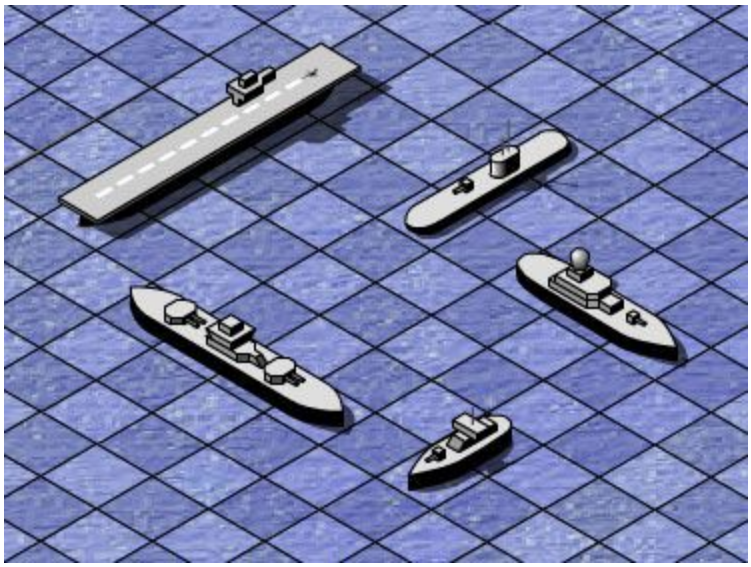
Introdução a Técnicas de Programação - 2016.1

Descrição de projeto

Batalha Naval



Introdução

Batalha naval é um clássico, que pode ter surgido ainda no período da 1ª guerra mundial. É um dos jogos mais conhecidos do formato de “papel e caneta”, no qual jogadores precisam apenas do suporte de papel e caneta para jogar. Ao longo dos anos, suas versões foram evoluindo em função do suporte tecnológico presente, chegando mesmo a extrapolar a mídia de jogo e alcançar as telas de cinema (“Battleship: a batalha dos mares” (2012)).

As regras do jogo Batalha Naval, como a maioria dos jogos no estilo “papel e caneta”, são simples e de fácil entendimento. É um jogo para dois jogadores (1x1) baseado em turnos. Cada jogador possui uma esquadra de navios, que deve evitar de ser afundada, e tem como objetivo afundar a esquadra adversária.

Inicialmente, cada jogador deve distribuir seus navios em uma malha (grid), normalmente quadrada. Os navios possuem tamanhos diferentes e ocupam na malha, portanto, um número de células consecutivas correspondente ao seu tamanho. O número de navios e tamanho de cada navio varia de acordo com a versão do jogo. O jogador oponente não deve ver a distribuição dos navios, pois eles precisam “adivinhar” a localização de cada um dos navios da esquadra. Os navios podem ser distribuídos aleatoriamente no tabuleiro, desde que as células de diferentes navios não se sobreponham e nem possuam células adjacentes. Ou seja, deve haver, no mínimo, o espaço de uma célula vazia entre dois navios.

