Relatório - EP Fase 2
Laboratório de Programação 2

28 de outubro de 2013

1 Integrantes

- Victor Sanches Portella - Nº USP: 7991152
- Lucas Dário - Nº USP: 7990940
- Ruan Costa - Nº USP: 7990929

2 Introdução

Neste relatório temos como objetivo dar uma explicação concisa sobre a organização das classes no funcionamento do programa atualmente. Para isso, explicaremos o funcionamento do progrmaa inteiro, e iremos nos aprofundar nos assuntos que forem necessários.

3 Funcionamento

Nessa fase o EP ainda não têm o funcionamento de um jogo em si, mas já é possível ver diversos robos agindo em paralelo. Para isso, o script em Perl[1] traduz diversos programas para um arquivo chamado **Main.java**.

Todos os robôs traduzidos serão colocados no mesmo time. Ao rodar o programa, os programas de cada robô será executado até o fim. Quando todos os robos tiverem acabado de executar, o programa será finalizado.

Cada robô imprime as instruções que execultou para que seja possível que nós possamos acompanhar a execução de cada robô.

[1] Explicação do mesmo feita mais a frente

4 Classes

4.1 Classe Arena

Essa classe representa a Arena em si. Nela, temos os seguintes atributos:

- **Mapa**: Objeto do tipo **Mapa** que representa o mapa em si. Nesse objeto existe uma matriz de Terrenos, cada um contendo informações como robôs naquele terreno, assim como cristais, bases, etc. Melhores explicações sobre esse objeto serão dadas mais a frente.
- **List<Robos>**: Objeto do tipo **List** usado como uma lista de objetos do tipo **Robo**. Essa lista contêm todos os robos em jogo.
- **Tempo**: Uma variável inteira representando o número de rodadas já passadas. O valor inicial dessa variável é 0 e, a cada rodada, é incrementada em 1.

Principais métodos dessa classe:

- **Arena(Mapa)**: Construtor da classe. Recebe uma mapa, e retorna uma arena vazia, ou seja, com nenhum robo, o tempo é iniciado com 0, e o mapa passado é definido como o mapa da arena.
- **void atualiza()**: Função que faz a Arena executar uma rodada. Atualmente, essa função percorre a lista de robôs de forma aleatória, executando em cada robô uma instrução de máquina. Em caso de conflito de instruções, tem preferência o robô que for executado primeiro naquela iteração.
- **void sistema(Operacao)**: Função chamada pelos robôs para fazerem chamadas ao sistema. Toda vez que os robôs tentam executar uma ação que influencia seu exterior, ele só pode fazê-lo chamando esse método. O objeto do tipo **Operacao** recebido descreve qual operacao está sendo executada, e qual objeto Programável está tentando executar aquela ação. Dependendo da operação, esse método mexe na pilha do robo, e executa as ações pedidas, caso possível.
- **void insereExercito(Programa[ ],Posicao[ ],int)**: Função para inserir um exército de robôs na Arena. Para isso, recebe três argumentos:
  - **Programa[ ]**: Um vetor de objetos do tipo Programa. Para cada programa, será criado um robô a ser inserido na Arena.

+ **Posicao[ ]**: Um vetor com posições x,y indicando onde cada robô criado deve ser posicionado, seguindo a odem do vetor de programas.
+ **int time**: Um inteiro que indica a qual time esse exército pertencerá.

4.2 Classe Maquina

Um objeto da classe Máquina possui os seguintes atributos:

- dados Objeto do tipo Pilha. É a pilha de dados da maquina.
- mem Vetor de Empilhaveis usado para o robo armazenar constantes.
- prog Objeto do tipo Programa. É o vetor de comandos que aquela máquina executará.
- vars Objeto do tipo variáveis. Guarda o estado atual das váriaveis de determinada máquina.
- arena Referência para a Arena para a realização de chamadas ao sistema.
- index Inteiro que guarda o índice do próximo comando do programa a ser executado.
- obj Objeto do tipo Programável. À princípio é apenas uma referência para o robô que contém aquela máquina.

Construtor: Recebe o sistema (arena) com o qual a máquina vai se comunicar e o Programável que a possuíra. Mais uma vez, nesse ep, Programaveis são sempre robos. Inicializa mem com tamanho 10 e index setado em 0(zero). Funções mais importantes: void executaProx(): executa o comando cmd da posição index do programa. Para isso chama a função executaCmd(cmd), caso o programa já não tenha chegado ao fim.

int executaCmd(Comando cmd): Executa o comando passado como argumento. Verifica qual o código do comando e realiza a devida ação. Retorna o índice do próximo comando a ser lido. O sistema é chamado por meio da função sistema(), caso o comando passado precise de permissão da arena(i.e, se for ATK, COLLECT, DROP ou WALK).

void sistema(Comando cmd): Pede para o sistema executar o Comando cmd.

As outras funções são, em geral, auxíliares e estão bem explicadas nos comentários.