PARIS
LODRON
UNIVERSITÄT
SALZBURG

End report of *Spatial Simulation*

Lecturer:	Wallentin, Gudrun
Stu-Name:	Chen, Yuzhou
Stu-number:	s1104123
Major:	Applied Geoinformatics

Winter semester, 2023

Contents

Background introduction

GAMA is an open-source platform specifically designed for developing, running, and analyzing Agent-Based Model (ABM). Coded in Java, GAML makes it possible for us to build integrated models, do parameter calibration works and generate visual results within the platform.

One of the main advantages of GAMA is that the models are driven by the quality and accessibility of datasets. GAMA can not only support normal file types like CSV and image, but also has the capability to manipulate and utilize GIS data as the base map or self-defined Species for modeling. Shapefiles, grids and even 3D files are all supported in this platform, which can be implemented by some built-in statements and related geographical data types. This platform has been designed for utilization across various application domains, including but not limited to the climate change, disaster evacuation strategy design and urban related research. We users can develop models tailored to each specific scenario considering the powerful functionality and declarative user interface of the platform, and what we have done throughout this semester showcased the versatility of them.

Similar to other languages, gaml has a strict order of execution implemented by initiation and reflex actions, and can also be impacted by global definitions, species and grids. When it comes to the modeling practices, GAMA plays an import role in simulating the attributes and behaviors of creatures in ecosystems, like what OOP languages do. In gaml code, different creatures can be defined as different ‘Species’ with their own moving patterns or lifecycles based on the time steps, independent of the others. Even if in the same species, all the members can be treated individually. Apart from user-defined actions, GAMA has provided us with some built-in actions to make the coding work easier. In real-life scenarios, species have their different interaction patterns, such as Predation and Prey or Symbiosis. By utilizing built-in capabilities for modeling movement and geometry, the model can visualize how these animals perceive and interact with their surroundings based on factors, like speed, preferred directions and sensing abilities, thus demonstrating the patterns and potential interactions among these animals in their environments. Additionally, the interaction can also be built between species and the ground elements. For the representations of agent-based model design, UML diagrams is a good choice, which use nodes and edges to show how different parts and actions of a system are connected.

To sum up, GAMA platform is suitable for providing solutions for various modeling requests from the natural ecosystems and human society.

1# Order of Execution

1.1 Introduction

The task of this assignment was to get start with GAMA coding and comprehend the order of code execution based on the given model "OrderofExecution", which is logistically complicated. The answer was find by executing the given code step by step.

1.2 Methods

There are mainly 4 code parts included in the given model: 1) Global part; 2) Species definition; 3) Grid definition; 4) Experiment part. Apart from these basic parts, there are also 2 action type, which is *init* and *reflex.* While executing the program, the order sequence may not only be determined by the priority of statement parts, but also depends on the actions. Once clicking on the Execution button, the first block of output can be seen in the *Console*. And every new click will give a new output block in the console. The model runs as follows:

- In the initiative step: 1) **Experiment**, similar to the 'main' entrance of programming language like C & Python, this is where the program begins; 2) **Grid**, a special type of agent differs from regular agents(species), which can be initiated automatically at the beginning of the simulation. The CA grid has a size of 2*2, indicating that CA consists of 4 instances; 3) **Global**, a way to define global elements. It represents a specific agent called *world*, which is created and initialized first when a simulation is launched, where all the global variables and init{} sections are initiated; 4) **Species**, a prototype of agents, defining the attributes, behaviors and aspects of agents. Creating species is an essential part of the global statement (Fig 1; Fig 3).

- Let the simulation step ahead 1 *cycle*. All *update* and *reflex* statements are triggered in each time step. *reflex* can be written in global part and agent parts. **Global reflexes** take the top priority to execute, which may run with sequential order. **Agent reflexes** follows, which contribute to specific species. Needs to be mentioned, the grid and species have been created, so the execution order of reflexes won't follow the order in global init part but follow the code itself (Fig 2; Fig 4).

- How to rewrite the model: Agent_A should have values 1, 2, 3, 4 and 5.

Answer: Use "ask" operator for 5 times in global init block, give each agent in agent_A a specific value. For instance, ask A[0]->1, ask A[1]->2, etc.

1.3 Results

CA variable: 2 CA variable: 2 CA variable: 2 CA variable: 2 time step:0 global variable: 1 Agent_B variable: 1 Agent_B variable: 1 Agent_B variable: 1 Agent_A variable: 1 Agent_A variable: 1 Agent_A variable: 1 Agent_A variable: 1 Agent_A variable: 1 Agent_C variable: 1 Agent_C variable: 1 Agent A variable from global: 2	time step:0 global variable: 2 Agent_A variable: 3 Agent_A variable: 2 Agent_A variable: 2 Agent_A variable: 2 Agent_A variable: 2 Agent_B variable: 2 Agent_B variable: 2 Agent_B variable: 2 Agent_C variable: 2 Agent_C variable: 2 CA variable: 95 CA variable: 4 CA variable: 4 CA variable: 4	CA variable: 0 CA variable: 1 CA variable: 0 CA variable: 1 time step:0 global variable: 1 Agent_B variable: 3 Agent_B variable: 3 Agent_B variable: 6 Agent_A variable: 1 Agent_A variable: 2 Agent_A variable: 3 Agent_A variable: 4 Agent_A variable: 5 Agent_C variable: 0.0 Agent_C variable: 0.0	time step:0 global variable: 3 Agent_A variable: 3 Agent_A variable: 4 Agent_A variable: 5 Agent_A variable: 6 Agent_A variable: 7 Agent_B variable: 5 Agent_B variable: 5 Agent_B variable: 8 Agent_C variable: 2.0 Agent_C variable: 2.0 CA variable: 97 CA variable: 3 CA variable: 2 CA variable: 3
Fig 1. Initialization (Origin)	Fig 2. Step ahead (Origin)	Fig 3. Initialization (New)	Fig 4. Step ahead (New)

