Paris-Lodron University Salzburg
Department of Geoinformatics

28.02.2022

End-of-term Portfolio

Spatial Simulation

Proseminar Course: Spatial Simulation
Lecturer: Prof. Dr. Gudrun Wallentin

Simon Meyer
ID: 01612616
simon.meyer@stud.sbg.ac.at

Table of Contents

1 Introduction & Background 3
2 Weekly Reports 4
2.1 Report – Weekly Task 1 5
2.2 Report – Weekly Task 2 6
2.3 Report – Weekly Task 3 7
2.4 Report – Weekly Task 4 8
2.5 Report – Weekly Task 5 9
2.6 Report – Weekly Task 6 10
2.6 Report – Weekly Task 6 - Revision 11
2.7 Report – Weekly Task 7 12
2.8 Report – Weekly Task 8 13
2.9 Report – Weekly Task 9 14
2.10 Report – Weekly Task 10 15
3 Discussion & Conclusion 17
4 References 19

1 Introduction & Background

Understanding complex and dynamic systems and detecting the driving factors can be challenging. This is especially true for systems that are made of many individuals that have complex behaviours and not only interact with the static environment but with each other as well. This falls true for human societies and ecology. The drive for understanding the underlying processes and relationships in these complex systems makes it necessary to develop respective models which can depict and explain these noticeable phenomena in our environment.

Spatial Simulation is one of these approaches. In contrast to other types of modelling, Spatial Simulation explicitly incorporates the spatial dimension which is necessary when analysing spatial patterns and individuals who interact in a geographic environment. The usage of Spatial Simulation additionally makes it possible for researchers to experiment with these systems in a controlled environment instead on having to rely on field experiments which can be expensive, unethical, or otherwise impossible to realize in real systems (Z_GIS, 2021).

The aim of this course is to experiment with selected types of Spatial Simulation models to get an understanding of the complexity of such modelling processes and to gain experience on modelling by creating new models. Within this course the focus was put on agent-based modelling (ABM) and cellular automata (CA) modelling.

Agent-based modelling (ABM), also called Individual-based modelling (IBM) is a computational modelling paradigm that has been developed and used to study complex systems which are composed of large numbers of interacting individuals. The basic concept of ABMs is the creation of induvial agents with defined properties and to simulate them in the following step to reproduce real phenomena. In this model, the individual agent has its own internal behaviour which reflects its perception and interaction with other entities. By letting the individual agents interact locally with simple rules, more complex dynamics in populations can be modelled. Examples for successful use of ABMs can be seen in the modelling of economic interactions, social phenomena, swarm intelligence and ecology. Advantages of ABMs, compared to other modelling strategies, are that agents are individually represented and monitored. This makes it possible to ask questions during the simulation like “what is the age distribution of the population?” and other statistical requests (Sotomayor et. al., 2020).

Cellular Automata models are grid-based models which work according to predefined set of rules. They were originally invented by von Neumann and J.H. Conway. Since then, different types of CAs have been developed. All CAs have four things in common, the grid, the neighbourhood, the states of a single cell and the local function. The concept of neighbourhood is crucial and distinguishes CAs from other mathematical models. Based on the definition of the rules, complex systems and pattern can be modelled, ranging from ecological models to fluid simulations (Muller et al., 2017).

The software used for the creation of the ABMs and CAs was GAMA in its version 1.8.1. GAMA is a modelling and simulation environment for building spatially explicit simulations. Scripting is done by using GAML, a custom high-level agent-based programming language. One distinct feature of GAMA is the ability to integrate GIS data into the models which allows the creation of agents based on existing feature datasets. This includes the possibility to connect to databases as well as to use various datatypes commonly used in the GIS domain, such as shapefiles, CSV files or a selection of raster datatypes. For the visualization, multiple model representations can be selected ranging from statistical charts, over classical maps, to complex 3D scenes (Taillandier et. al., 2019).

2 Weekly Reports

Until the 31. January 2022 five reports were produced, each covering the respective weekly tasks which had to be solved. The weekly tasks were structured to gradually introduce new and more complex features and mechanics of agent-based modelling. The topics and the goals of the respective weekly tasks are listed in the table below:

Weekly Task	Topic	Goal
1	Creating the first model	• Creating the first ABM • Create basic functions and agent behaviour • Create an aging process with conditions
2	Advanced coding principles	• Extend model by implementing lists • Update lists • Report agents' statistics
3	Moving herds	• Create multiple species • Create and visualize actions areas • Implement different movement types
4	Agent interaction	• Implement species specific agent interactions • Create basic predator-prey model • Read and load GIS datasets • Create perception areas • Create hunting and flight behaviours based on relative locations between agents
5	Predator-prey model	• Create new predator-prey model • Integrate charts that display statistical information about the species
6	Predator-prey model with species metabolism	• Addition of "energy" parameter to species • Attempt to stabilize the model
7	Predator-prey-grass model	• Addition of environmental component (grass) to the ABM • Combination of ABM and cellular automata • Stabilization of the populations • Oscillating species numbers
8	Basic cellular automata - Geyser model	• Create a geyser with water diffusion from a single origin cell • Water distribution over certain geographic space • Transfer of values between individual cells
9	Mountain well & basic runoff model	• Create a well and water runoff • Creation of rivers and lakes
10	River /flood modelling	• Create stable river runoff • Implementation of monitoring station and runoff statistics • Modelling of an extreme event • Model validation

2.1 Report – Weekly Task 1

Introduction

The task was to create a model of a lion population in GAMA. At the start of the experiment 30 lions should be displayed with a random age between zero and seven years. With every round the population should age a year. Lions below five years should be displayed in orange, lions over 5 years should be displayed in red. Agents over 15 years should die. Additionally, the lions should report their age in the console.

