Fourth International Conference
Modelling and Development of Intelligent Systems
October 28 - November 1, 2015
"Lucian Blaga" University
Sibiu - Romania

Agent-Based Computational Models Implemented in 3D Space

Florentin Bota, Dan Dumitrescu

Abstract

The purpose of this project is to design and develop a unique state-of-the-art graphical platform that can be used to simulate and render different computational models in a 3D world. Using this platform we will be able to observe processes like emergent behaviours in a new perspective, using Unity3D as a graphics engine and C# as the main programming language. The finished product will be called VisualAgents and it will provide us with a new tool in our study of Complex Systems and Computational Models. In this paper we will present an agent-based approach of the platform, specifically a multi-agent ant colony. This project intends to create a better simulation tool that will be used in further experiments and for educational purposes.

1 Introduction

A computational model is a mathematical model used to study the behaviour of a complex system by computer simulation.[1] Common computational models are weather forecasting models, earth simulator models, flight simulator models, neural network models etc. Operation theories of the model can be derived or deduced from these experiments by adjusting the parameters of the system in the computer and studying the differences in the results.

An agent-based model (ABM)[2] is a class of computational models for simulating the actions and interactions of autonomous agents. They can be individual or collective entities, with a view to assessing their effects on the system as a whole. An agent-based model combines elements from game theory, complex systems, emergence, computational sociology and evolutionary programming. ABMs can also be considered individual-based models (IBMS), simpler agents than a fully autonomous model. The goal of ABM is to search for explanatory insight into the collective behaviour of agents obeying simple rules, as usually found in nature.[3]. These models are used to simulate operations and interactions of multiple agents in order to re-create and predict more complex phenomena. A recurring notion is that *simple rules can create a complex behaviour*. Also, the fact that the whole is greater than the sum of its parts is embraced.

Custom software was written in order to implement the specific ABM from this project with decision-making heuristics, an interaction topology and a non-agent environment. We studied how changes in individual behaviours affect the emerging overall behaviour of the system. Because we used a real 3D system instead of a simple 2D one and 3D meshes, we encountered several intriguing problems and issues that did not appear before in simpler systems. Forces like collision and gravity were not taken into account in our formal experiments and they did have several effects in the final outcome of the simulations.

2 Motivation and objectives

This project was inspired from another project, *Sotirios*[4], an effective e-learning application (Digital Game Based Learning) which combines cutting-edge graphics with learning puzzles. The video game follows the story of a future robot who dared to question the axioms that ruled his world and the player has to solve different learning puzzles (Fig.1) in order to progress in the immersive world of the game.

Fig.1 Different outcomes, based on the player's answer in Sotirios

It was an important step in developing 3D applications and we proposed a wasp behaviour model that steps in and allows the dynamic assignment of puzzles in the multiplayer mode of the game, taking into account the specialty of the students, their former experience and the complexity of the puzzles. An adaptive method allows students to pass through level 1 and determine their needs for the first time.

The idea for this particular simulation, an ant colony, came while reading a book written by Paul Ormerod, (Butterfly Economics) [5].The ant example was proposed to explain "herding" and "epidemics" described in the literature on financial markets as corresponding to the equilibrium distribution of a stochastic process rather than to switching between multiple equilibria.[6].

The question that picked our interest was *how would an ant colony split itself between two sources of food?* We tried to reproduce the actual results by using a simple agent model composed of a basic ant.

Regarding the practical application, several perspectives were addressed:

1. Creating a basic platform using Unity3D graphics engine
2. Creating one or multiple worlds as non-agent environments
3. Developing a system where the models and agents can be easily changed
4. Implementing an ant colony simulation and observe their behaviour

Secondary objectives in the above examples were:

a. A better understanding of Unity technology and DirectX/OpenGL functions
b. Create a simple scene with basic elements
c. Test the platform viability on multiple systems and operating systems
d. Test the ability to create and import objects from Blender
e. Animate the agents
f. Implement natural movement for the agents

The ant colony system, first proposed by Marco Dorigo in his PhD. thesis [7] is well known and used throughout the industry in multiple areas such as network optimisation, image processing, biology (protein folding), routing vehicles, etc. The algorithms based on the ant colony optimisation have an advantage over the simulated annealing and genetic algorithm approaches. They can adapt in real time to graph changes, a major interest in network routing and transportation systems.

Visual representations of the algorithm are mostly used for teaching purposes and for a better understanding of the mathematical functions. We wanted to take this a step further and allow a regular user to better imagine how these algorithms really work and even permit them to change the parameters in real time.

In figure 2 we can observe a similar approach, but in 2D space and it serves the purpose, but it is highly abstract and linear.

