VERTIEFUNG

Creating a Reference Agent for the SiMA Agent a Benchmark

Submitted at the
Faculty of Electrical Engineering and Information Technology,
Vienna University of Technology

under supervision of

Dipl. –Ing. Alexander Wendt
Institute number: 384
Institute of Computer Technology

by

Bsc. Adam Cziko
Matr.Nr. e1427866
Romanogasse 12-14
1200, Wien

23. 01. 2016

_____

Abstract

The presence and importance of artificial intelligence is significantly rising. At the Technical University of Vienna, there is a project, called SiMA, which is focusing on developing a new approach for artificial intelligence. Its name stands for Simulation of the Mental Apparatus & Applications. The development of the SiMA agent shall be verified. For this purpose, a reference agent shall be provided. In the project of this documentation the mentioned reference agent is developed. This new agent will provide a benchmark in a selected game. The game was developed at the university and it is a last man standing, Pacman like game. The goal of the referent agent is to be as effective in this game, as possible. The chosen cognitive architecture for the agent is the LIDA, the Learning Intelligent Distribution Agent. The game is analysed and a strategy is built up. The decision making of the agents is developed and explained. One of the main part of the solution is to move in a swarm and overwhelm all the enemy agents. The swarms' desires are preferred to the desires of one given agent. This strategy is proved to be successful against the other agents. This strategy was mapped to the LIDA Framework. The framework provides a solution for implementing the LIDA Model. The game and the LIDA Framework is connected via a software gateway. The new reference agent was successfully implemented in the LIDA Framework and tested in the game. The initial goal to provide a benchmark has been achieved.

Acknowledgements

I would like to thank my supervisor, Alexander Wendt who supported me, whenever I needed his help and made it possible to deliver my work.

Table of contents

1. Introduction

The presence and importance of artificial intelligence is significantly rising. This trend can be seen on many fields of life. Parallel with it, the already existing implementation methodologies and architectures of the artificial intelligence is developing and changing. In the Technical University of Vienna, there is a project, called SiMA [1], which is focusing on developing a new approach for artificial intelligence. SiMA stands for Simulation of the Mental Apparatus & Applications.

1.1 Motivation

The project has two main motivations. The main motivation is creating a benchmark for the SiMA agent in a given scenario in order to measure its success. As a new approach is implemented, it shall be checked against other solutions. Moreover, with creating this benchmark, a deeper look into connecting a simulation environment with an artificial intelligence framework can be made. On the other hand, experience is gained in computing artificial intelligence.

1.2 Problem Statement

The main problem statement is that how to develop an agent that can win most, or nearly all of the games in a Pacman like game compared to the current state of the SiMA. This agent must be operating that good, that it can be a real benchmark for the SiMA agent.

1.3 Task Setting

The main goal of the project is to develop and implement an agent that wins as many games as possible against the actual implementation of the SiMA agent. For achieving this goal, a proper strategy has to be created, which is more efficient than the strategy of the SiMA agent. Based on this strategy the benchmark agent for this game shall be created.

1.4 Methodology

The first step, is to gather knowledge in the field of artificial intelligence. Gain a good understanding of the basics and the design decisions for those. Parallel to it, the Java programming language shall be learnt. A good exercise for it is to analyse the game and get to understand the way it works. The second step during this work, is to define, what kind of cognitive architecture will be used for creating the benchmark agent. This architecture can be chosen from already existing solutions or can be created from scratch as well. The third step is to design and implement the agent. It must be developed further, till it is more successful in the game, than the other agents.

2. State of the Art and Related Work

At the very beginning of this documentation, the state of the art development and environment is analysed together with the already accessible related works.

2.1 Cognitive Architecture

At the beginning, the term agent shall be clarified. The agent as a term can be used in different fields of studies. As an example, the agent in economics can be mentioned.

2.1.1 Agent in Economics

An agent in economics is an actor and more specifically a decision maker in a model of the aspect of the economy. Typically, every agent makes decisions by solving a well or ill-defined optimization/choice problem. [OEC08, p. 2]

2.1.2 Agent in Artificial Intelligence

In artificial intelligence, an intelligent agent (IA) is an autonomous entity which observes through sensors and interacts with its environment using actuators. The goal of an agent is to direct its activity towards achieving predefined goals. And addition to it, the intelligent agents may also learn or use knowledge to achieve their goals. They may be very simple or very complex. As an example a reflex machine such as a thermostat is an intelligent agent. [RSNP03, p. 31]

The intelligent agent definition can be made more precise. As intelligent agents are often described schematically as an abstract functional system, one that is similar to a computer program. As a result of it, intelligent agents are sometimes called abstract intelligent agents (AIA) to distinguish them from their real world implementations as computer systems, biological systems, or organizations. Some other perspective of intelligent agents emphasize their autonomy, and so prefer the term autonomous intelligent agents. [RSNP03, p. 35]

With this, bearing in my, it can be stated, that the goal is to implement in software an autonomous intelligent agent.

2.1.3 Cognitive Architecture

For the implementation of the agent a structure is desired. One of the task is to decide, if the structure is created from scratch or an already existing one is used. For that, the existing structures, the so called cognitive architectures shall be analysed.

