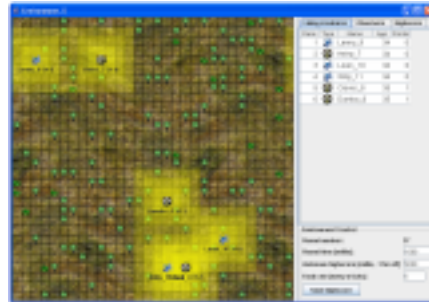


Examples

The Jadex software includes several small to medium size examples demonstrating different aspects of BDI agent programming. In the following the most interesting of them are described shortly. If you have [Java Webstart](#) installed, you even can launch the examples directly from your browser.

Hunter Prey

The hunter prey scenario consists of two kinds of creatures living in a grid world. The basic task of hunters is to chase, while preys move around looking for food. Both kinds of creatures have to act autonomously in the environment on basis of their current local view, experiences made in the past and communications with others. Besides hunter and preys the environment accomodates other passive world objects. On the one hand there are trees on many squares that prohibit creatures running on such fields and on the other hand little plants grow at random squares at the map. These plants can be eaten by the preys if they are on the same field. The scenario is round-based with a fixed time slot for each round. This means that all creatures in the world have to issue their next action (moving to some adjacent square or eating something on the current square) with that round time. If no action is announced no action will be executed. The environment will decide in each round if an action succeeds or fails.



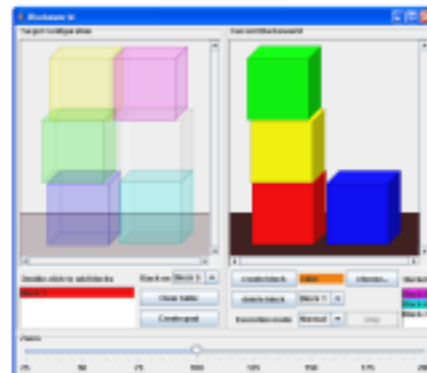
BlackJack

This example allows for playing BlackJack with one or more computer opponents. Basically, it consists of three different agent types: the manager, the dealer and the player. The Blackjack-Manager can be used to start the game. It shows up with a user interface that can be used to start and stop the local dealer as well as local players. The user may also customize the players, i.e. give them a name and an initial account, choose a strategy and assign a different color. When a player agent is started it will search a dealer to play a game. After at least one player has registered at the Dealer the dealer starts playing games. The player may stay at the table as long as it owns money for playing.



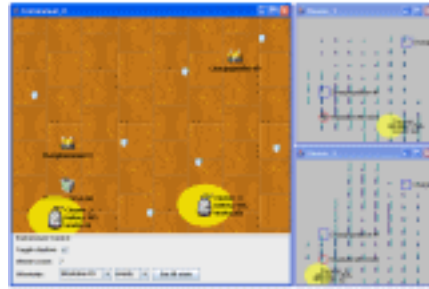
Blockworld

In the blockworld coloured blocks are placed on a table as towers of single blocks. This example only consists of a single agent, which has an internal representation of the blockworld. When the agent is started, a window is opened that shows the current state of the blockworld, and another panel where you can construct a desired target configuration. When the "create goal" button is clicked, the agent tries to achieve the target configuration by restacking blocks on the table. For this purpose a simple solution strategy is applied. Basically, all bad blocks are put onto the table and are stacked on towers that fit the target configuration.



Cleanerworld

The cleanerworld is based on the idea that an autonomous cleaning robot has the task to clean up dirt in some environment. In our scenario of the cleaner world the main system objectives are to keep clean a building at day, e.g. a museum, and to guard the building at night. To be more concise we think of a group of cleaning robots that are located in the building and try to accomplish the overall system goals by pursuing their own goals in coordination with other individuals. Therefore, three key goals for an individual cleaning robot were identified. First, it should clean its environment at day by removing dirt whenever possible. The cleaning robot therefore has to pick-up any garbage and carry it to a near waste bin. Secondly, it has to guard the building at night by performing patrols that should be based on varying routes. Any suspicious occurrences that it recognises during its patrols should be reported to some superordinated authority. Thirdly, it should keep operational by monitoring its internal states such as the charge state of its battery or recognised malfunctions. Whenever its battery state is low it has to move to the charging station.



Garbage Collector

The garbage collector example is a modified and simplified version of the cleaner world scenario. It consists of two kinds of agents in an environment covered with garbage. Collector agents search for dirt and if found bring it to one of the burner agents. The burner agents do not move. They pickup garbage at their position and burn it. The environment in this example is represented as simple grid world. All agents can only see things that are situated in their currently taken position. For this reason the collectors simply run in a predefined pattern through the environment and try to pickup garbage as soon as they reach a dirty square. If a collector managed to pickup a piece of garbage it heads directly to one of the garbage burners and lays down the garbage. Thereafter it returns to the point it has found the waste and continues its search for garbage from there.



Marsworld

Several interacting agents have the task to explore the environment for ore resources and bring as much ore as possible to the agents homebase. When the mission time has expired the agents have to abort their current actions and return to the homebase. The different agent types include sentry agents, production agents and carry agents. The sentry agent has the task to find ore resources inspect them if they can be exploited. Therefore the sentry agent has the greatest vision of all agent types. To find the ore resources more quickly all other agents report to the sentry about resources they explored. The production agent is called to a target from a sentry to produce as much ore as the capacity of the resource permits. When finished the agents calls for carry agents to bring the ore to the homebase. The carry agent has the task to bring ore from targets to the homebase. It is called by the production agent.



Puzzle

This example is adapted from the commercial agent platform JACK(TM) from Agent Oriented Software. The Jadex implementation is very similar to allow performance measurements between both platforms. The example shows a puzzle game played by one agent. It consists of a board with white and red pieces. The objective is to switch the positions of both kinds of pieces whereby the following rules for making a move exist:

- white pieces move right or down to an adjacent free field.
- white pieces jump right or down over a red piece to a free field.
- red pieces can only move up or left with the same restrictions as white pieces.
- the color of a piece to move is not specified.

The agent uses meta-level reasoning to solve the puzzle. It creates a goal to make a move and find the solution. For this goal a plan for each possible move is created. To decide which move plan to test first a meta-level goal is created. The choose move plan handles the meta goal and decides according to a specified strategy.

