UNIVERSITÉ DE TECHNOLOGIE DE BELFORT-MONTBÉLIARD

Lemmings Sapiens

Report VI51 – P12

SALVI Julien
DESOCHE Clément
MANIGOLD Gaëlle
LE GUERROUE Thomas

Computer Science Department
Imagery Interaction and Virtual Reality

Tutor in VI51

GALLAND Stéphane

Contents

1. Requirement Analysis

1.1. Authors and project's context

This project takes place during our studies for VI51 course. During this semester, we had to manage a project which deals about *Virtual Life Simulation*. A group of four people, Julien Salvi, Gaelle Manigold, Thomas Le Guerroué and Clément Desoche, decided to implement a project which deals about the multi-agent system and the learning algorithms.

1.2. Goals

The main goal of this project is to implement a software that manage a lemmings-like game. Instead of dealing directly with the lemmings, an artificial intelligence will be implemented thanks to a multi-agent system. The lemmings will move in a environment where he can walk, jump, dig, excavate or fill a hole. In case where he would be in free fall, he can be able to open a parachute to avoid dying. His main goals will be to learn how to reach the exit and avoid dying by the way. That's why, a learning algorithm has been implemented to solve this problem. A such algorithm is based on the QLearning studied during the semester.

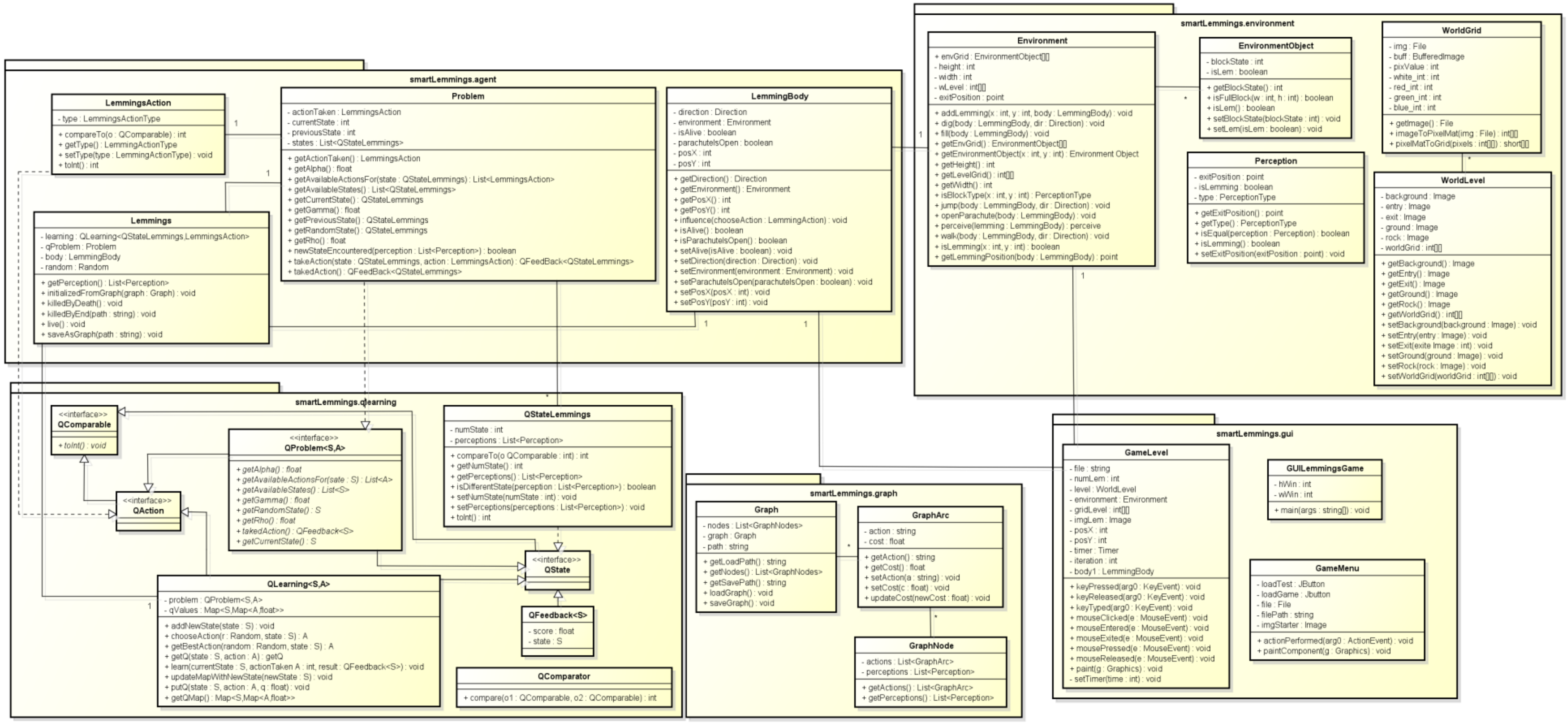
The initial goal is to develop a simulation software of the environment and of lemmings of a second generation, that is lemmings that are more intelligent than the individuals of the current breed. Such advanced lemmings should **learn from their past experience** how to avoid dying. In any case, lemmings must move together, as fishes do when they move as a school.

