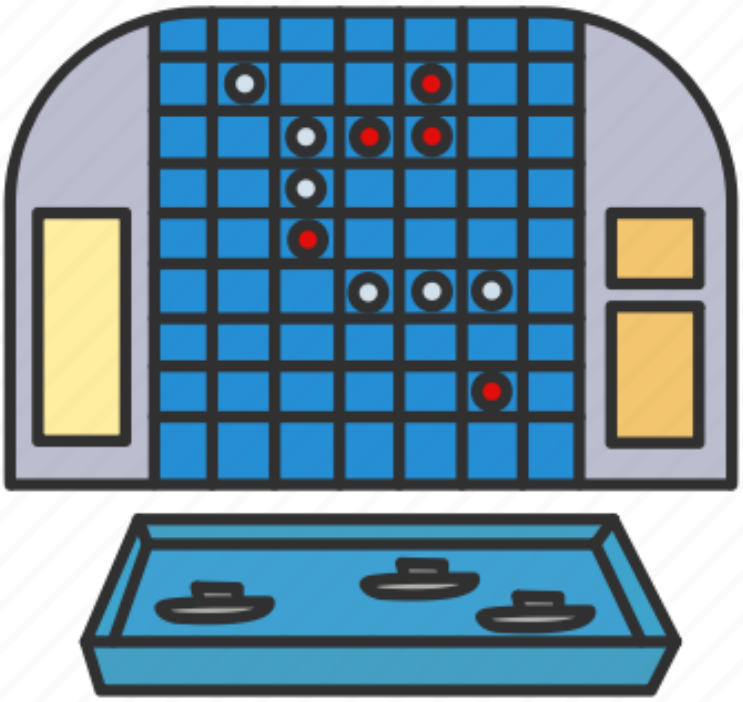
U.PORTO

FEUP FACULDADE DE ENGENHARIA
UNIVERSIDADE DO PORTO



Batalha Naval

RELATÓRIO DE PROJETO

Ivo Saavedra
Rodrigo Reis
| LCOM | 06/01/2020

1 – Instruções de Utilização

1.1 - MENU

Figura 1 - Menu

No início do jogo é mostrado o menu onde se pode selecionar, com o rato, uma das 3 opções:

Single Player – Inicia o modo de jogo *single player*. O jogador é direcionado para o modo de inicialização do tabuleiro (fig. 2), uma matriz aleatória para a AI é gerada, posteriormente.

Multiplayer – Inicia o modo de jogo *multiplayer*. Ambos os jogadores são direcionados para o modo de inicialização do tabuleiro (fig. 2);

Exit – Sai do jogo.