1.4 Discussion

The value of individuals can differ from each other in one species. The initiation of agents' value in a species is executed sequentially, not in parallel, enabling us to specify a special initialization sequence for the agents. Described as asynchronous in Javascript.

2# Cows behavior model

2.1 Introduction

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”.

2.2 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. 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. 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. Another actions cows would take in a new time step are as follows:1) grow older(that’s normal); 2) shout; 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.

The conclusion is that age growth must be the first change in a new time step. Then die_catch follows to judge the live, birth or death of the cows agents in a new year. Then reflexes named shout, move, age_report come into use, which show the actions of cow agents individually. All these outputs are determined only at the time when cows’ growth status are settled. While coding the die_catch reflex, I need 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 reaches 15 years old, which 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. If control structure is taken 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 the color on the age of 5 (Fig 5-9).

2.3 Results

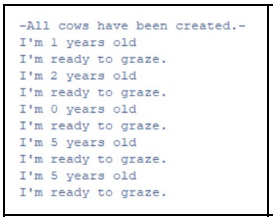

Fig 5. Initialization

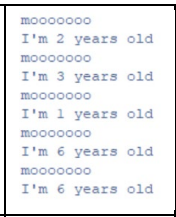

Fig 6. First step

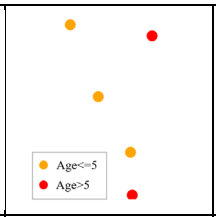

Fig 7. GUI of 1 step

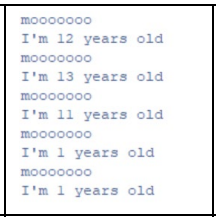

Fig 8. New Gene.

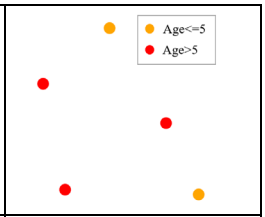

Fig 9. GUI of new Gene.

2.4 Discussion

The model practice provide a deep insight into the execution order and structure for me. I think 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 number of instances 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.

3# Species Movement model

3.1 Introduction

The task of this assignment is to track the movement of 3 species and visualize their action neighborhoods associated with the time steps. In order to better depict the typical movement and covering area, a model named “Ass3-MovementModel” is designed.

3.2 Methods

There are 3 typical intrinsic types of movement in GAMA coding, including wander, move and goto, which can be called by *do* statement. These movement can’t be visualized to action neighborhood without the help of Geometry data types. At the beginning of the code, a global part is defined to initiate cows, sheep and goats. Each species consists of two parts: reflex and aspect. Where reflex is used to define actions, specify the range of movement or perception, and assign values to geometry features. In this assignment, each move pattern matches with a movement behavior of a species: 1) cows do wander; 2) sheep do move; 3) goats do goto, leaving different covering areas compared to each other. Cows wander within a limited range of 90° with a speed of 2, using amplitude facet to specify, so the intersection is used to create a logical AND sector region by circle and cone with heading direction. Sheep move continuously to the south by setting the heading angle and can smell wolves at a distance of 3. Move action can be tracked using line feature with a length equal to the speed. The position of wolf can be defined as a point, just in-line with the sheep. Goats goto the origin. Unlike the others, goto needs a target facet while being called, thus indicating a fixed direction for the goats. So in every time step, goats move in a straight line, which can be create using “circle(speed)” to cut the line that runs from current position to the origin. In visualization, the predefined aspect types of each species can be called in experiment part and properties like “transparency” can be customized (Fig 10-12).

3.3 Results

Fig 10. Result of Time Step 2

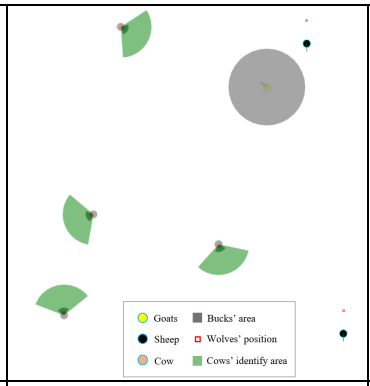

Fig 11. Result of Time Step 4

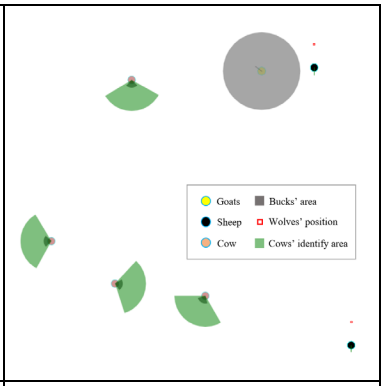

Fig 12. Result of Time Step 7

3.4 Discussion

1. The aspect statement is akin to the font family in Python chart drawing. In GAMA, this feature enables users to customize a graphical style and call it in the experiment using key-value pairs of aspect facet, where the value represents the name of the family.
2. In this assignment, sheep move in a due direction. What if there is an angular deviation on the direction? The vector “{0,speed}” should be calculated according to the specific angle with trigonometric functions.

