Introductory Guide to Agent-Based Simulation Development on the GAMA Platform[2]

Aline Rodrigues Santos[1], Fernando Santos[1]

[1]Departamento de Engenharia de Software
Universidade do Estado de Santa Catarina (UDESC)
Ibirama – SC – Brasil

aline.rs@edu.udesc.br, fernando.santos@udesc.br

Abstract. *The use of agent-based simulations is becoming common in the research field, being employed to abstract complex concepts through visual demonstrations. This has driven the emergence of platforms for developing these simulations. In this context, GAMA stands out as a good option due to its wide range of features. However, GAMA still lacks materials to guide beginner developers. With the aim of filling this gap, this paper presents an introductory guide to simulation development in GAMA. The paper describes the main functionalities and the structure for developing a simulation in GAMA. Beside that, the paper exemplifies these elements through the development of the 'Sugarscape' simulation, known in the community. Finally, the challenges that a beginner developer may encounter are reported, along with recommendations to address them.*

2

1. introduction

Agent-based simulations (ABS) have been used to replicate and study different phenomena. According to [Wilensky e Rand 2015], the adoption of ABS not only assists in visualizing complex scenarios but also in simplifying and abstracting them. An example of this approach is illustrated by [Costa 2023], who developed an ABS to demonstrate some of the main components of the complex Tupinambá society, allowing for the explanation of phenomena and demonstrating a tribal society that lived in Brazilian territory. Another example is the simulation of Covid-19 spread by [Teixeira e Santos 2020], which allowed for studying the effects of isolation measures on the pandemic.

[Kleiboer 1997] describes the different functions of ABS, two of which are research and support in teaching. This highlight the need for learning concepts and programming languages for ABS development. Although it is possible to develop ABS using general-purpose programming languages like Python or Java, the use of a specialized platform significantly simplifies learning and makes it more intuitive. For this reason, there are platforms that facilitate agent-based simulations development, such as NetLogo [Wilensky 1999] and, especially, GAMA [Taillandier et al. 2019].

[2]Copyright © 2024 for this paper by its authors. Use permitted under Creative Commons License Attribution 4.0 International (CC BY 4.0).

The GAMA platform offers a programming language called GAML (Gama Modeling Language). Through GMAIL it is possible to develop simple or complex simulations, such as 3D models that provide control over various visual aspects such as lighting, texture and camera position or just 2D simple models, like Sugarscape. Additionally, the platform prioritizes the development of intuitive interfaces and allows for the provision of different visualizations that can be customized and updated dynamically to observe the simulation progress. GAMA also allows specifying monitors to collect and display data about the simulation during its execution. The presence of these features illustrates how the GAMA platform can streamline the development of ABS [Taillandier et al. 2019].

Despite offering various elements, a beginner in ABS interested in developing simulations on this platform may face challenges. The available documentation[1] for GAMA is extensive; however, the explanation of the essential elements for developing an ABS is dispersed across different topics, which poses comprehension challenges, especially for beginners. The learning curve can be steep for those who do not have prior experience with ABS platforms, requiring additional time and effort to familiarize themselves with the necessary elements for implementation. Therefore, there is a lack of an introductory guide that presents the essential elements for implementing ABS in GAMA using a simple and widely known simulation. Despite the GAMA documentation offering a tutorial on implementing the *Predator-Prey* simulation, it provides few explanations about the platform, its organization, the reasons for use, and the functionality of each element used.

This paper presents a guide to developing ABS in GAMA, which is based on the development of the *Sugarscape*, simulation, widely known in the ABS community. It explains the essential elements for implementing ABS in GAMA and exemplifies them through the *Sugarscape* simulation. The paper also provides recommendations for addressing some limitations of GAMA perceived during the simulation development.

The remaining of this paper is organized as follows. Section 2 presents the required background on ABS, the GAMA platform, and the *Sugarscape* simulation. In section 3 we present the introductory guide to the GAMA platform throghout the development of the *Sugarscape* simulation. Next, section 4 points out recommendations for beginners to start developing in GAMA. Finally, section 5 presents concluding remarks and future work.

