Batalha Naval

Jesuino Vieira Filho
Departamento de Engenharias da Mobilidade
Universidade Federal de Santa Catarina - Centro Tecnológico de Joinville
Joinville, Brasil
jesuinovieira_@hotmail.com

Resumo

O intuito deste relatório é apresentar ao leitor o desenvolvimento do jogo Batalha Naval na linguagem de programação C++, realizado como trabalho da disciplina de Programação III do segundo semestre do ano de 2017. Dividido em quatro seções principais, o documento inicialmente apresenta uma breve introdução ao jogo, tais como suas regras e ilustrações do seu funcionamento. A segunda parte contém todo o desenvolvimento do programa, expondo desde a primeira abordagem do problema, a codificação e a aplicação da orientação ao objeto até a estrutura de decisão utilizada como inteligência computacional. O próximo tópico apresenta uma breve discussão sobre o projeto, como por exemplo, algumas dificuldades encontradas durante a sua realização e ocasionalmente, o esclarecimento de decisões que tiveram de ser tomadas. Finalmente, uma sessão com as conclusões encerra o documento, abordando o ponto de vista do autor sobre o desenvolvimento do projeto, assim como o progresso obtido com a realização.

I. INTRODUÇÃO

A. *O jogo*

Criado durante a primeira guerra mundial, Batalha Naval era inicialmente jogado em folhas de papel, no qual dois jogadores tentavam adivinhar em quais locais do tabuleiro estavam os navios do oponente. O passatempo tornou-se popular entre prisioneiros e soldados em meados de 1920, sendo comercializado pela primeira vez nos Estados Unidos, em 1931. Passado alguns anos, com o desenvolvimento de novas tecnologias e a chegada da guerra fria, surge a primeira versão em tabuleiro e, posteriormente, variações do jogo. Atualmente, há diversos softwares de Batalha Naval, até mesmo adaptações para um usuário jogar contra um computador - o qual é um dos objetivos deste trabalho. Tendo em vista a diversidade das variações do jogo, os tópicos a seguir se baseiam nas regras propostas para este projeto, podendo haver discordâncias com os estilos mais populares.

B. *Funcionamento*

O programa possibilita ao usuário dois tipos de jogos:

- Jogador contra Jogador: esse modo necessita de dois usuários, no qual o jogo se mantém até que um deles destrua toda a frota de navios do outro ou até que um deles desista - em caso de desistência de qualquer jogador, a vitória é atribuída para quem acertou mais tiros em navios;
- Jogador contra Ambiente: nesse modo, o usuário joga contra o computador, possuindo setenta tiros para destruir as embarcações - caso acabem, é declarado vitória para quem acertou mais tiros em navios. Antes disso, vence quem destruir toda frota de navios do outro. Possui dois níveis de dificuldades, fácil e difícil;

A decisão de tipo de jogo é realizada em tempo de chamada do jogo - por passagem de parâmetros ao programa. Por padrão, caso o usuário não passe nenhuma parâmetro ao programa, será adotado o tipo de jogo "Jogador contra Ambiente"no nível de dificuldade "difícil".

Inicialmente, independente do tipo de jogo, o programa usa o gerador de números aleatórios para posicionar os navios em uma grade 15x15, escolhendo uma coordenada como posição inicial e em seguida, uma direção (horizontal ou vertical) na qual a embarcação será marcada conforme o seu respectivo tamanho. O programa não permite a sobreposição de navios e estes também não podem ficar para fora da grade. Os tipos de navios, tais como a quantidade e o número de posições que ocupam no tabuleiro são descritos em seguida:

- Três fragatas: 2 posições;
- Três balizadores: 2 posições;
- Dois contratorpedeiros: 3 posições;
- Dois cruzadores: 3 posições;
- Um porta-avião: 4 posições;

Na tela é impresso os tabuleiros dos dois jogadores, os quais são atualizados todas as rodadas. As representações de cada célula da grade são dadas a seguir:

- Caractere '~ ' (til): representa a água;
- Caractere 'X': representa tiros que acertaram alguma embarcação;
- Caractere ' ' (espaço): representa tiros que acertaram a água;

Após os barcos serem posicionados, inicia-se a série de tiros. Os jogadores tem como objetivo descobrir em quais coordenadas estão os navios do oponente. Desse modo, cada jogador realiza um tiro por vez (no tabuleiro do oponente) até o final do jogo. Para cada tiro é impresso uma mensagem na tela relatando se o tiro foi certeiro ou não, em casos em que um navio é destruído, o programa emite uma mensagem dizendo qual o tipo do navio que foi destruído. Além disso, o usuário possui a opção de desistir a qualquer momento, dando como entrada "S"ou "Sair". Dado o fim de jogo, é impresso na tela o campeão e a situação atual do tabuleiro. Logo após, o programa é encerrado.