4# Cows Graze model

4.1 Introduction

The task of this assignment is to learn the operations of Geometry type and field, and to simulate the spatial interaction among cows in the Vierkaser pasture area. To better depict the cows' movement, a model named "Ass4-CowGrazeModel" is designed.

4.2 Methods

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 and attributes, including the initiation of area range, regrowing height considering logistic model and the height of each kind of the grazeland using loop; 4) Give birth to 5 cows with random locations; 5) Define the process of biomass growth with a proprietary iteration method for field. Since the range of grazeland is a combination of 4 grass types, they should be looped and assigned respectively. The key issue is how to identify the field cell with the maximum biomass within the range, so a reflex named graze is defined. Firstly, a range variable and a best_spot point need to be declared, indicating the moving range and next destination for cow's moving. Secondly, to get the position of the spot, an expression should be formed following the logic: 1) Get the moving neighborhood; 2) Collect the cells' value within the neighborhood using collect; 3) Maximize the biomass in reach. Then, let the cows do move to the best destination and eat. Moreover, field needs to be visualized by mesh statement with green color to make it clear (Fig 13-15).

4.3 Results

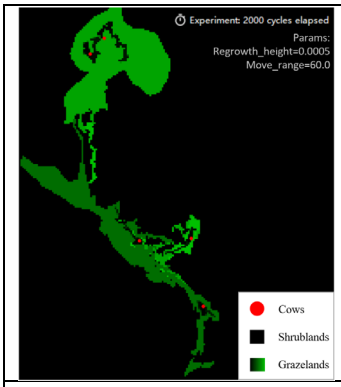

Fig 13. Regrowth_height = 0.0005

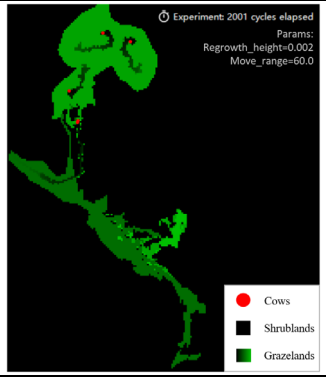

Fig 14. Regrowth_height = 0.002

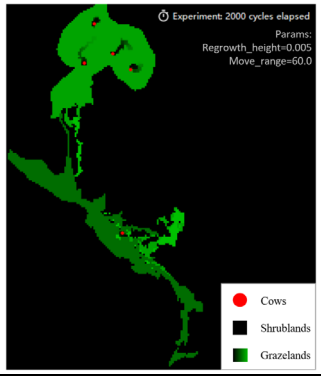

Fig 15. Regrowth_height = 0.005

4.4 Discussion

From this coding task, some differences are found between field and grid. Field acts as a light-weight alternative to grids, holding spatialized discrete values without having to build agent. So the field can cover the whole environment like grid. Actually, field was added in GAMA 1.9.1, which performs better than grid in larger sizes (above 500*500).

5# UML of Cattle Graze model

5.1 Introduction

The task of this assignment is to learn the basics of the UML notations and design a conceptual model using its diagram with agent-based components. To better visualize the grazeland ecosystem model, an UML diagram is designed in a sequential flow chart.

5.2 Methods

There are 2 species interacting with each other in the ecosystem. Considering that grass occupies the lowest level in the food chain, it functions as a producer for the consumer, cattle. Cattle and grazeland have their own attributes and behavioral patterns, representing by Action components in UML. These behaviors will lead to different results, commonly distinguished by Decision Nodes, and other characteristics will produce corresponding feedbacks on the results. Considering that plants will only grow when there is a partial loss, the Grow behavior is designed as a trigger, limited by the Logistic model where the line graph shows an S-shaped curve. When it comes to the definition of cattle, 4 sequential actions need to be taken:1) identify spot; 2) move; 3) graze; 4) gain energy. While searching for a spot, cattle will obtain the optimal intake. Once the best spot is found, it will move towards the spot and start grazing. During this time, the physical state of the cattle cannot be ignored, where a characteristic named Energy is used to define in UML: 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. After recognizing all characteristics and actions of both Biomass and Cattle species, we need to add the Control Flow to connect all existing nodes (Fig 16-17).