2. Background

This section introduces concepts used in this paper. Firstly, it presents what ABS are. Then, the GAMA platform is explained, along with some of its various features. Later, the Sugarscape simulation is described, highlighting the main parts of its development.

2.1. Agent-based simulation

An agent-based simulation (ABS) is a practical approach to studying complex systems [Klügl e Bazzan 2012]. This approach allows researchers to predict and to optimize, these systems more effectively. Essentially, the simulation offers a tangible way to represent and explore complex situations, where agents play crucial roles, such as individuals, animals, or entities, interacting with each other and with a predefined environment.

[1]`https://gama-platform.org/wiki/Home`

According to [Klügl e Bazzan 2012], an ABS is composed of three elements: a set of autonomous *agents*, endowed with rules governing their behavior; the specification of *interactions* between agents and the environment, responsible for producing the overall output of the system; and the simulated *environment*, which contains all other simulation elements, such as resources and other objects without active behavior. The result of an ABS can be observed through visualization of agents behavior or through graphs presenting data collected during simulation execution.

Because they focus on individuals, ABSs offer researchers the ability to make specific adjustments to agent behavior so that they can respond adaptively to the environment and other agents, influencing the observed results in the system under study. An example of this is the ABS developed by [Pereira et al. 2011] to study human behavior in natural disaster situations and how to improve the performance of rescue teams.

Additionally, ABSs are highly scalable and modular, which means that models can be easily adapted and extended to include new elements or details [Macal 2016]. This makes this approach especially useful for simulating dynamic and evolving systems, such as ecosystems, financial markets, or social networks.

2.2. GAMA platform

Agent-Based Modeling (ABM) platforms, like Cormas and GAMA, are specialized tools for creating agent-based simulations (ABSs), particularly focusing on explicit spatial representation. Cormas, developed in 1998, stands as one of the pioneering ABM platforms [Bousquet et al. 1998]. Similarly, GAMA, built in Java, is a prominent open-source ABM platform. These platforms aim to offer a scientific approach for modeling a wide range of scenarios, making them accessible to both scientists and non-scientists alike [Taillandier et al. 2019].

GAMA provide a programming language called Gama Modeling Language (GAML). Through GAML, it is possible to develop simple or complex simulations, including 2D and 3D models that provide control over aspects such as lighting, texture, and cameras. Additionally, the platform prioritizes the development of intuitive interfaces, enabling the provision of various customizable displays for the same model, dynamically updated to visualize the simulation [Taillandier et al. 2019].

ABSs for studying various phenomena have been recently developed in GAMA. [Gaudou et al. 2020] present the COMOKIT, an ABS developed in GAMA to analyze and compare interventions to deal with the Covid-19 epidemic at the scale of a city. [Daudé et al. 2019] present the ESCAPE ABS, to study mass evacuation strategies in crisis situations. These simulations highlight the potential of the GAMA platform.

2.3. Sugarscape

The Sugarscape is a model of an artificial society proposed by [Epstein e Axtell 1996]. It is widely used in the ABS community as an example of a simulation that exhibits emergent phenomena. In essence, the Sugarscape model simulates a population that depends on limited resources existing in the environment. The population consists of *ants* searching for food (sugar) present in the environment, which is composed of different regions, some rich and others poor in sugar. Each ant in the simulation is represented by an artificial agent with distinct attributes, such as vision, energy, and metabolism. These agents have the ability to move in the environment and collect sugar. The objective of each ant is to identify nearby regions with a higher quantity of sugar and that are free (without another ant). When a region is identified, the ant moves there. When an ant interacts with a sugar region, it consumes the resource, and this sugar may or may not have an immediate growth. The variation in resources allows for exploring population dynamics and their concentration in different parts of the environment.

