FACULDADE DE ENGENHARIA DA UNIVERSIDADE DO PORTO

Batalha Naval

Implementação em Minix

Vasco Gonçalves e André Freitas

9 de Fevereiro de 2012

Índice

Introdução

O jogo da Batalha Naval é muito conhecido entre todos nós. Este é jogado por dois jogadores em que cada um, através de uma matriz, tenta afundar todos os navios do adversário. A complexidade do jogo é muito simples, usando-se coordenadas para atacar.

Ora partindo dos pressupostos do jogo é necessário propor uma implementação, que descreva esta abstração em algo concreto no *Minix* na linguagem C, no que toca às estruturas de dados e aos periféricos para os *inputs* e *outputs* do jogo. Uma visão simplista seria logo uma solução em que dois computadores comunicavam por porta série e o jogo era representado no modo gráfico em cada jogador e, os *inputs* das coordenadas vinham do teclado. Porém, há que ver até que ponto a solução é eficaz e ser plausível de ser implementada, tendo especial foco na porta série pelo sincronismo da comunicação, de poder identificar o emissor e recetor e a partir disso tomar as decisões em termos de algoritmo do jogo.

Desenvolvimento da Solução

Antes de partir para a solução em C, é necessário enumerar as entidades que pertencem a este jogo. Basicamente é necessário representar cada tipo de embarcações e associar a cada um as coordenadas das suas posições. Existem então os seguintes:

1. Porta-aviões – 5 posições;
2. Couraçado – 4 posições;
3. Submarino – 3 posições;
4. Contratorpedeiro – 3 posições;
5. Barco de patrulha – 2 posições;

Cada jogador na sua tela tem uma matriz onde regista os seus ataques, colocando azul na coordenada se calhou água ou vermelho se acertou numa embarcação.

Implementação em C

A implementação em C passa por traduzir estas entidades em estruturas de dados (*structs*), visto que o C não suporta classes, o que torna a tarefa ligeiramente mais difícil.

```
/*********************************************************
 Definition of a ship that have the type of ship,
 the health and the positions in the matrix of the game.
 *********************************************************/
/* Attributes */
struct ship {
	char type;
	int sizeShip;
	int health;
	int positions[5][3];
};

/* Functions */
int ship_build(struct ship *pShip, char type, int positions[5][3]);
int ship_hit(struct ship *pShip, int x, int y);
int ship_destroyed(struct ship *pShip);
void ship_print(struct ship *pShip);

/*********************************************************
 Definition of a game that has a player, the ships
 *********************************************************/
/* Attributes */
struct game {
	struct ship ships[5];
	char map[10][10];
	char mapOpponent[10][10];
};
/* Functions */
int game_build(struct game *pGame);
```

```
int game_attack(int x, int y);
int game_defend();
int game_isOver(struct game *pGame);
void game_printMap(struct game *pGame);
void game_printOpMap(struct game *pGame);
void game_hit(struct game *pGame, int x, int y);
```

Periféricos usados

Os periféricos usados nesta implementação foram o teclado e o modo gráfico do *Minix*, servindo, respetivamente para os *inputs* e outputs do jogo. Estava previsto a utilização da porta série, porém, devido a problema recorrentes da mesma, foi decidido descontinuar o seu desenvolvimento, apesar de termos atingindo algumas metas na utilização da mesma.