1.2– INICIAR O TABULEIRO

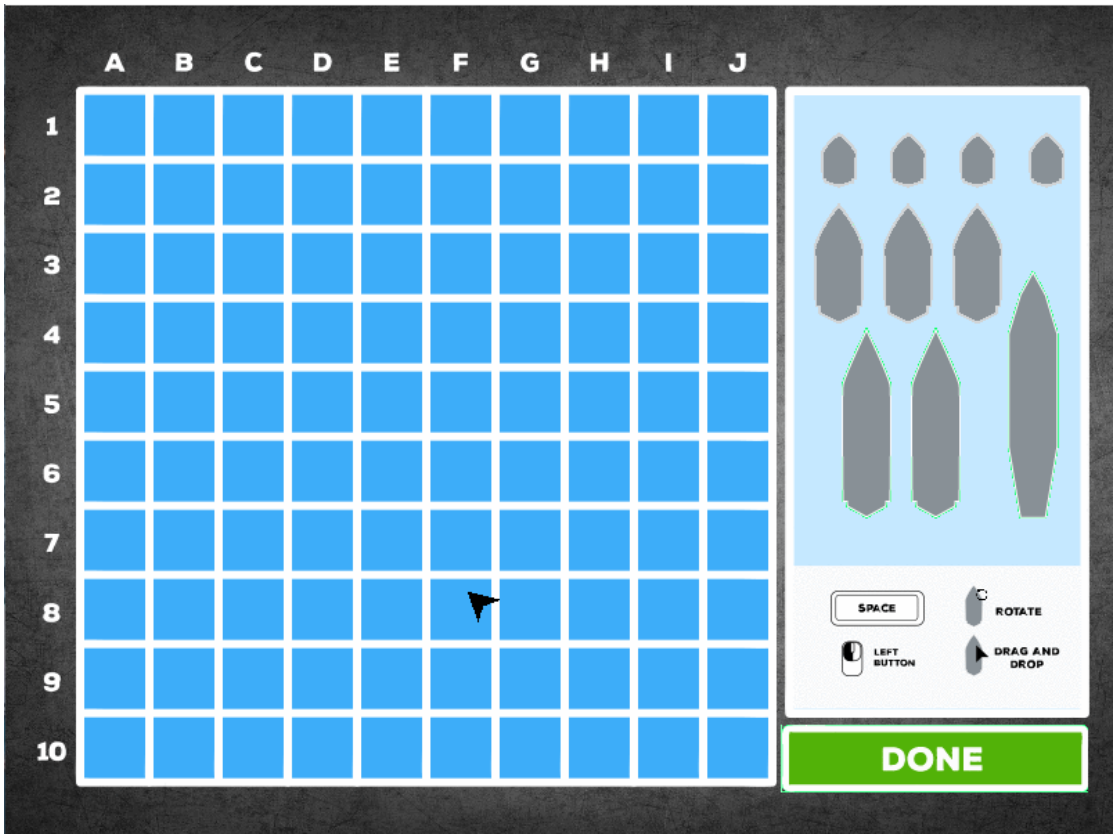


Figura 2 - Modo de Inicialização do Tabuleiro

Neste modo o jogador deve posicionar os dez barcos na matriz e no fim deve carregar no botão "*DONE*".

Posicionamento na matriz

Para posicionar os barcos basta carregar no barco, arrastá-lo para a matriz e largá-lo. Se a posição não for válida o barco retoma a posição inicial.

Para rodar o barco, é necessário selecionar o barco e carregar no espaço (enquanto o barco estiver selecionado).

Se pretender remover um dos barcos já posicionados na matriz, basta arrastá-lo para fora da matriz.

Importante: Se algum dos barcos não foi colocado na matriz não será possível sair deste modo.

