Exercício 3

No **Tutorial Project II**, vamos estudar um agente, "nosso herói sem medo", na "floresta dos hamburgers" que é uma matriz disreta de 10x10 células. As propriedades desse sistema estão abaixo e o *report* das atividades.

Project II Environment & Agent

Environment Attributes	Agent, evil monkeys, trees, rocks, hamburgers 10 x 10 grid (cells can contain zero or more objects) Monkeys move randomly and cannot enter cells with rocks or trees Agent cannot enter cells with rocks
Sensing	Objects in current and facing cell, health
Actions	Move forward, turn left, turn right, turn around, eat, flee
Agent Health	+ eat hamburger - attacked by monkey, moves into rock, moves out of bounds, time

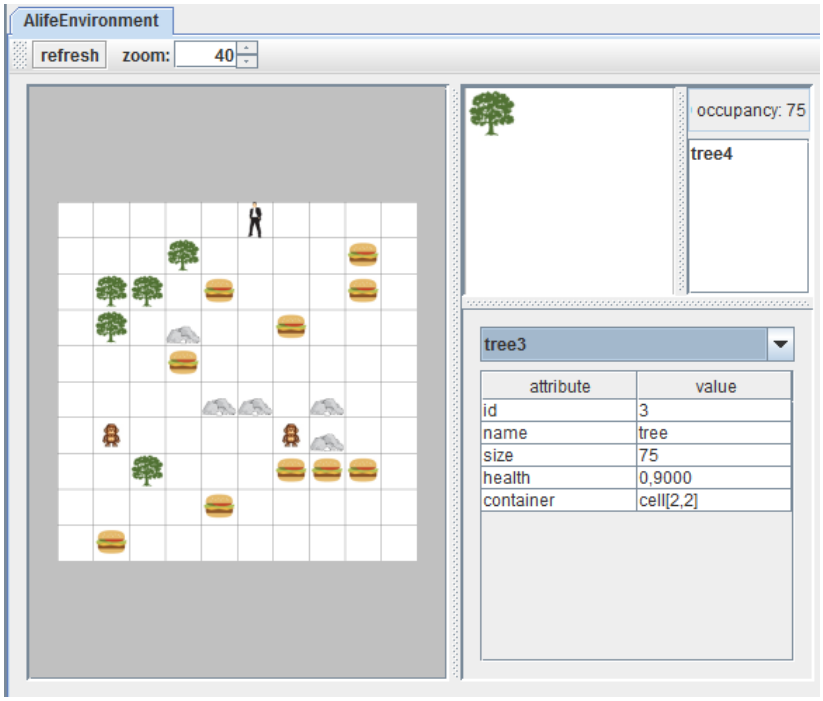
Environment Icons

Agent	
Monkey	
Food	
Tree	
Rock	
Multiple Objects	

Agent Exercise 1

TAREFA 1

A primeira atividade solicita uma exploração do **GuiPanel**. Observamos que ao clicar em uma célula aparece do lado direto o conteúdo (na imagem, **tree4**). Também temos uma combobox dos elementos e os atributos deles (na imagem **tree3**).