Methods & Approach

The modelling process was conducted in three general steps. The first step was to create and fill the global function which contains the init function and the reports that all agents have been created. The second step was to create the species function for the lions, which defines the agent's behaviour (random wandering movement, age and aging process, roaring, visual aspect). For the integration of the aging process, a new function called aging was created which contained an if…else condition. If the age of the individual agent was over 15 years the agent was killed with the "do die;" function, else the age was increased by one and the roar function was executed. To handle the colours, the aspect function was adapted to check the age. If the age of the agent was over five years, the colour was changed to red. The third step was to add the experiment function to create the display output, a map containing the species with its previously defined aspect.

Results

The result of the script was a basic agent-based model that consists out of 30 agents which move randomly on a map and age with every cycle until they die of old age (see Figure 1). After up to 15 cycles, all agents have died, as agents are only created with the initialization of the experiment.

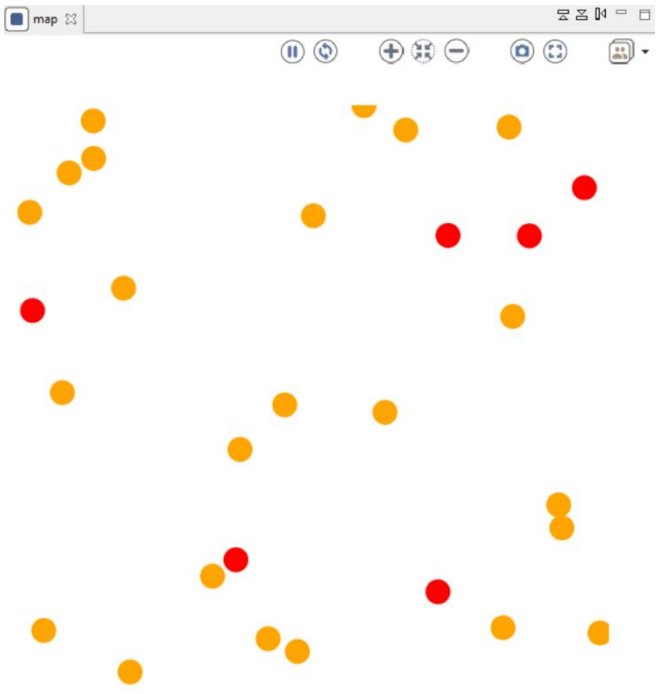

Figure 1: Result of the model. Agents over 5 years are displayed in red, agents below 5 years in orange.

Conclusion & Discussion

The model shows the basic characteristics of agent-based models. Individual agents are simulated based on predefined rules. In this case they move in a random manner and age until they programmed death. Additionally, some randomness is implemented by creating agents based on a range of possible starting ages (0-7 years). Based on the functions and rules used in this task, more complex species behaviours and therefore more complex models can be created.

2.2 Report – Weekly Task 2

Introduction

The task was to report demographic population data of the lions created in the week before in GAMA. This should be done by declaring a list containing the age values. With every time step the list should be updated accordingly, removing the old values and adding new age values. Additionally, population statistics should be reported in the console such as minimum, maximum, and mean values. In difference to the existing model, agent colours should be changed to represent the mean age of the population.

Methods & Approach

The modelling was conducted in GAMA by modifying and extending the existing script for the lion population. The list was added in the global function. To add the agent's age to the list, the init function was modified. Because this only adds the initial age of the agents to the list, an update function was added below the init which clears the age list after each time step. To add new age values for new time steps, another function was added to the species function which adds the current age of each agent to the age list. For the integration of the statistics report in the console, a function was added in the global function which writes the age list, means/max/min values in the console. Also, the lion colour was redefined here by using the RGB operator which was linked to the mean of the age list. To improve the visualisation, the mean age value was normalized by 12, as the age range does not cover the whole RGB value range.

Results

The result of the script was a lion population with fluctuating mean age as some lions get older and die, new lions are born. This was visualized by the new colouring which represents the mean age of the population. This colouring changed slightly showing mean values between around 8 and 11 years (see Figure 2 and Figure 3). In comparison to the model before, without the lion babies added to the model, the mean age was oscillating in the population, because younger lions lowered the mean age. In the long term, after about 870 cycles, the mean age seemed to level off at around 10.3 years.

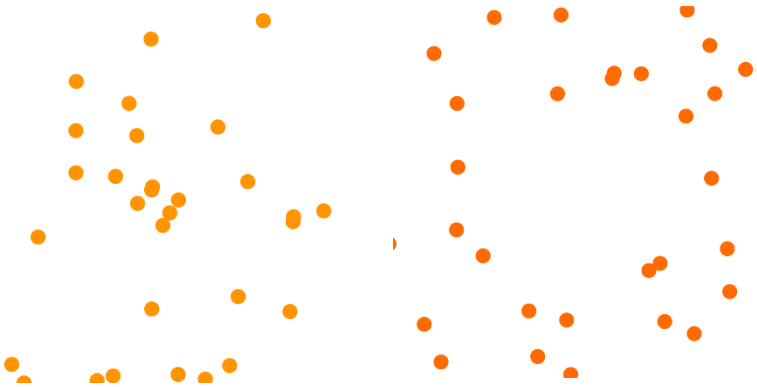

Figure 2 (left): Old population with a mean age of around 11 years.

Figure 3 (right): Younger population with a mean age of around 8 years.

Conclusion & Discussion

The statistics report enables the user to monitor the age of the agents in a list, which is constantly updated, which makes it easier to find interesting developments in the model. The colouring of the population shows an initial oscillation in the mean age of the population, as new lions are born, but after some time the oscillation levels off. This could be, because in the beginning there is only an aging process until the first agents die, this creates a high mean age, which is then countered by many young lions. This is balanced out after some time.

2.3 Report – Weekly Task 3

Introduction

The task was to create a new model in GAMA that simulates and displays three different species: cows, sheep, and goats. Each species should have a unique movement behaviour (correlated random walk for the cows, movement towards the south for the sheep, movement to the origin coordinate of the map for the goats) as well as unique movement speed. Additionally, the action neighbourhood should be visualized.