1.3 – MODO DE JOGO

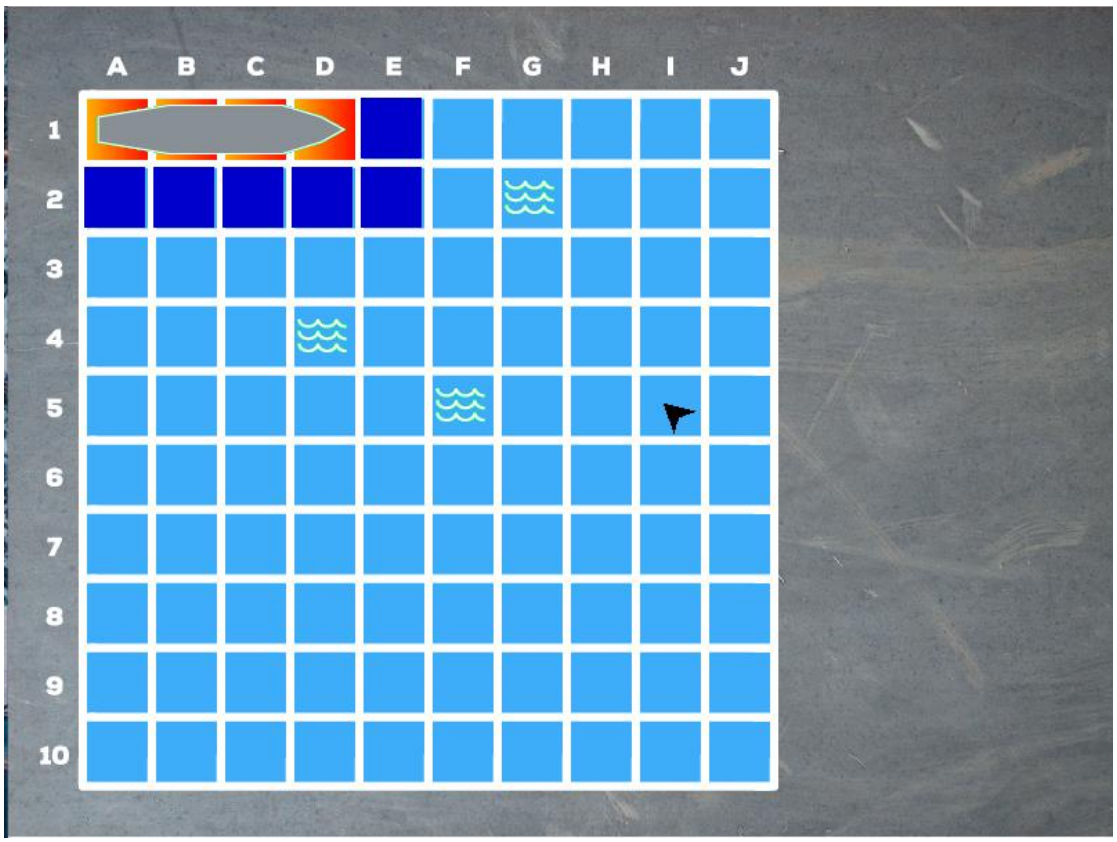


Figura 3 - Modo de Disparo

O jogo é baseado em turnos. Em cada turno o jogador pode efetuar apenas um "disparo" sobre o tabuleiro do oponente. No fim de um turno o jogador irá ver o seu tabuleiro (com os barcos posicionados) e a célula em que o inimigo disparou. Este ciclo é repetido até que todos os barcos de um dos jogadores tenham sido destruídos.

Neste modo o jogador deve "disparar" para o tabuleiro do oponente, carregando numa das células da matriz (fig. 3). Cada jogador tem direito a um disparo por turno.

Após um disparo, a célula atingida muda de estado de acordo com a disposição dos barcos do inimigo. Uma célula pode ter os seguintes estados:

1) **Água (ondas) -** quando a célula atingida não contém nenhum barco;
2) **Fogo (vermelho) -** quando a célula atingida contiver um barco;
3) **Protegido (azul escuro) –** quando um dos barcos for destruído as células adjacentes ficarão protegidas, não permitindo futuros disparos.

2 – Estado do Projeto

2.1 – Funcionalidades Implementadas

Dispositivo	Função	Int.
Timer	Atualizar os frames e cronometragem	S
KBD	Posicionamento dos barcos e navegação	S
Mouse	Seleção de objetos e navegação	S
Video Card	Representação gráfica do menu e dos vários elementos do jogo	N
RTC	Mostrar a hora no menu	N
Serial Port	Modo de jogo multiplayer	N

2.2 – Descrição dos Módulos implementados

No plano propusemos os uso da serial port para o modo de jogo multiplayer, mas acabamos por não conseguir implementar. Foi implementada uma versão mais simples em que ambos os jogadores têm que jogar no mesmo computador.

Após a subscrição dos dispositivos necessários é chamada a função **int interrupt_cicle()** onde é inicializada a struct do tipo **interrupt_data** com a informação das respetivas interrupções. Esta informação é, posteriormente, usada nas funções que necessitam destes dispositivos.

Timer

Uso das interrupções do timer, para atualizar os frames nas funções:

- **int game_init_board()**
- **int game_round_sp()**
- **int menu()**

Uso das interrupções do timer, para cronometrar tempo através da função **int idle_time()** :

- **int game_sp()**
- **int game_round_sp()**
- **int game_round_ai()**

Keyboard

Usado apenas para o controlo do jogo, por interrupções, nas funções:

- **int game_init_board()**
- **int game_round_sp()**
- **int game_round_mp()**

Video Card

A gráfica foi iniciada no modo 0x115, com uma resolução de 800x600 e modo de cor direto, sendo que cada pixel possui 24 bits. É usado double buffering, com exceção na função **int game_round_sp(),** onde se recorre ao uso de um buffer auxiliar, para melhorar o desempenho na atualização dos frames.

Em relação aos objetos, recorreu-se ao uso de Sprites, para representar o cursor e os barcos. A colisão entre os objetos ocorre quando o ponto de colisão de um objeto se encontra numa posição interna de outro objeto. Para arrastar os objetos, é utilizada a função **int check_sprite_collision()** que verifica se o ponto de colisão do cursor está no interior do Sprite que se pretende arrastar; se uma colisão tiver ocorrido a posição do Sprite é incrementada com o deslocamento do rato, provocando assim o "movimento" do Sprite.

