Tutorial 1

How to configure and run the Feed-Flee-Mate game

Introduction

In this tutorial the configuration possibilities available in the Feed-Flee-Mate game is discussed.

Configuration

Configuration is as of now performed in the *Configuration.java* file. This implies that you have to recompile this file every time you want your configuration to be used by the game. In the *Configuration.java* file you can add agents you want to be in the game from the very beginning as well as configure the general agent characteristics with more.

All the constants used in this file have self explanatory names. However, comments are provided to further spell out their meaning.

Run the Game

In order to run the game, simply invoke the “main” method in *Main.java*. This will initialize your game with the settings and agents provided in the *Configuration.java* file.

Tutorial 2

How to create an agent in the Feed-Flee-Mate game

Introduction

In this tutorial a basic, non-intelligent, agent for the Feed-Flee-Mate game is developed and put into the game. The agent is called KillerAgent and the source code for this agent is available in the *yaes.application.eel6938.classagents* namespace. This tutorial will not cover details on how the intelligence is implemented for the KillerAgent.

The process of creating a user defined agent consists of 6 steps:

1. Create a new Java file with an appropriate name for your agent (*KillerAgent.java*, *FleeAgent.java*, etc.).
2. Inherit, by using the Java keyword *extends*, from the Agent class located in the game.objects namespace.
3. Create the following functions in your agent class:
    a. String getName()
    b. int[] getColor()
    c. void intelligence()
4. Write your custom made intelligence.
5. Update the Configuration.java file with your newly created agent.
6. Compile and run!

Step 1

Create a new Java file. Place this file in the *yaes.application.eel6938.classagents* namespace. I name my file *KillerAgent.java*.

Step 2

Inherit from the Agent class located in the *yaes.application.eel6938.game.objects* namespace.

```
package yaes.application.eel6938.classagents;
import yaes.application.eel6938.game.Constants;
import yaes.application.eel6938.game.objects.Agent;

public class KillerAgent extends Agent {

}
```

Step 3

Implement the abstract function required by the Agent class. These are:

- *String getName()*
- *int[] getColor()*
- *void intelligence()*

getName() - should return the name you want to display for your agent.
getColor() - should return an array of integers describing the RGB components of the color you want to display for your agent.
intelligence() - should contain the intelligence for your agent. This function is called every cycle of the game.

```
package yaes.application.eel6938.classagents;
import yaes.application.eel6938.game.Constants;
import yaes.application.eel6938.game.objects.Agent;

public class KillerAgent extends Agent {

	public void intelligence()
	{

	}

	public String getName()
	{
		return "KillerAgent";
	}

	public int[] getColor()
	{
		int[] rgb = {255,0,0};
		return rgb;
	}
}
```

Step 4

Create the intelligence for your agent in the intelligence function you created in the previous step. The commands available are inherited from the Agent class. See the API

for more specific information about the available commands and helper functions. The command set currently consists of:

- Move
- Eat
- Flee
- Attack
- Mate

A simple random move behavior is created in the example below.

```
package yaes.application.eel6938.classagents;
import yaes.application.eel6938.game.Constants;
import yaes.application.eel6938.game.objects.Agent;

public class KillerAgent extends Agent {

	public void intelligence()
	{
		int speed = (int)(Math.random()*5);
		int direction = (int)(Math.random()*4);
		move(direction, speed);
	}


	public String getName()
	{
		return "KillerAgent";
	}

	public int[] getColor()
	{
		int[] rgb = {255,0,0};
		return rgb;
	}
}
```

Please refer to the *KillerAgent.java* file in order to get details on how path-planning, sensor reading and other commands are being used.

Step 5

Update the *Configuration.java* file to indicate use of your newly created agent. Simply open the *Configuration.java* file and append your agent to the Agents array. In the example below three KillerAgent agents have been added to the array.

```
public final static String [] Agents =
{
        "KillerAgent",
        "KillerAgent",
        "KillerAgent"
};
```

Step 6

Compile the files and run the main function in *Main.java*.