A cognitive architecture is a hypothesis about fixed structures that provide a mind, whether in natural or artificial system, and how they work together to yield intelligent behaviour in a diversity of complex

environments [4]. For the work, at the very beginning, the supervisor provided three main options for cognitive architectures. These were the LIDA, the SOAR or development from scratch.

Developing from scratch is extremely interesting, but hard and complex as well. Even the very basic problems of creating a new structure must be taken into consideration, as it must function properly in case of a very complex task as well. This part was neglected because of the deadlines.

One of the proposed architecture is SOAR. It was first created by the professors of the Carnegie Mellon University and now it is maintained at the University of Michigan. It is a general cognitive architecture for developing systems that exhibit intelligent behaviour. [5] SOAR will also not be described in detail, as after the first month of the project, a re-estimation for the required time for the project were made. Based on it, the third option, the LIDA have been taken. At the University, there is a lot more experience with LIDA that turned out to be the reason for selecting this cognitive architecture.

2.1.4 LIDA

The LIDA cognitive architecture is developed and maintained at the University of Memphis by the Cognitive Computing Research Group. The abbreviation stands for Learning Intelligent Distribution Agent. As in its name it is already stated, the agents are able to learn and use this knowledge. The predecessor of LIDA is IDA, the Intelligent Distribution Agent. It was developed by the USA Navy to fulfil some assignments that earlier were in the responsibility of humans. [2]

The LIDA Model is both a conceptual model of minds and a computational model that provides an architectural design for autonomous agents. [JRSS1, p. 2] It means, that LIDA Model is not just a composition of concepts, but it is also a mathematical model that requires resources to study the behaviour of the system via simulation. This model will be used within a framework during the fulfilment of the assignment.

To understand the way this model works, more details are needed. The model is based primarily on Global Workspace theory [JRSS1, p. 2] and implements and fleshes out a number of psychological and neuropsychological theories. The LIDA computational architecture is derived from the LIDA cognitive model. The LIDA Model and its ensuing architecture are grounded in the LIDA cognitive cycle. Every autonomous agent must frequently sample (sensor) its environment and select an appropriate response (actor). More sophisticated agents, such as humans, process the input from such sampling in order to facilitate their decision-making. This means making sense of the input and the decision based on that. The agent's "life" can be viewed as consisting of a continual sequence of these cognitive cycles. Each cycle consists of units of sensing, attending and acting. A cognitive cycle can be thought of as a moment of cognition, a cognitive "moment." During each cognitive cycle the LIDA agent first makes sense of its current situation as best as it can by updating its representation of its current situation, both external and internal. This is made by the Sensory Memory module and the Perceptual Associative Memory. By a competitive process, as specified by Global Workspace Theory, it then decides what portion of the represented situation is most in the need of attention. Which means that there are parallel running and competing modules, so called Codelets, which are about to seize the attention. This portion of attention is the current content of consciousness. Broadcasting this portion of attention enables the agent to choose an appropriate action from its Procedural Memory in

combination with its Action Selection Module and execute it via the Sensory-Motor Memory, completing the cycle. [JRSS1, p. 3] Learning also takes place with the broadcast. Learning will not be taken into consideration in this work. More detailed explanation will be done at the implementation part of the agent.

