Relatório - EP Fase 4
Laboratório de Programação 2

9 de dezembro de 2013

1 Integrantes

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

2 Introdução

Nesta fase foram implementados o compilador da linguagem de alto nível do jogo, algumas instruções adicionais para melhor desenvolvimento dos robos através do nosso *Assembly*, e foram implementados melhoramentos de certas instruções. Este relatório tem como objetivo dar uma visão geral de todas essas mudanças.

3 Compilador

Optamos por adequar o código do professor ao nosso assembly. Em alguns casos foi necessário incluir instruções novas ao nosso assembly, como é o caso da instrução LRCL, LSTO e outras. O compilador está no arquivo *src/Parser.jj*. Na linguagem de alto nível, além das funcionalidades que já haviam no compilador do professor, também temos os seguintes métodos:

OBS: Todas as funções abaixo retornam zero, caso os argumentos não sejam do tipo certo.

- **move(int)**: Move o robô para a direção passada, caso esteja entre 1 e 6 e a Arena permita o movimento do robô.

- **look(int)**: Retorna o terreno da direção passada, caso esteja entre 1 e 6 e a Arena permita a ação.
- **getRobo(int)**: Retorna o robo da direção passada, caso exista um robô lá.
- **look(int)**: Retorna o terreno da direção passada, caso esteja entre 1 e 6 e a Arena permita a ação.
- **myTime()**: Retorna o objeto time do robô que está executando a ação.
- **atk(int)**: Executa um ataque na direção recebida. Caso haja um robô nessa direção, o mesmo perde 10 de vida(por padrão).
- **drop(int)**: Caso o robô esteja carregando um cristal, abandona-o na direção dada, caso esta aponte para a base inimiga. Se não for, o cristal volta para seu depósito de origem.
- **collect(int)**: Pega o cristal da direção dada, caso haja cristal no terreno daquela direção. Lembre-se que cada robô só pode carregar um cristal.
- **getTime(Robo/int)**: Retorna um objeto time. O argmento pode ser tanto um robô quanto um inteiro. No segundo caso, retornará o time cujo ID é o inteiro.
- **timeId(Time)**: Retorna o ID do objeto time passado.
- **timeRb(Robo)**: Retorna o objeto time do robô passado.
- **getX(Posicao)**: Retorna a coordenada x de um objeto posição.
- **getY(Posicao)**: Retorna a coordenada y de um objeto posição.
- **getPos(Terreno)**: Retorna o objeto Posicao do terreno passado.
- **getBase(Time)**: Retorna a base do time passado.
- **dropBomba()**: Coloca uma bomba na posição atual do robô. Esta explodirá daqui a 5 rodadas. TODOS ao redor tomam 40 de dano, portanto cuidado.
  obs: dropBomba() ainda está em versão pré-alfa.

- **x isa y**: Retorna 1 caso x seja do tipo y. y tem que ser uma das palavras reservadas declaradas no final do compilador. Ex: x isa ROBO, x isa BASE, etc..
- **x mod y**: Retorna o resto da divisão de x por y.

4 Interface Programável

Nessa fase começamos a usar a interface Programável. Ela define quais objetos podem ter comportamento programável. Por enquanto apenas Míssil é programável(Robo também é, mas não usamos), e este foi implementado de última hora. Portanto o objeto míssil ainda está em versão "alfa".

5 Compilação e Execução

Compilação e execução devem ser feitas dentro da pasta principal, isto é, pasta que contém as pastas bin, dist, etc... Para compilar, execute o seguinte comando:

```
$> make
```

OBS: supomos que o javacc está instalado.
Com isso, o jogo será compilado, e será gerado um arquivo .jar localizado na pasta dist/. Assim, para executar o jogo, a sintaxe é:

```
$> java -jar dist/EP4.jar 1 robo1 robo2 ... roboN 2 robo1 robo2 ... roboM
```

Por exemplo:

```
java -jar dist/EP4.jar 1 roboEzio 2 roboAndador
```