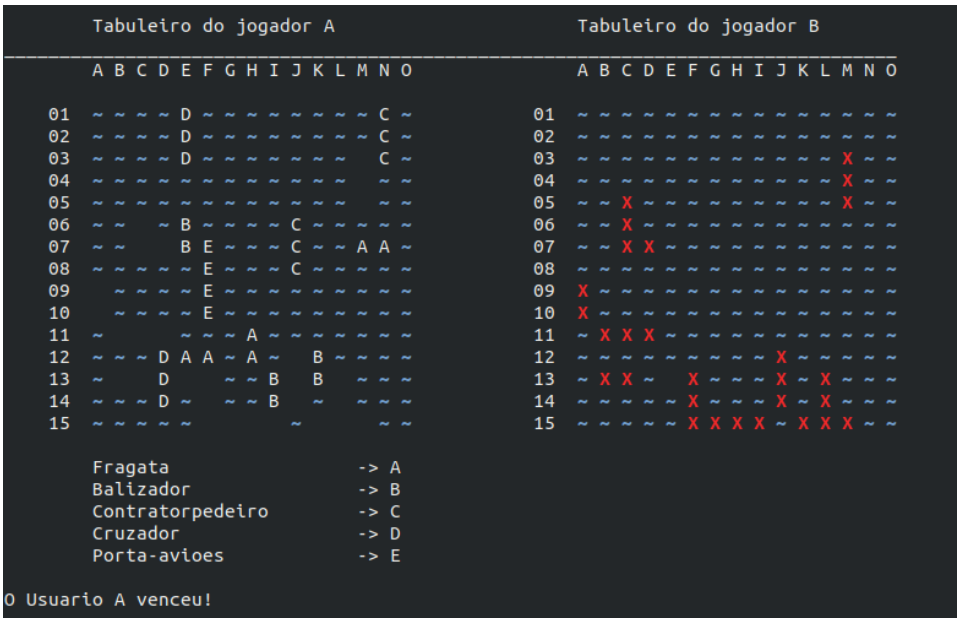

Figura 1. Final de uma partida.

II. DESENVOLVIMENTO

A. Estruturação do programa

Primeiramente, antes de inicial a implementação do código, a base do projeto foi sendo desenvolvida no papel, com uma breve análise das classes que iriam compor o jogo e seus respectivos relacionamentos. Isso foi extremamente importante para criar uma ideia do funcionamento do código e para facilitar a codificação. Para a realização desse exame preliminar, foi buscado responder a seguinte pergunta: "Quais métodos a classe precisa disponibilizar para que seus objetos tenham o melhor proveito, respeitando os conceitos da programação orientada ao objeto?". Desse modo, inicia-se a implementação a partir da classe mais simples até a mais complexa.

B. Implementação

1) Classe BatalhaNavalException: com o objetivo de dar mais clareza ao código, foi desenvolvida uma classe para representar as exceções que o programa pode apresentar ao decorrer de sua execução. Além do foco de tratar erros incomuns, a classe também foi utilizada para melhorar o fluxo do jogo. Sendo recebido como parâmetro um inteiro no lançamento da exceção e criação do objeto desta classe, mostramos a seguir um trecho do código com as possíveis exceções que a classe pode gerar.

```
const char* BatalhaNavalException::what() const throw()
{
        switch(codigo){
                case 1:
                        return "Excecao: tipo de jogo indefinido! Tente novamente.";
                case 2:
                        return "Um jogador abandonou o jogo!";
                case 3:
                        return "Seus tiros acabaram!";
                default:
                        return "Excecao desconhecida.";
        }
}
```

2) Classe Tabuleiro: a classe tabuleiro é fundamental e simples, foi utilizada para representar a grade dos dois jogadores. Possui uma matriz de caracteres que é inicializada com um caractere recebido como parâmetro no seu construtor. Um método setXT(int, int, char) que recebe um marcador e a coordenada em que a matriz será alterada. Um método getXY(int, int) que retorna o caractere localizado na coordenada recebida como parâmetro. Por fim, um método reiniciaTabuleiro() que insere o caractere recebido como parâmetro em todas as posições da matriz.

3) Classe Navio: a classe navio foi utilizada como base para as classes derivadas: Fragata, Balizador, Contratorpedeiro, Cruzador e PortaAviao. Cada uma de suas derivações possuem em seu construtor o nome e a quantidade de células do tabuleiro que cada tipo de embarcação utiliza. Assim, além de seus métodos getters e setters, possui o método decrementarVida() que é utilizado quando a embarcação é atingida.