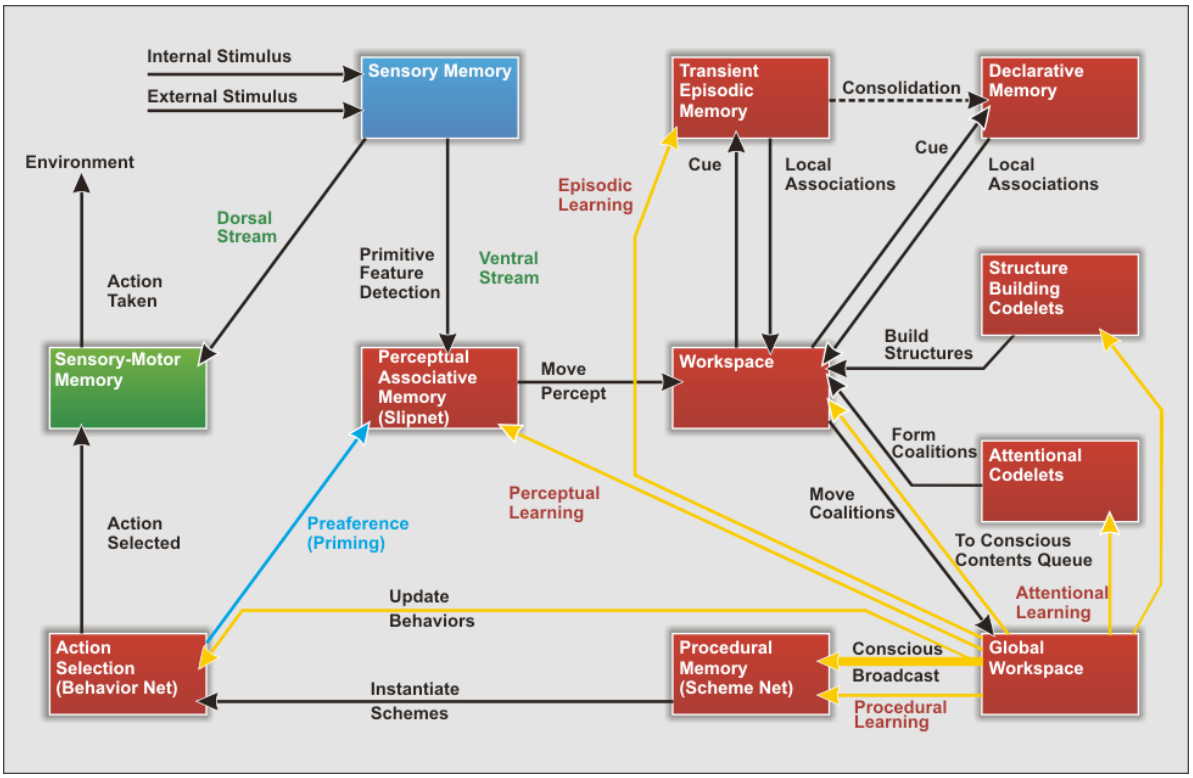

Figure 2.1: The LIDA Model Diagram [JRSS1, p. 3]

2.2 Tools and Development environment

As an intelligent agent is about to be developed, its environment and the game itself shall be taken as a basis. It is important to analyse the structure, as different environments results in different needs and the focus in this work is to develop a successful agent in one given environment. The game itself was developed at the Institute of Computer Technology via the supervisor, Dipl. –Ing. Alexander Wendt and some of his colleagues. This is a Pacman like, last man standing game. The game itself consists of 7 different projects in the context of an eclipse base development environment.

2.2.1 JGameGrid

The basis of the game is the JGameGrid Framework. It is a class library specially designed for programming courses with the focus of object oriented programming. The main purpose of this framework is to provide the student the possibility of game based learning. So in the game there are grids, where one object can take place and the number of grids makes up the world. The framework provide

a setup for defining the number of grids, just as well additional function, for example a background picture. [3]

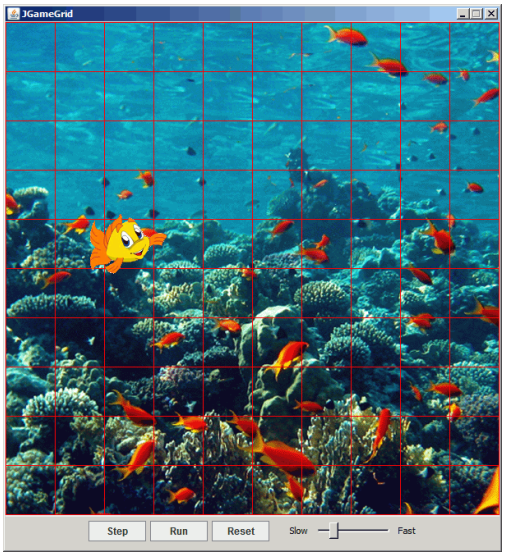

Figure 2.2 Screenshot of the JGameGrid

2.2.2 The Miklas Game

The main project from the seven is called the "Miklas". This is the backbone of the whole game. The different states and scenarios of the game are controlled via conditions. The configuration of the whole game is also happening here. The basis of the game configuration is outsourced from the eclipse environment itself, as there are separate .ini files for defining the setup. This setup then will be read and processed by the modules of this project. In general, it can be said, that the game is built a way that as much part as reasonably possible is not coded, but configured via the .ini files. The creation of the entities of the game are also within this project. An entity consists of two main parts, its body and its mind. The body is the so called physical representation of the entity in the games world, while the mind defines its behaviour based on its perception. The mind is the part of the game that controls the intelligent agent. Moreover, the states can be separately asked –for example health, position etc.- of one given entity via an evaluator module. The game engine and the games event handling are the modules that make the separate parts functioning together. Last but not least, there is a visualization module for the graphical user interface.

The "Miklas Events" project is responsible for the actions of a given entity. These action events are grouped into two the big groups. One group is responsible for the interactions with or on bodies. It can be the own body or another agents body. The other groups are simple actions, like commands for the body without interactions or changing position on the map.

The "Miklas External Minds" project is responsible for defining minds outside of the game. The external minds are provided with suitable interfaces for the whole game. This is the project, the mind of the agent is implemented as well. There are additional utilities for debugging purposes. The "Miklas Launcher" project is the one that starts each project together and makes the system run.

The Gameplay

The game is a last man standing game. The size of the word and the number and type of players, the agents' number can be freely configured. The goal is to attack and kill all the enemy agent. Agents can have different types. These different types can be allies or enemies. It means, that if there are 2 different type of agents, that are enemies, and during a gameplay only these two types of agents survive, then they have won the game. Each agent, regardless the type has the same area of grids in its vision, in which they can percept. This means, that none of them can see the whole map and none of them is capable of seeing more grids than any others. The agent can cooperate, if they are aware of each other, it is only depending on the agents mind implementation. The cooperation must be predefined in scenarios, as the agent are not allowed nor capable of communicating with other agent. At the beginning of the game, each agent has the same, maximum level of health. If any of them is attacked, then this health will be decreased. Each agent is capable of decreasing the same amount of health during one attack and each agent losses the same amount of health when it is once attacked. This means, that there are no different types of agent, which can attack or defend better than the others. If an agent runs into an obstacle then the agent is damaged. This damage has the same value as being attacked by another agent. Last but not least, the agents can increase their health. It is possible via eating a food object. The health can only be increased when the agents is already damaged, there is no possibility of gaining extra health.

