Aula 8, 9, 10 e 11 - Clarion: Controlando o WorldServer3D

Objetivo

O objetivo destes exercícios foi utilizar a arquitetura cognitiva **Clarion** para controlar uma criatura artificial no **WorldServer3D**, de modo análogo ao que foi feito com o **SOAR**. Assim, realizamos o estudos pelo guia do **Clarion** para entender o funcionamento dos diversos módulos e capacidades do **Clarion** e desenvolvemos uma instância da arquitetura **Clarion** para controlar a criatura.

Testes

Teste 1

No início não era possível saber se o leaflet estava completo. De última hora foi proposta uma solução pelo professor e a implementação de última hora foi adicionada e ficou funcional. Para a criatura parar a verificação é testar se todos os itens do **consolidatedLeaflet** estão com **collected >= required**.

Teste 2

Para este teste, é interessante iniciar primeiro a versão do Clarion para inicializar o ambiente do WS3D e em seguida inicializar o Soar.

Detalhes de Implementação

Solution files

Para simplificar a implementação em Windows (laptop pessoal) e Linux (faculdade), foram copiados arquivos com **- windows** no nome de projetos e soluções com pequenas correções de bibliotecas.

Debug control

Em **ClarionDemo.Main** existem duas variáveis para o controle de impressão de tela, *lastDebug* armazena a última data/hora de impressão e *debugInMilliseconds* armazena o tempo em milissegundos até a próxima. O resultado na tela é o combustível do agente e o leaflet consolidado.

Funcionamento

Segui o exemplo existente e trabalhei com o ACS (ver método *SetupACS*) utilizando regras fixas (*FixedRules*). Adicionamos no conjunto de regras as ações e sensores detalhados nas seções abaixo para controlar a criatura em busca do objetivo.

Itens adicionados

Temos as ações em *ClarionAgent.cs*:

- *LookForSomething:* procurar por algo no mapa, se não tiver algo no campo de visão
- *GoToJewel:* ir ao encontro de uma jóia
- *GoToFood:* ir ao encontro de uma comida
- *GetSomething:* pegar algo (jóia ou comida próximas)

Temos os sensores em *ClarionAgent.cs*:

- *JewelAhead:* indica presença de uma jóia (não necessariamente de alta prioridade devido o leaflet)
- *FoodAhead:* indica presença de uma comida
- *NothingAhead:* indica ausência de jóia ou comida
- *SomethingReady:* indica algo muito próximo sujeito a uma ação (comer ou capturar)

Processamento de informação sensorial

Primeiro criamos análises dos dados observados em *ClarionAgent.cs*:

- *somethingToGet*: alguma comida ou jóia próxima, isto é, distância <= 22.0
- *somethingIsSeen*: alguma comida ou jóia no campo de visão
- *foodIsSeen:* alguma comida à vista
- *jewelIsSeen:* alguma jóia a vista
- *fuel:* combustível atual da criatura (e o nível de alerta chamado FUEL_WARNING_LEVEL)

A lógica de análise pros níveis de ativação em *ClarionAgent.cs* é:

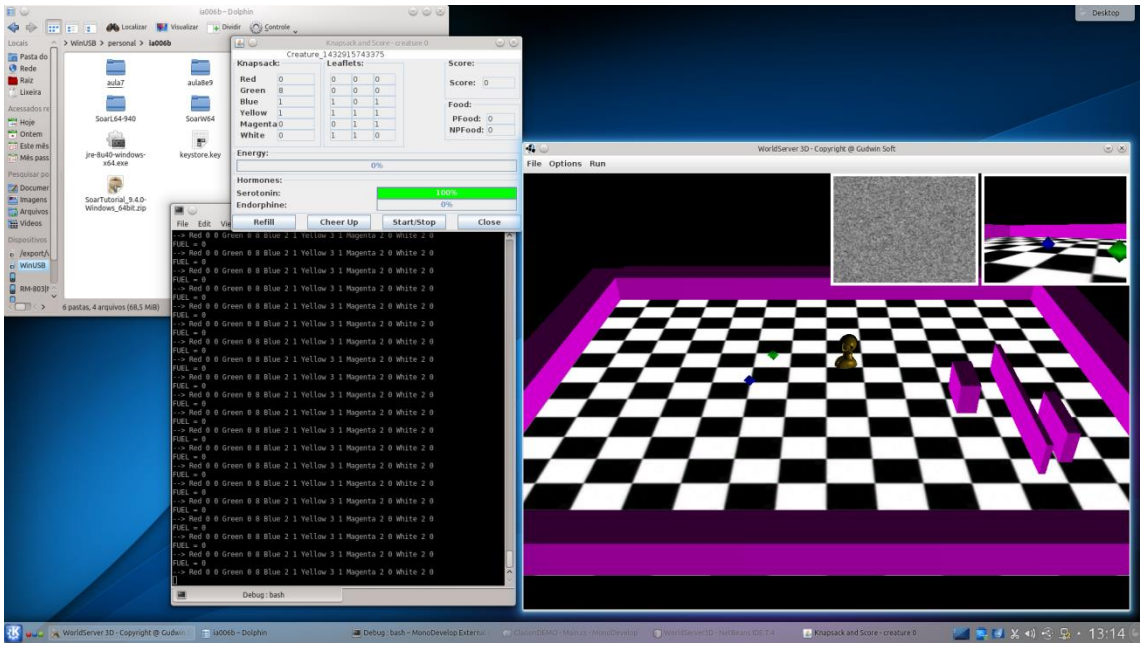
SE tem algo próximo ENTÃO coma comida/pegue jóia
SE campo de visão limpo ENTÃO rotacione
SE tem jóias e comida na visão ENTÃO siga a comida se precisar de energia, ou jóia
SE tiver ou jóia ou comida ENTÃO siga o que tiver

A lógica de seleção dos itens visualizados em *Main.cs* no método *agent_OnNewVisualSensorialInformation* é:

- *closestFood:* sempre a comida com menor distância da criatura que controlamos, caso a busca por comida seja selecionada
- *targetJewel:* sempre a jóis com maior prioridade. Definimos prioridade maior quanto mais falta itens para capturar e no mínimo a prioridade é zero, ou seja, mesmo que capturemos mais do que o necessário (para atrapalha o outro jogador) as jóias que estão completadas no leaflet tem a mesma prioridade e sempre menor do que jóias ainda não completadas.
- *somethingToGet:* qualquer comida ou jóia no caminho que esteja muito próxima, isto é, distância <= 22.0. Vale dizer que isso resolve o problema de focarmos a criatura em uma comida por ter o nível baixo e ela travar em uma jóia no meio do caminho, desta forma a criatura “tromba” na jóis, captura ela e continua caminhando pra comida. Outros casos semelhantes são resolvidos com este *approach*.

Imagem de funcionamento

Na imagem, observa-se que a criatura observa duas jóias e estabelece prioridade na azul que está mais distante se estar completada.


Desktop
Knapsack and Score - creature 0
Creature_1432915743375
Knapsack:
Red 0
Green 8
Blue 1
Yellow 1
Magenta 0
White 0
Leaflets:
Score:
Score: 0
Food:
PFood: 0
NPFood: 0
Energy:
0%
Hormones:
Serotonin:
100%
Endorphine:
0%
Refill
Cheer Up
Start/Stop
Close
WorldServer 3D - Copyright @ Gudwin Soft
File Options Run
Debug : bash
13:14