Methods & Approach

The modelling was conducted in multiple steps. The first step was to create the main functions (global function, functions for the behaviour of each species, experiment function defining the output of the simulation). Inside the global function, the speed variables of the individual species were defined and the init function was created which handles the initial creation of all agents (5 cows, 3 sheep, 2 goats). The code for each species was structured very similarly. First, the reflex function which handles the movement was added. After that, the action neighbourhood was defined by creating a geometry variable and an update function that updates this variable. Last, the visualization was defined with two aspect functions (one which defines the agent, one which defines the action neighbourhood. Finally, the experiment function was added, which adds a map as display output containing all species aspect functions.

Results

The result of the script (see Figure 4) was three animal species which were displayed in the map in different colours. Every agent had specific movement behaviour which depends on the species which it belongs to. During the simulation every species moved according to its programmed behaviour. The cows were moving randomly only restricted somewhat to their field of view, the sheep were moving towards the defined coordinate, and the goats were moving only southwards. The movement ended for the sheep and goats when their respective destination, the map frame, was reached.

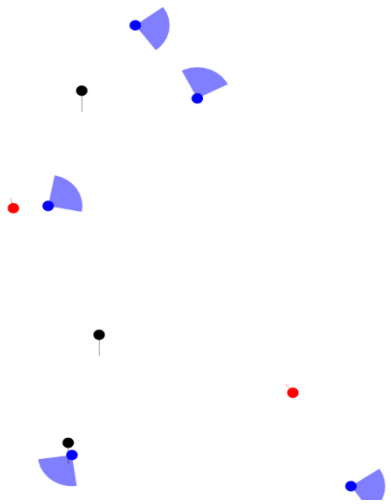

Figure 4: Result of the model. Cows are displayed in blue with a field of view. Sheep are displayed in black. Goats are displayed in red. Both sheep and goats indicate their path with a line.

Conclusion & Discussion

The model shows the different basic types of movement supported by GAMA. Based on these movement characteristics, more complex behaviours can be modelled. The action neighbourhoods visualize the ways which the agents can choose for their movement. Generally, these options are either linear paths to a certain destination or direction or based on a certain randomness with limiting factors (field of view).

2.4 Report – Weekly Task 4

Introduction

The task was to create a basic predator-prey model in GAMA (lynx and roedeer). For this, two new concepts were introduced. The first being the implementation of geographic data (shapefiles) into the modelling process. The second being the agent interaction between the different species. Lynx should hunt roedeers within their perception and roedeers should flee from lynx. Everything should happen within the defined habitat (forest).

Methods & Approach

The geographic environment was set up by loading the shapefiles into GAMA and creating a new geometry containing an envelope with the files inside the global function. Additionally, a new habitat geometry was created. Inside the init function, the new species were created from the loaded files and the habitat geometry was filled with the forest file. To visualize the forest on the map, a new species called "forest" was created which only contains the aspect function to visualize the habitat geometry. To archive the agent interaction, the lynx and roedeer species were given an action area (area where they can move) and a perception area (area where they can perceive other agents). Then, two movement types were implemented (walking and running). The walking movement was set to random for both species. In difference, the lynx running was defined to be a movement towards the nearest roedeer while the running of the roedeer was set to be a movement in the opposite direction of the nearest lynx. Both species move faster while running. To handle the change in behaviour a new function was implemented for both species. The lynx got a searchDeer{} function, that incorporates an if…else condition. If the perception area overlaps a roedeers location, then the running movement should be initiated, else the default movement is set active. The roedeer got an escapeLynx{} function that incorporates an if…else condition. If the perception area of the roedeer overlaps a lynx location, the deer runs away from the current lynx position, else it moves in its default movement. Unlike the roedeer species, the lynx species also contained a killDeer{} function that kills the deer that are within the action area of the respective lynx. To restrict the movement of the lynx and roedeer species to the habitat, bounds were defined in the functions containing the movement. This prevents agents from escaping the defined habitat.

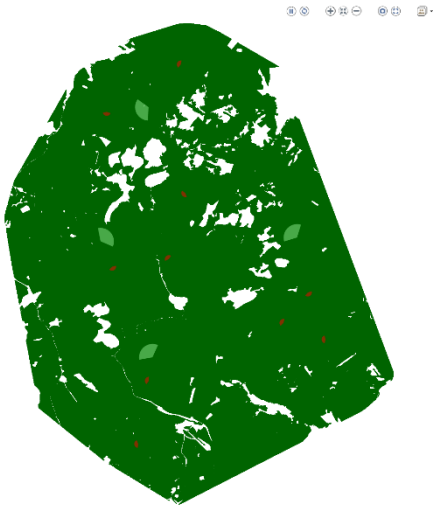

Figure 5: Final model. The forest is shown in dark green. The light green geometry represents the perception areas of the lynx, the red geometry the one of the roedeers.

Results

The result was a model that is limited to a geographic environment which incorporates real geographic data (see Figure 5). The different species have unique behaviours and react to each other in different ways by changing their movement type and speed. The roedeers try to evade the lynx and the lynx try to catch the roedeers. If the experiment is played, the lynx slowly kill all the deer. As the habitat is very big, this can take a few thousand cycles.

Conclusion & Discussion

The model of predator-prey interaction is a basic way of interaction between agents in ecology. Based on this type of modelling more complex ecological models can be modelled. It also implements environmental constraints into the map.

2.5 Report – Weekly Task 5

Introduction

The first task was to create a model in GAMA that implements a simple predator-prey model (lynx & roedeers) in a defined space of 1000x1000 units. The numbers of the agents were to be set to 20 lynx and 150 roedeers. Additionally, age variables should be defined for each agent. The second task was to visualize the obtained results with 3 different graphs. This should be implemented within GAMA with one chart showing the population numbers over time, one chart plotting the populations of the two species against each other, and one chart which shows age related information.

