Loïc Vauvrécy B00054517

Loïc Vauvrécy B00054517

**A Lemmings 3D game**

Creation of an Artificial Intelligence

Computing – Year 4 – [Project (Part 1)](https://amsprd0302.outlook.com/owa/redir.aspx?C=pFPg0UXdPUWj8ZHexPZOD0_kuY-ppc4I-Yh0eALjHBdRfbNAA1XLCC5agNLkMjjpqs_PdnHRMUo.&URL=http%3a%2f%2fcourses.itb.ie%2findex.cfm%2fpage%2fcourse%2fcourseId%2findex.cfm%2fpage%2fmodule%2fmoduleId%2f1379)

COMP H4027

January 2012

# Aim of my project

The goal of my project is to create a *Lemmings game*[[1]](#footnote-1) in 3 dimensions with an *artificial intelligence*[[2]](#footnote-2).

# Description of the Lemmings game

The goal of the “Lemmings 3D” game is to bring a lemmings team from the starting point to the end. A lemming walks until it meets an obstacle: then it turns right. The player can interact with a lemming with actions:

* Block: the lemming stops to move and becomes like a wall for the other lemmings.
* Build: the lemming builds a small bridge.
* Mine: the lemming digs down forward.
* Explode: the lemming explodes.
* Dig: the lemming digs down.
* Float: the lemming does not die if he falls from a high point on the map.
* Climb: the lemming can climb a wall.
* Bash: the lemming digs in front of him.

There will be 2 teams by map: 1 human player and 1 computer player.

My part in the “Lemmings 3D” project was to build an artificial intelligence for the computer team.

The project is based on the *“Unity” software*[[3]](#footnote-3).

# Search for a solution

My first idea to build the artificial intelligence was to begin with a path finding algorithm. I decided to implement the *A star algorithm*[[4]](#footnote-4) in Unity.

It worked well for a “classic” pathfinding, in real time.

But when I decided to modify it to implement the lemmings’ actions, I realized that the Lemmings solution would be too complex and that the A-start algorithm wasn’t designed for this.

I also discovered that the Lemmings game is a known problem for an artificial intelligence, and it is really complex: it is a NP-complete problem.

*Wikipedia*[[5]](#footnote-5) says about the NP-complete problems:

“The time required to solve the problem using any currently known algorithm increases very quickly as the size of the problem grows. As a result, the time required to solve even moderately sized versions of many of these problems easily reaches into the billions or trillions of years, using any amount of computing power available today.”

It means we can only solve the easiest problems and only by using approximation algorithms.

Then, since I didn’t know a lot of artificial intelligence algorithm, *I read a lot of articles about it on internet and on books.*[[6]](#footnote-6)

I chose to implement a genetic algorithm because:

* It seemed to be a really powerful algorithm to deal with complex problems.
* It also seemed not too hard to understand it and to implement it in Unity.
* I found a paper describing a previous attempt *to use GAs to solve Lemmings problems* [[7]](#footnote-7)
* And finally, it seems very interesting and I really wanted to know how it worked.

# Description of the solution

A genetic algorithm is an algorithm based on the process of natural evolution. It uses the same techniques, such as inheritance, mutation, selection and crossover.

An explanation of the algorithm is given by the following diagram:

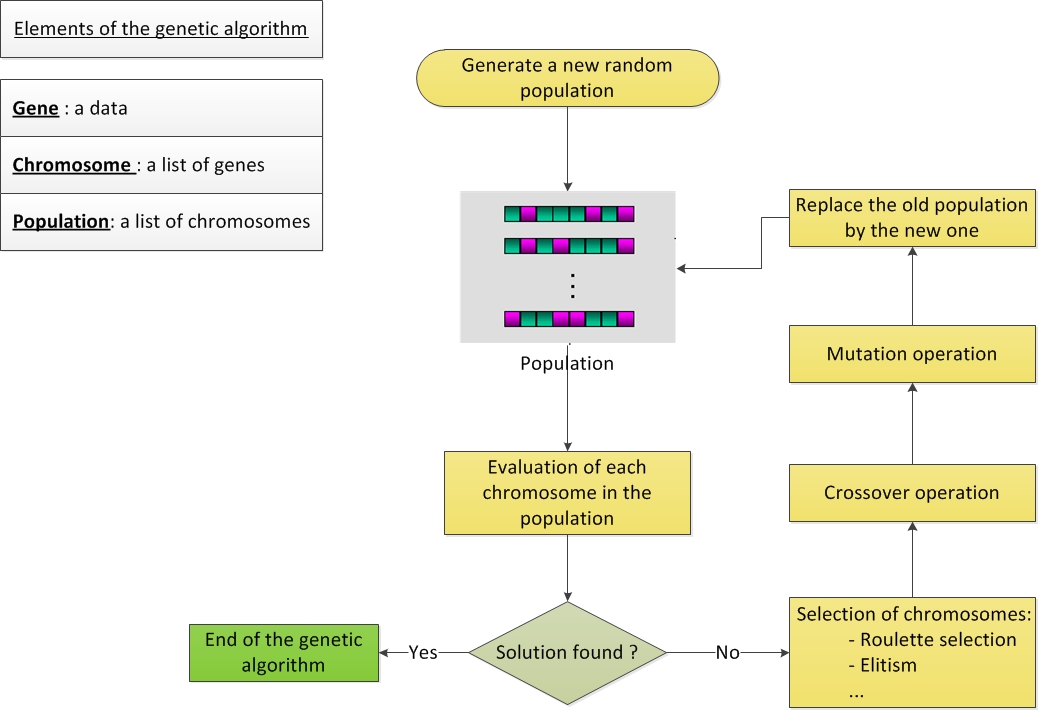


Figure 1: Genetic Algorithm approach

For my approach, a gene is a data pair (T, A), consisting of time “T” and a action “A”. A chromosome (made up of a sequence of genes) represents a complete set of actions for a team – i.e. a “solution” to the current map. The first action will be applied to the first lemming in the team at the time defined by the gene, the second action for the second lemming, etc.

To evaluate a chromosome, I couldn’t use the straightforward Unity 3D simulated physics because it is too slow when there are hundreds of chromosomes to test. So I decided to simulate the game myself in the algorithm.

The terrain is represented by an array of integers.

* 0 means a lemming can walk.
* 1 means a lemming can’t walk and will turn right.
* 2 means a lemming built a bridge.
* 9 is the end door.

By simulating a team of lemmings walking in the array, we can evaluate a solution more easily.

Each solution is evaluated by the closest point of the end door they walk to. The highest evaluation is when a lemming succeeds to arrive at the end door.

A word about the mutation in a chromosome: in this case, the length of the chromosome can change, since we don’t know how many actions will be needed to achieve the goal. So I coded 3 types of mutations:

* A changing mutation, which changes the time of a gene.
* A deleting mutation, which deletes a random gene
* A creating mutation, which creates a new random gene.

# Review

The solution works very well for basic levels, where the solution only requires a small number (e.g. less than 3) of actions. Moreover, it is faster than I expected: it usually takes only a few seconds to find a solution.

However, there is a lack of precision in the simulation and sometimes, a simulated solution doesn’t work in the game. Indeed, there are some details that are simplified in the simulation:

* Unity’s time-step is 0.02 second. The simulation’s time-step is 0.1f. I could increase it, but the time necessary to find a solution increases really fast.
* A lemming is a simulation is simplified as a square. In Unity, it is more complex than that.

For example, the simulation can find a solution where, in fact, the lemmings will move just next to the end door (see Appendix B).

One possible solution to solve this could be an additional test we a solution is find, by running the chromosome in Unity and if it does not work, then the algorithm would continue to search for another solution. This way, the additional test would not be too much time-consuming because only 1 chromosome would be tested in Unity.

Moreover, my evaluation function is not yet complete: I woud like to add the number of lemmings used to find a solution and the time used by the team to arrive at the end.

At present the simulation of the game is currently only in 2 dimensions and it includes only 2 actions: the block and the bridge. Further work would include both increasing the number of different actions, and also extending the system to solve 3D Lemmings mazes.

My solution works (for example, see Appendix A) and has several differences from the existing attempt published by Kendall and Spoerer (2004):

* A lemming here is not just a pixel, like in the classical Lemmings game: it is a complete model and should not be represented only as a cell in an array.
* My genes are different than the ones used by Kendall and Spoerer: they used data pair (I, A), consisting of a lemming ID “I” and an action “A”. They used one gene by time-step. I use one gene by lemming.
* Moreover, the maps they used to test their algorithm were really small (20x20). Here, my map used in the simulation is a 100x100 map; I even tried a 500x500 map and it worked (solution found in 2min).
* Finally, since my solution is working in Unity, it would be just a matter of time before changing it to work in 3 dimensions.

