

**————————————————————**

Weekly Assignment in *Spatial Simulation (5)*

**————————————————————**

Lecturer： Wallentin, Gudrun

Stu-Name： Chen, Yuzhou

Stu-number： s1104123

Major： Applied Geoinformatics

Winter semester, 2023

Assignment #5 Calendar week 47 / 2023, Winter semester Course Date: 23.11.2023

## Introduction

A Unified Modeling Language (UML) diagram works as a partial representation of a system model, which contains graphical nodes connected with edges. The task of this assignment is to learn the basics of the UML notations and design a conceptual model using UML Activity diagram with agent-based components. To better visualize the conceptual model of the cattle-grazeland ecosystem, I designed a UML diagram in the form of a sequential flow chart, which named “CattleGrazingEcosystem”.

## Methods

There are 2 species that interact with each other in the ecosystem. Considering that grass, measured by biomass-content, occupies the lowest level in the food chain, it functions as a producer, while cattle makes a living by consuming grass, establishing a producer-consumer relationship between these two. In general, cattle and grazeland agents in our ecosystem have their own characteristics and behavioral patterns, which can be represented by *Action* components in UML notation. These behaviors, depending on the execution mode, will produce different results, which are commonly distinguished by *Decision Nodes* in UML notation, and even, other characteristics will produce corresponding positive/negative feedbacks on the results determined by *Decision Nodes*. In fact, *Decision Node* is also used as *If* conditions in a normal flowchart. For biomass Object, the most important characteristic is biomass-content. Considering that plants in nature have a maximum growth height, if the grass is still at the maximum height, it will *Maintain* the current condition, but if there is a partial loss, it will trigger the *Grow* behavior, which will be limited by the *Logistic Growth Model*, with the following formula: *dN/dt=r\*N\*(1-N/K)*, where *N* is the number of populations, *t* is the time, *r* is the intrinsic growth rate and *K* stands for the max capacity. As a result, the line graph shows an S-shaped curve. When it comes to the definition of of cattle, there are 3 sequential actions that cattle would take as follows:**1) find spot**; **2) move**; **3) graze**. While cattle are searching for a spot, they will always obtaining the optimal intake. Once the cattle have found the best spot, it will move towards it and start grazing. During this time, the physical state of the cattle cannot be ignored, where I use a characteristic named *Energy* to define in the UML notation: if the cattle does not find a suitable spot, it will be starved, leading to a greater grass intake in subsequent *Graze* action and vice versa.

## Methods (continue) \*Result

|  |  |  |
| --- | --- | --- |
| After recognizing all characteristics and actions of both *Biomass* and *Cattle* species, we need to add the *Control Flow*, an arrow-shaped component, to connect all existing nodes, creating 2 parallel sequential *Swimlanes* these species. At the same time, conditions or *yes/no* statement need to be attached to these arrows, thus making the UML notation easier to understand. Then, we can visualize the producer-consumer relationship. The *Graze* action of the cattle will lead to a partial absence of the biomass-content inside the corresponding *field*, which in turn makes the *Grow* action of the grass inevitable. Therefore, in the UML flowchart, the arrow is drawn from the *Graze* action of cattle and ends at the *Decision Nodes* that determine whether the bio-content is the maximum value. | |  |
|  | Fig 1. UML diagram: CattleGrazingEcosystem | |

## Discussion

While drawing the UML of an ecosystem model with species interactions, we can view each species as an abstract Class in Java. So that, the core components of the UML notation are the attributes and methods of the Objects, and the judgement condition corresponding to the control flow is mainly distributed in the *reflex* functions of the species. I think that’s why GAMA recommends to write each action in a different *reflex* function while programming.