Methods & Approach

The modelling process was based on previous approaches. The two species were created with the code for the behaviour (hunting, flight) transferred from the previous task. Both species were programmed with defined action and perception areas. In difference to the previous tasks, the species functions were extended by an aging function which handles the age of each agent. Agents over 30 years were programmed to die. Every agent started with a random age between zero and seven years. The agents ages were documented in a list in the global function. The second part was to create the charts for statistics visualization. This was done by creating new experiment functions for each chart. The type of chart was then defined and the data values to be displayed.

Results

The results of the script were a basic map with the predator-prey model and three new charts containing additional information about the model (see Figure 6-8). It is visible that the number of deer decreases slowly as they are hunted until they rapidly die of old age. This development is not as clearly visible in the mean age of the populations where the high numbers average out single deaths.

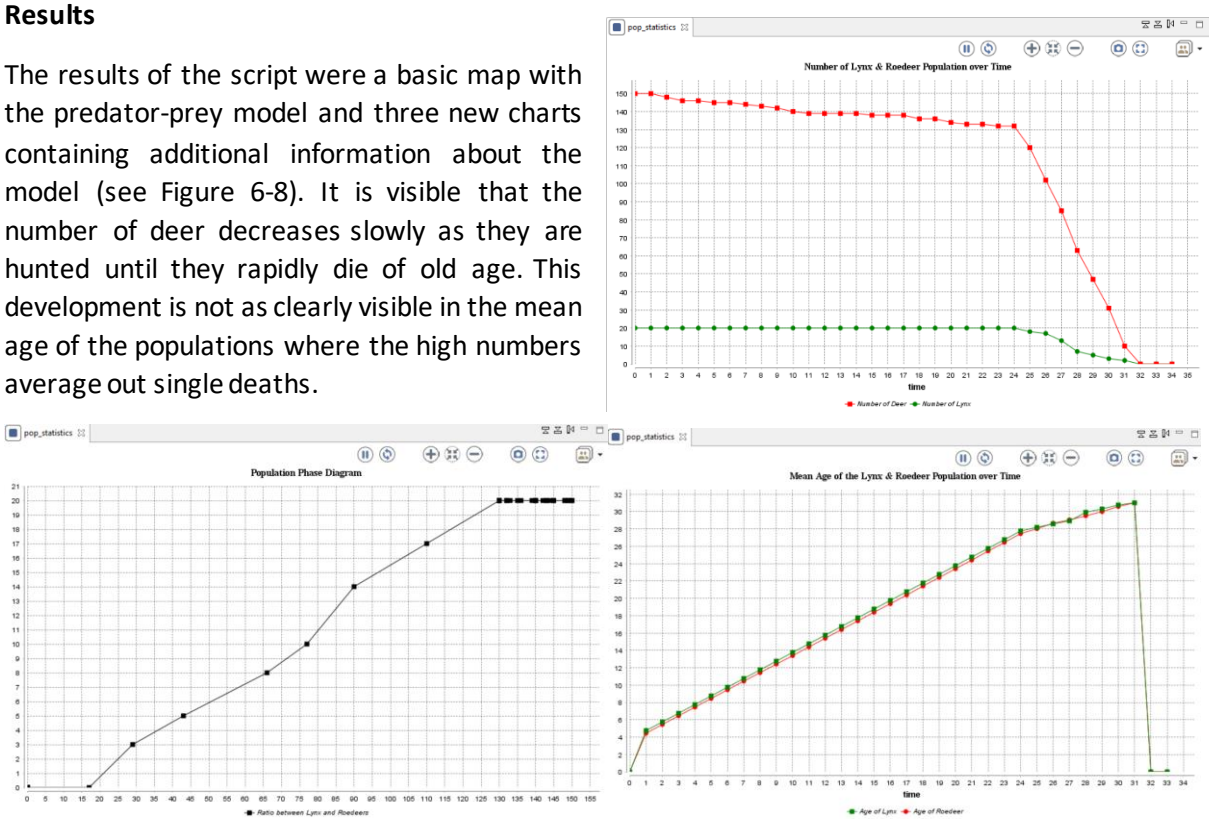

Figure 6-8: Statistics about the two species. Top: Comparison of populations numbers between lynx and roedeer. Left: Comparison between lynx and roedeers. Right: Comparison of mean age.

Conclusion & Discussion

The model shows possible visualization options beside the map view. Charts are a good way of conveying statistically relevant information and can support the model. Displaying charts about the agent's statistics is one strength of agent-based modelling.

2.6 Report – Weekly Task 6

Introduction

This task was about creating predator-prey population oscillations for lynx and deer. To archive this, a new parameter was to be added called "energy". Births and deaths of both populations should be governed by these energy statistics. Both species should lose energy through metabolism every time step. At zero energy the agent should die. Energy should be gained through grazing (when not hunted) for the deer and through killing prey for the lynx. At a certain energy level both species should give birth to a new agent, which transfer some of their energy to their offspring.

Methods & Approach

The general implementation of energy to the model was similar to the implementation of age variables in previous models. In the global function an empty list was added containing the energy values. In the init function every new agent was assigned a random energy value. Additionally, a new update energy function was written below the init to clear the energy list after a time step. Within the species function, a new energy variable was created. Below a new function with conditions was added that defined the energy of the agents. Every time step metabolism decreases the energy. If the energy reaches zero, kill the agent; if the energy reaches a certain (high) value, create a new agent, and remove energy from the parent. Additionally, in the killDeers function energy should be increased if the agent killed a prey. Alternatively, the prey should lose energy by fleeing from the predator and increase energy from grazing (default movement). Last, the current energy of the agent should be reported to the global energy list.

Results

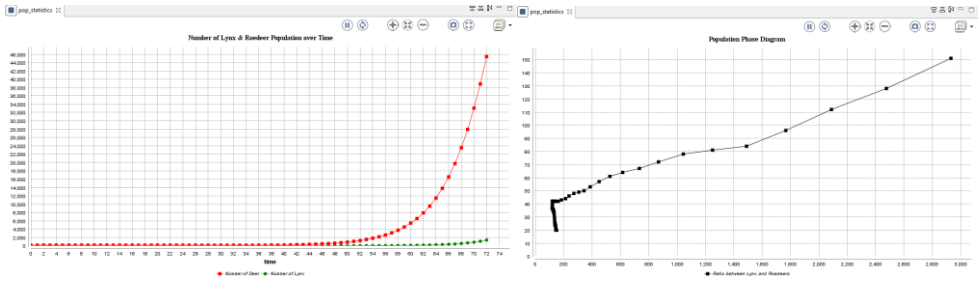

Figure 9-10: Population numbers over time (left) and Phase Diagram (right).