In the model created by [Epstein e Axtell 1996], the environment consists of a set of 2500 cells, representing the regions, organized in a grid of dimensions 50 x 50. Each cell has a certain amount of sugar and a maximum storage capacity. The initial amount of sugar in each cell is defined to form two sugar peaks, in the northeast and southeast of the grid. An external file provides the initial amount of sugar for each cell. As the simulation progresses, the sugar consumed by the agents is replenished. Two replenishment rules are considered: *immediate replenishment*, where the amount of sugar consumed by an agent is fully replenished at each simulation step, and *partial replenishment*, where only a percentage of the initial amount of sugar is replenished at each simulation step.

Although simple, the Sugarscape simulation highlights, for example, the emergent ecological phenomenon of carrying capacity for a species—according to which a given environment can only support a finite population [Epstein e Axtell 1996]. This ABS was chosen in this work due to its simplicity and popularity. The NetLogo platform, for example, provides well-documented implementation of Sugarscape with the two mentioned sugar replenishment rules [Li e Wilensky 2009a, Li e Wilensky 2009b].

3. Development of ABS Sugarscape in GAMA

This section presents the development of the Sugarscape simulation on the GAMA platform. The essential elements for developing ABSs are explained through the Sugarscape simulation, making this section an introductory guide for ABS development in GAMA.

To develop an ABS in GAMA, it is essentially necessary to implement three elements: the global species, the regular species, and an experiment. Figure 1 presents a diagram with these elements, their relationships, and components.

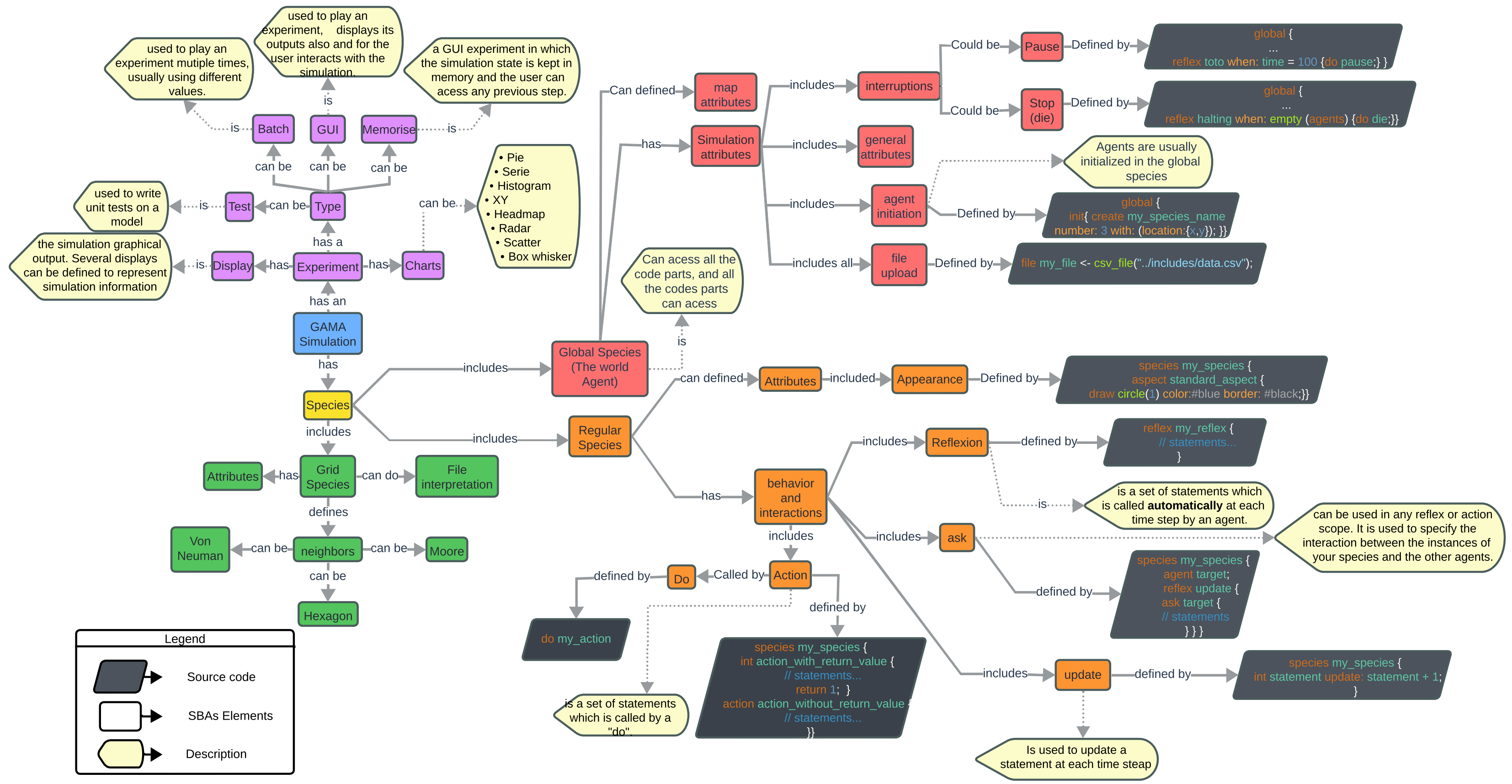
It is worth noting that the diagram in Figure 1 is a contribution of this paper. The GAMA documentation provides a learning tutorial[2] that only describes the general structure of the GAML language metamodel, not offering a concrete view of the elements required in the simulation and their components. GAML is still under development, which may explain its documentation. This can create difficulties for beginner developers to understand the structure of ABSs on the platform. The following will detail the essential elements of ABSs in GAMA and how they were implemented in Sugarscape. Due to space limitations, code snippets are not included in this paper. However, we provide the complete ABS implementation online[3]. We suggest that the reader download the ABS so they can verify the implementation of the elements as they are described.

