Segundo Trabalho de Programação Orientada a Objetos 2005.1

Prof. Renato Cerqueira

Objetivo

Neste segundo trabalho, deve-se desenvolver um jogo inspirado no sistema utilizado na disciplina Introdução à Engenharia do ciclo básico do CTC/PUC-Rio.[1] Esse jogo é uma disputa entre robôs submarinos para prospecção de petróleo.

Os robôs submarinos são lançados ao mar, a partir de um helicóptero, sobre uma região onde se acredita haver petróleo. Esses robôs atingem o solo submarino em posições aleatórias. Duas ou mais companhias competidoras lançam robôs semelhantes para competir por poços na mesma região. Vence o jogo a companhia cujos robôs conseguirem produzir mais petróleo durante um período predeterminado.

Os robôs são equipados com diversos sensores que fornecem constantemente informações, tais como a posição do robô, a pressão subterrânea no local, e o gradiente da pressão no local. Além disso, cada robô pode perfurar o solo e iniciar a prospecção de petróleo em sua posição corrente. Sabe-se que a produção de petróleo em um local é diretamente proporcional à pressão subterrânea no local. Por isso, os robôs devem encontrar os locais com maior potencial de produção e iniciar perfurações nesses locais.

Descrição

Seu programa deve receber como entrada dois arquivos:

- um arquivo com a descrição do terreno submarino (criado com o editor implementado no primeiro trabalho);
- um arquivo com a configuração do jogo, que inclui informações como
  - o tempo de duração da partida;
  - a descrição de cada equipe que irá disputar o jogo, com o nome da equipe, a cor de identificação da equipe, os nomes de seus robôs, e os tipos de controladores associados a cada um de seus robôs (os tipos de controladores de robôs serão discutidos mais adiante);

Esse segundo arquivo de entrada pode ser definido com a mesma linguagem que você utilizou para descrever o terreno (XML ou Lua).

Ao se começar o jogo, os robôs são distribuídos aleatoriamente pelas células do terreno. Dois robôs nunca podem ocupar uma mesma célula em um mesmo instante do jogo. Os limites extremos do terreno funcionam como paredes, isto é, os robôs não conseguem sair desses limites.

Uma vez que todos os robôs estejam posicionados, estes dão início a suas estratégias de prospecção, que devem ser executadas concorrentemente até o término do partida.

[1] Mais detalhes em `http://www-di.inf.puc-rio.br/~hermann/introeng/introeng.htm` e `http://www.inf.puc-rio.br/~mascarenhas/robots/`.

A Interface com o Usuário

Durante a execução da partida, o usuário deve poder acompanhar a evolução do volume de petróleo prospectado por equipe e por robô. O usuário também deve ser capaz de consultar os valores das propriedades das células do terreno durante o jogo, assim como acompanhar a posição corrente de cada robô. Ao final da partida, deve ser exibido ao usuário o relatório final com as quantidades prospectadas por equipe e por robô.

Os robôs devem ser desenhados com a cor de sua equipe e devem exibir seu nome, para que o usuário possa acompanhar a evolução dos robôs individualmente. A representação gráfica do robô deve indicar sua orientação, isto é, se ele está voltado para o norte, sul, leste ou oeste (essa informação é importante pois, como será visto mais adiante, as operações de movimentação dos robôs são relativas à sua direção corrente).

O Terreno

Em uma região submarina pode haver mais de um campo petrolífero. A quantidade de petróleo a ser explorado é diretamente proporcional à pressão de uma célula do terreno. Sabe-se que a distribuição de pressão em um campo petrolífero obedece a uma função suave, isto é, sem rupturas ou discontinuidades. Entretanto, devido à geografia do subsolo (existência de maciços rochosos, transições tectônicas e etc.) a distribuição de pressão pode corresponder a somente uma porção limitada da função. Se uma célula do terreno for perfurada, então não pode haver outra perfuração nessa célula e nem em suas células adjacentes; isto é, após começar a perfuração, nenhum outro robô deve conseguir perfurar o terreno nessas células.

Deve ser definida uma fórmula que relacione a pressão de uma célula com o volume de petróleo que um robô consegue extrair nessa célula. Essa fórmula irá determinar quantos barris de petróleo um robô irá contabilizar para sua equipe quando fizer uma prospecção nessa célula.

Como foi visto no primeiro trabalho, ao se configurar os valores das propriedades das células do terreno, é possível definir que esses valores serão determinados dinamicamente em função das coordenadas (x, y) da célula sendo consultada. Para isso, no editor de terreno, é indicado o nome da classe do objeto que deve ser responsável por determinar o valor da propriedade. Sempre que se quiser consultar uma dessas propriedades com valores dinâmicos, seu programa deve verificar se é a primeira vez que a classe associada vai ser utilizada. Se for a primeira vez, seu programa deve carregar a classe e criar uma instância (objeto) dela. Esse objeto deve ser guardado para futuras consultas a propriedades associadas a mesma classe. Por exemplo, considere que várias células têm seu valor de pressão associado a uma classe `X`. Na primeira vez que o jogo precisar consultar o valor da pressão em uma dessas células, o seu programa deve carregar a classe `X` e criar uma instância dessa classe para responder à consulta. Nas próximas vezes que for necessário consultar a pressão em qualquer uma dessas células, essa mesma instância de `X` deve ser utilizada, ao invés de se criar uma nova instância.