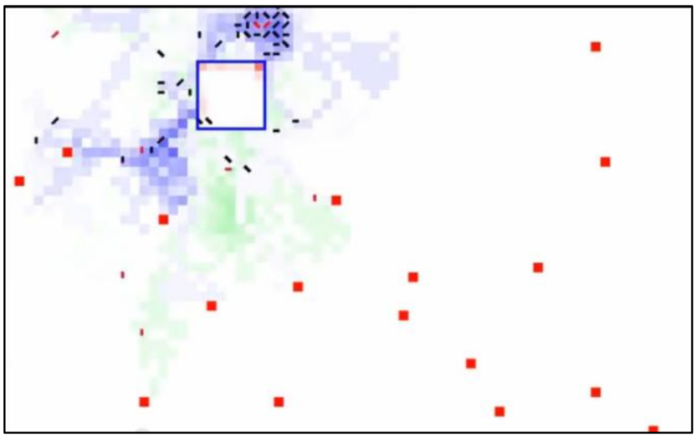


Fig.2: Ant colony simulation in 2D
Author: Gregory Brown
Source:https://practicingruby.com/articles/ant-colony-simulation

3 Implementation

3.1 Software environment

Our approach was implemented within the *Unity engine*, a cross-platform game creation system which includes a game engine and an integrated development environment. The reason we chose Unity was because it can deploy our results on more than 15 systems and operating systems, including Microsoft Windows, Apple iOS, Android, Linux, consoles and even web browsers. Also, a free version of the software is available for educational purposes or for individuals and companies with less than US$100,000 of annual gross revenue. Some characteristics of the engine that we used include:

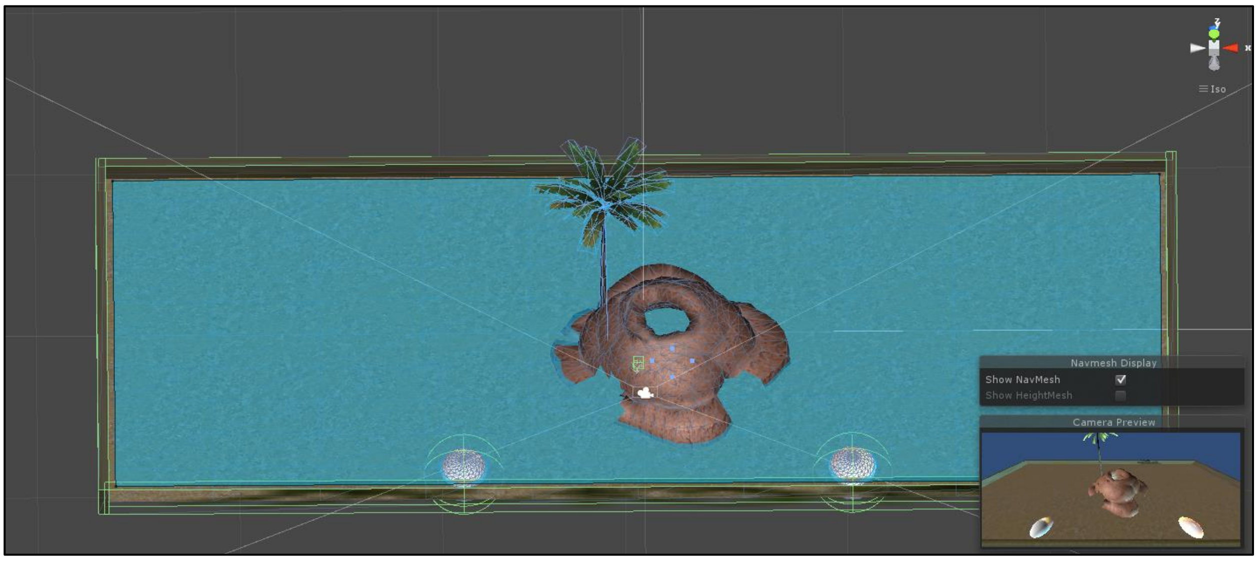
- Direct3D, OpenGL and some proprietary API's
- texture compression and adaptive resolution for different platforms
- bump mapping, reflections, screen space ambient occlusion
- easy world and character creation

The scripting is built on Mono, an open-source implementation of the .NET Framework and different programming languages can be used, like JavaScript, C# or Boo. In this project, *C# was mainly used*, because it was also possible to use Visual Studio as an IDE

Because we had to create some of the 3D models, we also used *Blender*, an open-source 3D computer graphics software where we did the actual modelling.