Models of these social creatures, of the environment's physical laws and possibly of the possible interactions from the user will have to be designed. For instance, a user interaction with the lemmings' environment would consist in either saving them by building a bridge between the edges of two cliffs, or killing them by digging a hole into a mountain...

2. Design Implementation

2.1. Software architecture

Here is a class diagram which represents the architecture of the software:



2.2. Design choices

This software has been designed with an interface and a visual perception to satisfy the user. Moreover, several techniques and technologies has been employed to manage this project.

Create the world:

In order to create the world where the lemmings will evolve, it has been decided to create a text file where a sequence of number will be stored to represent the type of blocks which belong to the environment. A 0 represents a empty block where the lemming can stood. A 1 represents a ground block which is breakable, the lemming can destroy this kind of blocks or fill the world with one of them. A 2 represents a rock block which is unbreakable, the lemming cannot go through this kind of block and he cannot destroy it. A 3 and a 4 represent, respectively the input and the ouutput of the world.

The file text, composed with this sequence of number, has a width of 27 numbers and a height of 18 lines. To create the world, the file is parsed and the numbers are stored in an array of integer that represent the world. Then, these number will be a part of the environment objects. You can see below the representation of a world in a text file:

```
000000000000000000000000000
000111111111111110000000000
011110000000000011111000000
011000000000000000111000000
011300000000000000011000000
011111111111111111111000000
000000000012210000000000000
020002020012210002220002000
002020020012210002000022000
000200020012210002220202000
000000000012210000020002000
000000000012210002220002000
000000000012210000000000000
000000000012210000000000000
000000000012210000000400000
000111111111111111111110000
000112222222222222222110000
000111111111111111111110000
```

Environment design and perception:

The lemmings' environment is represented by an array of environment objects which are composed of an integer and a Boolean. The first attribute represents the block type with respect to the world established previously, and the second is a Boolean to know if there is a lemming on the block.

When the lemming evolves in the environment he is able to perceive the elements around him. The frustum has a length of one that means the lemming perceive a square of eight blocks around him. Moreover, the lemming is able to know the position of the exit door in the world in order to reach it more easily. With respect to the block type, the environment will react with respect to the action taken by the creature. The actions excavate or dig will destroy the breakable block in the corresponding direction. Another action can alter the environment, it is the fill function that allows the lemming to fill an empty block in front of him on the ground. To reach the exit, the lemming is able to walk, jump or open a parachute. These actions do not alter the world, they just have a consequence for the lemming's position.

A gravity-like is applied to the environment. Indeed, when the lemming's position is into an empty set of blocks, he is going to fall till he reach a breakable or unbreakable block. If the fall is greater than 3 blocks, the lemming will die, unless the parachute is open, and a new creature will appear in the entry block.

Graph:

The first implementation of Q-Learning was based on a graph structure which nodes represent the learning state, the arcs represent the action and the weight on the arcs represented the Q-Values. But after some research on Q-Learning implementation it has been decided to move the Q-Value as a Map structure and the current implementation. As the Graph structure was already implemented and designed to be loaded and save via serialization we decide to keep it and to built our new class via this structure.