3.1. Global Species

The global species, represented by the red color in Figure 1, plays a fundamental role in the simulation, being the main entity that defines the essential characteristics of the

[2]`https://gama-platform.org/wiki/LearnGAMLStepByStep`

[3]`https://github.com/agentbasedsimulations/sugarscape-gama-simulation`


GAMA Simulation
has an
Experiment
has a
Type
can be
Batch
is
used to play an experiment mutiple times, usually using different values.
GUI
is
used to play an experiment, displays its outputs also and for the user interacts with the simulation.
Memorise
is
a GUI experiment in which the simulation state is kept in memory and the user can acess any previous step.
can be
Test
is
used to write unit tests on a model
has
Display
is
the simulation graphical output. Several displays can be defined to represent simulation information
has
Charts
can be
• Pie
• Serie
• Histogram
• XY
• Headmap
• Radar
• Scatter
• Box whisker
has
Species
includes
Grid Species
has
Attributes
can do
File interpretation
defines
neighbors
can be
Von Neuman
can be
Moore
can be
Hexagon
includes
Global Species (The world Agent)
is
Can acess all the code parts, and all the codes parts can acess
Can defined
map attributes
has
Simulation attributes
includes
interruptions
Could be
Pause
Defined by
Could be
Stop (die)
Defined by
includes
general attributes
includes
agent initiation
Agents are usually initialized in the global species
Defined by
includes all
file upload
Defined by
includes
Regular Species
can defined
Attributes
included
Appearance
Defined by
has
behavior and interactions
includes
Reflexion
defined by
is
is a set of statements which is called automatically at each time step by an agent.
includes
ask
can be used in any reflex or action scope. It is used to specify the interaction between the instances of your species and the other agents.
defined by
includes
update
defined by
Is used to update a statement at each time steap
includes
Action
defined by
Called by
Do
defined by
is a set of statements which is called by a "do".
Legend
Source code
SBAs Elements
Description

```
global {
...
reflex toto when: time = 100 {do pause;} }
```

```
global {
...
reflex halting when: empty (agents) {do die;}}
```

```
global {
init{ create my_species_name
number: 3 with: (location:{x,y}); }}
```

```
file my_file <- csv_file("../includes/data.csv");
```

```
species my_species {
aspect standard_aspect {
draw circle(1) color:#blue border: #black;}}
```

```
reflex my_reflex {
// statements...
}
```

```
species my_species {
agent target;
reflex update {
ask target {
// statements
}}}
```

```
species my_species {
int statement update: statement + 1;
}
```

```
species my_species {
int action_with_return_value {
// statements...
return 1; }
action action_without_return_value {
// statements...
}}
```

```
do my_action
```