5.3 Results

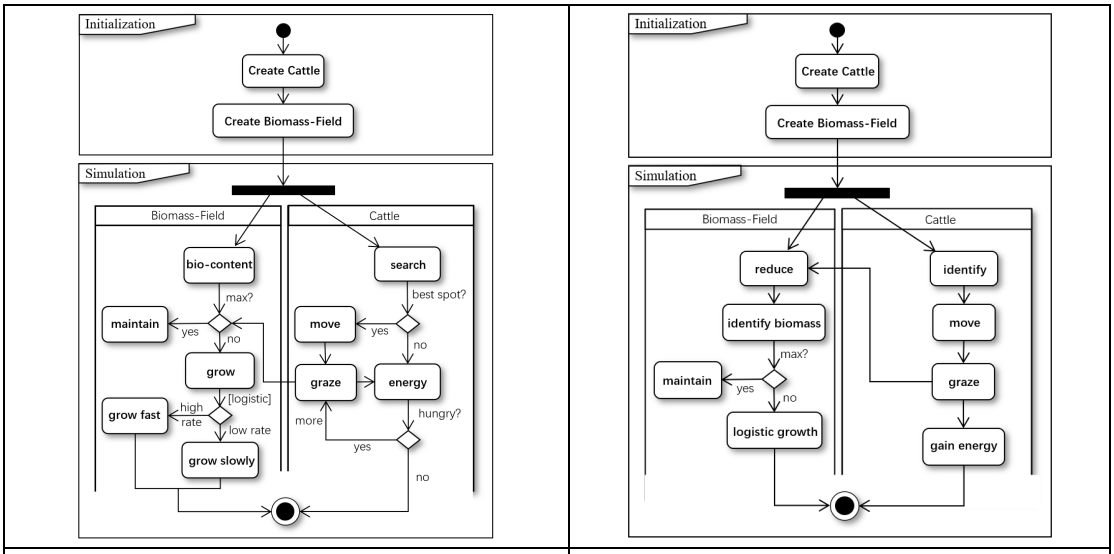

Fig 16. ***Old version*** of UML diagram

Fig 17. ***Revised version*** of UML diagram

5.4 Discussion

While drawing the UML of a model with species interactions, each species can be viewed as an abstract Class in Java. So that, the core components of the UML are the attributes and methods of the Objects, and the conditions corresponding to the control flow is mainly distributed in the reflex functions of the species. That's why GAMA recommends writing each action in a different reflex function while programming.

6# UML of Urban Growth model

[This document was collaborated by Yuzhou Chen and Haoyu Cao.]

6.1 Introduction

Urban growth is the spatial growth of city limits over time. The study of urban growth is of great significance in several fields like urban development and urban planning. Therefore, being able to accurately predict urban growth becomes a challenge for current society. Agent-based spatial simulation methods have considerable potential for modeling urban growth and support the input of geographic data. To discover the urban expansion of Salzburg, we design a scenario that contains many factors affecting urban expansion, such as surface slope, city center, roads, and water bodies (Kumar et al., 2021). The data inputs, simulation process, and data outputs are clear in the UML diagrams, presenting the simulation as UML diagrams helps in the subsequent development of the urban growth program.

6.2 Methods

Considering the requirements in the instructions, our project aims to simulate the urban growth based on the theory of an integrated urban index evaluation system. There are mainly 6 essential constituents to be taken into account: 1) The land use plots, indicating the extent of urban expansion; 2) The road, including different road types; 3) The slope, a limiting factor for urban expansion, especially in mountainous areas; 4) The city centers, consisting of hypothetical transportation hub locations, are indicative of the multi-center pattern that may result from urban development; 5) The water regions, no urban growth in these areas; 6) Neighbors of non-urbanized plots, which are the ranges dynamically delineated in the simulation to evaluate the contribution of the level of urbanization within the neighborhood. At each simulated time step, the model calculates factor-weighted composite scores for unurbanized points and filters the range of urbanization for the next time step on the basis of the scores and additional conditions, which may result in a long time simulation of urban growth in Salzburg(Fig 18-19).

When it comes to the Input Data of this project, all the datasets above can be imported into the GAMA model in the type of raster or vector files. So, the land use plots can be derived from the land use type map in Salzburg, 1830; the road can be a line vector stored in a shapefile with an attribute table indicating the road type (or maybe speed); the slope can be calculated by FABDEM in 30m; the city centers may be assumed by ourselves depending with some support evidence; the water regions can be derived from some basic geodata for Europe.

6.3 Validation patterns

Check that the model arithmetic logic is correct. As the simulation proceeds, it is crucial to ensure that the model logic is correct, and this can be done by means of a single-step adjustment program to ensure that the model does not make errors. The results should be checked at every fixed number of model steps, and errors are extreme, unchanging or recurring results.

Parameter sensitivity test. Changing the value of a variable while holding the values of other variables constant and observing the change in the simulation results. If the result is stable, it is indicates that it is making a valid contribution to the model. On the contrary, the variable maybe too sensitive under the current parameters and further consideration should be given.

Comparison with OSM or other data types. The OSM data provides a more direct representation outcome of the building areas. Therefore, the OSM data can be used as validation data. And the luminous remote sensing raster records the distribution and brightness of surface

lights, which can be regarded as reflecting the scale and economic status of the city.

Calibration and validation with existing land use data. Land use data is an accurate man-made data, in which the urban land can be completely equated to the area where the urban buildings are located, therefore, the land use data can not only be used to check the model results, but also to adjust the parameters of the model by comparing it with the actual urban area in the early stage of the simulation. In general, the goal of this process is to adjust the parameters in such a way that the sum of the differences between the simulation results and the actual results at the same locations in the raster values is minimized.