A gráfica é usada para a representação gráfica da interface nas funções:

- **int menu()**
- **int game_init_board()**
- **int game_round_sp()**
- **int game_round_ai()**

Mouse

O rato foi subscrito por interrupções e usado em todas as funções que envolvessem um cursor.

Recorreu-se ao uso da posição do rato para "movimentar" o cursor no ecrã e para arrastar objetos aumentando a sua posição com o deslocamento do rato. O único botão usado, foi o botão esquerdo, para selecionar objetos, para disparar e para navegação.

RTC

Utiliza-se as interrupções do timer para verificar se houve alguma alteração na hora. Se houver, os valores das horas e dos minutos são atualizados.

Este módulo é utilizado no menu para mostrar as horas do sistema.

3 – Organização do Código

timer.c

Contém as funções relativas à subscrição das interrupções do timer e o interrupt handler do timer.

kbd.c

Contém as funções relativas à subscrição das interrupções do teclado, o interrupt handler, a função que retorna o scan_code lido, para facilitar a comunicação com os outros módulos.

mouse.c

Contém as funções necessárias para a subscrição das interrupções do rato, o interrupt handler, usado para inicializar a **struct packet pp**, para subsequente uso noutros módulos.

rtc.c

Contém as funções responsáveis pela leitura das horas atuais.

graphic.c

Contém a função que aloca o espaço necessário para a *video memory* e inicia o modo gráfico e as funções responsáveis da representação de imagens no ecrã.

interrupt_cicle.c

Ficheiro com o ciclo de interrupções, que trata das interrupções enviadas pelos dispositivos.

sprite.c

Contém as funções fornecidas para criar e destruir objetos do tipo *Sprite*; as funções responsáveis da representação dos *Sprites* no ecrã e também a função que verifica a colisão entre sprites **int check_sprite_collision()**.

proj.c

Contém o proj_main_loop que retorna o menu.

menu.c

Contém a função que representa graficamente o menu e permite ao utilizador selecionar o modo que pretende.

game.c

Inclui as funções referentes aos modos de jogo *single player e multiplayer*; a função de inicialização do tabuleiro - **int game_init_board();** a que simula um disparo da AI (computador) para a matriz do *player* e a que efetua um disparo do jogador contra o oponente (computador ou jogador).

state_machine.c

Ficheiro com a máquina de estado responsável pelo posicionamento dos barcos na matriz - **int select_boat_sp() -** e a função que simula uma estratégia para a AI em modo de jogo single player – **int game_round_ai_sm()**.

player.c

Módulo da classe **Player**, com o construtor, a função que valida e diferencia o tipo de disparo do jogador – **int player_shot()** – e a função que gera uma matriz aleatória – **int generate_random_matrix()**.

boat.c

Ficheiro da classe **Boat**, com o construtor e a função que inverte a orientação do barco – **void rotate_boat()**.

board.c

Módulo da classe **Board**. Contém o construtor, as funções que manipulam a matriz do jogador de acordo com as ações efetuadas durante o jogo, e , ainda, as funções que validam o posicionamento dos barcos ou as seleções no tabuleiro, através da colisão entre Sprites, ou coordenadas.

Funções da internet

No módulo **rtc.c** recorremos a uma função para converter os valores das horas de binário para decimal - **uint32_t BCDtoDecimal()**. Foi alterado o tipo de valor de retorno e o tipo de valor recebido por argumento.

