

**————————————————————**

Weekly Assignment in *Spatial Simulation (2)*

**————————————————————**

Lecturer： Wallentin, Gudrun

Stu-Name： Chen, Yuzhou

Stu-number： s1104123

Major： Applied Geoinformatics

Winter semester, 2023

Assignment #2 Calendar week 41 / 2023, Winter semester Course Date: 12.10.2023

## Introduction

GAMA has wide-ranging applications in modeling complex ecological systems and various aspects of biological behavior. The task of this assignment was to simulate the behavior patterns and growth status of cows associated with the time steps. In order to better depict the typical characteristics of this species, I designed a model called “Ass2CowsModel”.

## Methods

Given the basic information in the instructions, cows’ initial state, behavior patterns, growth changes, removal of old members, and increase of new members should all be taken into account. There is a code provided that has 1 global part and 1 species called “cows”. At the very beginning of the code, global part is defined to do initiation works, which indicates the birth, age and the ready status of the 30 cows, where the age was randomly assigned by “age<-rnd(0,7);”. I choose to use a “cows\_num” variable to receive the quantity of cows rather than using a fixed number in the create statement, which may enhance the portability of the program. When it comes to the “species” definition for cows, the first thing is to add “skills:[moving]” outside the curly brackets, so that “wander” action can be triggered, combing with speed and amplitude attributes. Another actions cows would take in a new time step are as follows:1) grow older(that’s normal); 2) shout, like “mooooooo” ; 3) age\_report, reporting current age of an individual; 4) die\_catch, if older than 15 years old, with a fresh-baby cow coming into birth.

I consider the order consequence of these functions carefully and draw a conclusion that, *age growth* must be the first change in a new time step through “update:age+1;”. Then *die\_catch* follows to judge the live, birth or death of the cows agents in a new year, which may have a profound impact on the cows’ number and the following output. Then reflexes named *shout*, *move*, *age\_report* come into use, which show the actions of cow agents individually. So why these 3 specific reflexes arranged to the end of the section? Because all these outputs are determined only at the time when cows’ growth status are settled. And it is really a challenge to code the *die\_catch* reflex, which needs to identify a dying cow and create a new one based on it. That means a fresh-baby cow with age 0 will “come into birth” when a cow reach 15 years old. In other words, it can also be regarded as the appearance of a 1 year old cow in the next year, when the dying cow last year truly passed away. This can help to maintain the overall cows population unchanged. I take *if* control structure to find proper cows, use *create* and *ask* statement to create and initialize a new agent(with all action copies inside *ask* to ensure the new one act as normal), and finally “do die;” for dying one. When it comes to visualization, there are 2 parts requiring to be set: The *aspect* inside *species* format setting, where I differs the color on the age of 5. And the *display* set in *experiment* part, using “output{display map{species cows aspect:default;}}” according to the document.

## Results

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  |  |  |  |  |
| Fig 1. Initialization | Fig 2. First step | Fig 3. GUI of 1 step | Fig 4. New gene. | Fig 5. GUI of new gene. |

\* Gene. stands for generation

## Discussion

With the help of the practice on the model construction with “cows”, I now get a much deeper insight into the execution order and the coding structure of GAMA. From my point of view, the relationship between species and agents can be regarded just as the relationship between class and objects in ***Java***, which separately reflexes the generality and the specificity of a normal object in the world. Also, the *init{}* part matches with the self-defined *constructor*, while the *reflex* statements match with *methods*. Nevertheless, the main difference lies in the number of instances, which can be fixed in *.gaml* but not *Java*, so that we can get access to the specific instance(object) with *ask* facet and order number in GAMA.