Reinforcement:

To implement the AI of the lemming we chose to use the Q-Learning algorithm. This is a reinforcement learning algorithm. The algorithm consists in dividing the problem in state and action. Each action in a current state has a given Q-Value that measure the quality of choosing the action. At each step of the simulation the agent is in a given state s and chooses a given action a. Then a reward is given to the agent and it update the Q-Value related to the action in the state via this function.

$$Q(s,a) = Q(s,a) + \alpha\,(r + \Upsilon max\,(Q(s',a')) - Q(s,a))$$

Where:

- *Q(s,a) : Q-Value for the state s and the action a*
- *α : learning rate*
- *ϒ : discount rate*
- *r : reward*

The set of actions is composed by 6 of them: walk, jump, dig, excavate, fill and open a parachute. Indeed, a lemming dies when he falls during over 3 blocks so the opening a parachute permits to avoid his death.

A state in the lemming’s problem contains three different data: the lemming’s perception, lemming’s direction and the direction of the exit (computed from the lemming’s position and the exit position). The choice of the perception as the state’s core is obvious so that we can for instance reward him if it opened is parachute if the case below him is empty or punish him if he chooses to walk out of the world’s bounds. Direction are used to evaluate if the lemming shorted his distance to the exit for example if he decides to walk and it is in the right direction it is rewarded. The use of direction is more generic than directly use the exit’s position because it changes from a world to another.

The action selection method is an epsilon-greedy function, if a random number is inferior to a given epsilon, we choose a random action else we choose a random action among the best of the current state.

The lemming’s life:

The lemming’s life follows those steps:

1. Perceive the environment

2. Check if it reaches a new state

    a. If so we add the new state to the problem and the Q-Value store with 0 for all Q-Values
    b. Else we change the current and previous state

3. Learn from the previous state and the taken action by applying the Q-Learning formula.

4. Choose an action via epsilon-greedy

5. Influence the environment

After the lemming received a very bad reward when it dies and it is put back at the entry of the world and this while it has not reach the exit or until it got no more times.

Calibration

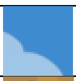
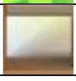
For the epsilon-greedy considering that even if we choose the best action is in fact a random choice among the best the rate of choosing a random action has to be medium or low. Here a rate of 30% gives a good equilibrium between exploration and convergence.

The learning rate evolves during the simulation as it measure how much the lemmings learn it will learn a lot in the first third of the simulation then the learning rate will decrease to reach convergence.

The discount rate evolve too, as the lemming know nothing about the environment at the beginning we will explore and have a low discount rate. Then when many states as been reach we increase the discount rate to reach the better states.

2.3. GUI implementation and technology

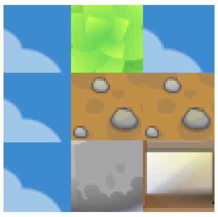
In order to implement the graphical interface of our application, the use of Swing/AWT has been chosen to display the world, the behavior and the changes which will occur during the simulation. In each world of this game, there is an entry and an exit. There are also some breakable and unbreakable blocks.To build a world we use a text file which contains some numbers from 0 to 4. These numbers represents the different blocks as described in the legend below:

	Code	Image
Empty	0	
Breakable	1	
Unbreakable	2	
Entry	3	
Exit	4	

The rendering is done in the class thanks to the WorldLevel.java, where the text file is loaded and parse into an array of integer. Then, in the GameLevel class, the rendering is done by parsing the array and a mapping is done with the corresponding blocks. Here is an example of mapping of a text file and its graphic rendering:

```
030
011
024
```

$\Longrightarrow$



When the lemming is falling, he has the possibility to open a parachute to avoid dying. In order to see on the screen if the lemming has or not the parachute opened, two images have been set to compare the two states:

Lemming	
Lemming with the parachute	

2.4. Problems encountered

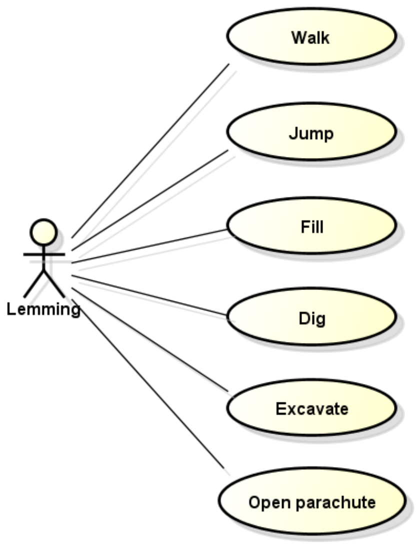
During the implementation of this project, we encountered several problems.

The first one is the comprehension and the implementation of the Q-Learning. Indeed, the seminars about the learning were at the end of the semester so we had to do some research by ourselves to understand this method. We had a problem with the perceptions which were not stored in the way they were collected. Another persisting problem was the repaint of the level panel. Indeed, as swing is mono-threaded the implementation of a *Thread.sleep()* method aborted the panel refresh. Thus, a timer had been implemented in order to solve this problem.

3. User Guide

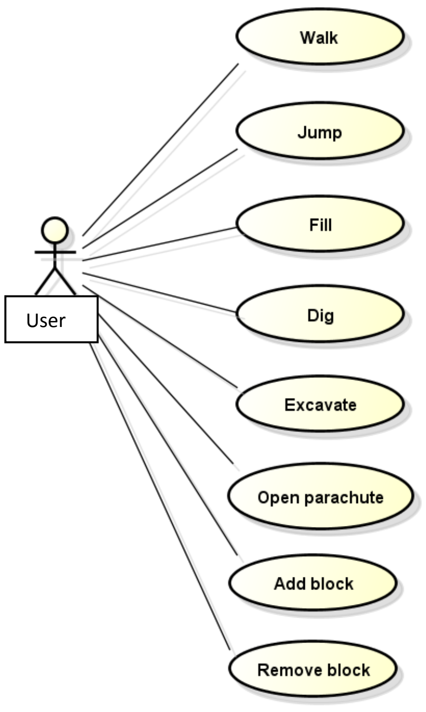
3.1. User cases

A lemming can choose what to do among 6 actions:



He can walk or jump to the left or the right. He can also fill or excavate the block in front or behind him and dig the one which is below him. Finally, he can open a parachute, action very useful in case of falling.

About the player, he will be able to interact with the environment and the lemmings thanks to the following actions:



The actions executed by the lemmings, may be done thanks to keyboard events except for opening the parachute which is done thanks to an event on the mouse wheel. These interactions are disabled in the agent mode but active in the player mode.

The environment interactions are done thanks to some event with the mouse buttons.

3.2. GUI controls

When the application is launched, you can see a menu which offers 2 options: the test level and the game level.