url: http://www.mbeddedc.com/2017/03/decimal-to-binary-coded-decimal-bcd.html

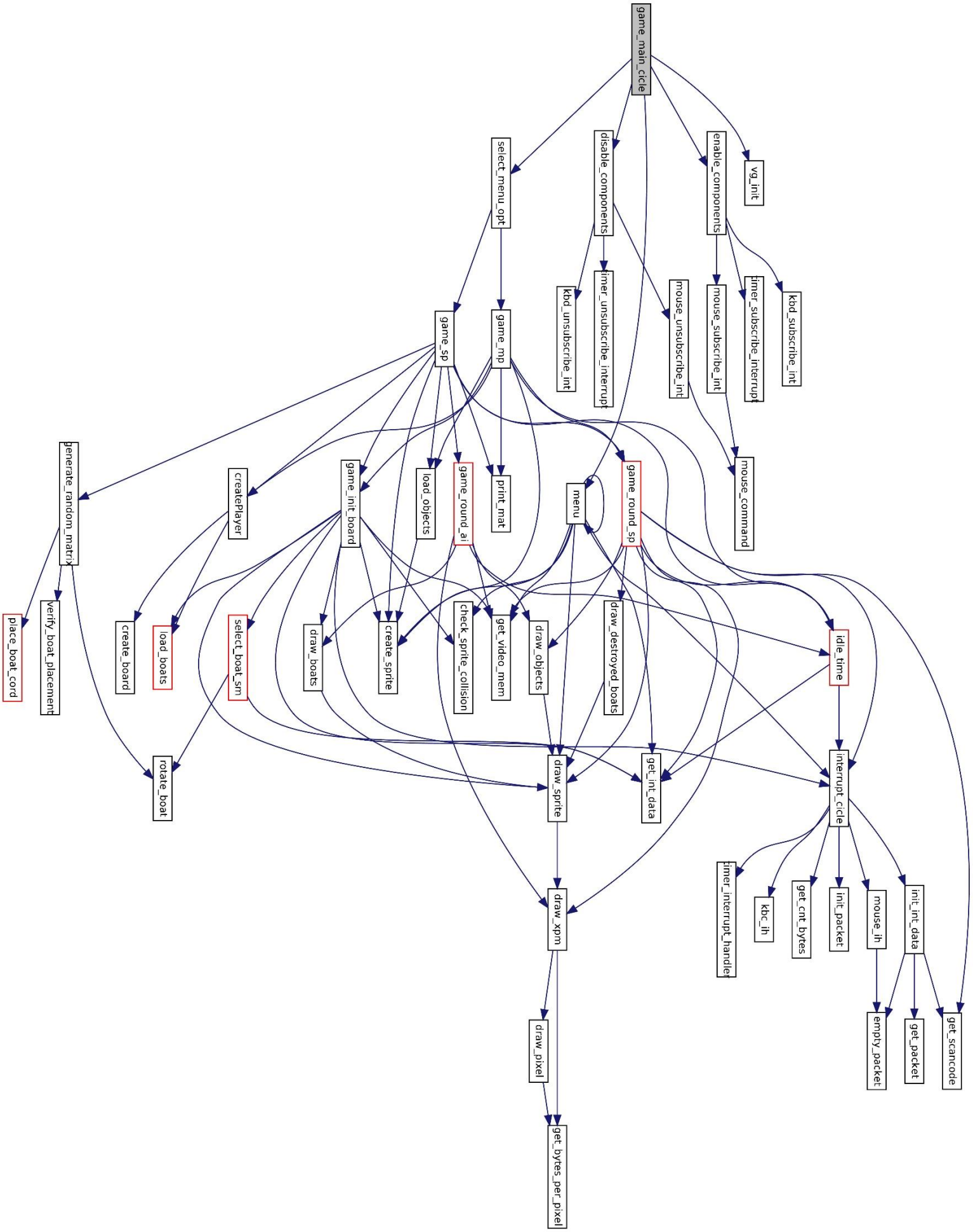


Figura 4 - Gráfico de chamada de funções

4 – Detalhes da Implementação

4.1 – Detalhes Gerais

O projeto começa por chamar a função **int game_main_cicle()**, onde são iniciados todos os dispositivos e onde se entra no modo gráfico. De seguida é chamado o menu, onde o jogador pode escolher entre uma das três opções (single player, multiplayer, exit). O utilizador é depois direcionado para o modo escolhido. No final, o menu é chamado novamente. Se o botão *exit* for premido, o ciclo acaba e os dispositivos são desativados. Sai do jogo.

Ao longo do desenvolvimento do projeto teve-se em atenção a modularidade do código, de modo a torná-lo mais legível e organizado. Todos os dispositivos foram desenvolvidos em módulos diferentes para facilitar a comunicação entre as funções relativas a esses ficheiros e para reduzir a quantidade de variáveis globais de projeto.