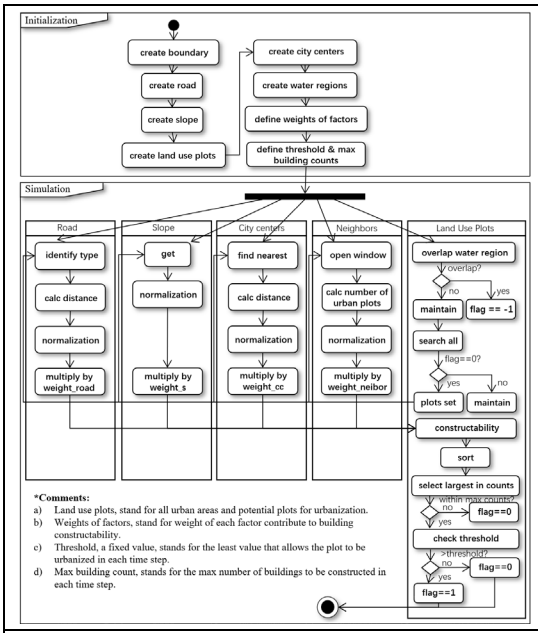

Fig 18. ***Old version*** of UML: Urban Growth Model

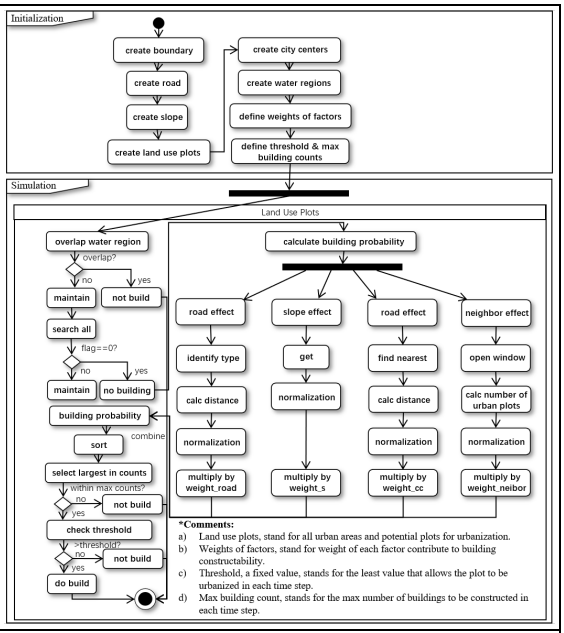

Fig 19. ***New version*** of UML: Urban Growth Model

6.4 Result

Scenarios: To explore the performance of the model in different contexts, we chose to conduct multiple sets of experiments by setting the value of the weights of a certain variable and the weights of the rest of the variables to 0, and finally compare the experimental results. For example, when exploring the effect of slope on urban growth, we set the weights of all other factors to 0, and then adjusted the factor weights of slope to observe whether there is a significant trend in the change of the results.

According to experience, the undulation of the terrain will greatly affect the process of urban sprawl, therefore, in the final results, we firstly reclassified the operation based on the slope, and secondly counted the number of buildings in different slope intervals, if the number of buildings in low slope intervals is high, then it is in line with our perception of urban growth.

6.5 Discussion

While discussing the issue of urban expansion under the influence of multiple factors, it's important to acknowledge that certain cities may also undergo the process of de-urbanization, characterized by a population shift away from urban areas towards suburban or rural regions.

6.6 Reference

Kumar V, Singh VK, Gupta K, Jha AK. 2021. Integrating Cellular Automata and Agent-Based Modeling for Predicting Urban Growth: A Case of Dehradun City. Journal of the Indian Society of Remote Sensing 49 : 2779–2795. DOI: 10.1007/s12524-021-01418-2

Final Project

6.1 Division of labour

Our team consists of Yuzhou Chen and Haoyu Cao. The brainstorming work about the model topic was done together.

The work Yuzhou did in this project: 1) Data collection and UML design, which refers to the locations of city centers and road network, and the fully design and revision of UML diagram. 2) Construction of the code framework, including the various data input and display, pre-definition of model parameters and definition of agent and species. 3) Result visualization, including the visualization of urban expansion processes and the statistical value in regular steps in graphs. 4) Parameter sensitivity evaluation. We input a group of parameters to generate a series of output rasters, representing the generic scenario of urban growth, then evaluated the output and the statistical value with different input parameters to evaluate the parameter sensitivity.

The work Haoyu did in this project: 1) Data collection, more specifically, it refers to process scope of the study area and calculate slope derived from DEM. Then, integrating other data sets, including origin building region, water region and areas outside the study area into one base raster as model input. 2) Programmed from the section that calculates the distance from the current raster to roads and city center. The distance was calculated in a reasonable way because we consider that the Manhattan distance applies to the city. 3) Validation with confusion metrics. We output the results of urban simulation once every fifty circle and evaluated the performance of the model by confusion metrics method comparing the 2014 Salzburg building data downloaded from OSM with the modelling results. Finally, a series of confusion metrics was calculated, which reflect not only the accuracy of final simulation results, but also the trends in simulation accuracy during the simulation process.

6.2 Introduction

Urban growth stands for the spatial growth of city within a period of time. The agent based spatial simulation solution provided by GAMA platform is capable of assuming the construct possibility of new buildings. However, along with the integrity of the urban structure and the increasing diversity of the functional areas of the city, the index system of the factors affecting the construction of the city has become increasingly complex. To simplify the scenario, several factors affecting urban expansion of Salzburg, including surface slope, city center, roads, and water bodies, were considered. The simulation of the urban growth model utilized the 25m raster data of the Salzburg city area as a base map, combining the contributions of the aforementioned factors to building construction, and depicting the expansion progress through the significant changes in the color of the cells in each time step of the simulation.

