PROJETO FINAL - OCMA

IMD1012 - INTRODUÇÃO ÀS TÉCNICAS DE PROGRAMAÇÃO - 2021.2

Docente: CHARLES ANDRYE GALVAO MADEIRA

Discente: AFONSO JULIO MOREIRA NETO **Turma:** T02

RESUMO

A solução desenvolvida consiste na busca de um foco (ponto de pesca ou porto) baseado na distância entre o meu bot e dos outros bots na execução. Caso a distância do meu bot ou para a tainha, ou a cioba ou o robalo mais próximos seja menor ou igual a dos outros bots, um foco é definido. Dessa forma, o bot avança até o ponto e executa a ação de pesca, respeitando os limites mínimos. Se estiver com o limite máximo de pesca - 10 peixes - procura o porto mais próximo, vai até ele, vende os peixes e busca um novo foco. Assim sucessivamente até que sejam encerradas as rodadas ou que o jogo termine.

INTRODUÇÃO

No contexto de um jogo de simulação em uma área de pesca gerenciada pelo OCMA - Órgão de Controle do Meio Ambiente, é preciso desenvolver um *boat*, ou melhor, um *bot* de pesca que respeite as condutas discriminadas de uma pesca mais sustentável. Entre estas condutas, temos de principal para o desenvolvimento da aplicação o respeito aos limites da região onde a pesca é sediada e à população dos cardumes de peixes.

Além do supracitado, é preciso lembrar que a solução é testada e avaliada dentro de um âmbito coletivo/competitivo, porque simula a ideia de uma prática pesqueira regulamentada pela autonomia tecnológica, em que a pesca atinja a sua máxima eficiência em meio a outros barcos pesqueiros, respeitando o controle ambiental.

DESENVOLVIMENTO

O desenvolvimento do meu bot seguiu às seguintes fases:

1. Leitura e definição das variáveis

1.1. Estruturas heterogêneas

Para a leitura e definição de variáveis foram criadas 3 estruturas de dados heterogêneas: *Position*, *Bot* e *NextObject*. Abaixo segue um breve explicação sobre cada um delas:

- *Position* armazena dois inteiros, um para guardar a linha - largura - (*width,* w) e outro para a coluna - altura - (*height,* h). Dessa forma, *Position* serve para auxiliar no gerenciamento das posições dos.
- *Bot* armazena uma variável do tipo *Position* e um ID do tipo matriz de caracteres - *string*. O tipo *Bot* é utilizado para definir todos os bots.
- *NextObject* armazena as mesmas informações que *Bot* com a adição de um campo inteiro para a distância.

As demais variáveis são tipos já definidos na linguagem C.

1.2. Leitura da entrada

A leitura da área de pesca e tanto a leitura, quanto a definição dos bots é feita através das funções readData(), defineOtherBots() e defMyBot(). Como são funções de leitura, o único destaque é a utilização lógica de ponteiros para leitura.

2. Buscar foco

Após realizada a leitura das variáveis, o bot passa a buscar um foco. Foco é uma posição no mapa que é definido na lógica que será apresentada a seguir.

Para definir um foco é necessário saber se a quantidade de peixes totais no barco é menor que o máximo permitido, 10 peixes. Caso seja, vai tentar buscar um ponto para pescar como foco. Caso não seja, busca o porto mais próximo e vai até ele para vender os peixes.

2.1. Foco: ponto de pesca

Para buscar um foco de pesca é necessário verificar se não existe um foco e está *rotacionando ao redor do mapa* buscando um foco e se é a primeira rodada ou se o meu bot não está no foco.

Caso as condições sejam atendidas haverá a tentativa de definição de um foco pela função mapFocus(), buscando saber os 3 peixes - um de cada tipo - mais próximos do meu bot, e os bots mais próximos desses peixes, através de um cálculo de distância feito pela função distance().

Existe uma lógica diferente para cada caso em que meu bot seja de nenhum a todos os bots mais próximos dos peixes. A definição de prioridade de foco, caso meu bot seja algum desses bots, inclusive todos, é pela menor das distâncias. Caso sejam iguais, o foco será do R$/kg mais caro.

Caso seja nenhum dos bots, o bot vai *tentar*[2], através das funções minExtLimit() e maxExtLimit(), entrar em um eixo “circular” na região de pesca, viajando entre os pontos (¼ largura, ¼ altura), (¾ largura, ¼ altura), (¾ largura, ¾ altura) e (¼ largura, ¾ altura), que definem o eixo de rotação quando não há foco.

Definido um ponto de pesca como foco, o algoritmo passa a verificar se o bot está no foco - função onFocus() - e se estiver, verifica se ainda há a possibilidade de pesca naquele ponto, através de onFocusFishable(). Caso não esteja no foco, se move até ele, através de moveToFocus(). Se estiver, mas não é “pescável”, tenta definir um novo foco.

2.2. Foco: porto

Através da função nextHarbor() procura e define o porto mais próximo. Caso não esteja no foco, em um porto, se move até o foco através da função moveToFocus(). Caso esteja no foco, vende os peixes, zera o total de peixes e “anula” o foco ao porto.

CONCLUSÃO

O meu bot cumpre com as normativas propostas para uma pesca mais sustentável, porém ainda é necessário um melhor debugging nas funções de rotação do mapa, minExtLimit() e maxExtLimit().

Acredito que está ainda longe dos termos de uma solução ótima, mas, ainda assim, é uma solução funcional.