3.2 Experiments and platform elements

We started with a basic world, a confined space with a ground layer for navigation and several walls so our agents will not fall over the edge during their interactions, as can be seen in figure 3. A central spawn point is the hive, where the agents will gather to leave their *food* in the event that they find it on the map. There are two identical food sources, shown in figure 3 as white spheres that can be moved by the users around the map and observe the different behaviour of the agents. The users can also change their view of the camera much like in a video game by using a mouse or even by touch input for handheld devices.



Navmesh Display
Show NavMesh ✓
Show HeightMesh

Fig.3: Scene view of our simple world in Unity editor

Our basic agent, which mimics the ant behaviour, follows few basic rules, ant the behaviour can be expressed like a finite state machine (Fig. 4) with a probabilistic determination of the choices.

At first, an ant is in idle state, just as the simulation begins. After that, it can start searching food on the map, or follow the *pheromone* trail of another ant returning with food. This pheromone can evaporate over a *t* period of time, avoiding the convergence to a locally optimal solution [8].

Without the evaporation, the path chosen by the first ants would tend to be excessively attractive to other ants and the exploration of the solution space would be constrained. Of course, this also means that shorter paths between food and the colony will have a higher pheromone density and positive feedback will determine a lot of the ants to follow that path.

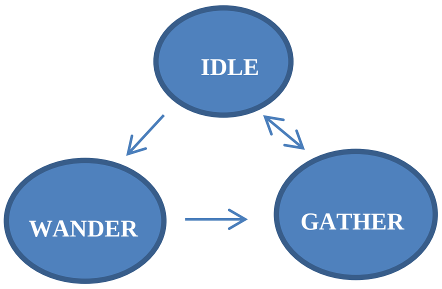


Fig.4: Simplified agent states

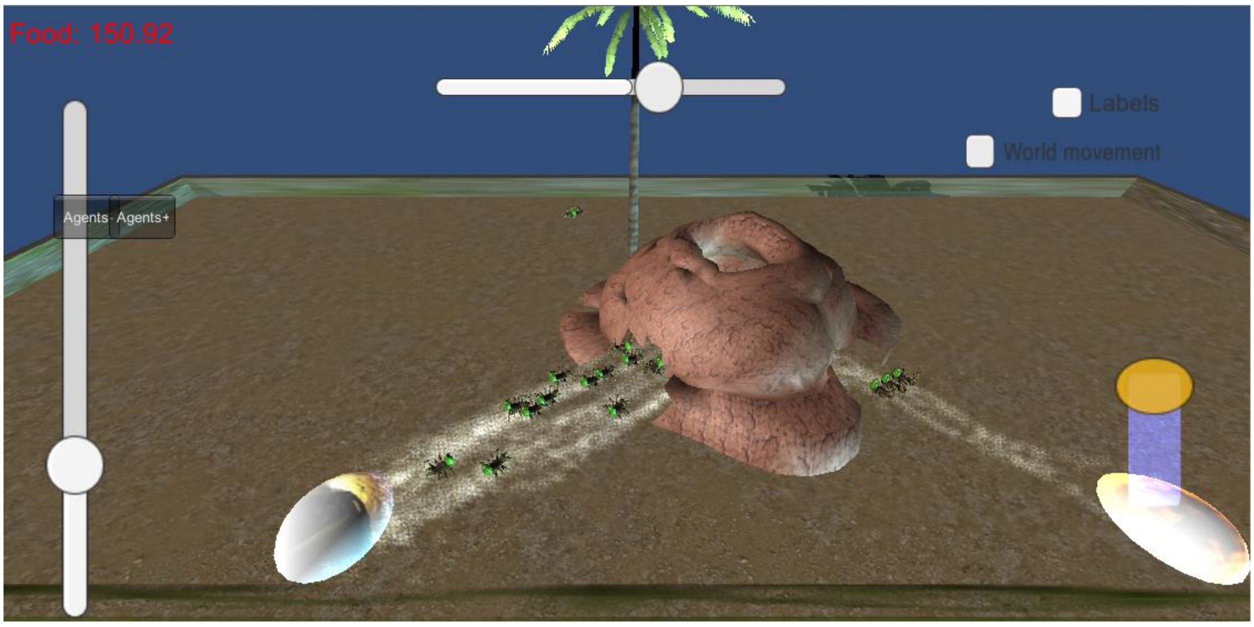


Fig.5: Experiment result showing different pheromone density and uneven distribution of the agents

The pheromones were implemented using a particle emitter attached to the agent container. If a collision between an agent and a pheromone particle is registered, the agents have the possibility to follow the trail to the food source, but only if it is not currently gathering food by itself. If an agent delivers the food to the colony and encounters another pheromone trail, there is a *p* probability that it will change the food source with the new one.