The results of the model can be seen in Figure 9-10. Both populations increase exponentially. While the deer population decreases in the first cycles while it increases its energy level, no further decrease can be seen. After a few circles the deer population has increased so much that it slows down the simulation. Therefore, no (potential later) decrease could be noticed. The oscillation stays unstable even when adjusting the energy parameters.

Conclusion & Discussion

With the parameters given in the task it was not possible to create a stable oscillation. Even adjusting the variables did not create a lasting change. This might be due to the fact, that the prey population is not bound to any environmental carrying capacity like grass. The result is an exponential increase in population, followed by an exponential increase in predators. But they can't catch up in numbers. This results in a linear graph in the phase diagram. A more realistic model would add an environmental limiting factor that controls the populations numbers of the prey.

2.6 Report – Weekly Task 6 - Revision

After progressing throughout the semester, the questions asked in the conclusion of the weekly task 6 (predator-prey model) could be answered. The problems experienced when trying to achieve a stable oscillation in population numbers was indeed caused by the missing environmental constraints of the prey population. The follow-up weekly task introduced a simple CA representing grass into the model. This reduced the extreme population increase and following depopulation, resulting in stable numbers, as seen in the following section. Especially interesting is the comparison between the two graphs of the phase diagrams, which widely differ. Interesting new questions would be the influence of additional environmental constraints on the model results, such as unpassable mountains or rivers or new population parameters such as births and eventual social components.

2.7 Report – Weekly Task 7

Introduction

The goal of this task was to modify the predator-prey model to incorporate an environmental factor in the form of grass. The grass should be implemented as a cellular automaton grid. Like previous models, the deer should gain energy by grazing. This time the grazing function should be integrated by interacting with the cells of the grid. Also, grass should regrow after some time. The results should be visualized in a map and charts.

Methods & Approach

The model was modified by adding a new grid function which adds a cellular automaton. Every cell was assigned a random value up to 256. Then a new grass growing function was added, which checks if the grass value has reached a value of 256, if not the grass value increases 5 values every cycle until it reaches the threshold. The grass cell value was then visualized by different states of greenness. Last, a deer grazing action was added, that checks the value of the nearest grass cell and sets it to zero if the grass is fully grown. If the grass is not fully grown, then the deer will not eat it and not gain energy. The energy gain from grass was increased to 2.

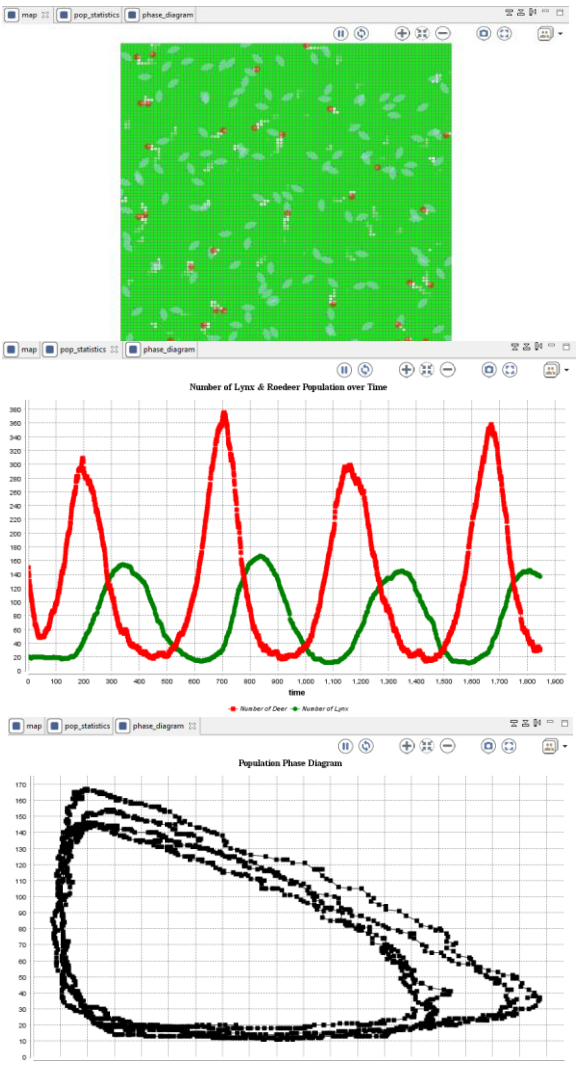

Figure 11-13: Results of the adjusted lynx-deer-grass model.

Results

The result of the model are stable predator-prey population dynamics, which can be seen in the Figures 11-13. On the map (Figure 1) deer leave behind grazed areas, that regrow after some time. This limits the potential number of deer in the area. In this model deer can only graze from fully grown grass but the amount of energy which can be gained from grass has been increased as well.

As the simulation is based on individual agents, the exact course of the population numbers varies to some degrees. This can be seen in the phase diagram where the lines are less precise in comparison to other diagrams created for example by mathematical models. The seemingly higher variance of the ABM in comparison to mathematical models seems to better represent the dynamics of real populations as they incorporate a certain randomness and unpredictability which can be observed in nature.

Conclusion & Discussion

In comparison to previous models, the predator-prey-grass model which includes cellular automation for the simulation of environmental factors (grass growth) stabilized the population dynamics and by that closer represents the dynamics observed in nature.