6.3 Data

Various types of datasets were utilized in this urban growth model, which mainly fell into 2 categories depending on the purpose. One is for the simulation of city expansion: 1) Building area of Salzburg in 1830s, which is the starting point of the simulation; 2) Surface slope, calculated by 25m DEM, which is a limiting factor for urban expansion, especially in mountainous areas; 3) Road network, guiding the extension of Salzburg with its length, direction and road level; 4) Points of city centers, contain the most attractive urban hubs in Salzburg, which are the indicators of the multi-center pattern; 5) Water bodies, indicating the

areas with no construction.

The other is for validations: 1) OSM urban land use data in Salzburg, derived from the OSM archives of Austria, mainly used for validating the simulation result, which needs to be transferred from Shapefile to Raster type; 2) Weight parameters, indicating the weight of each factor contributing to the building constructability.

6.4 Methods & Results

General introduction:

In this project, the urban area will extend along with the simulation steps, and the main idea of identifying these new building plots lies in the value of constructability. The urban land use of Salzburg in the 1830s is used as the initial state for simulation, and a binary-value raster with 25m spatial resolution is constructed based on the range of built-up and non-built-up areas, where a value of 0 represents the non-built-up area, and a value of 1 represents the built-up area.

In each time step, all grid plots representing non-built-up areas are identified to calculate their constructability respectively, and pick up some plots eligible for building construction according to the constructability. Since all the factors above are input as files of type raster and vector, which belong to the geographic element data type, the analysis of spatial topology and distance calculation between them can be achieved by the built-in operators in gmal for geographic elements, such as calculating the distance between a non-built-up point and a road using *distance_to*, and searching for the nearest city center point using *closest _to*.

In the model validation phase, the outputs of urban simulation are saved every 50 time steps (cycle) for the evaluation of the model performance using confusion metrics, which are calculated by comparing the simulation results with 2014 Salzburg building area from OSM. Additionally, our model performs several simulations with different input parameters to clarify how these parameters affect city growth.

My role and achievements in the project:

As is mentioned above, I take the responsible for the design of urban expansion's algorithm, data processing, model framework construction, visualization, analysis and parameter sensitivity evaluation. Moreover, based on the feasible model, I evaluate the differences of urban expansion between scenarios dominated by roads and city centers respectively.

While building the conceptual model, I proposed two urban expansion patterns. One is a regional growth model, similar to flood inundation, in which the extent of built-up areas at each time step is the result of searching outwards from the previous one. The other is based on scoring, which finds the location of non-built-up areas and combines various factors to assess their suitability for building. The second pattern is chosen witha scoring variable named *constructability*, which can be used to find built-up areas by setting a classification threshold. Generally, the urban expansion is the result of multifactorial influences, so the constructability is defined as a weighted average of factors including road accessibility, road level, city center accessibility, slope cost and surrounding building density, with corresponding weights.

Based on the theory above, the construction of the model framework started. Firstly, the model consists of 4 agents. 2 of them are grids, including a base land use of Salzburg (plot) and a slope raster (slope), others are species, including road network (roads) and city center positions (centers), imported using *file* operator. Grid agents like the land use and slope are created outside the global part, while roads and centers are created inside the global init{} using *create...from:.....* Given that roads have a "level" field, *with* operator is added to read the field value. Secondly, I defined color list corresponding to the raster values of different land use

types, allowing the visualization of the land use status to be updated over the timeline. Interactive weight parameters were defined. After extracting all the current non-built-up plots, I defined the attributes of each species and grid and designed some of the interfaces corresponding to their behaviors. Thirdly, normalization functions is defined to ensure that the cost factors positively correlate with constructability and different factors keep the same magnitudes. In the simulation, various params were set for running generic model (Table 1).

Table 1. Input parameters of the generic model

Para Name	Value	Meaning	Param Name	Value	Meaning
w_neibor_dens	*0.1*	weight of neighbor density	*w_slope*	*0.55*	weight of slope cost
w_road_level	*0.05*	weight of road level	*prob_threshold*	*0.8*	min constructability
w_road_dist	*0.15*	weight of road accessibility	*plots_to_build*	*150*	max construction number
w_center_dist	*0.15*	weight of center accessibility	*radius*	*4*	size of surrounding area

After about 500 cycles of running, the model was stopped manually, mainly because the building number had reached its limit constrained by the parameter named *prob_threshold*, which stood for the minimum constructability threshold. In other words, all raster points with a constructability below this threshold can never be turned into built-up areas. So, in the simulation, I visualized the accumulated building areas along with the max and mean value of constructability in each time step (Fig 20). (*One abnormal value of Max Constructability occurred in "**Statistic of constructability**", so another line chart was drawn based on the same history data as an alternative in Fig 20). As is shown in the chart, the increase of building number slowdown at 300th time step, and the max constructability decreases from 0.9 to 0.8, while the mean constructability slowly declines from 0.73 to 0.68.

