

**————————————————————**

Weekly Assignment in *Spatial Simulation (4)*

**————————————————————**

Lecturer： Wallentin, Gudrun

Stu-Name： Chen, Yuzhou

Stu-number： s1104123

Major： Applied Geoinformatics

Winter semester, 2023

Assignment #4 Calendar week 45 / 2023, Winter semester Course Date: 13.11.2023

## Introduction

A variety of built-in statements and solutions provided by GAMA can support the construction of the interactive movement simulation of the agents. The task of this assignment is to learn the basic knowledge and operations of *Geometry* type and *field*, and to simulate the spatial interaction among several cows considering the distribution of grass biomass in the Vierkaser pasture area. To better depict the whole region and cows’ movement, I designed a model named “Ass4-CowGrazeModel”.

## Methods

Given the basic requirements in the instructions, there are 3 essential constituents to be taken into account: **1)** **The based field region**, including both graze area and shrubland; **2) The biomass field region**, indicating the nutrition value; **3) The moving cows**, which always identifying the harvest spot of grass as the target. The field areas mentioned above can’t be visualized without the help of Geometry data, which can be defined from external files like *.geojson*. There is a code provided presenting a global part, 1 species (cow) and 1 experiment part.

At the beginning, the global part is defined to: 1) Import *geojson* files, using *file* type to catch the value; 2) Convert *file* into *envelop* or *geometry* by homonymous operators, using *geometry* type to receive the value; 3) Initialize biomass field and attributes, including the initiation of area range using “field biomass <- field(ceil(shape.width/5), ceil(shape.height/5),0.0)”, regrowing height in each time step and the initial height of each kind of the grazeland using *loop*; 4) Give birth to 5 cows with random locations, usingbuilt-in variable *location* and method *any\_location\_in()* to achieve; 5) Define the process of biomass growth running in each time step, which can be done by *loop s* statement like “loop s over: biomass cells\_in <my\_geom>{}”, a proprietary iteration method for *field*. It is worth noting that the range of grazeland is a combination of 4 different grass covers, so its *geometry* should be obtained by summing up the geometries of the four kinds of grass. At the same time, these 4 kinds of grasses varies in biomass contents, so they should be looped and assigned respectively. When it comes to the definition of cow *species*, the key issue is how to identify the field cell with the maximum biomass within the range. I defined a *reflex* named *graze* to solve this problem. Firstly, a range variable named *range* and a *geometry point* named *best\_spot* needs to be declared while initializing, indicating the moving range and next destination for cow’s moving. Secondly, to get the explicit position of the spot, we should form an expression as the logic follows: 1) Get the moving neighborhood of the cows, containing the position of the grid where the cow is now and the maximum range it can move to in next step; 2) Collect the cells’ value within the neighborhood, using *collect* operator; 3) Maximize the biomass in reach, considering both the range and the biomass contents, using *with\_max\_of* operator and built-in variable named *z*. Based on the comprehensive analysis above, the expression is as follows, where “shuffle()” works to shuffle the order of the list of centroid point from *collect* operator “best\_spot <- shuffle(((biomass cells\_in (self + range)) collect (each.centroid))) with\_max\_of each.z;” Then, let the cows *do move* to their destination towards the *best\_spot* and eat grass using minus. In the experiment part, *field* needs to be visualized by *mesh* statement with green color to make it clear.

## Results: Model running 2000 steps with different Regrowth\_height.(No logistic model)

|  |  |  |
| --- | --- | --- |
|  |  |  |
| Fig 1. Regrowth\_height = 0.0005 | Fig 2. Regrowth\_height = 0.002 | Fig 3. Regrowth\_height = 0.005 |

## Discussion

1. Additional issues considered in my code: The *logistic* *model* of growth. *Growth rate next step= k⋅x⋅(max−x).*

2. According to this coding task, I found that both *field* and *grid* can express geographic grid, but different from grid agent, field is just a light-weight variable type, which can only be initialized in global. I think, the current way of defining and assigning field will cause the global part to look complicated, is it possible to manage field as an object like form for unification as well, such as in a code block?