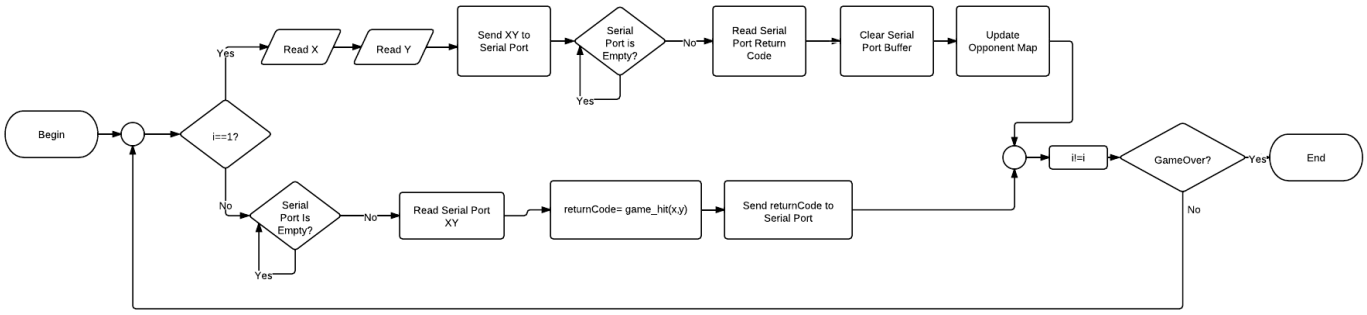


Ilustração 1 - Algoritmo de Sincronização do Jogo entre computadores

Um dos problemas com a porta série era o sincronismo sobre quando um jogador defendia e outro atacava. Na figura acima pode-se ver um fluxograma com a implementação de um algoritmo da solução do problema. Este algoritmo foi implementado mas devido às limitações da porta série ocorreram imensos problemas, pelo que poderia comprometer a conclusão do projeto.