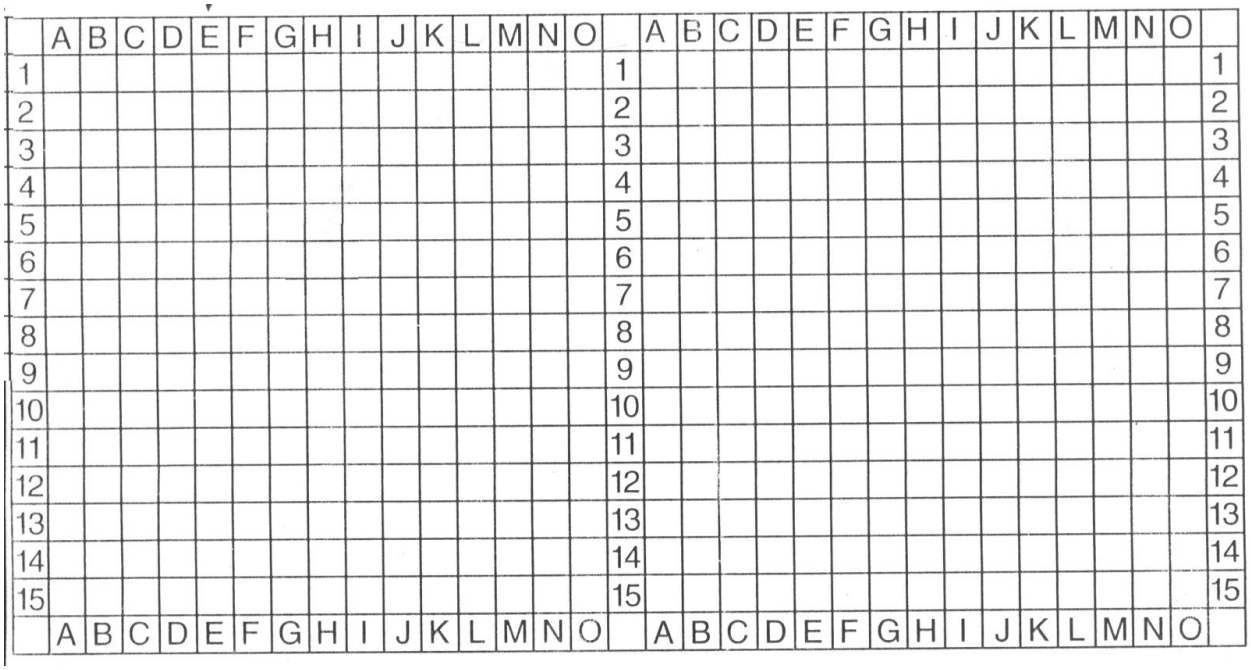


Figura 1 - Versão "papel e caneta" da Batalha Naval. Na malha à direita, definimos a configuração da nossa esquadra, e na malha à esquerda os tiros realizados.

Após o configuração inicial dos navios, a batalha se inicia, cada jogador lançando, a seu turno, um tiro no território do oponente. Um tiro é definido pelas coordenadas de uma das células da malha e atinge apenas a célula em questão. Caso a célula esteja ocupada por um dos navios do adversário, o jogador terá direito a mais um tiro. Ou seja, a cada turno, os jogadores continuarão tendo direito a tiros enquanto estiverem acertando navios adversários. Quando o número total de células de um navio for destruído, considera-se que o navio afundou e o jogador que afundar todos os navios adversários primeiro será o vencedor.

Para conhecer melhor as regras do jogo, você poderá testar uma versão simples online no link: http://pt.battleship-game.org/

Descrição do projeto

O projeto de Batalha Naval deve ser desenvolvido em linguagem C, a ser executado, em sua versão mais simples, através de linha de comando (entrada e saída em um console/terminal) e deve atender também os seguintes critérios:

1. **Usar de arranjos / matrizes;**
2. **Usar de registros e enumerações (`struct` e `enum`);**
3. **Definir novos tipos de dados através de `typedef`;**
4. **Realizar leitura e gravação de arquivos;**
5. **Ser modularizado em diferentes arquivos (uso de diferentes arquivos .c e .h, cada um com sua funcionalidade);**
6. **Possuir um padrão de indentação do código fonte e de nomenclatura das sub-rotinas e variáveis, e a adequação a este padrão;**

7. **Possuir documentação adequada do código-fonte.**

O projeto deve implementar um conjunto de regras para representar o comportamento de um oponente virtual. Ou seja, as regras irão ditar o processo de tomada de decisão do oponente do jogador. Esse conjunto de regras (IA do oponente) deve ser implementado em um módulo separado (.c), seguindo uma interface pré-definida, a ser encaminhada em arquivo anexo. A definição de um módulo separado com interface pré-definida será especificada em um arquivo em anexo.

O jogo implementado deve levar em conta que a esquadra terá os seguintes navios:

Navio	Número de células adjacentes	Quantidade a ser distribuída no tabuleiro
Porta-avião	4	1
Fragata	3	2
Corveta	2	3
Submarino	1	4

Observações:

- O projeto deve ser desenvolvido individualmente ou em duplas (grupos de dois alunos). Não serão permitidos grupos com três ou mais alunos.
- Para facilitar o acompanhamento do projeto, cada dupla deve estar associada a uma única turma de PTP (ou a um único professor de PTP). Ou seja, não serão permitidas duplas formadas por alunos matriculados em turmas de PTP com professores diferentes.
- Cada dupla deve desenvolver sua solução de forma independente das demais. Soluções idênticas serão consideradas plágios e, portanto, sanções serão devidamente aplicadas em todas as duplas com soluções similares.
- Códigos e algoritmos podem ser utilizados da web desde que devidamente referenciados. Caso sejam encontrados trechos de código na web equivalentes aos apresentados pelo grupo sem a devida citação, o código será igualmente considerado plágio e sanções serão aplicadas ao grupo. Vale salientar que a avaliação será realizada unicamente sobre o código produzido pela dupla. Códigos retirados da web, apesar de permitidos com a devida citação, serão desconsiderados dos critérios de pontuação.

Avaliação no componente curricular PTP (IMD0012.1)

Esta avaliação compreende a execução e desenvolvimento do projeto em 4 semanas. A cada semana, o grupo deve apresentar uma etapa do projeto desenvolvido, seguindo o calendário abaixo:

ETAPA 1 - semana de 11 a 17 de maio de 2016

1. Modularização do programa (quais os arquivos .c e .h)
2. Definição dos tipos de dados necessários (`typedef`, `structs` e `enums`);
3. Definição das malhas.