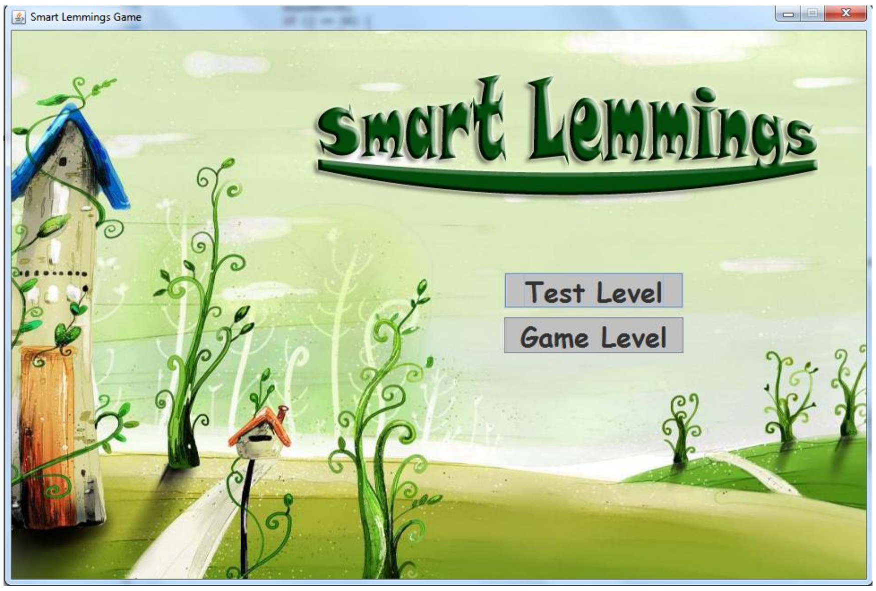


Figure 1: Game menu

The test level is a learning phase which permits to the lemmings to learn how to reach the exit of a world. For example, it's in the test level where a lemming will learn to open a parachute when he falls in an empty space. The game level is the "final" game of lemmings. By clicking on these button, an option panel is opened in order to ask to the use the path of the wanted world text file.

For each option the user has the choice to launch the game with an AI or play the lemming himself. About the game level, the user can choose the number of lemmings he wants in the chosen world.

In player mode, the user can apply several actions thanks to the following keys:

- Move forward : *right*
- Move back : *left*
- Jump to the right : *up*
- Jump to the left : *shift*
- Dig : *down*
- Excavate : *move on a breakable block with left or right*
- Fill : *space*

Moreover, the user can use the mouse in order to add or remove blocks in the environment. The listener on the mouse is available for the player and the agent mode. If the user does a right click, the block where the cursor is will be removed from the world and set to empty. But the user cannot remove a entry or a output. By clicking on the left button, the user will be able to add a breakable block to the world or modify the selected block to the breakable type, it works the same way with the unbreakable block by double clicking on the left button. You can see below, some screenshots from the application which represent the lemming's fall and the lemming's behavior in the environment:

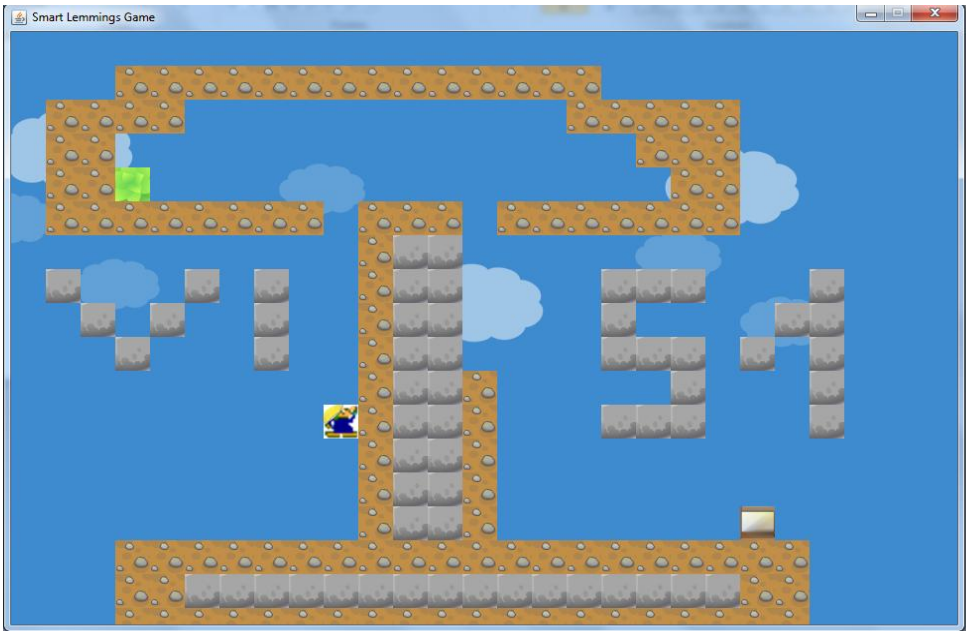


Figure 2: Lemming's falling

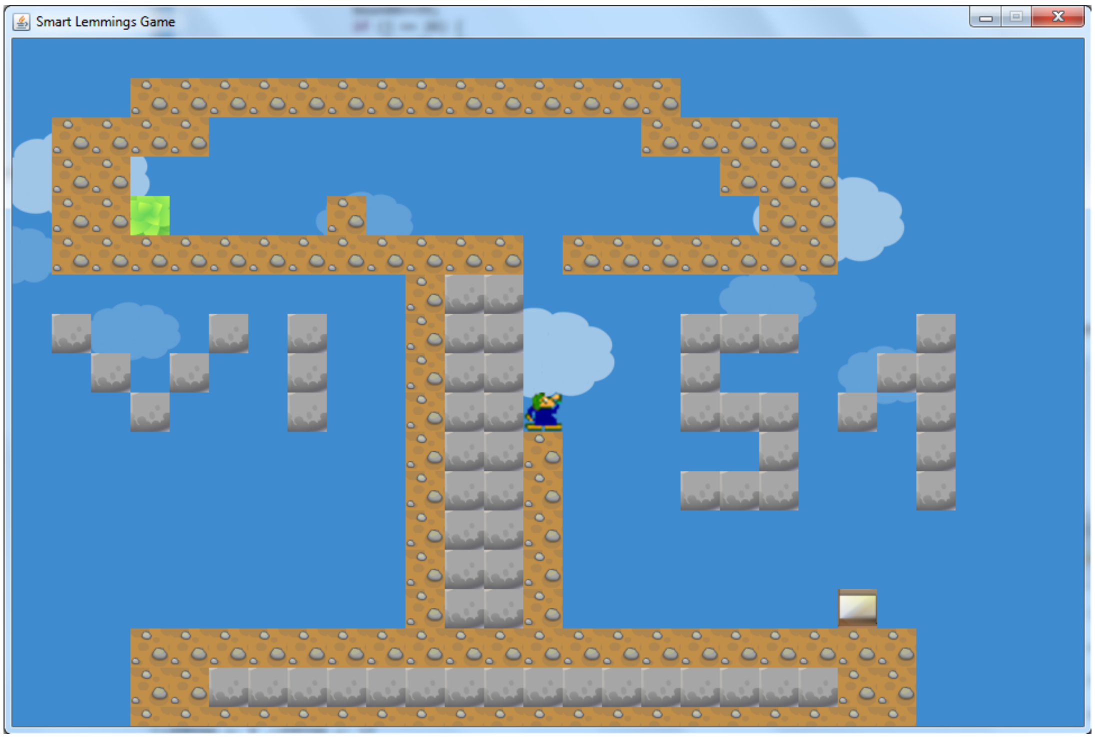


Figure 3: Lemming's moving in the world.

In order to refresh the lemmings' position and the changes applied to the environment, a timer has been set up. An interval of 400ms between each iteration to see the changes but this time may be modify in the *GameLevel* class. At each iteration and *live()* function, the repaint() method is called. The timer is stopped when the lemmings reach the exit or if the iteration overpass the 10000.

4. Conclusion

To conclude, this project provide us a real experience in the virtual reality domain with the conception of a environment model and in artificial intelligence thanks to the multi-agent system. Also, this project allows us to have some knowledge in learning algorithm based on the Qlerning. The use of Java and Swing/AWT has been very useful in order to create a dynamic interface.

About the improvement, we did not have enough time to implement the learning algorithm for a school of lemmings. Moreover, it would be nice to improve the environment by adding water, fire and maybe wind to increase the difficulty for the lemming. It could be interesting to implement the environment into a 3D world represented by a cubic grid.

Keywords

Virtual life simulation – Informatics – Development - Learning algorithms - Environment models - Basic behaviors – User interactions

SALVI, DESOCHE, MANIGOLD, LE GUERROUE

Report VI51 – P12

Summary

During this semester, the VI51 course has been followed in order to learn the bases of the Virtual life simulation. A project has been managed along this period to apply the knowledge learnt in lectures. Thus, the project *lemmings sapiens*' main goal was to implement an environment model, a basic behavior and a learning algorithm which will be apply to the creature for reaching the exit.

In addition, to these required elements. Some user interactions have been implemented which allow the user to interact directly with the lemmings or modify the world by clicking on the environment.

utbm
université de technologie
Belfort-Montbéliard