Manual do Programador para o Projeto de Programação Orientada aos Objectos

• U  C •

João Craveiro 2013136429
João Faria 2013136446

Professor Alexandre Pinto, POO 2014/2015

Introdução

O principal objetivo deste manual é descrever a estrutura do código usada no nosso programa e clarificar as razões por detrás de algumas das nossas escolhas. O programa simula o movimento de diferentes robots(agentes) num ambiente onde estão inseridos diferentes objectos. O objectivo dos agentes é percorrer o ambiente em busca de todos os os objectos, num número máximo de saltos. O agente move-se de acordo com a sua estratégia (objecto mais perto, aleatória e máxima diferença relativa em relação aos objectos em memória) sempre que tem algum objecto no seu campo de visão, quando não tem nenhum objecto no seu campo de visão os agentes movem-se aleatoriamente no seu campo de visão.

Estrutura Geral

Classe Environment

Esta classe representa o ambiente onde a simulação vai ocorrer, define os limites do campo e guarda todas es entidades.

Métodos principais

addToEnvironment()

Este método recebe uma entidade e primeiro verifica se as coordenadas deste se encontram dentro do ambiente. Se estiver dentro do ambiente verifica se é uma instância de InanimateObject ou se é um Agent. Antes de adicionar à respectiva ArrayList verifica ainda se há algum objecto na mesma posição e não o adiciona caso isso se verifica.

run()

Este método itera pelos diferentes agentes do Environment e chama as suas rotinas.

Classe Entity

A classe Entity é uma classe abstracta e é a Parent Class de todas as outras entidades. Tem todas as variáveis e métodos em comum com todos os tipos diferentes de entidades (InanimateObject, Agent, RandomStrat, Hamming e Closest).

Métodos principais

checkSamePosition()

Método que é chamado por uma Entity que verifica se as coordenadas de outra Entity são iguais às suas. O método retorna 1 se as entidades estão na mesma posição ou -1 caso contrário.

Classe Agent

Child Class de Entity e Parent Class de todos os agentes, Agent é uma classe abstracta que tem todos os métodos e variáveis comuns aos diferentes tipos de agentes tal como a memória e a sequência de coordenadas.

Métodos Principais

setNextPossibleObjects()

Este método vai iterar por todos os InanimateObjects no Environment e avaliar a distância entre eles e o Agent, se os objectos se encontrarem dentro do campo de visão do agente eles são adicionados à ArrayList de próximos objectos para onde o agente se pode deslocar.
O método vai retornar 1 se adiciona algum objecto à ArrayList ou -1 se não encontrar nenhum objecto dentro do campo de visão.

getNextObject()

Nesta classe este método é uma declaração abstracta do método que, consoante a política escolhida, vai seleccionar o objecto para o qual o agente se vai mexer de todos os objectos que se encontram no seu campo de visão.

routine()

Um dos métodos principais dos agentes, este método vai invocar outros métodos do agente para efectuar a simulação (getNextObject, moveTo, addToMemory). Este método vai também, a cada iteração, escrever para um ficheiro de log toda a informação da simulação, isto é as coordenadas do objecto, a sua percepção e a sua memória.

Classe Closest

Child class da classe Agent este método vai realizar a implementação da política de seleccionar o objecto mais próximo de si dentro do campo de visão.

Métodos Principais

getNextObject()

Este método vai seleccionar o próximo objecto para onde o agente se vai deslocar.
Se apenas existir um objecto no seu campo de visão ele selecciona-o logo, caso contrário vai percorrer todos os objectos do campo de visão e calcular a distância entre eles e o agente e selecciona o que tiver a menor distância, em caso de empate escolhe o primeiro que viu.

Classe RandomStrat

Child class da classe Agent este método vai realizar a implementação da política de seleccionar um objecto aleatoriamente dentro de todos os objectos no campo de visão de um agente.

Métodos Principais

getNextObject()

Este método vai seleccionar o próximo objecto para onde o agente se vai deslocar.
Se apenas existir um objecto no seu campo de visão ele selecciona-o logo, caso contrário vai escolher um número aleatório e menor ou igual ao número de objectos no campo de visão e selecciona esse objecto.

Classe Hamming

Child class da classe Agent este método vai realizar a implementação da política de seleccionar o objecto coma máxima diferença relativa em relação aos objectos em memória.