4) Classe Jogador: esta é uma das classes mais trabalhosas do jogo, possuindo como principais atributos os vetores que representam cada navio do jogador e o seu tabuleiro. Foi decidido colocar alguns membros como protected para a implementação de sua classe derivada: IA. Em vista disso, a classe é responsável por posicionar os navios aleatoriamente, fazer a verificação das jogadas, a realizar um tiro, entre outros.

5) Classe IA: utilizada para o tipo de jogo Jogador contra Ambiente, esta classe é responsável pelas jogadas do computador. Foi realizado uma derivação do tipo public da classe Jogador, tendo em vista que o computador também "é um"Jogador e está contém algumas funções úteis para o funcionamento da

classe. Além disso, possui funções adicionais necessárias para administrar a estrutura de decisão utilizada, a qual será abordada posteriormente. Para o tipo de jogo Jogador contra Ambiente no nível de dificuldade fácil, é realizado apenas tiros aleatórios.

6) Classe Coordenadas: surgiu com a necessidade de salvar as coordenadas das futuras jogadas do computador no tipo de jogo Jogador contra Ambiente no modo difícil. Possui dois atributos inteiros que representam as coordenadas e suas funções getters e setters usuais.

7) Classe BatalhaNaval: aqui foi utilizado templates com o intuito de que cada tipo de jogo - Jogador contra Jogador ou Jogador contra Ambiente - possua um fluxo diferente. É composta principalmente por dois jogadores e um inteiro que representa a rodada atual. Desse modo, fornece ao utilizador da classe as funções necessárias para gerenciar um jogo de Batalha Naval.

8) Classe Interface: com a necessidade de uma comunicação com o usuário, surgiu a classe Interface. Esta também é um template, devido ao diferente fluxo que cada tipo de jogo exige. Os dois templates criados são compostos por um membro batalha naval, e principalmente, pelo método inicia(), o qual realiza o tratamento do jogo até o final. Um deles realiza o tratamento do jogo no modo Jogador contra Jogador e o outro, no modo Jogador contra Ambiente.

C. Aplicação dos conceitos de POO

1) Associação: a utilização de associação do tipo composição foi fundamental e muito utilizada no programa, sendo utilizada em quase todas as classes. Ao longo do desenvolvimento do programa, procurou-se respeitar o tipo de relacionamento tem-um, o qual governa esse mecanismo da linguagem C++.

2) Herança: a utilização de herança do tipo pública foi utilizada em duas classes com o objetivo de desfrutar do reaproveitamento de código que esse recurso oferece. Além disso, trouxe mais clareza e facilidade na implementação do código.

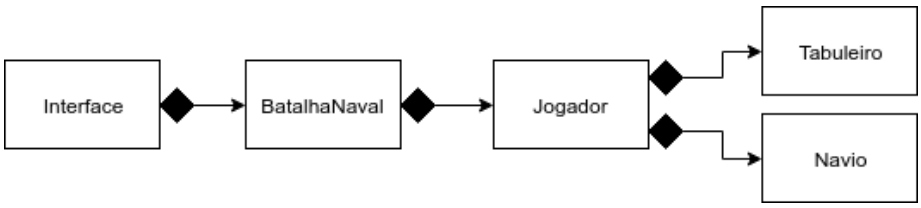

Figura 2. Representação das composições utilizadas no programa.

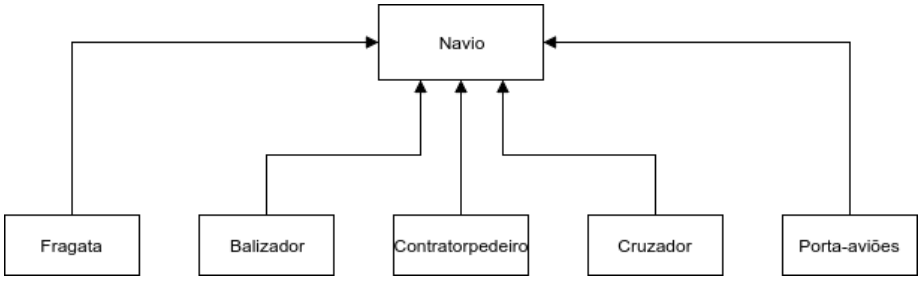

Figura 3. Classe base Navio e suas respectivas derivadas.

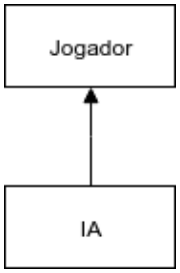

Figura 4. Classe base Jogador e sua respectiva derivada.