À semelhança dos dispositivos, as classes dos objetos também foram implementadas em ficheiros diferentes, pelas mesmas razões.

Na ocorrência de uma ou mais interrupções (no ciclo de interrupções), a informação é tratada nos ficheiros dos respetivos módulos e os dados necessários são guardados na struct **interrupt_data d**. Esta estrutura pode ser acedida por outros módulos através da função **get_int_data()**.

Nas funções que requerem movimento de objetos os frames foram atualizados à mesma frequência que as interrupções do timer (60 vezes/segundo). A informação proveniente das interrupções do rato foi utilizada para a movimentação dos objetos, incrementando a posição atual dos objetos com o deslocamento do rato.

4.2 – Máquinas de Estado

A máquina de estados **select_boat_sm** é semelhante à que foi feita no âmbito do lab4, e possui os seguintes estados:

1. INIT – se o botão esquerdo do rato estiver premido, verifica se algum dos barcos foi selecionado; se for esse o caso, avança para o próximo estado.
2. DRAG – arrasta o barco selecionado até que o botão esquerdo do rato seja largado, ou a barra de espaços seja premida. No primeiro caso, a máquina transita para *DROP*, no segundo para ROT.
3. ROT – troca a orientação atual do barco selecionado e transita para *DRAG*.
4. DROP – verifica se o posicionamento efetuado é válido, se for o barco é posicionado na matriz, senão o barco retorna para a posição inicial. Transita para INIT.

A máquina de estados **game_round_ai** executa uma estratégia semelhante à de um jogador comum no que toca aos disparos para o tabuleiro do oponente. Primeiro, é

efetuado um disparo sobre uma posição aleatória da matriz. Se essa posição contiver um barco, então a máquina de estados dispara para as células adjacentes até destruir o barco na totalidade.

Possui os seguintes estados:

1. RANDOM – estado inicial onde é gerada uma posição aleatória e verificada a validade do tiro para essa posição. Quando o tiro calhar numa célula já atingida, então a posição não é válida e deve ser gerada de novo. Se atingir uma célula de água, o tiro é registado e máquina permanece no mesmo estado para no próximo turno gerar uma nova posição. No caso de acertar num barco, o tiro é registado e máquina transita para o estado *MOV_R*.
2. MOV_R – dispara para a posição imediatamente à direita da célula atingida. Se acertar num barco continua no mesmo estado (disparar para a direita), senão transita para *MOV_L* (disparar à esquerda da célula acertada).
3. MOV_L - dispara para a posição imediatamente à esquerda da célula atingida. Se acertar num barco continua no mesmo estado (disparar para a esquerda), senão transita para *MOV_U* (disparar para cima).
4. MOV_U - dispara para a posição imediatamente acima da célula atingida. Se acertar num barco continua no mesmo estado (disparar para cima), senão transita para *MOV_D* (disparar para cima).
5. MOV_D – se o barco não está em nenhuma das direções anteriores, então só pode estar para baixo (disparar para baixo até que o barco seja destruído).

5 - Conclusão

O jogo da Batalha Naval pareceu-nos logo desde início um projeto interessante, quer pela envolvência do jogo, quer pela lógica subjacente ao mesmo, na implementação de uma Inteligência Artificial que decide as jogadas automaticamente.

Apesar de o módulo mais desafiante, não ter sido implementado com sucesso (a porta de série), foi alvo de várias tentativas da nossa parte. Assim sendo, para a criação de um modo multiplayer, desenvolvemos uma versão mais simples que utiliza apenas um computador.

O RTC foi implementado com sucesso, mostrando um relógio digital gráfico no menu de jogo. O rato é o elemento mais utilizado do jogo, permitindo toda a interação com o tabuleiro. O teclado é utilizado para rodar os barcos na sua colocação no tabuleiro. E o timer para, a cada interrupção desenhar os objetos no modo gráfico. É utilizado o modo gŕafico 0x115, 800x600px.

Em suma, este projeto foi enriquecedor na medida em que nos permitiu consolidar os conteúdos abordados nos labs ao longo do semestre. A junção de todos os módulos num projeto desvendou, sem dúvida, o interesse e a pertinência da Unidade Curricular.