Batalha Terrestre

Marco Cezar Moreira de Mattos[1], **Rômulo Manciola Meloca**[1]

[1]DACOM – Universidade Tecnológica Federal do Paraná (UTFPR)
Caixa Postal 271 – 87301-899 – Campo Mourão – PR – Brazil

{marco.cmm,rmeloca}@gmail.com

Resumo. *Este relatório apresenta o processo de desenvolvimento do jogo* Batalha Terrestre*, apresentando duas entregas, uma com cliente stand-alone e outra com a utilização de sockets e threads para a execução em computadores diferentes.*

1. Stand-Alone

Decidiu-se fazer o jogo Batalha Terrestre, como uma variação do clássico batalha naval, contanto com armas terrestres do pertencentes a frota do Exército Brasileiro, como por exemplo o blindado Guarani e o lança mísseis Astros 2020. O diferencial deste jogo, é que cada jogador pode montar sua própria estratégia, escolhendo com quais e com quantas armas deseja jogar, além de poder plantar minas no campo adversário antes de inciar a batalha. Quando o confronto é iniciado, todas as minas plantadas estouram, eliminando os campos inimigos em que as minas estiverem.

O desenvolvimento do projeto iniciou-se com a modelagem do diagrama de classes e, logo no início buscou-se fazer uma sólida base que fosse independente da interface gráfica, rebuscando os conceitos de Engenharia de Software e Orientação à Objetos de modo a maximizar o reaproveitamento de código, diminuir o acoplamento entre as classes e, consequentemente, pacotes e manter as classes coesas e encapsuladas. Desse modo, permitiu-se que o projeto facilmente fosse alterado e/ou adicionado de funcionalidades.

Contudo, a decisão por um projeto altamente consistente, tornou a programação fácil e natural. Tanto está independente de interface o projeto que existem dois modos de jogo, um textual (que foi utilizado como teste e teve seu desenvolvimento descontinuado) e outro em swing. A preocupação com um core bastante independente possibilitou a aplicação estar preparada para receber outras funcionalidades. Congregou-se os objetos em três pacotes: Controller, View e Jogo. Em jogo existem dois pacotes Tabuleiro e Alvo. No pacote Controller existe uma lista de jogos e uma lista de armas disponíveis para que o jogador as escolha. Também constam lá as funcionalidades e persistência e recuperação das partidas iniciadas.

No pacote Jogo existem as classes Jogo, Jogador e Estrategia. Jogo possui duas estratégias que por sua vez, cada qual, pertence à um Jogador e tem uma Grelha e uma lista de Armas utilizadas.

No subpacote Alvo existem as armas M60Patton, Astros2020, L118, Guarani como um CarroCombate, cada um com seu tamanho e imagem, e M15 como um Explosivo. Todos essas classes são especialização de Arma, que juntamente com os objetos concretos Terra e Borda, são epecializações de Objeto. São classes abstratas Objeto, Armas, CarroCombate e Explosivo. Em Objeto existe a sobrescrita do método toString(),

sendo este modificado para um método abstrato, obrigando todas as subclasses concretas a sobrescreverem o método.

No subpacote Tabuleiro existem as classes Grelha, Campo e Coordenada. Cada Campo possui uma coordenada, obrigatoriamente e nele consta alguma instância de Objeto (que são inicializadas como Terra) uma vez que a classe Grelha constrói toda a Grelha.

No pacote View existem as classes textView, que executa o jogo no modo texto e a classe GUI que, assim como FrameConfronto herdam todos os atributos de JFrame, de modo a abrirem janelas e exibirem aos jogadores a lógica do game. A classe FrameConfronto pertence à um Jogo e possui dois painéis do tipo PainelGrelha.

O jogo inicializa com a tela de escolha das armas, na qual cada jogador precisa preencher dez campos. Ao clicar em jogar, as minas são explodidas e cada jogador deve efetuar o seu disparo alternadamente, a menos que tenha acertado o tiro. Ganha o jogo o jogador que conseguir destruir todas as armas inimigas.

2. Cliente-Servidor

Segue o diagrama criado para trabalhar com as threads e socket.

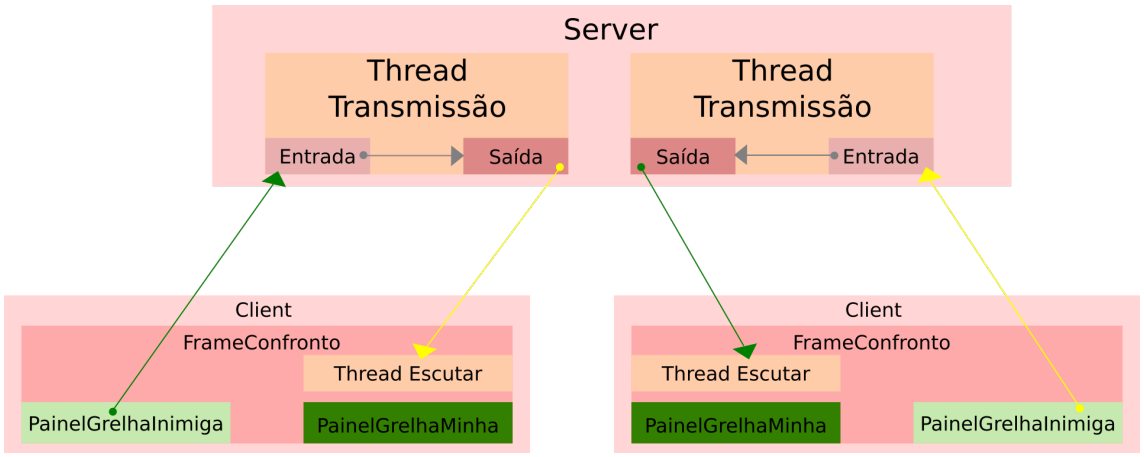

Figura 1. Fluxo de dados