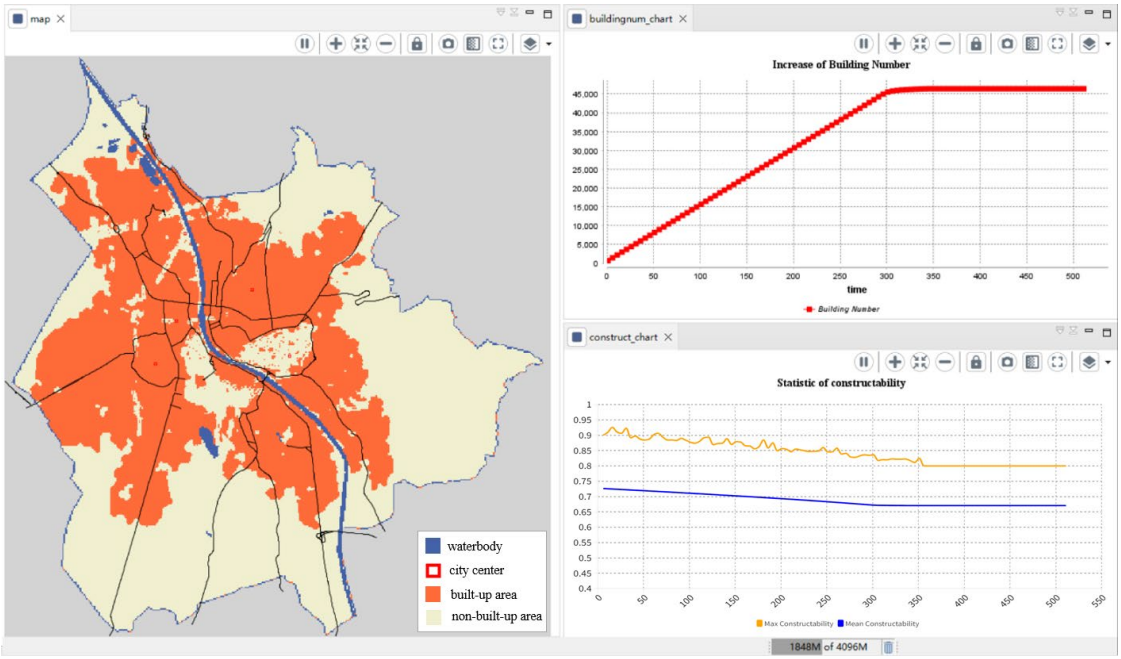

Figure 20. Simulated built-up area in general condition

After that, the simulation results with various *prob_threshold* input were evaluated to indicate how this minimum threshold influenced the urban expansion process. The values were set to 0.7, 0.75, 0.8, 0.9, respectively. The process of buildings number increase and the total area of final built-up zones were evaluated (Fig 21). Salzburg city covers an area of 65.65 km^2, when *prob_threshold* is set 0.7, the simulated built-up area occupies 80% total area of the city.

Threshold: 0.9 ->Total built-up area: 0.45km^2	Threshold: 0.8 ->Total built-up area: 29km^2

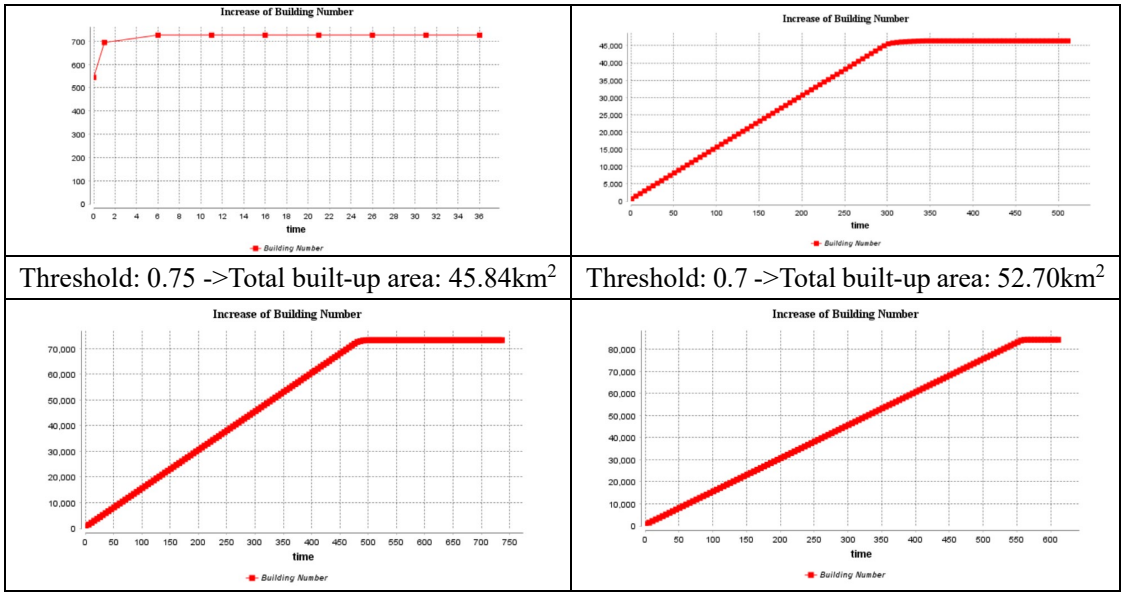

Figure 21. Sensitivity evaluation of the *prob_threshold* parameter

Furthermore, the model parameters are adjusted to simulate the urban expansion in 2 different scenarios, containing the Multiple Nuclei Model (Harris *et al.*, 1945), dominated by the cost distance from the city centers; the Hoyt Model, known as the Sector Model, dominated by the cost distance from the roads (Hoyt et al., 1939). In the simulation, the weights of some other unimportant factors are reduced accordingly. The consistency between simulated built-up area based on the Hoyt Model and surpasses that of the Multiple Nuclei Model. This indicates a stronger influence from main roads to urban expansion. However, the role of the city centers cannot be ignored, given the subjective nature of urban centroid placement (Fig 21).

