Relatório II – Laboratório de Arquiteturas Cognitivas (IA941A)
Danilo Cominotti Marques – RA 208986

Introdução

Este relatório contém as instruções de uso do programa 'DemoJSOAR', que foi customizado conforme os requisitos da atividade, e as respostas solicitadas nos requisitos.

Pré-requisitos

- Oracle Java versão 8 deve estar instalado e configurado no bash para ser executado com o comando 'java'.
- O WS3D (World Server 3D) não deve estar em execução, pois o mesmo será automaticamente inicializado pelo programa.

Instruções de Uso

Extraia o conteúdo do arquivo compactado para uma pasta de destino de sua escolha. Depois, abra a pasta de destino no terminal e execute o arquivo run.sh por meio do comando

sh run.sh

Alternativamente, execute diretamente o arquivo JAR do programa por meio do comando

java -jar DemoJSOAR.jar

Por fim, aguarde alguns instantes até que o WS3D seja inicializado e a janela "DemoJSOAR" da GUI (Graphical User Interface) referente ao MindView da criatura apareça.

Ajustes Implementados

Para atender aos requisitos da atividade, foram implementados os comandos e operadores referentes às ações de Hide e Stop. O comando/operador Hide foi implementado de modo que a criatura esconda todas as joias de cores que não fazem parte de seu leaflet e também as joias de cores que fazem parte mas cuja quantidade necessária já foi obtida. Desse modo, a criatura irá obter somente as joias de que realmente necessita e, ao mesmo, tende a obter uma vantagem sobre seus possíveis oponentes, já que esconderá joias que podem interessar a eles. Por sua vez, o comando Stop, utilizado por um dos operadores de Halt, foi implementado de modo que a criatura pare assim que o objetivo de seu primeiro leaflet seja atingido.

Conclusão

Do loop principal de simulação do DemoJSOAR (executado após a inicialização do ambiente, da interface de comunicação com o SOAR e da visão da mente da criatura), pode-se descrevê-lo como responsável pela execução dos seguintes passos: 1) se o modo debug não estiver ativado, isto é, se a visão da mente da criatura não estiver pausada na respectiva GUI, será executado um ciclo completo de simulação no SOAR; e 2) a visão da mente da criatura será atualizada com os dados do input-link, provenientes do ambiente WS3D, e do output-link, provenientes da simulação do SOAR. Por outro lado, caso o modo debug esteja ativado, o estado da criatura será meramente atualizado e a thread de execução será pausada momentaneamente para então verificar novamente o estado do modo debug.

Do acesso ao WS3D pela classe SoarBridge, pode-se dizer que na preparação do input-link, a qual é realizada pelo método prepareInputLink(), é feita a leitura do estado da criatura, dos objetos em seu campo de visão, etc. a partir de uma referência para um objeto do tipo Creature do WS3DProxy, sendo essas informações colocadas no input-link do SOAR. Por outro lado, o método processOutputLink() lê o output-link do SOAR e gera uma lista de comandos reconhecidos como ações passíveis de serem executadas pela criatura a partir da mesma referência para um objeto do tipo Creature do WS3DProxy, sendo a execução de cada um desses comandos solicitada no método processCommands().

Por fim, do princípio lógico das regras originais contidas no arquivo soar-rules.soar, pode-se dizer essencialmente que 1) a criatura deve girar até que uma comida ou joia entre em seu campo de visão; 2) a criatura vai registrar em sua memória o objeto encontrado e vai movimentar-se de encontro a ele; 3) caso haja obstáculos no caminho, a criatura vai tentar desviar desses; 4) chegando ao objeto alvo, a criatura vai comê-lo, caso seja uma comida, ou pegá-lo, caso seja uma joia, e removê-lo de sua memória; 5) entre uma comida e uma joia, a criatura vai priorizar a comida; e 6) a criatura vai tentar repetir esses passos enquanto puder.