2.8 Report – Weekly Task 8

Introduction

The task was to create two simple cellular automata (CA) models. The first model should introduce the concepts of grids and grid values as well as neighbourhoods by creating a small grid with randomly increasing grid values and assigning neighbouring cells of cell [5,5] a red colour. The second model was intended to represent a geyser which breaks down and defuses water into its neighbourhood. The geyser cell should start with 4000mm height and diffuse its value into eight neighbouring cells (adding 10mm each time step to every neighbouring cell as long as its value is over 90mm).

Methods & Approach

Both models were created by first creating a new grid function and defining its neighbours. The first model was assigned the "von Neumann" interaction type, while the geyser model was set to a hexagonal grid structure. The first model was structured to include functions that increase the grid values randomly (0-10 values per cycle), update their respective colours, and handle the colour change of the neighbours. The geyser model was created by creating a diffusion function to check if the value of a cells is greater than 90mm. If yes, then the grid value of the neighbouring cells should be increased by 10 and the own grid value should be reduced by 10 times the number of neighbours. Also, the initial geyser height was defined in a global function and was added to the specific cell within a new init function.

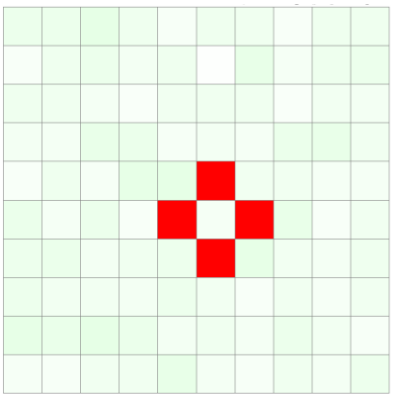

Figure 14: Basic CA model.

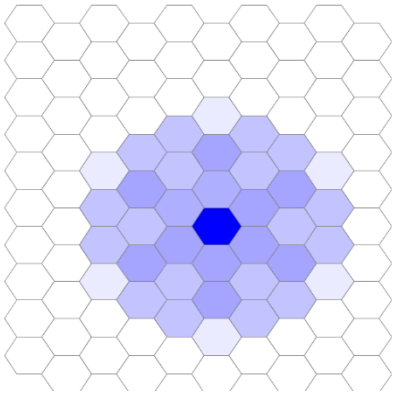

Figure 15: Geyser CA model.

Results

The result were two different models. The first model visualizes a basic grid with individual cells which increasing grid value. The values are shown by a green colour that is progressively getting darker (see Figure 14). Additionally, the neighbours of cell [5,5] are shown in red. The number of neighbouring cells changes when the type of interaction neighbourhood is changed ("von Neumann" to "Moore"). The second model shows the geyser cell and its water diffusion into the neighbouring cells (Figure 15). The neighbours then increase in grid value and diffuse water themselves increasing the spatial spread of water until the initial geyser value is spread so much over the grid that no more diffusion is possible.

Conclusion & Discussion

The purpose of these models was to introduce the basic characteristics of cellular automata models and concepts of neighbourhood, interaction, and different grid types. The results show emergent patterns based on simple rules. In the geyser model diffusion leads to the spread of water (grid values) over multiple cells until a steady state is reached. The outcome of the simulation also depends on the chosen neighbourhood type. Unfortunately, some irregularities could be observed in the geyser model, where the spread was spatially delayed in some cells.

2.9 Report – Weekly Task 9

Introduction

The goal of this task was to create a new model in GAMA that extends the previous geyser model by adding terrain and incorporating water run-off based on the terrain height. If water runs into a sink, the water level should rise until it overflows the bounding ridges.

Methods & Approach

The model extended the previous geyser model by adding data in the global function from an existing file ("local-elevation.asc"). Then a geometry object was created to display the elevation on the map. As in the previous model, the spring was initialized in the init function. In total three value fields were added. The elevation from the imported dataset, the water value which was empty except the spring cell and the total elevation which is the sum of ground elevation and water value(/height). In the grid species, a function was created which updates the total elevation each round. Additionally, two main functions were created: The first functions checks if enough water is available for a diffusion (more than 100mm). If true, the function "diffusion" is activated, which asks the surrounding neighbour with the lowest total elevation to take the water value of the origin cell (if its total elevation is lower than the origin cell). Also, the function changes the cells colour to a blue tone. The last function in the grid updates the elevations colour if the water value is greater than 0.

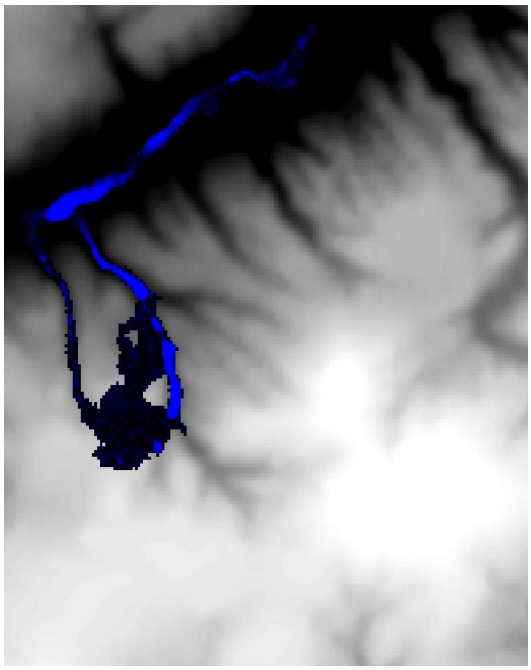

Figure 16: Result of the runoff model (terrain in black/ white; runoff from the spring in blue, with brighter colours when more water has been accumulated)

Results

The results of the model can be seen in Figure 16. The model diffuses water from a central cell (the spring) to neighbouring cells, depending on the elevation and water level. The value change happens between the origin cell and its lowest neighbour if the respective neighbour is located lower than the origin cell. This creates unique spatial patterns based on the geographic location of the well on the terrain. In the example, the waterflow split into two neighbouring valleys until it reunited to one single stream in the next lower depression. If the water flows into a sink, water value is accumulated until the elevation is high enough to spill into neighbouring areas.