Eclipse

The development environment used for the game is Eclipse. The version used is the Java EE IDE that stands for Java Enterprise Edition Integrated Development Environment. It provides all the needed functionalities for the development and can be downloaded for free from the official Eclipse website. [6]

2.2.3 The LIDA Framework

The other tool that is going to be used during the work is the LIDA Framework. The LIDA Framework software was developed by the Cognitive Computing Research Group in the Institute for Intelligent Systems at the University of Memphis. It is distributed under the Non-Commercial License Agreement. For more detail, the official site of the framework shall be visited. [7]

The LIDA framework is for creating a generic and configurable version of the domain independent modules and processes of LIDA implemented as a software framework in Java. It is easily customizable for different domains, for different environments. It permits changes to the implementation of each module and allows for XML definitions of the data structures, processes, parameters, and modules of the system. The framework embraces parallelism and employ multithreading. As a complex system, it provides users with tools to make it more manageable. A GUI display of the inner workings of the system, a logging of important events, and utilities to parse XML and Properties files for easy customization are each provided with the framework together. [7] The usage of the framework is very advised, as it supports to develop exceedingly complex systems with the proof of the right implementation structure. The developers only have to build up the inner part of a new module, as the backbone is already provided. The framework is modular, it means, that all modules from the LIDA Model –

introduced one chapter before - can be implemented, but it is not compulsory for creating a working system. [JRSS1, p. 5] This attitude will be visible during the implementation

NetBeans and Tutorial

The LIDA Framework is provided with a tutorial and a description together. The tutorial consists of different exercises on already created projects. These projects are created via NetBeans. It is an eclipse based Integrated Development Environment. This tool was used during the tutorial. These exercises helps to understand the system and the connection between the different modules in an interactive way. Completing this tutorial was a great help for being able to implement the agent in the framework.

3. Models and Concepts

The first step before implementing the intelligent agent is to create a strategy. This strategy is going to be the key for creating the benchmark agent. This strategy shall consider as many aspect of the games as possible to make it the best last man standing agent. The focus of mine was on thinking in a system of agents and trying to achieve the best result as a system. It means, that it can happen, that for one given agent the probability of surviving for a longer time could be higher with another attitude, but the goal was not to optimize one agents chance for survival, but to maximize the chance, that the only type of agent, surviving a gameplay is mine. It also means, that at first, it had to be defined on swarm level, what the goal is, and not in the other way. Described it with an example it would be as followed. What is better? Go and eat some food, what can perceived or stay in a swarm formation. This question will also be described in details.

Moreover, the other was the KISS, so to Keep It Short and Simple, or with another often used definition, as Keep It Simple Stupid. This tactics does not mean, that the agent is going to be stupid. It means, that the implementation of the tactics will be as simple and straight forward as possible, even if the reasoning behind it can be complex.

3.1 Basic Perception

For being able to interact with the environment and make the right decision, information shall be fetched from the environment itself. This is possible via the perception of the agent. The perception of the agent consists of a number of grids and the objects within a grid. The agents can percept in the x direction between +3 and -3, while in the y direction they are able to see from 0 till +4. The position is the grid with the coordinates of (0,0), which means, that they can see both the left and the right side of theirs, but they are not able to perceive what is behind them. Moreover, the perception can not be extended via communication with other agents, so this described area is all, what they can work with.

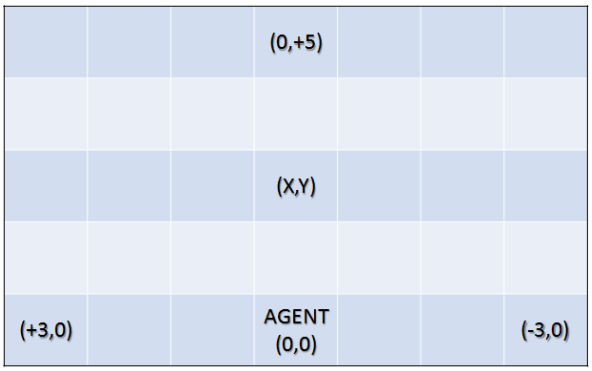

Figure 3.1 Area of the perception

3.2 Movement Pattern and Object Perception

The simplest case that has to be handled is the one, when the agent see an empty map ahead. In this case, the agent will follow a predefined order of movement. This is on purpose not a random movement as it is explained in the following use cases. This is the movement pattern, what is called cruising. This scenario of movement is consisting of moving forward, or left or right. This scenario can also happen, if more agents are together, but the one that is ahead will see only the map. There is no follow the leader option implemented for the agents behind. The agents will automatically move together, as they are following the same pattern of movement. This is also a part of the strategy that makes sense for the more complex use cases. This pattern can only change, when the agent starts to percept objects in its perception.