The emission of particles is enabled only when an agent found food and is returning to the colony.

The user also has several controls at his disposal, as can be seen in fig.5. They have a basic design and will probably look different or will be completely changed in the final version of the project, within a more polished graphical user interface (GUI).

We used a spider mesh in these simulations, but further developments will include an ant model or perhaps an array of agents and models that can be chosen before the simulation starts. We also observed several emergent behaviours, like swarming, route optimisation and even line formations (fig. 6)

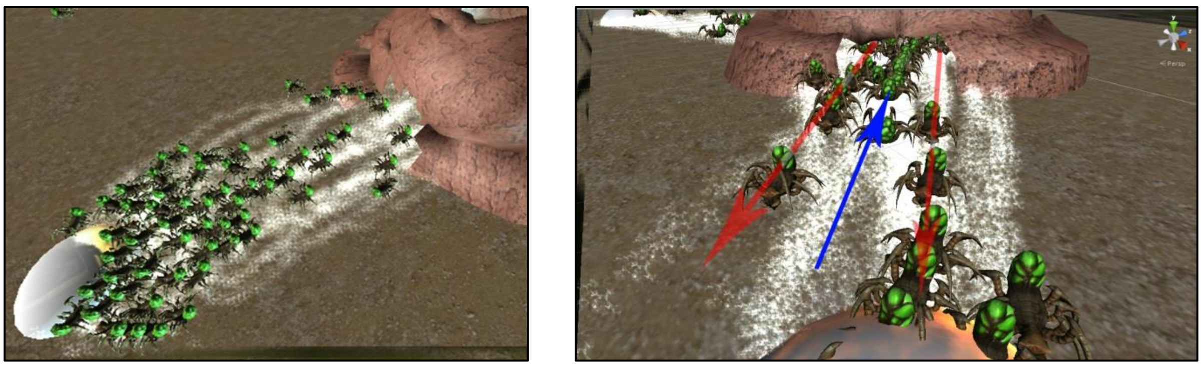


Fig.6: Swarming and line formations

4. Conclusions and future development

In conclusion, most of the objectives have been met and we successfully created an agent-based simulation using a 3D graphics engine. We were able to observe emergent behaviour and the adaptability of the ant colony algorithm. In the final iteration we created an augmented reality application and used it to run our simulations on any surface by using a specific marker. The latest version of the platform can be accessed at http://botashop.ro/va/.

Future developments follow:

- Modular design
- New computational models

Acknowledgement: This paper was realized with the support of POSDRU "QUALITY, EXCELLENCE, TRANSNATIONAL MOBILITY IN DOCTORAL RESEARCH" ID 187/1.5/S/155383 project, co-funded by the European Social Fund.

REFERENCES

[1] Computational model, Wikipedia article, http://en.wikipedia.org/wiki/Computational_model (accessed on: 10.02.2015)
[2] Agent-based model, Wikipedia article, http://en.wikipedia.org/wiki/Agent-based_model (accessed on: 10.02.2015)
[3] Grimm, Volker; Railsback, Steven F., Individual-based Modeling and Ecology, Princeton University Press, 2005, p. 485.
[4] F. Bota, Game Based Learning, Project Sotirios, Proceedings of the second International

Students Conference on Informatics, "Imagination, Creativity, Design, development", ICDD 2012, Sibiu, Romania, pp. 44-50, 2012.

[5] Paul Ormerod, Butterfly Economics: A New General Theory of Social and Economic Behavior,Faber and Faber Limited, 1998, p. 50-52.

[6] Alan Kirman, Ants, Rationality, and Recruitment, The Quarterly Journal of Economics, Vol. 108, No. 1, Feb., 1993

[7] M. Dorigo, Optimization, Learning and Natural Algorithms, PhD thesis, Politecnico di Milano, Italy, 1992

[8] X Hu, J Zhang, and Y Li, Orthogonal methods based ant colony search for solving continuous optimization problems. Journal of Computer Science and Technology, 23(1), pp.2-18, 2008

[9] Unity official website, http://unity3d.com/ (accessed on: 4.11.2014)

Florentin Bota,
"Babeş-Bolyai" University
Faculty of Mathematics and
Computer Science
Mihail Kogălniceanu, nr. 1, Cluj-Napoca,
Romania
botaflorentin@cs.ubbcluj.ro

Dan Dumitrescu,
Department of Computer Science,
Faculty of Mathematics and Computer
Science,
University Babes-Bolyai,
Cluj-Napoca, 3400, Romania
ddumitr@cs.ubbcluj.ro