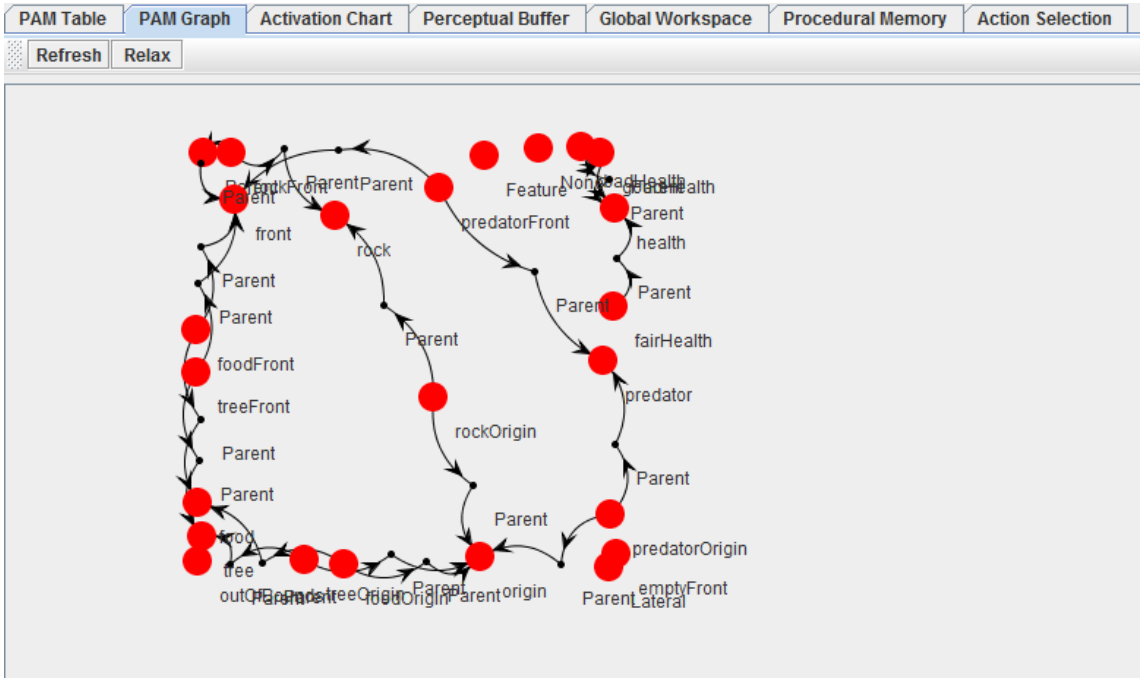


STUDY QUESTION 2.1: Compare the GUI of the alifeAgent to that of the basicAgent. What are the differences, and why are they different? As propostas são diferentes e os ambientes são diferentes para melhor permitir explorar o domínio de problema. Como aprendido no Tutorial Project I, a GUI é configurável.

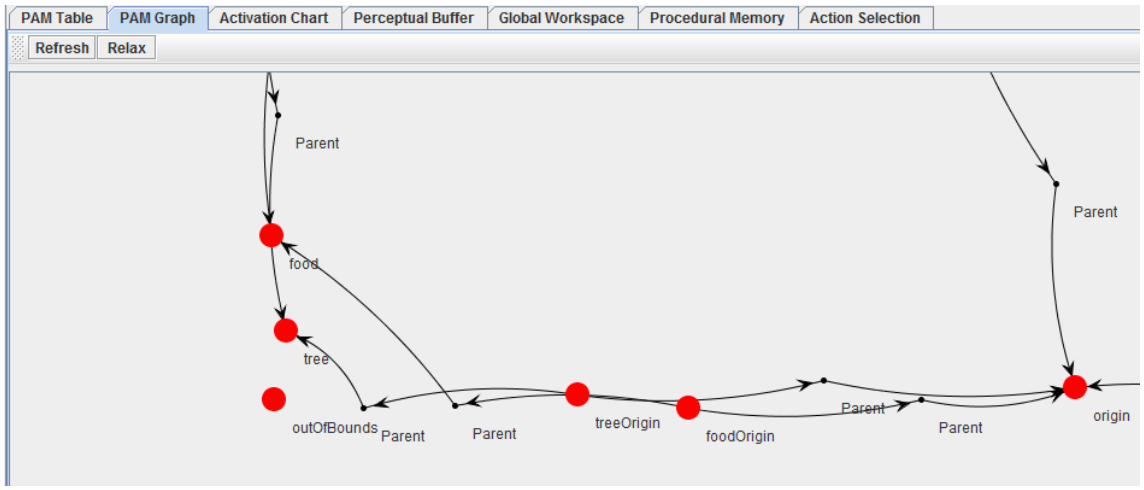
TAREFA 2

Na segunda atividade, exploramos o **PAM Graph.** O botão **relax** espalha os elementos, e podemos fazer zoom in e zoom ou com o mouse para explorar as relações entre links e nodes.