At first, it must be defined, what can be perceived and how these objects can be grouped together. An agent can percept for example another agent. As described in the games section, the agent is going to percept the body of the other agent, what is its physical representation. The bodies can be grouped in two big groups. The agent with the same body, will be the allies. All the other agent will be enemies and will be attacked. On the map, there are obstacle objects as well, for example walls. These objects must be taken care carefully, as bumping into them causes a decrease in health. The one additional object that can be perceived is the food. Taking care the food object can have a significant change on the tactics. It will described in this chapter.

Now, there is a perception and the object can be interpreted as well. It must be defined, that based on the perception, what the agents are willing to do. With these points, bearing in mind, the movement pattern can be reflected back. This predefined movement also has the advantage that the agent can cruise next to each other as well, what would not be possible with a following option, as the agents could easily lose the leader agent from the perception, when they have to take a turn because of an obstacle. Moreover, if they are following an agent, then the agent ahead will always be the one attacked for the first time. This will not be an efficient use of the agents. What is needed, is to have as many agent alive, as possible. More agent means more attacking power. If there is one agent ahead all the time then it will die first. With that, the group of agents have lost a whole unit of attacking power. If the one ahead is not predefined, but the agent are still moving in the same pattern, resulting in a swarm, then it is more likely, that they will have two agent with half of the life, that means the same amount of health loss, but they do not lose any attacking power this way. The special scenarios that can be handled, and shall support this swarm strategy are going to be described now.

3.3 Obstacle Ahead

If there is an obstacle ahead, then it must be avoided. It can be done via turning left, or turning right, and then moving forward. In the movement pattern for cruising, after a turn there is always a move forward action as well. So it is checked, if the agent is already turning left, or right. If yes, then there is nothing to do in addition. In case, that next movement would be forward, then the agent will jump to another step of the cruising pattern. This step will always be the turn left action. The agents will

simply prefer left to right. With this, the agents can be kept moving together at the same direction, forming a swarm.

3.4 Enemy Ahead or on the Side

If there is an enemy ahead, then the decision is very easy to made, the agent will simply start to attack it. If an agent, that is an enemy, standing directly next to the agent on the left or the right side, then it will turn to that direction. It will always be the same scenario, regardless the agents health. It is assumed, that the agents are moving in a swarm, and the agents will also join each other in attacking the enemy. The means, there is no point of running away from the enemy, as that would lead to leaving the swarm.

3.5 Ambushing

Another main strategy of the agents is the so called ambushing. When the agents perceive an enemy agent within an area, that is -2 to +2 in x direction and 0 to +2 in y direction, then they will ambush it. It means, that the agent will stop moving and will wait. If the enemy moves in the path ahead, then the agent will run it down. It means, that it will start moving in its direction till it reaches the enemy, and then it will start to attack it. With this strategy, if an enemy appears ahead, the possibility for the enemy to attack us in the back is minimized, while the other ally agents can catch up with the agent that was ahead.

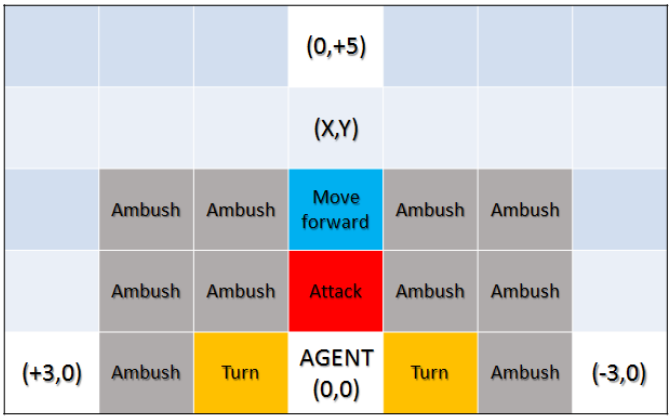

Figure 3.2 Strategy based on closest enemy

3.6 Eating the Food or not

An interesting topic is how to handle the food. Eating the food can increase the health of one given agent, on the other hand, as the agent will stop to follow the cruising movement pattern, it can result in losing the swarm. To make a decision, a world with more food were created. For the first use case,

the world had provided the enemy with more food, and the food was neglected from the new agents' side.

For the second use case, the both agents took the food into consideration. The strategy was to take the food only into consideration, when there is no enemy agent in the perception at all. The agents ate the food all the time, even if they were with full health or not. The reason for that was, that this way all the agent, that noticed the food started to go for the food, so that they stayed close to each other in swarm. In addition, they can eat the food rather than the enemy agent. This is a valid tactic, as the enemy agents are not that strongly focused on the swarm, and for these agents the food and extra health means more extra. In this case, the goal is to keep the agents close and have as high attacking power in the swarm as possible, not as high health as possible. In the final implementation the strategy that considers no food is used.

4. Implementation

The implementation of the concept can be divided into two parts, the one without the LIDA Framework and the one with it.

4.1 Implementation without the LIDA Framework

For trying out the strategy in a real environment, an external mind was created that runs the same strategy as the future agent is going to use. The main difference is in the implementation. In this case, a fixed algorithm performs just exactly the strategy. A fixed algorithm is good, if the complexity can be kept low, while for an extensibly growing complexity the cognitive architecture is the real, long term solution.