D. Estrutura de decisão

Como batalha naval é um jogo estratégico, criar uma estrutura de decisão inteligente para o programa não é algo fácil. A opção de utilizar uma árvore de decisão foi descartada devido as numerosas possibilidades de jogadas que o jogo possui, o que se tornaria inviável. Com o objetivo de contornar esse problema, a estrutura implementada realiza tiros aleatórios até acertar alguma embarcação. Sendo assim, é baseada em duas principais características:

1) Paridade: como o número mínimo de células que uma embarcação possui é dois, se considerarmos a grade como um tabuleiro de xadrez, com células pretas e brancas se revesando ao longo dele, aumentamos as nossas chances de acertar uma embarcação se atirarmos apenas nos locais fictícios brancos ou somente nos locais pretos.

2) Tiro certeiro: como dito anteriormente, os tiros aleatórios são realizadas até que uma embarcação seja acertada. Quando isso ocorre, é inserido em uma lista quatro jogadas: uma com as coordenadas da célula localizada no norte da que foi realizado o tiro, uma com as coordenadas do sul, uma com as coordenadas do oeste e outra com as coordenadas do leste. Caso já foi atirado em alguma dessas células ou elas ficariam para fora do tabuleiro, o algoritmo não insere. Assim, os próximos tiros são realizados nessas coordenadas e caso uma embarcação seja destruída, a lista é esvaziada e o algoritmo volta a atirar aleatoriamente. A seguir, o trecho de código responsável pelo fluxo mencionado acima.

```
void IA::getTiros(int* x, int* y, bool _IA)
{
        if(!_IA || lTirosVazia())       // Se o nivel de dificuldade e facil ou a lista
        tiroAleatorio(x, y, _IA);       // de tiros esta vazia

        else
        {
                int xx = 0, yy = 0;
                Coordenadas tmp;

                tmp = _lTiros.front();  // Pega o primeiro elemento da lista
                tmp.getXY(&xx, &yy);    // Pega as coordenadas
                _lTiros.pop_front();    // Remove o primeiro elemento da lista

                *x = xx;
                *y = yy;
        }
}

void IA::tiroAleatorio(int* xx, int* yy, bool _IA)
{
        std::random_device gerador;
```

```
        std::uniform_int_distribution<std::size_t> x(0, 14);     // Coordenada x
        std::uniform_int_distribution<std::size_t> y(0, 14);     // Coordenada y

        *xx = x(gerador);
        *yy = y(gerador);

        // Caso negue a paridade e IA esteja ativada, chama novamente esta funcao
        if(_IA)
                if(!confereParidade(*xx, *yy))
                        tiroAleatorio(xx, yy, _IA);
}
```

III. DISCUSSÃO

O desenvolvimento do programa iniciou-se com a implementação das classes fundamentais, seguido do posicionamento aleatório das embarcações, que ocorreu rapidamente, sem nenhuma dificuldade. A partir do momento que o programa começou a ficar extenso, exigiu-se uma maior organização, de modo que a implementação da classe Interface tornou-se necessária para realizar a interação com o usuário e o controle do jogo, facilitando o desenvolvimento das outras classes. Outro ponto que merece destaque é a complexidade de implementar um algoritmo estratégico e eficiente em um tabuleiro consideravelmente grande, além de tudo, podemos dizer que essa parte torna-se mais complicada considerando que o jogo é altamente influenciado pela sorte. Por último, aplicando as técnicas da orientação a objeto, ficou evidente como a linguagem C++ permite implementar códigos mais robustos do que, por exemplo, a linguagem C. Isso porque seus recursos de encapsulamento permitiram criar classes com comportamentos e estados bem definidos, que depois de implementadas e testadas, não são afetadas por outras implementações. Além disso, com a utilização do recurso de herança, o código tornou-se mais simples, organizado e compacto.

IV. CONCLUSÃO

O projeto apresentado permite desenvolver e familiarizar-se com os conceitos da programação orientada a objetos e assim, aprimorar as habilidades no desenvolvimento de softwares. Além disso, também foi possível compreender o por que a herança é um dos recursos mais importantes da linguagem de programação C++, ficando evidente o poder que ela possui e exaltando suas características de reusabilidade e criação de um código mais robusto. Finalmente, tanto na prática quanto na teoria, a conclusão do projeto foi enriquecedora, incorporando ao autor técnicas e competência para a elaboração de programas cada vez mais sofisticados.

REFERÊNCIAS

[1] https://en.wikipedia.org/wiki/Battleship_(game). Acessado em 05 de novembro de 2017.
[2] https://pt.wikipedia.org/wiki/Batalha_naval_(jogo). Acessado em 05 de novembro de 2017.
[3] http://www.datagenetics.com/blog/december32011/. Acessado em 05 de novembro de 2017.