[visão com zoom out]



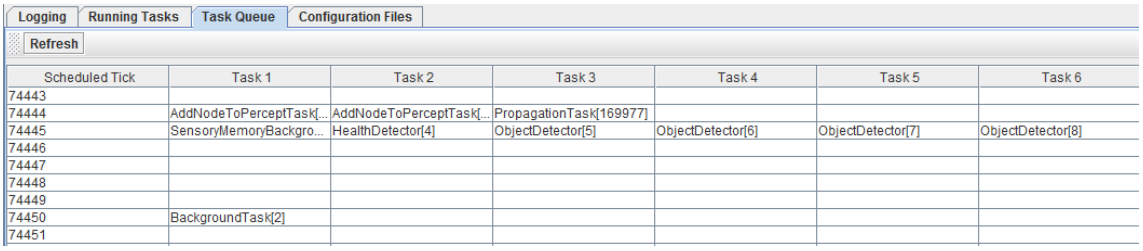
[visão com zoom in]



TAREFA 3

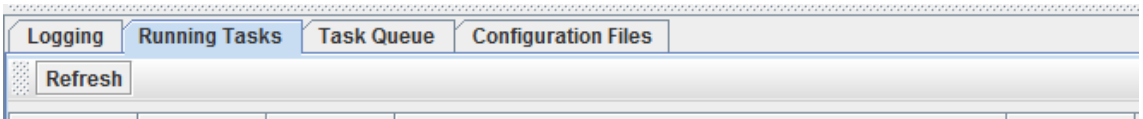
Na Terceira atividade, iniciamos a simulação como solicitado e analisamos a forma que o agente age. O agente fica no mesmo local, move-se para fugir dos macacos ou quando a saúde estiver baixa e come o hamburger caso encontre. Analisamos as abas **Task Queue** e **Running Tasks** em diversos momentos (ticks) como solicitado.

[Task Queue]



Scheduled Tick	Task 1	Task 2	Task 3	Task 4	Task 5	Task 6
74443						
74444	AddNodeToPerceptTask[...	AddNodeToPerceptTask[...	PropagationTask[169977]			
74445	SensoryMemoryBackgro...	HealthDetector[4]	ObjectDetector[5]	ObjectDetector[6]	ObjectDetector[7]	ObjectDetector[8]
74446						
74447						
74448						
74449						
74450	BackgroundTask[2]					
74451						

[Running Tasks]



Task ID	Activation	Status	Description	Next sched...
0	0,0100	RUNNING	EnvironmentBackgroundTask[0]	74600
2	0,0100	RUNNING	BackgroundTask[2]	74450
3	0,0100	RUNNING	SensoryMemoryBackgroundTask[3]	74445
4	0,0100	RUNNING	HealthDetector[4]	74445
5	0,0100	RUNNING	ObjectDetector[5]	74445
6	0,0100	RUNNING	ObjectDetector[6]	74445
7	0,0100	RUNNING	ObjectDetector[7]	74445
8	0,0100	RUNNING	ObjectDetector[8]	74445
9	0,0100	RUNNING	ObjectDetector[9]	74445
10	0,0100	RUNNING	ObjectDetector[10]	74445
11	0,0100	RUNNING	ObjectDetector[11]	74445
12	0,0100	RUNNING	ObjectDetector[12]	74445
13	0,0100	RUNNING	ObjectDetector[13]	74445
14	0,0100	RUNNING	ObjectDetector[14]	74445
15	0,0100	RUNNING	CueBackgroundTask[15]	74445

ALIfe Agent Exercise 2

TAREFA 0

Após alterar o **lidaConfig.properties** para utilizar o **alifeAgent_ex2.xml** e o **objects.properties**, observamos como o agente falha em mover-se quando tem energia baixa.

TAREFA 1

Criar um detector de saúde baixa, chamado **BadHealthDetector** e adicionar ao pacote **alifeagent.featuredetectors**.