4.2 Connecting the game and the LIDA Framework

The first challenge was connecting the Framework with the game. The LIDA Framework consists of one eclipse project and its additional configuration files. The game consists of 7 eclipse projects. For the connection between the framework and the game a new workspace were created and the 8 projects were added to it together. The game is the master, so it will work as it worked before. A new type of agent were defined within. The body of the agent is provided by the game, but the mind of it is via the LIDA Framework. It means, that a connection were created between the external mind of the game and the LIDA Framework. It was solved via a gateway. This gateway is the part of the LIDA project. Here, data can be transmitted from the game to the LIDA Framework. With that, the LIDA agent will have the needed inputs for its sensor. On the other hand side, the LIDA Framework can also interact with the game via this gateway and it can set the proper actions for the agents' body. A synchronisation between the two parts is needed. As the game, just as the LIDA Framework can be initialized, the simulation times can be altered, but the simulation time does not reflect to the extra time needed for running the programs. These programs are quite significant from the size point of view (>100MB) and this delay can not be precisely predicted. For that, the LIDA Framework must run faster than the game, as it must be able to sense the environment for each simulation cycle. The action handling function of the gateway were restricted for that. It means, that an action is set only once, and it is then instantly cleared and remains so, until the next cycle comes.

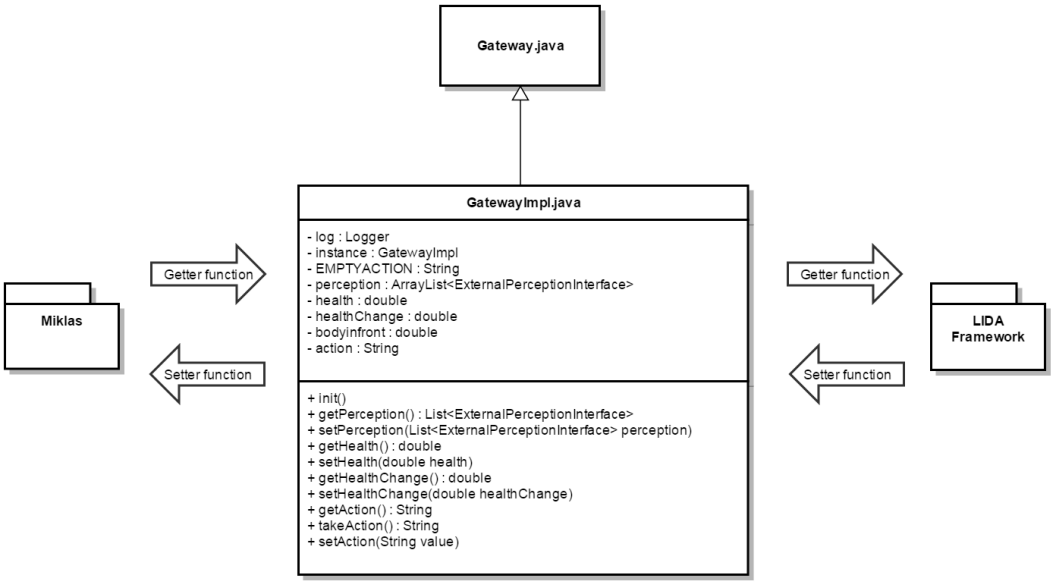

Figure 4.1: The gateway in a UML class diagram

4.3 Implementation with the LIDA Framework

For the creation of the agent in the LIDA Framework a strategy has already been provided and the gateway is created as well. It means, that the core of the task was to map the functionalities properly to the LIDA Framework and make it run appropriately.

At first, values shall be passed between the game and the framework. For that the setter and getter function shall be defined within the gateway and then comes the implementation in the gateways implementations module. It must also be handled properly from the games implementation side with passing or fetching the right values.

Now, the framework has all the provided data to work on it. Both external and internal stimuli must be passed to the Sensory Memory module. The sensors shall run periodically and in addition it is set, how these sensor information are provided to the other modules of the framework.

The sensor information will be taken into consideration via feature detectors. The feature detector is responsible for interpreting the sensor value and provide a result for a given interpretation scenario.

The creation of an agent and the setup of the framework is outsourced into xml files. After the implementation of the feature detectors in code, these must be properly configured. As a first step, the new tasks of the feature detectors has to be added to the Factory Data. This xml is responsible to set the non-agent specific configuration.

For the setup of an agent an agent relevant xml file is created, that is usually called appropriate to the agents attribute. For the Primitive Feature Detection, the task of the feature detectors must be added to Perceptual Associative Memory. In addition, the nodes for the Perceptual Associative Memory are

created and linked to the proper feature detectors. The node is the element of the framework, a data structure that has an activation and salience as well. It will be used for storing the information of a detected feature and to decays it value over time.

The information from the Perceptual Associative Memory shall be moved upon need to the Global Workspace. For that, the tasks combined with the nodes are added to the Attention Module. An initial activation level can be set for each added task. This way, different detections will have different priority as their activation level is different.