Figure 1. Essential elements of an ABS in GAMA

environment and agents. [Taillandier et al. 2019] define it as a *master species*. Only one instance of the global species can exist, and it is where characteristics such as values and methods accessible to all other species and the experiment are defined.

In addition to user-defined data, the global species incorporates attributes that define vital aspects of the simulation, such as the size of the map, as well as fundamental attributes like simulation interruptions, which can pause or terminate the simulation. These attributes can be accessed and modified by other species and the experiment, providing a flexible basis for configuring the simulation environment [Taillandier et al. 2019].

In Sugarscape, the global species plays an important role because it defines the general parameters of the simulation, such as the minimum/maximum values for some ants attributes. These parameters can be associated with visual components, allowing the user to input their values when running the simulation. Considering the heterogeneity of ants in the simulation, with frequently different attribute values among them (for example each ant may have a different range of vision or metabolic rate), the use of global parameters and their editing through visual components facilitates the execution of the simulation.

It is in the global species that the type of simulated environment should be defined. GAMA supports two types of environments: grid and graph. The grid environment resembles a matrix formed by cells, where it is possible to define specific values for specific regions of the environment. In GAMA, the attributes of each cell can be defined manually or initialized from files. When the simulation depends on external files to initialize the environment, it is in the global species that such files should be loaded. In the case of Sugarscape, we chose to initialize the environment from a CSV file, which provides the initial amount of sugar and the maximum storage capacity for each cell.[4].

The visual aspect of the simulated environment is also defined in the global species, where it is possible to implement which colors will be used for the elements of the environment. In Sugarscape, darker shades of yellow were defined to represent cells with a higher concentration of sugar and lighter ones to represent cells with a low concentration of sugar. This visual representation facilitates the interpretation of simulation results and provides important insights into the behavior of agents in relation to their environment.

Finally, in the global species we can define which data will be collected during the simulation. In Sugarscape, it is necessary to collect data on the average vision and average metabolism of ants, as implemented by [Li e Wilensky 2009a]. The collected data aggregate values from various agents, highlighting the importance of the collection being performed in the global species, which has access to all elements of the simulation.

3.2. Regular Species

The regular species, represented by the orange color in Figure 1, within the GAMA framework, are considered as a complement to the master species, each composed of specific characteristics and values that will serve as fundamental parameters for the creation and manipulation of the necessary instances during the simulation. Analogous to object-oriented languages, we can understand regular species as classes that define a set

[4]The CSV file used was adapted from [Li e Wilensky 2009a]

of properties and behaviors to be shared by their agents [Taillandier et al. 2019]. This organization is essential in GAMA to define and structure the entities that compose the simulation, providing them with identity and functionality.

When creating a species in GAMA, some attributes are automatically integrated, such as the name (by default, the same name by which the species was defined), location and shape. Attributes related to the context of the ABS under development must be explicitly defined, such as actions and behavioral characteristics of each species. Actions, in relation to object orientation, can be compared to methods, and they are what enable the interaction and behavior of the agents.

The instantiation of species (agents) for the simulation execution, as recommended by the language documentation, should be implemented in the `init` block of the global species [Taillandier et al. 2019]. This block is responsible for determining the quantity of agents to be created and, optionally, their initial location. To implement interaction between agents, GAMA provides the `ask` command. This command allows an agent to interact with instances of its own species or other species. For example, the `ask` command can be used to check which agent is the closest in a specific situation.