Conclusion & Discussion

The runoff model is a simple way to model flows in a cellular automaton. It incorporates the concepts of neighbourhood, interaction and exchange and accumulation of different values of the grid's cells. Additionally, this model includes external data sources, which are used for the terrain values and the gird size. The model could be further refined by adding a drain at the lowest elevation, so the valley isn't flooded after some time.

2.10 Report – Weekly Task 10

Spatial Simulation – Final Project

Introduction

The goal of the final model was to create a new runoff/river model in GAMA. In difference to the previous models, this model should represent and simulate an actual historical flooding event of the Weisspriach river in the state of Salzburg, Austria. For this task, elevation data, sensor location, and precipitation statistics were provided. The main challenge was to integrate the datasets into GAMA, to implement the rain events (addition of water values to random cells), and to adjust the units of measurement to the resolution of the elevation model. The results should be visualized in form of a map and graphs, that measure water level and runoff at the defined sensor location. To achieve this task, several interim targets were set.

The first target was the basic runoff model where single raindrops accumulate in depressions and form rivers and lakes. The second target was to implement the sensor location in the river in the bottom of the valley that measures the amount of water passing by in m3 / sec. Then, the river height should be calculated and adjustments to the input values should be made until the measured values of the historical flooding event are matched. References of the historical flooding data were taken from the Hydrographical Yearbook 2018. Finally, comparisons between the model output and the observed flooding event should be made and the accuracy of the modelling approach should be validated.

Methods & Approach

The general code structure of model is based on the previous basic runoff model. It is divided into three parts: the global function, the grid species function, and the experiment function.

The global function contains the main functions that handle the loading of the external datasets into the simulation (Weisspriach DEM), the init function that initializes the DEM shading and the colouring of the cell which functions as measuring station, as well as a function that monitors the water level of the individual cells. The function that handles the rainfall is also located inside the global function. Rain generated in this model in form of a water value added to random cells in the grid. To achieve a steady river stream, a default rain (4 mm per cycle, as that matches the average daily rainfall measured over a year) is implemented that changes to increased rainfall (340 mm for one cycle) after 90 cycles to simulate the heavy rainfall event and further flooding.

The grid species function contains mainly the water runoff handling function. Runoff is defined by asking the neighbouring cell with the minimum total elevation (DEM height + water level). If the neighbour has a lower total elevation than the origin cell, then water value is distributed. Additionally, if the origin cell has less than 8 neighbours (Moore Neighbourhood is implemented) than the cell is drained. This creates drains at the edge of the DEM, so it does not act as a water container itself. The last function of the grid species handles the updating of the colours where cells are coloured in shades of blue if the water level is > 0. Otherwise, the cells are shaded according to the DEM values.

The final experiment function handles the experiment output. The output is defined in form of a map as well as a simple line chart that monitors the water level of the river. After running an initial simulation, values were adjusted, and the model was adapted to the target event to represent the flooding. For this, values were set to match the average precipitation per day over a year to fill up the rivers to a default level. Then, the rain event was implemented which changes the input for the defined cycle, causing a temporal increase in the water level. As a reference a rainfall event in October 2018

was chosen. The biggest challenge was to match the measured results in the model with the measured water discharge of the event.

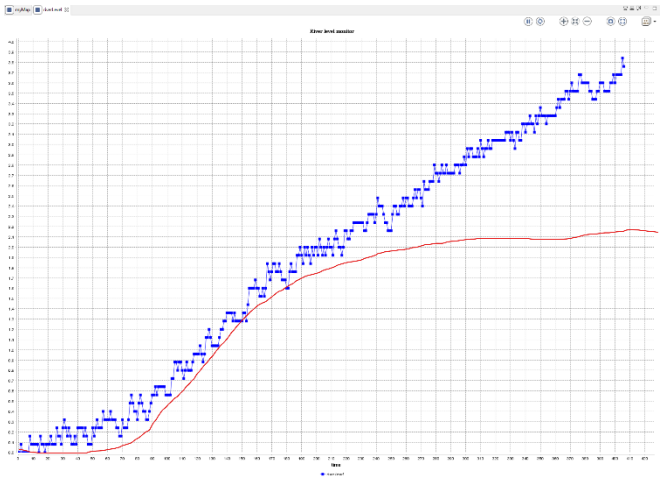

Figure 17: Displayed is the chart that monitors the water height (in mm) at the defined measuring station. The measured values are depicted in blue, while the red line roughly shows the expected progression of water height without any extreme event.

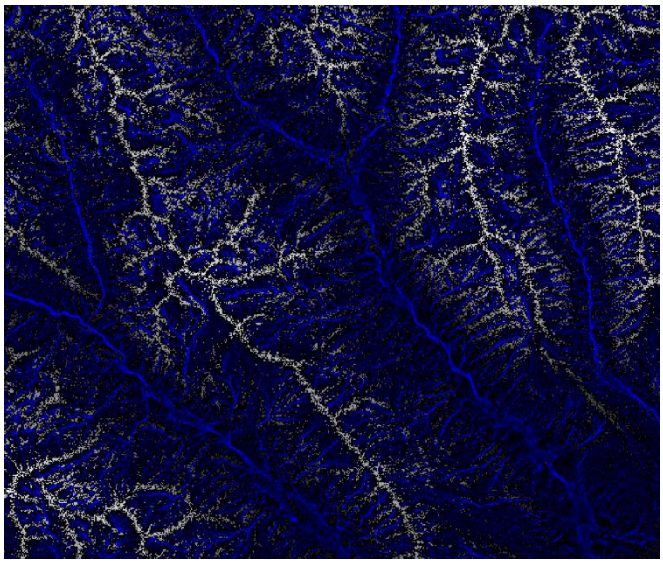

Figure 18: Geographic distribution of river streams and water accumulations in geomorphological depressions (lighter shades of blue indicate nigher levels of water.