As an event can already be detected and attention is also given to it, the next step is to set up the Procedural Memory. This module is responsible for storing the desired actions, the so called schemes. A scheme is the connection between the Procedural Memory and the Sensory Motor Memory. In the LIDA Model, there is the Action Selection module in between, but in this implementation it is not used. As it was also taught via the framework as its exercises, the users of the framework do not always need to use each modules of the LIDA Model. In the future, this Action Selection Module shall be used for defining an easy and proper overview for the action priorities. At the end, the selected action must have an implementation in code in the Environment module.

The implemented version of the agent is as follows. It uses 7 feature detectors. One for detecting if an agent is within the ambush are, one for detecting an enemy that is directly in front of the agent, one for detecting an enemy in the path straight ahead, two for detecting enemy directly on the left or the right side, one for detecting an obstacle directly in front of the agent and finally one, that detects, if the agent is free to cruise.

Each feature detector has one node in the agents' xml definition. The following schemes are built upon it with the corresponding activities in the Sensory Motor Memory. If an enemy is ahead, then it is going to be attacked. If an enemy is on the left or the right side, the agent is going to turn to it. If an obstacle is ahead, the agent is going to jump to the proper step of the cruising pattern. The cruising pattern is implemented in the Sensory Motor Memory and is equivalent to the algorithm based solution. If an enemy agent is within the ambush area and is in the path, the agent will move forward to it. This is a unique scheme in comparison with the others, as here the agent has two nodes for activation. If an enemy is in the ambush are, the agent is going to wait for the next movements of the enemy. Last, but not least the agent has the free to cruise detection, for which the cruising action is executed.

In the validation of the development the provided LIDA graphical user interface was a great help. It has separate tags for different modules from the LIDA Model. With the help of it, we can follow in runtime the whole chain beginning from the detection till the action execution. It covers our whole use case described above.

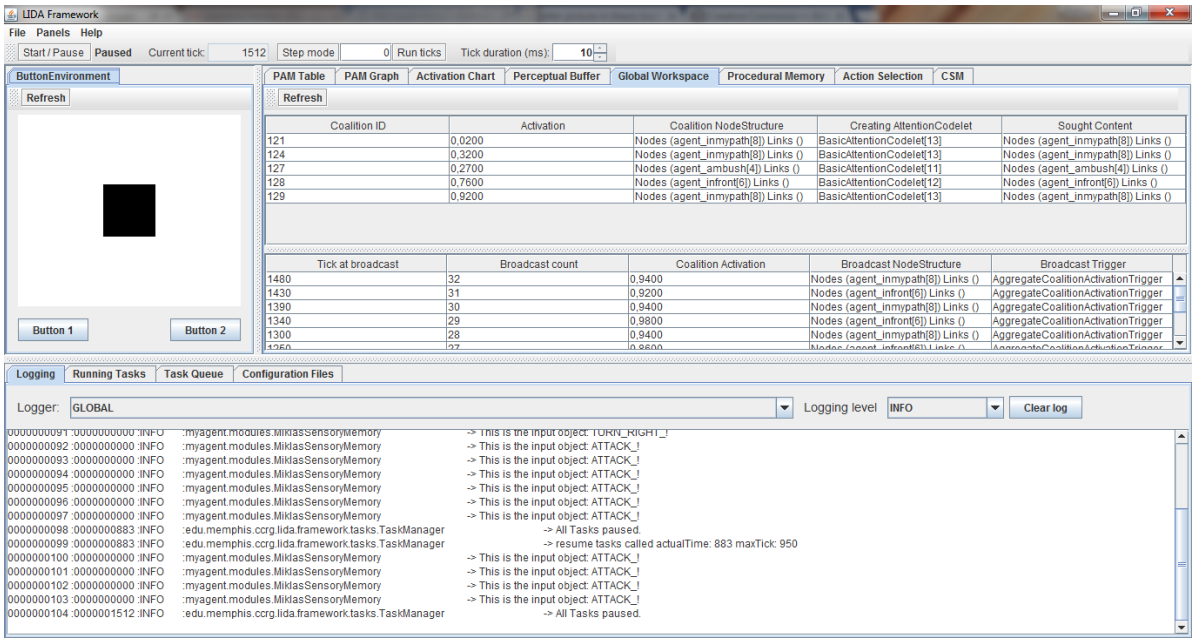

Figure 4.2: Graphical user interface for the LIDA Framework

5. Simulation and Results

The simulation of the implementations was quite straight forward, as both the simple game and the LIDA Framework is provided with its own graphical user interface. For the implementation without the framework the graphics of the game were adapted. It was desired to have an easier overview on the system, even if a high amount of agents are on the map.

With the strategy, that is totally following the Keep it a simple and short attitude, an agent has been created, that is more effective than the reference agent. The swarm was that much more effective than the enemy agents, that 25 of the new agents were still overwhelming 44 enemies.

As the consideration of the food is a significant factor in the success, it was additionally tested. If the food is considered only from the enemy side, then even the high amount of food did not make a real change to the game, as when the agents were moving in a swarm, then they killed the enemy agent earlier, than it could refill its health. As the health can not be overflowed, the enemy were not really able to make the use of the food. When the food was taken into consideration, it became clear, that the agent will start to separate themselves into smaller swarms. That is totally reasonable, as there is no follow the leader tactic implemented, nor it is intended to be done, as that would be against the basis of the strategy. The result of the test showed, that it is not advised to search for food, as that will cause smaller swarms and a smaller swarm is not as effective as a bigger one. This means, that the agents were easily overwhelmed by the enemies, which had a tactic based on their actual own status and perception.