Os Robôs

Os robôs devem ser capazes de receber comandos, aos quais eles reagem apropriadamente. Cada robô possui um *controlador*, que é responsável por controlar suas ações, isto é, um controlador é a entidade encarregada de enviar comandos para um robô, de acordo com sua estratégia de jogo. Em um determinado instante do jogo, um robô só pode ter um controlador associado a ele, mas esse controlador pode mudar ao longo do jogo.

Para simular uma situação mais real, os comandos devem possuir um custo de tempo para serem executados. Durante a execução de um comando, o controlador que enviou o comando fica parado esperando sua conclusão. Esse custo de tempo pode variar de acordo com o comando e com o contexto em que o comando deve ser executado. Por exemplo, um comando para movimentar um robô pode levar mais ou menos tempo em função da diferença de profundidade entre as células de origem e destino.

O conjunto mínimo de comandos que um robô deve ser capaz de processar pode ser dividido em 4 subconjuntos:

- comandos de sensoriamento: `posicao`, `pressao`, `gradientes`,`tempo`
- comandos de movimentação: `anda`, `volta`, `esquerda`, `direita`
- comando de prospecção: `sonda`;
- comandos de comunicação: `enviamsg`, `recebemsg`

Através dos comandos de sensoriamento, um controlador pode obter a posição (x, y) atual de seu robô (comando `posicao`), assim como obter a pressão em sua posição (comando `pressao`) e os gradientes de pressão em sua região (comando `gradientes`). O comando `gradientes` informa o gradiente de pressão na direção da orientação de movimento do robô (diferença entre as pressões na célula a sua frente e na célula atrás dele) e o gradiente na direção normal ao seu movimento (diferença entre as pressões na célula a sua direita e na célula a sua esquerda). Utilize o intervalo do coeficiente de erro de leitura, definido na configuração do terreno (primeiro trabalho), para introduzir erros aleatórios nas leituras de pressão. O comando `tempo` permite um controlador obter o tempo decorrido em segundos desde o início da partida.

Os comandos de movimentação permitem deslocamentos apenas para células adjacentes à posição corrente do robô. Esses comandos são relativos à posição corrente do robô que, além de sua posição (x, y) no terreno, também indica em que direção o robô está apontando. Os comandos `anda` e `volta` fazem com que o robô ande para frente e para trás, respectivamente. Os comandos `direita` e `esquerda` fazem com que o robô vire 90^o para direita ou esquerda.

O comando `sonda` perfura um poço na célula onde se encontra o robô. Como foi dito anteriormente, quando um robô começa a extrair petróleo em uma determinada célula, não é mais permitido começar uma perfuração nessa célula e nem nas suas células adjacentes. Isto é, robôs posicionados nessas células não poderão mais executar o comando `sonda`.

Robôs de uma mesma equipe podem trocar mensagens entre si. A idéia é que essa troca de mensagens, feita através dos comandos `enviamsg` e `recebemsg`, permita que os robôs de uma mesma equipe troquem informações ou enviem comandos para seus companheiros, com o intuito de colaborarem na estratégia do jogo.

Os Controladores

Como foi visto anteriormente, cada robô possui um controlador, que é responsável por comandar suas ações no jogo. As associações entre controladores e robôs devem ser definidas no arquivo de entrada que descreve as equipes que irão participar do jogo. Mas seu programa também pode oferecer algum mecanismo para fazer novas associações entre controladores e robôs durante a execução do jogo.

O seu programa deve oferecer pelo menos os seguintes tipos de controladores:

- controladores pré-programados: são controladores que possuem sua lógica de controle já programa diretamente em Java;
- controladores interativos: são controladores que recebem comandos interativamente do usuário (via a interface do seu programa);
- controladores remotos: são controladores que enviam seus comandos via RMI;

Um outro tipo de controlador interessante, mas que é opcional, seria um controlador interpretado, que recebe a sua lógica de controle em tempo de execução, descrita em alguma notação que seria interpretada pelo controlador. Por exemplo, esse controlador poderia interpretar um *script* em Lua, com o auxílio da ferramenta LuaJava, que definiria sua lógica de controle.

Observações

- **Data de entrega:** 29/junho.
- Trabalhos atrasados perdem 1 ponto por dia de atraso.

- O trabalho pode ser feito em grupo de até 3 participantes.
- Serão marcadas apresentações dos grupos entre os dias 29/junho e 1/julho. As notas dos membros do grupo podem ser diferenciadas, em função do conhecimento demonstrado sobre o projeto durante a apresentação do trabalho.
- Na correção do trabalho, não será avaliado apenas se o programa funciona. Também será considerada a qualidade do projeto, principalmente sob o ponto de vista do emprego de técnicas de programação orientada a objetos.
- Cola, total ou parcial, implica em grau zero para todos os envolvidos.
- Os trabalhos devem ser enviados por email para `rcerq@inf.puc-rio.br`.
- O email deve conter apenas os arquivos `.java`, alguns exemplos de arquivos de entrada, um arquivo texto explicando como compilar e executar o seu programa, e um relatório explicando seu projeto (classes e interfaces definidas, os relacionamentos entre os objetos, padrões de projeto utilizados, etc.). Opcionalmente, os arquivos podem ser enviados em um único arquivo `zip` ou `tgz`.
- Só considere o seu trabalho entregue após receber uma mensagem de confirmação minha.