BadHealthDetector.java

Source | History

```java
package alifeagent.featuredetectors;

import java.util.HashMap;
import java.util.Map;
import edu.memphis.ccrg.lida.pam.tasks.BasicDetectionAlgorithm;

public class BadHealthDetector extends BasicDetectionAlgorithm {

    private final String modality = "";
    private Map<String, Object> detectorParams = new HashMap<String, Object>();

    @Override
    public void init() {
        super.init();
        detectorParams.put("mode", "health");
    }

    @Override
    public double detect() {
        double healthValue = (Double) sensoryMemory.getSensoryContent(modality, detectorParams);
        double activation = 0.0;
        if (healthValue <= 0.33) {
            activation = 1.0;
        }
        return activation;
    }
}
```

TAREFA 2

Adicionar o novo detector ao arquivo **factoryData.xml**

```xml
<!-- INSERT YOUR CODE HERE ************************** -->
<task name="BadHealthDetector">
    <class>alifeagent.featuredetectors.BadHealthDetector</class>
    <ticksperrun>3</ticksperrun>
    <associatedmodule>SensoryMemory</associatedmodule>
    <associatedmodule>PerceptualAssociativeMemory</associatedmodule>
</task>
<!-- ********************************************** -->
```

TAREFA 3

Adicionar o novo detector ao arquivo **alifeAgent_ex2.xml**

```xml
<task name="BadHealthDetector">
    <tasktype>BadHealthDetector</tasktype>
    <ticksperrun>3</ticksperrun>
    <param name="node" type="string">badHealth</param>
</task>
```

TAREFA 4

Adicionar uma tarefa com nome **predatorFrontDetector** com modelo semelhante à **predatorOriginDetector** como indicado no tutorial.

```
<task name="predatorFrontDetector">
    <tasktype>ObjectDetector</tasktype>
    <ticksperrun>3</ticksperrun>
    <param name="node" type="string">predatorFront</param>
    <param name="object" type="string">predator</param>
    <param name="position" type="int">1</param>
</task>
```

Ao salvar e executar, observa-se o agente mover-se quando a saúde está baixa no limiar estabelecido 0.33 e ele foge quando o macaco está na célula adjacente à sua frente.

STUDY QUESTION 2.2: Why does the agent's behavior change in this way after the new feature detectors were added? As respostas aos estímulos de saúde baixa e predadores próximos estavam na memória procedural mas não tinham detectores, ou seja, o agente não percebia.

ALIfe Agent Exercise 3

TAREFA 0

Após alterar o **lidaConfig.properties** para utilizar o **alifeAgent_ex3.xml** e o **objects.properties**, observamos como o agente falha em reagir aos predadores.

STUDY QUESTION 2.3: What are the possible reasons the agent doesn't flee? Neste exercício, o agente não possui codelet de atenção transmitindo a informação percebida para a consciência do agente.

TAREFA 1

Criamos a tarefa de nome **predatorAttentionCodelet** no módulo de codelets segundo as configurações indicadas pelo tutorial.

```
<task name="PredatorAttentionCodelet">
    <tasktype>NeighborhoodAttentionCodelet</tasktype>
    <ticksperrun>5</ticksperrun>
    <param name="nodes" type="string">predator</param>
    <param name="refractoryPeriod" type="int">5</param>
    <param name="initialActivation" type="double">1.0</param>
</task>
```

STUDY QUESTION 2.4: How might this affect the agent's cognition and behavior? Com a informação percebida na consciência permite uma reação.

TAREFA 2

Alterando a configuração de **initialActivation** para o **FoodAttentionCodelet=0.01**, o agente deixa de comer a comida que encontra.

STUDY QUESTION 2.5: How does this parameter change affect the agent's cognition? A sensibilidade de percepção para comida é reduzida e o valor muito pequeno faz com que o agente ignore a comida.

TAREFA 3

Alterando a configuração do GoodHealthAttentionCodelet no parâmetro refractoryPeriod=10, observa-se que a frequência de coalizões formadas com **goodHealth** aumentam consideravelmente.