The LIDA implementation of the agent could still poses possibilities for additional development. With the current implementation, the framework can handle only one agent in the game. It could be enhanced via property files. With the action selection module of the LIDA Model, the agents' decisions making can be made more sophisticated. The fact, that there is still further development possibilities is absolutely natural in such an artificial intelligent project, when the system and the possible strategies together poses an infinite combination of setup and behaviour.

All in all, the initial goal of the project has been successfully achieved and a referent agent can be provided for the SiMA agent.

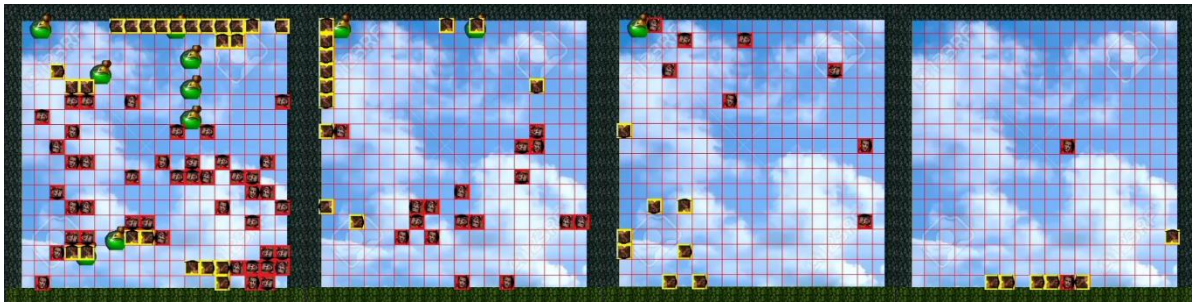

Figure 5.1 Screenshot from a game, where 25 new agents (yellow) overwhelms 44 enemies (red)

6. Conclusion

The main task of the project has been successfully achieved. As a good basic understanding were gained, a proper cognitive architecture has been chosen for the agent implementation. A successful strategy has been created. This strategy was implemented at first in the game, then in the LIDA Framework. The result of the semester can provide a benchmark, as well as a good basis for further developments. The methods implemented can be reused in further university projects.

Regarding the technical solutions, used in this work, it can be clearly stated, that in case of an algorithm, which was quite easily programmable just inside the game, the effort to implement it into a cognitive architecture was significantly higher. Besides the fact, that using the architecture desires to implement at least a given predefined number of modules even for a very simple function. If somebody gets more and more experience in one given architecture, just as it is done in simple algorithm programming, then the effectivity will significantly rise. What shall also be considered, is that the cognitive architecture makes it easy to adapt, even if the initial effort is a bit higher. That is why for a very complex system, that is about to be continuously adapted, the cognitive architecture is a clear advantage and shall definitely be used. The conclusion from the results point of view is that a simple solid strategy with a good reasoning in its behaviour is an advised solution for even complex scenarios.

Literature

[JRSS16] Javier Snaider, Ryan McCall, Steve Strain, Stan Franklin – The LIDA Tutorial, Verison 1.0 from the Cognitive Computing Research Group at the University of Memphis, 2016

[OEC08] Enrico Giovannini - Understanding Economic Statistics: An OECD Perspective, 2008

[RSNP03] Russell, Stuart J.; Norvig, Peter, Artificial Intelligence: A Modern Approach (2nd ed.), 2003

Internet References

[1] SiMA, Simulation of the Mental Apparatus & Applications, http://sima.ict.tuwien.ac.at/, 8. February 2016

[2] IDA: A Cognitive Agent Architecture, IEEE Internation Conference on Systems, Man and Cybernetics, http://ieeexplore.ieee.org/xpl/login.jsp?tp=&arnumber=725059&url=http%3A%2F%2Fieeexplore.ieee.org%2Fxpls%2Fabs_all.jsp%3Farnumber%3D725059, 12. February 2016

[3] Aplu (Aegidius Plüss) – JgameGrid, http://www.aplu.ch/home/apluhomex.jsp?site=45, 02. December 2015

[4] University of Southern California - Institute for Creative Technologies, http://cogarch.ict.usc.edu/, 15, January 2016

[5] University of Michigan, SOAR Cognitive Architecture, http://soar.eecs.umich.edu/, 20. January 2016

[6] The Eclipse Foundation open source community website, https://eclipse.org, 24. January 2016

[7] CCRG – Cognitive Computing Research Group, University of Memphis, LIDA Framework, http://ccrg.cs.memphis.edu/framework.html, 24. January 2016

Declaration

Hereby, I declare, this present work has been drawn up without inadmissible aid of third parties and without usage of other than mentioned resources. Further sources or indirectly appropriated data and concepts are identified by stating the source.

This work has not been presented to other examination procedures, neither nationally, nor in foreign countries, in the same or in a similar form.

Vienna, 25th January 20136

BSc Adam Cziko