Hoyt Model-core factors			Multiple nuclei model-core factors	
road level	road distance cost	slope cost	center distance cost	slope cost

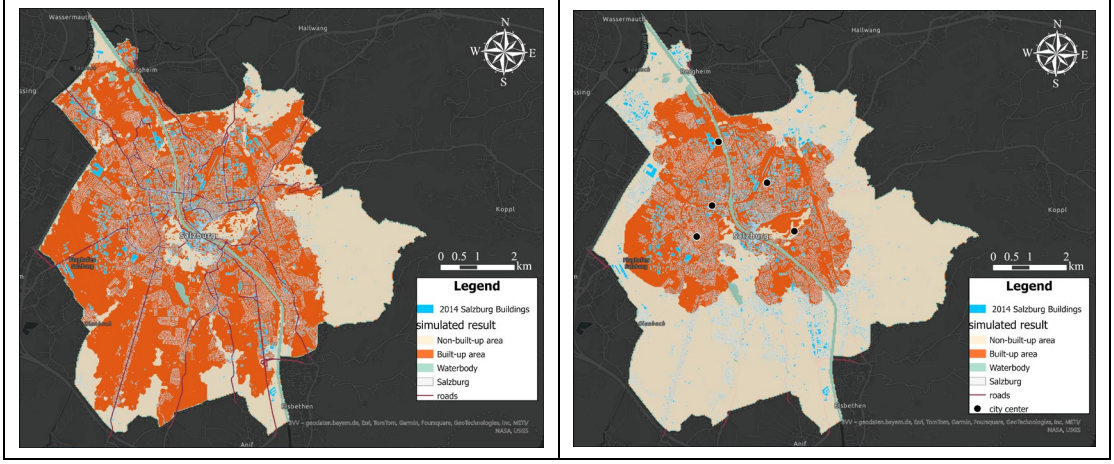

Figure 22. Comparison of simulated results from different theories

6.5 Discussion

In the global part, the code "geometry shape <- envelope(...)" is used to limit the geographical boundary of simulation. However, at the very beginning, I had used dem as the ‘envelope’, which resulted in the disappear of roads and city centers, after trying various solutions such as projection transformation and data format conversion to no avail, I extracted the vector boundary of the study area as the envelope and managed to visualize the roads and city centers.

6.6 References

Harris, C. D., & Ullman, E. L. (1945). The nature of cities. The annals of the American academy of political and social science, 242(1), 7-17.
Hoyt, H. (1939). The structure and growth of residential neighborhoods in American cities. US Government Printing Office.

Wrap-up contents

The provided tasks contain many patterns or instances of complex systems in the environment, mainly concentrating on the spatio-temporal changes, distribution, and interactions among various species. After finishing these specific tasks and generate reports based on my individual work, some achievements and deep insights about the spatially explicit modeling process are gained.

Firstly, the original task helped me to get the basic knowledge about the grammar of GAMA language in the form of .gaml. While comparing the coding hierarchical structure of .gaml file with some programming or scripting languages, such as Python, JavaScript and Java, some elementary problems with data types are solved. And after completing the 2nd task on modeling animal behaviors, I started to generate a deeper understanding on the 'function' concept, which may fall into different working pattens, like action, reflex, ask and other built-in movements, just as object methods and self-defined functions in Java or OOP coding pattern in JavaScript. They accurately represent behaviors and growth patterns of the species over time providing insights into ecological systems and the simulation of biological behavior.

Our understanding of how a species moves through space is not limited to where it is currently located. We start to consider the trajectory of the species and the extent to which a specific behavior influences its movement. During the coding process, obtaining trajectories for these behaviors might involve intersecting geometries, which adds complexity to the model but also makes it more realistic. For example, while determining areas where wolves exist, using point based identification is recommended instead of drawing lines.

Moreover, the geometry variable type and field operations are used to simulate the relationship between grazing cows in a spatial context in the following task. It explains how two important species, grass as producers and cattle as consumers have interactions that create a producer consumer relationship in a space or ecosystem. The benefits of utilizing fields for modeling are discovered in GAMA, providing valuable insights into optimizing simulations for larger scale scenarios and improving efficiency when representing complex spatial dynamics, within simulated environments.

After building UML diagrams for ecosystem models in the 5th and 6th, each species can be represented as a Class. This viewpoint emphasizes the importance of attributes, methods and control flow that correspond to functions of species in UML notation. All species contained in a space container will not only continue their own life cycle, but also interact with other species concurrently. This idea explains the relationship between UML notation and programming paradigms demonstrating how UML diagrams can accurately represent intricate ecosystem interactions and act as a blueprint for implementing agent-based models in GAMA.

Based on the lessons from tasks and outcomes, my own project is embarked, which involves studying the expansion of Salzburg city. The study on how cities expand under factors is a complex task where multiple elements may contribute to the growth with an unknown ratio. So abstracting the scenario into conceptual models can simplify the influencing factors in urban change, which is really crucial for coming up with strategies for urban development.