ETAPA 2 - semana de 18 a 24 de maio de 2016

4. Entrada do posicionamento inicial da esquadra do jogador;
5. Geração do posicionamento da esquadra da IA;
6. Execução do jogo baseado em turnos, simulando o adversário através da entrada do usuário.

ETAPA 3 - semana de 25 a 31 de maio de 2016

7. Substituição do adversário pela implementação de uma IA;
8. Leitura e gravação do estado do jogo em um arquivo.

ETAPA 4 - semana de 01 a 07 junho de 2016

9. Implementação de elementos extras, definidos pelo próprio grupo.

Sobre as duplas

Os alunos têm ATÉ o dia **10 de maio de 2016** para comunicar aos professores de ITP e PTP se farão o trabalho em dupla (e a composição da mesma) ou se farão individualmente.

Critérios de pontuação

O desenvolvimento do projeto aqui descrito vale **100%** da nota da terceira unidade de PTP.

- Uso da Linguagem C (Total: 2,0)
  - Arranjos (0,5)
  - Registros, Enums e Typedef (0,4)
  - Ponteiros e Passagem Referência (0,5)
  - Leitura/Escrita Arquivos (0,4)
  - Estruturas e comandos de controle básico (0,2)

- Organização do Projeto (Total: 1,0)
  - Modularização (0,5) (“O programa está devidamente modularizado em diferentes arquivos?”)
  - Padronização (0,2) (“A indentação e uso de { } seguem um padrão?”).
  - Documentação (0,3) (“o código está documentado?”)

- Estrutura do Projeto (Total: 4,0)
  - Modelagem dos Dados (0,5)
  - Impressão de Menu Inicial (0,1)
  - Novo Jogo (Total: 1,0)

        - Inicialização do tabuleiro (Total: 0,8)
            - Jogador (0,3)
            - Computador (0,5)
        - Impressão do Tabuleiro Inicial (0,2)
    - Loop Principal do jogo (Total: 2,0)
        - Leitura da jogada (0,2)
        - Validação da Jogada (0,7)
        - Verificação da condição de vitória do jogo (Em caso de afundar navio) (0,6)
        - (Na vez do jogador) Salvar e interromper a partida (0,3)
        - Alternância de turno e atualização do tabuleiro (0,2)
    - Continuar (Carregar Tabuleiro previamente salvo) (0,2)
    - Créditos (0,1)
    - Sair (0,1)

- IA do computador (Total: 2 pontos)
    - Aleatório (0,3)
    - Inteligente (joga geograficamente próximo do acerto) (1,0)
    - Inteligente + (joga geograficamente próximo do acerto e considera os navios já afundados) (1,5)
    - Integração com API fornecida (0,5)

  OBS: os modos Aleatório, Inteligente e Inteligente+ são excludentes, apenas um será considerado para avaliação. No caso, o mais complexo.

- Tópicos Avançados (Total: 3,0)
    - Interface Gráfica (1,0)
    - Jogo em rede (1,0)
    - Alocação Dinâmica do Tabuleiro (0,5)
    - Uso de recursão nas operações de alocação de navios (0,5)

  OBS: A pontuação a ser dada pelas funcionalidades “Outros” não é definida *a priori*. Cada caso será avaliado em função da complexidade envolvida. Itens extras de baixa complexidade serão desconsiderados na pontuação.

Entrega do projeto

O projeto deve ser submetido pelo SIGAA até a data **07 de junho de 2016** em um **arquivo comprimido (.zip)** contendo os arquivos fontes do projeto (.c e .h) e um arquivo README.TXT. Este arquivo deve ter as seguintes informações:

- O que foi feito (o básico e as funcionalidades extras implementadas);
- O que não foi feito (caso não tenha sido possível implementar alguma funcionalidade);
- O que seria feito diferentemente (quando aprendemos à medida que vamos implementando, por vezes vemos que podíamos ter implementado algo de forma

diferente. Assim, esse tópico é para vocês refletirem sobre o que vocês aprenderam durante o processo que, se fossem fazer o mesmo projeto novamente, fariam diferente);

- Como compilar o projeto, deixando explícito se foi utilizada alguma biblioteca externa que precise ser instalada, que versão e quais parâmetros extras são necessários para o compilador.
- Em caso de duplas:
    - Identificação dos autores;
    - Contribuição de cada integrante no desenvolvimento do projeto (quem fez o quê).

Recomendações diversas:

1. Solução de *backup*: não será tolerada a desculpa de que um disco rígido falhou nas avaliações. Assim, é importante manter várias cópias do código-fonte desenvolvido ou usar um sistema de backup automático como o Dropbox, Google Drive, Box ou similares. Uma solução melhor ainda é fazer uso de um sistema de controle de versões como git, e um repositório externo como Github ou Bitbucket.
2. Especificar precisamente a interface e o comportamento de cada sub-rotina. Usar esta informação para guiar o desenvolvimento e documentar o código desenvolvido.