Descrição do problema

Implementar um jogo de batalha naval utilizando a biblioteca socket e o protocolo TCP, com um tabuleiro 10x10, sendo que cada tabuleiro conterá 10 barcos, sendo eles 4 submarinos, sendo que cada um possui tamanho igual a 2, 3 contratorpedeiros de tamanho 3, 2 navios-tanque de tamanho 4, e um porta-aviões de tamanho 5. Em suas rodadas os jogadores tentam adivinhar a posição de um dos navios do oponente, e ganha o primeiro a afundar todos os navios do inimigo. Serão implementados dois programas, sendo um deles o cliente, que será controlado pelo jogador, e o outro programa o servidor.

Descrição do programa

Os tabuleiros foram armazenados em um vetor 10x10, com cada posição armazenando um inteiro que indica o que está presente naquele quadrado. O 0 representa a água, 1 os tiros dados em água, as posições de 2 a 5 os navios que já receberam tiros, sendo representados em ordem crescente de tamanho, logo o 2 representa o submarino, o 3 os contratorpedeiros e assim por diante. Os números 6 a 9 representam os navios que ainda não foram alvejados.

Ao iniciar os programas, é solicitado o IP ou nome, e as portas que serão utilizadas para a comunicação entre o servidor e o cliente. As mensagens foram enviadas em ASCII, e ao serem recebidas foram decodificadas e armazenadas em uma string.

Depois de informar os IPS e as portas, o jogo tem início, e são lidos do arquivo navios.txt as posições dos barcos do cliente. No arquivo tem a posição inicial e orientação do barco (N – Norte, S – Sul, W – Oeste, E – Leste) de um barco em cada linha separados por espaços (ex: 1 A N), com ordem crescente de acordo com a quantidade do mesmo tipo de navio (na primeira linha será lido o porta-aviões, por existir apenas um, depois os dois navios-tanque, etc).

Depois disso é apresentado para o cliente os comandos possíveis. Caso p seja apertado, são imprimidos os tabuleiros. Se esc for apertado, o jogo termina, e se j for apertado é iniciada uma jogada, com o console imprimindo “coordenadas:”, para indicar que o jogador deve fazer uma jogada. As coordenadas devem ser inseridas no padrão “(linha)(espaço)(coluna)”, como por exemplo “1 A”, e ocorrem trocas de mensagens entre o cliente e o servidor para saberem o resultado de suas jogadas.

Se o cliente tentar fazer uma jogada repetida ou com coordenadas invalidas, será mostrada uma mensagem indicando isso, e será necessário indicar as coordenadas de uma nova jogada. Caso o servidor acerte um tiro, na próxima jogada ele irá atirar nas posições vizinhas, seguindo sentido horário caso algum tiro não possa ser feito por ser uma jogada repetida ou indicar uma posição que está fora do tabuleiro. Assim, caso não fosse possível atirar a norte, tentaria atirar a leste, e assim por diante.

Quando a quantidade de acertos do cliente ou do servidor for igual a 30, que é a soma do tamanho de todos os barcos, o jogo termina.

Principais Funções

Cliente

mostrarTabuleiros -> chama o método mostrarTabuleiro, que imprime o tabuleiro do cliente e também chama o método mostrarTabuleiroInimigo, que troca mensagens com o server receber as inforações sobre o tabuleiro dele e então as imprime.

montarTabuleiro -> Lê o arquivo navios.txt e chama o método adicionarNavio, passando como parâmetros o tamanho do navio, as posições iniciais e orientação, para adicionar os navios nas posições definidas no arquivo de texto.

atirar -> Lê as coordenadas do input e envia como uma mensagem ao servidor, para que ele indique se o tiro atingiu um navio ou a água. Caso seja uma jogada inválida por já ter sido feita antes ou as coordenadas indicarem uma posição que não existe, as coordenadas serão solicitadas novamente.

receberTiro -> Recebe uma mensagem do servidor com as coordenadas de sua jogada e retorna para ele se atingiu um navio ou a água.

Servidor

montarTabuleiro -> Chama diversas vezes o método adicionarNavio, que tenta adicionar navios do tamanho passado a ele como parâmetro. Caso a posição gerada aleatoriamente já contenha um outro navio ou passará dos limites do tabuleiro, como por exemplo um navio qualquer sendo inserido na primeira linha com orientação Norte, o servidor continuará tentando inserir o navio em outras posições aleatórias até gerar uma que seja válida.

enviarTabuleiro -> Envia uma mensagem que contém as informações já descobertas pelo cliente.

atirar -> Atira em uma posição vizinha caso tenha atingido um navio na última jogada, ou então em uma posição aleatória se o ultimo tiro tiver atingido água ou não for possível atirar em uma posição próxima do último tiro por todas serem jogadas já feitas ou passarem dos limites do tabuleiro.

receberTiro -> Troca mensagens com o cliente para indicar se o tiro recebido atingiu água ou navios.

Testes