# References

Buckland M. AI Techniques for Game Programming. Course Technology PTR, Florence, KY, USA. (2002)

Kendall G. and Spoerer K. Scripting the Game of Lemmings with a Genetic Algorithm. In proceedings of Congress on Evolutionary Computation 2004 ([CEC'04](http://cec2004.org)), Portland, Oregon, 20-23 June 2004, pp 117-124 <http://www.cs.nott.ac.uk/~gxk/papers/cec2004kts.pdf>

Spoerer K. The Lemmings Puzzle: Computational Complexity of an Approach and Identification of Difficult Instances. PhD Thesis, University of Nottingham, UK. (2007) <http://www.cs.nott.ac.uk/~gxk/papers/ktsthesis2007.pdf>

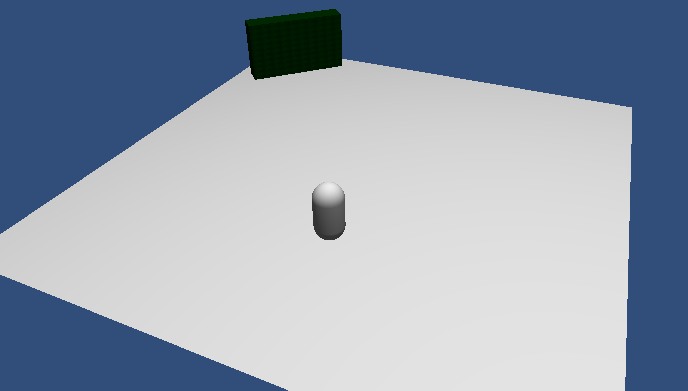
Howard Y. An explanation on genetic algorithm and a “Hello world” javascript example <http://www.puremango.co.uk/2010/12/genetic-algorithm-for-hello-world/>  
Official Unity web site <http://unity3d.com/>

Marshall D. An artificial intelligence course <http://www.cs.cf.ac.uk/Dave/AI2/AI_notes.html>

Obitko M. An introduction to genetic algorithms <http://www.obitko.com/tutorials/genetic-algorithms/>

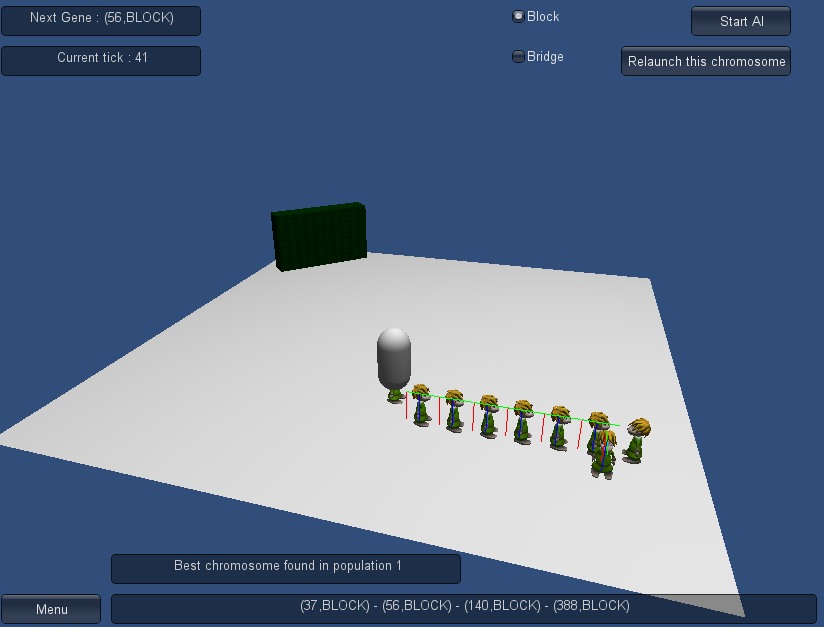
Various authors An explanation about the “machine learning” concept <http://en.wikipedia.org/wiki/Machine_learning>