Results

Unfortunately, the expected results differ from the outcome of the model (see Figure 17). In the model, rivers build up (visible in Figure 18) while the river height at the measuring station slowly increases, as additional water is accumulating. Slight spikes in the water level can be noticed. This is potentially related to the filling of sinks and other depressions that temporarily hold back water volumes until they overflow. In difference to the expected flattening of the increase to a threshold value in water height, the rise of water proceeds linearly. No stable equilibrium between the water input and output could be achieved. This meant, that the river level cannot be contained to a realistic height, which is crucial when measuring extreme flooding events as it functions as a reference. In theory any extreme rainfall event should then be visible as a significant spike in the water level, but due to the continious rising of the reference level no spike is visible.

Conclusion & Discussion

The conclusion of the results described previously is that modelling complex systems can result in unexpected outcomes where it is difficult to identify the exact cause of the error. Errors in simulations might be due to wrong input values or units of measurements, bugs or erroneous code, or simply because the environment has been simplified too much to get accurate results. As this simulation model only calculates on morphology and a random distribution of water onto the map, results could be inaccurate. Additional factors such as underground streams, soil saturation, different rainfall areas and fine geomorphological structures are not integrated into the calculation. But these additional factors can have a significant influence on the outcome. In conclusion, modelling real events in existing geographic spaces requires a lot of work and thinking. Corrections in the model code and changes to the input resolution of the DEM did not have a significant effect on the model outcome, therefore more complex problems seem to be the cause for the resulted output.

3 Discussion & Conclusion

Within the semester course, multiple models have been created. Every model has integrated new mechanics and features of GAMA to showcase the capabilities of different spatial modelling approaches. With every weekly task, the complexity of the models and their difficulty gradually increased. The first half of the semester was focused on the creation of ABMs, while the second half reserved for CA models.

In summary, the first two reports mainly dealt with getting to know the basic coding principles in GAMA and creating basic model frameworks, the following tasks built upon these structures to create the first interactive models by incorporating different movement types for different species and defining different behaviours for each species. Two following milestones in the creation of an ABM were the implementations of different movement types and the creation of action and perception areas. The creation of multiple movement types makes it possible to define differentiated movement patterns and thereby to create different behaviours. By creating action and perception areas, the agents get the ability to sense other agents and other species which is the foundation for agent interactions. Based on these features basic ecological models could then be created.

The first fully working ABM was created in 4. report when the interaction between the different species was implemented. The hunting and flight behaviours of the agents created a dynamic and emerging spatial pattern, where the prey was running away from the predators until they were eventually caught up and killed. This would not have been possible to recreate with other models, as these patterns emerge unpredictable (depending on where the agents are generated and how they move). The predator-prey models were further refined in the following weekly tasks by adding species metabolism parameters and environmental constraints (grass cells) which did affect population numbers. Through this, a stable oscillation in population numbers could be achieved in weekly task 7.

Another advantage of ABMs over other types of modelling was demonstrated in the 5. report, where statistics of the individual agents were displayed in various charts. This is possible, because every individual can have assigned properties like a specific age variable for example. This enables the user to create detailed statistics about the simulation. In case of the report, age and population numbers were displayed and compared.

CA was already introduced in the final predator-prey model as the environmental component to stabilize the prey population. Within the second part of the semester modelling focused on further possible applications CA. Basic value transfers between individual cells were introduced in weekly task 8 with a basic geyser model spreading water. Building on this, water runoff modelling was introduced, which was applied to a real-world scenario in the final project.

In conclusion, the individual weekly tasks have shown fundamentals as well as practical applications of Spatial Simulation. Both ABMs and CAs were demonstrated using practical examples, such as predator-prey models or flooding simulations. Especially interesting was the combination of both models in the predator-prey-grass model, where both models were in exchange with each other. These approaches laid the (potential) foundation for more complex agent-based and cellular automata modelling. This could be done by adding more agent properties, more types of behaviour, more interaction conditions, and more complex environmental conditions. Regarding CAs, a more defined flood model could be developed that takes in more data inputs or reflects different surface conditions (runoff on sealed surfaces versus grass for example). Besides the understanding and learning of various spatial models, the course finally also helped to understand the issues in validating models and unexpected results and troubleshooting errors in the code.

Regarding the broader field of Geoinformatics, Spatial Simulation provides tools to model and understand certain geographic problems or uncertain patterns in systems. This is especially true with complex systems. While ABMs and CAs can help understanding complex and dynamic systems, like social systems or various ecological systems, the incorporation of GIS data is a valuable feature. It makes it possible to create models based on existing representations of real-world objects and can reduce the work effort needed for preparing the simulation. Potential for a deeper integration of GIS and Spatial Simulation is also provided in CA which relies on the same rasterized grids as many remotely sensed datasets. While the ABMs and CAs used in this course may not provide a solution to all problems in Geoinformatics, they can help in modelling specific scenarios and improve our understanding geographic systems.

4 References

MULLER, J., HADELER, K.-P. (2017). CELLULAR AUTOMATA: ANALYSIS AND APPLICATIONS. SPRINGER MONOGRAPHS IN MATHEMATICS. SPRINGER

SOTOMAYOR, M., PÉREZ-CASTRILLO, D., CASTIGLIONE, F. (2020). COMPLEX SOCIAL AND BEHAVIORAL SYSTEMS. ENCYCLOPEDIA OF COMPLEXITY AND SYSTEMS SCIENCES SERIES. SPRINGER

TAILLANDIER, P., GAUDOU, B., GRIGNARD, A.,HUYNH, Q.-N., MARILLEAU, N., P. CAILLOU, P., PHILIPPON, D., & DROGOUL, A. (2019). BUILDING, COMPOSING AND EXPERIMENTING COMPLEX SPATIAL MODELS WITH THE GAMA PLATFORM. GEOINFORMATICA, (2019), 23 (2), PP. 299-322, [DOI:10.1007/S10707-018-00339-6]

Z_GIS (2021). SPATIAL SIMULATION. [ONLINE]. AVAILABLE: HTTPS://SPATIAL-SIMULATION.ZGIS.AT/ [ACCESSED 24.11.2021].