Métodos Principais

getNextObject()

Este método vai seleccionar o próximo objecto para onde o agente se vai deslocar.
Na primeira iteração escolhe o primeiro objecto que vê ou se apenas encontrar um selecciona-o. Nas restantes iterações vai calcular a diferença de Hamming entre os objectos no campo de visão e os objectos em memória seleccionando o menor valor. Depois de todos os valores selecciona o maior e desloca-se para esse objecto.

Classe Coordinates

Esta classe foi criada para guardar e tratar todas as operações relativas às coordenadas.

Métodos Principais

updateCoords():

Este método actualiza as coordenadas de um agente para as coordenadas recebidas.

addFov():

Este método é utilizado quando não existir nenhum objecto dentro do campo de visão de um agente. Nesse caso o método vai escolher uma coordenada aleatória dentro de todas as possíveis coordenadas para onde o agente se pode deslocar.

distanceBeetween():

Este método recebe duas coordenadas e calcula a distancia entre elas.

inEnvironment():

Este método recebe uma coordenada e verifica se esta está dentro do ambiente.

Classe InanimateObject

Child Classe da classe Entity os objectos desta classe são os objectos que vão popular o ambiente de simulação.

Métodos Principais

getHammingDistance()

Método estático que recebe dois objectos e calcula a diferença de Hamming entre eles. Ele compara os dois objectos e retorna um valor entre 0 e 3 dependendo das diferenças das características de ambos.

Classe FileHandling:

Classe criada para tratar de todas as operações com ficheiros, seja ela a escrita ou a leitura de ficheiros.

Métodos Principais

openFileRead():

Este método recebe um ficheiro e abre-o iniciando o BufferedReader. Caso o ficheiro não exista na directoria a função retorna -1.

openFileWriter():

Este método recebe um ficheiro que é aberto em modo de escrita e inicializa o BufferedWriter.

openFileAppend():

Este método recebe um ficheiro e abre-o para adicionar texto e inicia o BufferedReader.

closeRead():

Método que fecha um ficheiro que tenha sido aberto para leitura.

closeWrite():

Método que fecha um ficheiro que tenha sido aberto para escrita.

readConfig():

Este método lê o ficheiro de configuração e cria o ambiente. Na primeira iteração o método vai inicializar o ambiente e nas iterações seguintes vai colocar as entidades no ambiente. Caso o método openFileRead executado dentre de este método retorne -1, é retornado -1 e não é criado o ambiente.

initEnvironment():

Método chamado no readConfig recebe uma linha que contém o tamanho do ambiente, presente no ficheiro de configuração e cria-o.

initEntity():

Método que recebe o resto do ficheiro de configuração linha a linha. O método vai dar parse à linha separando as características da entidade, identificando-a, e no final adiciona-a ao ambiente.

writeToLog():

Este método recebe um agente e uma iteração e escreve no ficheiro log em cada iteração, as coordenadas, a memória e a percepção do agente.

writeStatsToLog():

Método usado na iteração final de um agente onde são escritas as estatísticas finais (distância total percorrida e número de objectos aprendidos). O número de objecto diferentes aprendidos não é imprimido pois o nosso agente nunca revê o mesmo objecto.

Notas adicionais

GUI

Para o projecto tentámos fazer uma User Interface simples de perceber e que seja intuitiva para o utilizador.
A frame UIInterface é a janela principal, esta dá a opção de escolher entre iniciar a simulação usando o ficheiro de configuração ja criado ou pode ser criado um novo ficheiro. Caso esta última opção seja escolhida vão ser abertas várias janelas, de cada vez, que permitirão ao utilizador escrever os dados.

Opções Tomadas

Para facilitar o nosso projecto optámos por definir o campo de visão dos agentes como um quadrado e não um circulo.
Outra opção tomada por nós foi a de quando não se encontra nenhum objecto no campo de visão de um agente, este desloca-se para uma posição aleatória dentro do seu campo de visão, isto significa que o agente pode passar por uma posição onde já esteve mas na nossa interpretação do enunciado nunca volta a aprender um objecto que já foi aprendido previamente.
Por fim se um agente encontrar todos os objectos do Environment antes do seu tempo de vida terminar a sua simulação termina.