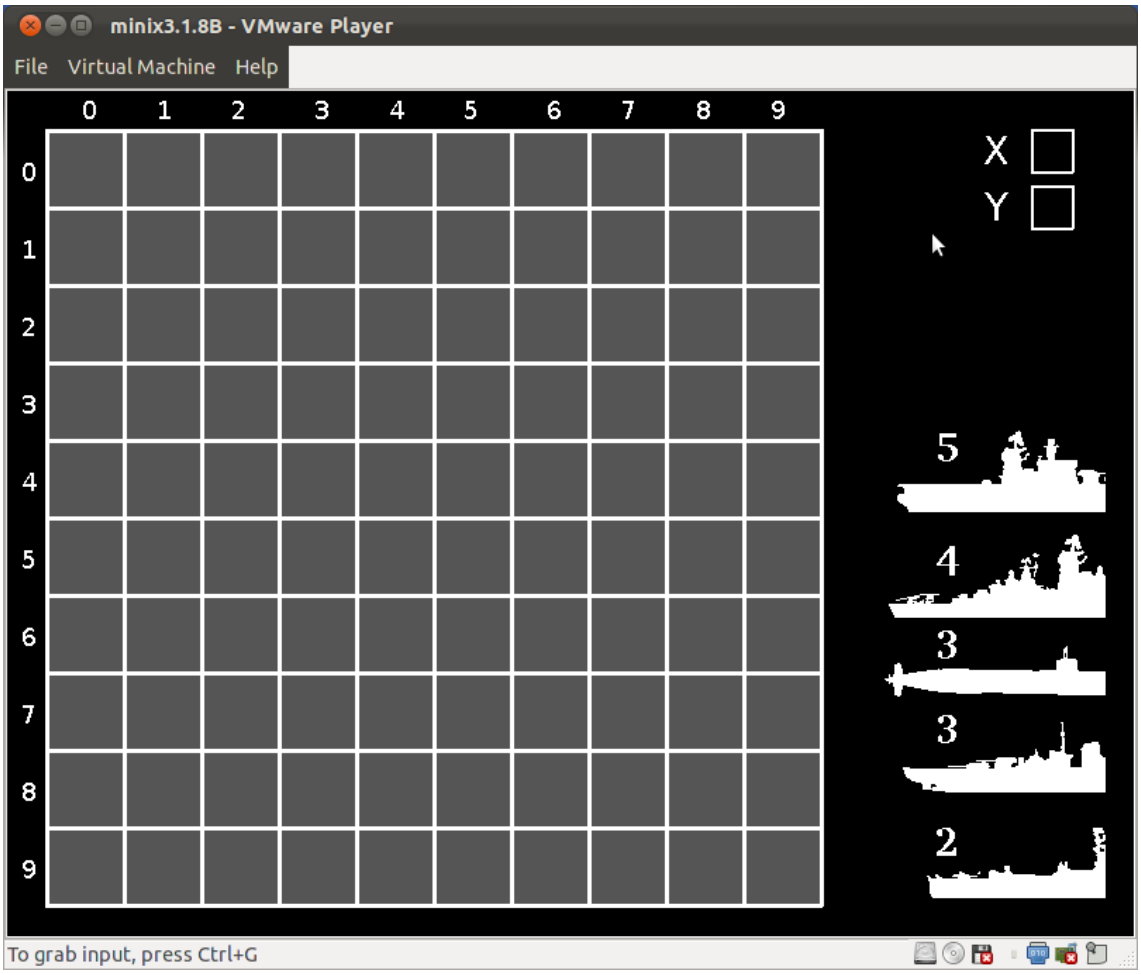
Interface do Jogo

A interface para o jogo é relativamente simples. Este começa com um menu inicial onde se escolhe para começar um novo jogo. De seguida aparece uma matriz onde a partir do teclado, se introduz as coordenadas. O jogador basicamente introduz o x e y do sítio que pretende atacar. Se acertou numa embarcação a célula fica a vermelho, porém, se acertou na água, a célula fica a azul. O jogo acaba quando o jogador tiver afundado todos os navios, aparecendo uma mensagem com o seu número de tentativas e voltando ao menu inicial. Durante o jogo, no lado direito aparece a informação de quantas células faltam afundar de cada um dos cincos navios, a fim de guiar melhor o jogador.

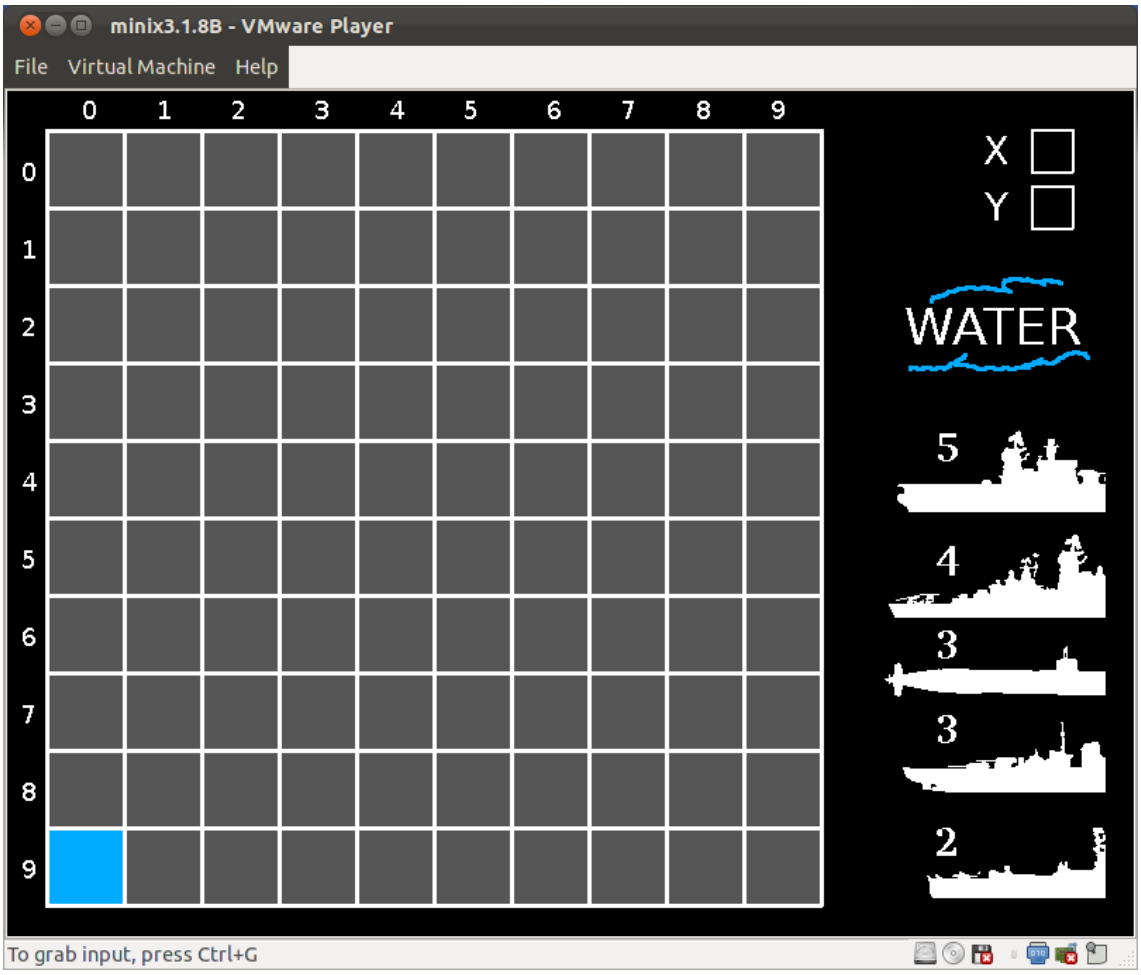
Casos de Utilização

- Menu principal:

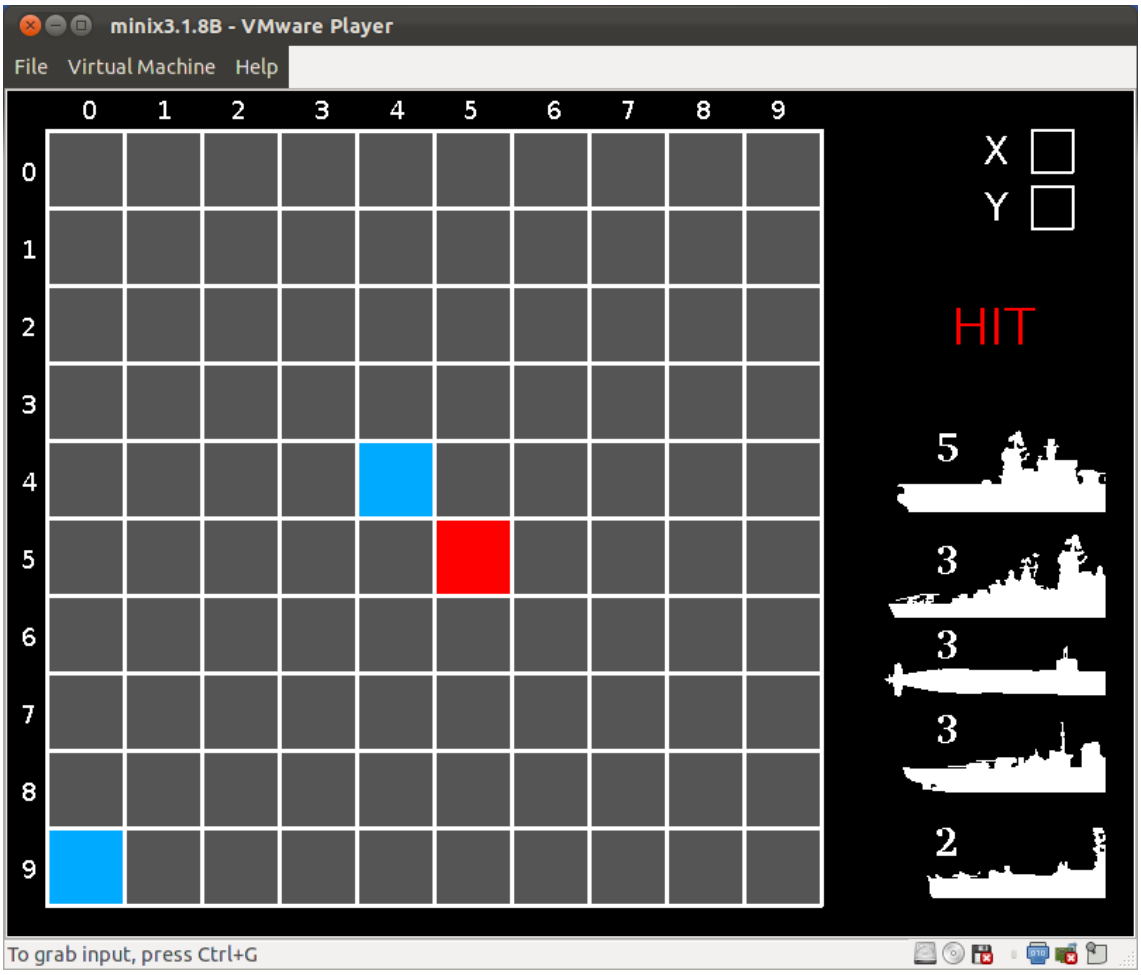
- Início de um jogo:



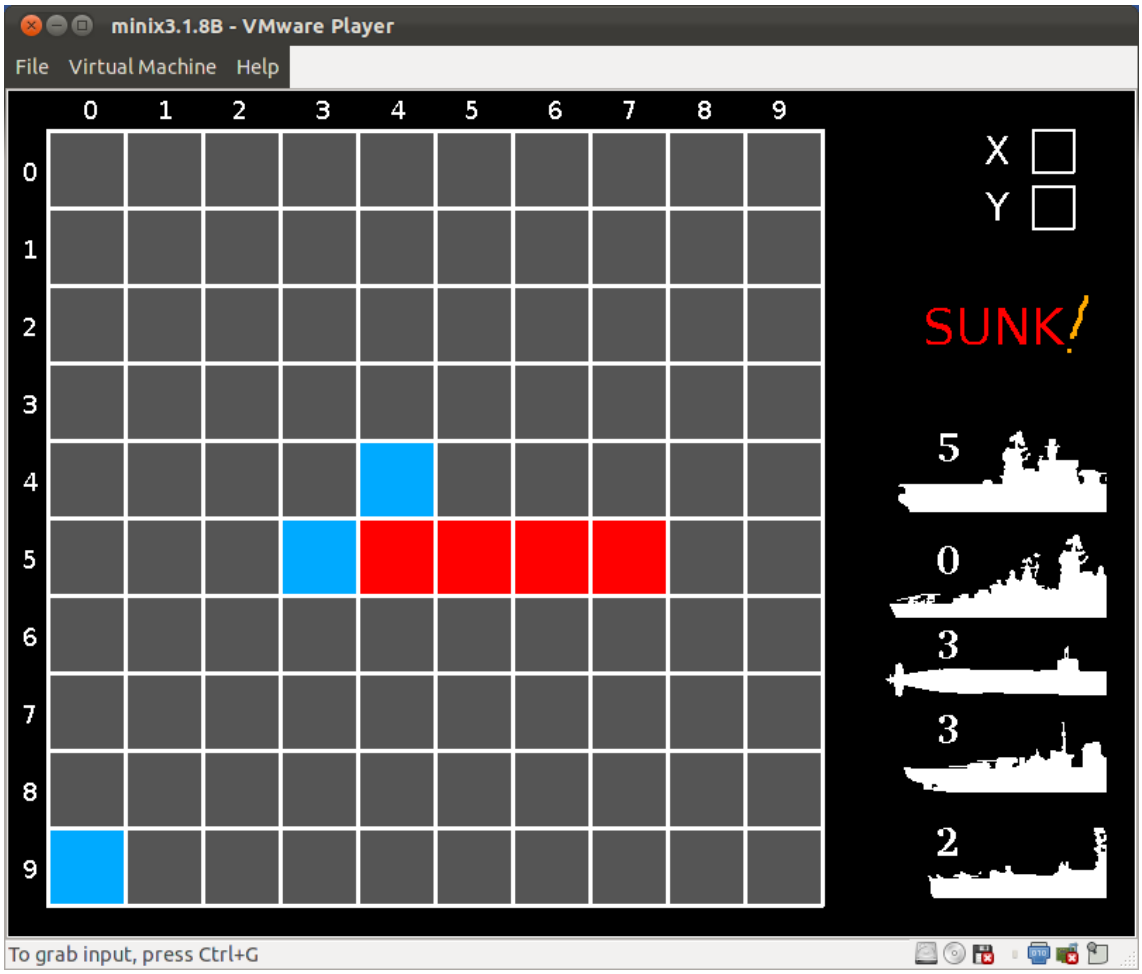
- Água:


minix3.1.8B - VMware Player
File Virtual Machine Help
0 1 2 3 4 5 6 7 8 9
X
Y
WATER
5
4
3
3
2
To grab input, press Ctrl+G