# APPENDIX A – a map and its solution



**Figure 2: Here is the map, with the starting point in gray and the goal in green.**

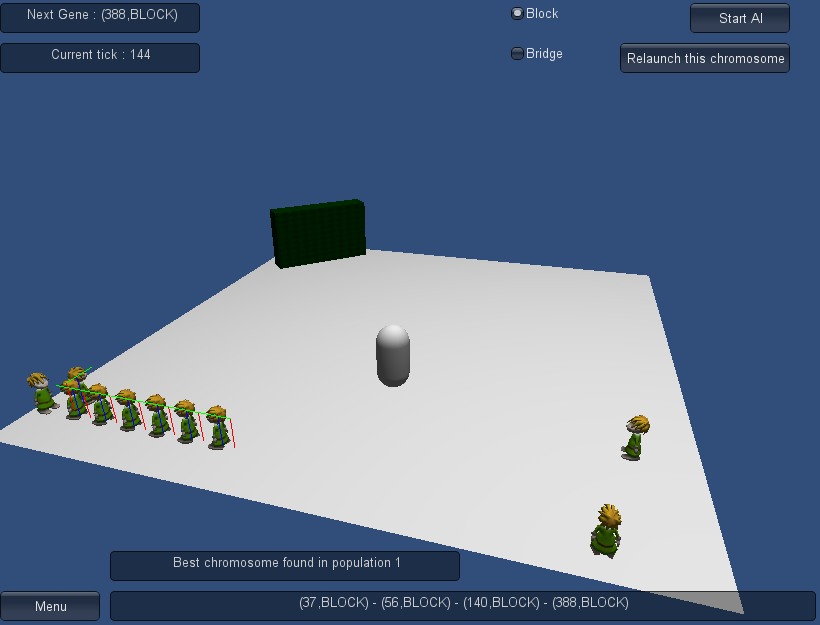
The chromosome the solution found for this map is: (37,BLOCK) - (56,BLOCK) - (140,BLOCK) - (388,BLOCK).



**Figure 3: First gene - the first lemming became a block at the step 37.**



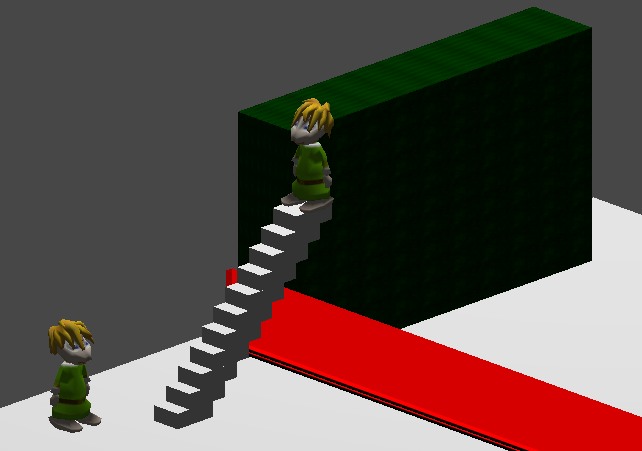
**Figure 4: Second gene - the second lemming became a block at the step 56.**

****

**Figure 5: Third gene - the third lemming became a block at the step 140.**

The last gene (388, BLOCK) will not be executed because all the other lemmings arrived at the end.

# APPENDIX B – an error on the final solution due to a lack of precision in the simulation

****

**Figure 3: a lemming became a blocker too early in the game.**

In the simulation, it was working well because of the longer time-step: the blocker was just after the stairs.

1. The original lemmings game: <http://en.wikipedia.org/wiki/Lemmings_%28video_game%29> [↑](#footnote-ref-1)
2. Definition and description: <http://en.wikipedia.org/wiki/Artificial_intelligence> [↑](#footnote-ref-2)
3. Official web site: <http://unity3d.com/> [↑](#footnote-ref-3)
4. Definition and description: [http://en.wikipedia.org/wiki/A\*\_search\_algorithm](http://en.wikipedia.org/wiki/A*_search_algorithm) [↑](#footnote-ref-4)
5. <http://www.wikipedia.com> [↑](#footnote-ref-5)
6. See references at the end of the report. [↑](#footnote-ref-6)
7. Kendall and Spoerer, 2004. See the references at the end of the report. [↑](#footnote-ref-7)