ALIfe Agent Exercise 4

TAREFA 0

Após alterar o **lidaConfig.properties** para utilizar o **alifeAgent_ex4.xml** e o **objects.properties**, observamos como o agente fica parado até que sua energia esteja abaixo de 0.66, apesar de ainda reagir aos macacos.

TAREFA 1

Segundo a atividade, alteramos o arquivo **alifeAgent_ex4.xml** no módulo **ProceduralModule** na tag **scheme.10b** para implementar a ação **action.moveAgent**. Ao executar, observamos que o agente passa a se mover mesmo com **saúde > 0.66**.

```
<param name="scheme.10b">if emptyFront move forward|(emptyFront)()|action.moveAgent|()()  |0.1</param>
<taskspawner>defaultTS</taskspawner>
```

TAREFA 2

Nesta atividade o tutorial sugere que façamos uma customização do inicializador de elementos da PAM. Para isso, acessamos o arquivo **alifeAgent_ex4.xml** no módulo **PerceptualAssociativeMemory** na tag **initializerclass** para configurar o valor **alifeagent.initializers.CustomPamInitializer**. Agora, no arquivo já existente no pacote **alifeagent.initializers** adicionamos a configuração solicitada.

```
child = pam.getNode("food");
pam.addDefaultLink(factory.getLink(child, objectNode, PerceptualAssociativeMemoryImpl.PARENT));
```

TAREFA 3

Nesta atividade o tutorial sugere que façamos um ajuste no tempo de decaimento da ativação de um **node** para a estratégia **slowDecay**. Tipos de decaimento estão disponíveis no arquivo **factoryData.xml**.

```
DecayStrategy decayStrategy = factory.getDecayStrategy("slowDecay");
objectNode.setDecayStrategy(decayStrategy);
```

Ao salvar o arquivo e executarmos, concluimos que quando o objeto **Node** é detectado a ativação ficará alta (1.0) por um período mais longo em comparação com os nós pedras e comidas.

Advanced Exercise 1

O agente não toma uma ação específica quando encontra uma árvore que é muito importante pois o protege dos macacos malvados. O exercício proposto pelo tutorial precisa das seguintes alterações em **alifeAgent_ex4.xml**:

- Adicionar uma **task codelet** no módulo **AttentionModule** para detectar a árvore

```
<task name="TreeFrontAttentionCodelet">
    <tasktype>NeighborhoodAttentionCodelet</tasktype>
    <ticksperrun>5</ticksperrun>
    <param name="nodes" type="string">treeFront</param>
    <param name="refractoryPeriod" type="int">50</param>
    <param name="initialActivation" type="double">1.0</param>
</task>
```

- Adicionar um esquema **scheme** no módulo **ProceduralMemory** para a ação

```
<param name="scheme.11">if treeFront move forward|(treeFront)()|action.moveAgent|()()|0.1</param>
```

Advanced Exercise 2

Neste exercício o tutorial sugere que o módulo de seleção de ações seja alterado. A versão atual é muito simples. O tutorial sugere que uma nova seja criada, extendendo a **FrameworkModuleImpl** e implementando a **ActionSelection e BroadcastListener**. Abaixo vamos ver as alterações de configuração no alifeAgent.xml apesar de não contemplar completamente a implementação desejada.

- No módulo **ActionSelection** alterar a classe carregada

```
<module name="ActionSelection">
    <class>edu.memphis.ccrg.lida.actionselection.BasicActionSelection</class>
    <param name="actionSelection.ticksPerStep" type="int"> 10</param>
    <taskspawner>defaultTS</taskspawner>
</module>
```

- Na configuração de **listeners**, adicionar um novo **BroadcastListener** entre o **GlobalWorkspace** e o **ActionSelection**, como abaixo:

```
<listener>
    <listenertype>edu.memphis.ccrg.lida.globalworkspace.BroadcastListener</listenertype>
    <modulename>GlobalWorkspace</modulename>
    <listenername>ActionSelection</listenername>
</listener>
```