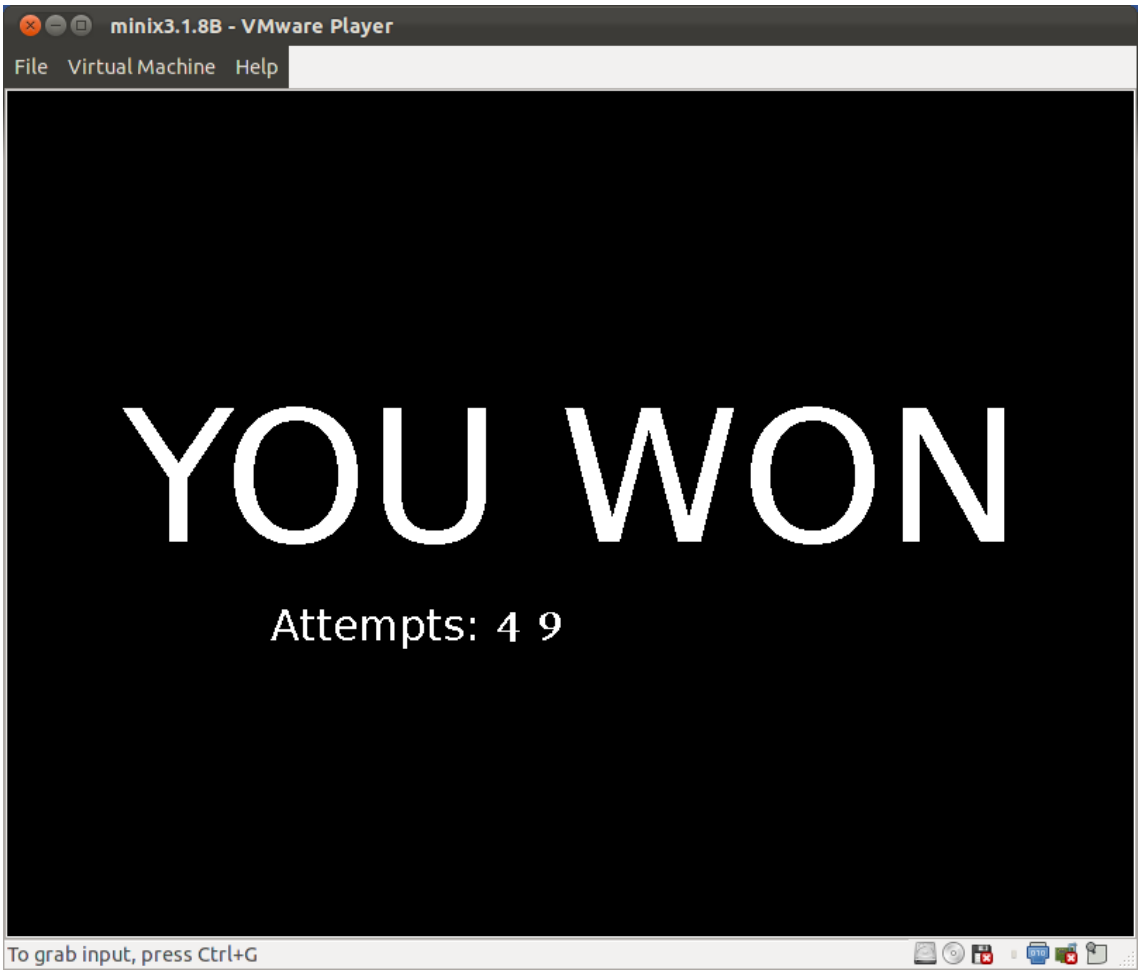
- Tiro:


minix3.1.8B - VMware Player
File
Virtual Machine
Help
0 1 2 3 4 5 6 7 8 9
0
1
2
3
4
5
6
7
8
9
X
Y
HIT
5
3
3
3
2
To grab input, press Ctrl+G

- Navio destruído:



- Jogo ganho:


minix3.1.8B - VMware Player
File Virtual Machine Help
YOU WON
Attempts: 4 9
To grab input, press Ctrl+G

Conclusão

Com os periféricos do rato e memória gráfica conseguiu-se implementar a solução que descreve o jogo da batalha naval. A interface final ficou amigável e o jogo corre de uma maneira fluente e sem quaisquer tipos de crashes. A porta série dada às suas limitações induziu em erro complicando e pondo em causa a conclusão do projeto, pelo que decidiu-se uma abordagem mais simplista mas que se mantivesse fiel à lógica do jogo.

Programar em Minix com uma linguagem que não é orientada a objetos (linguagem C) pode não ser o mais indicado, perdendo-se alguma das simplicidades que estamos habituados enquanto programadores, com um aumento significativo de erros, devido à perda de algumas abstrações. Quiçá numa outra instância usar o GCC seria uma melhor hipótese.

Bibliografia

- Wikipedia. Battleship (Game). Wikipedia, http://en.wikipedia.org/wiki/Battleship_game.