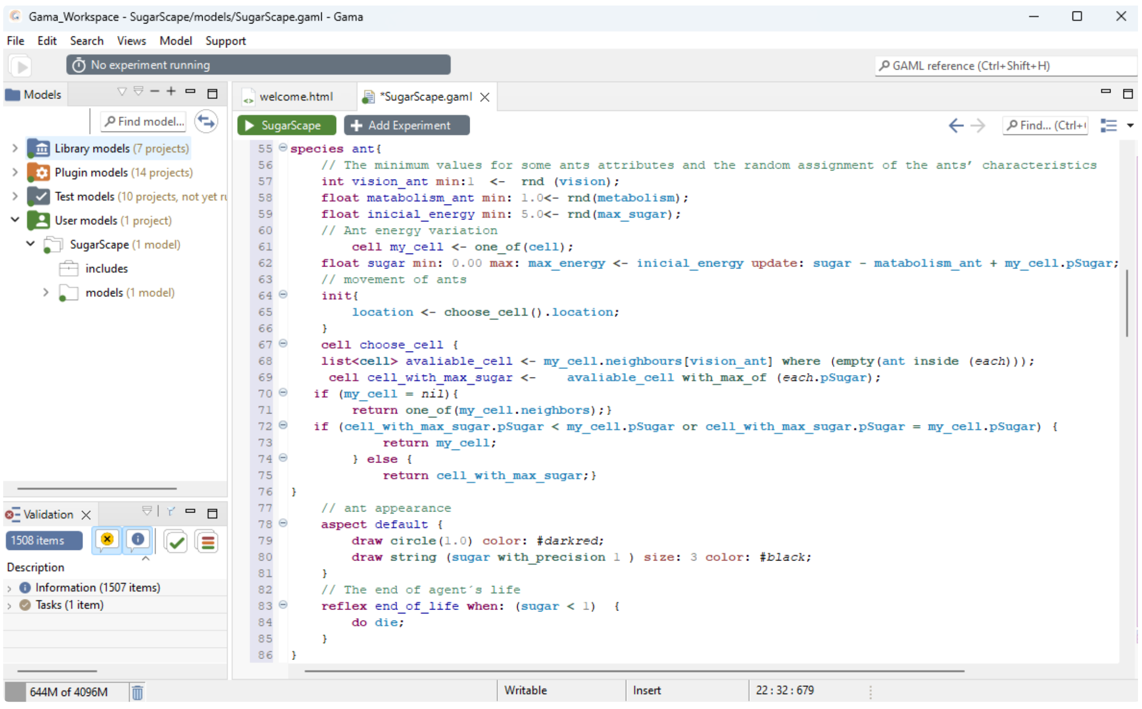
As a platform aimed at facilitating the creation of ABSs, GAMA offers additional resources that are not found in traditional object-oriented languages. This includes the availability of ready-to-use abilities, such as the movement ability, which provides predefined commands for agents to move and can be adjusted to change the speed and direction of agents as necessary. This flexibility allows for greater adaptation and customization of simulations according to the specific needs of each scenario.

In Sugarscape, the agents are ants. Therefore, it is necessary to define a species `ant` to instantiate the agents in the simulation execution. In the Sugarscape model, the ant species has three essential attributes: *vision*, which determines the number of accessible regions for sugar checking; *metabolism*, which symbolizes the energy expended to keep the ant alive; and initial energy, which defines the amount of energy with which the ant is born [Epstein e Axtell 1996]. In the ABS developed in this paper, minimum and maximum limits for initializing these attributes were specified. These limits are implemented in the global species. Upon creation, a random value is chosen between these limits for each characteristic of the ant using the `rnd()` command in GAMA.

In the `ant` species, the actions that the agents must perform during the simulation are also implemented. In GAMA, these actions can be implemented in the following ways: through the `Reflexion` command, which calls a method automatically at each cycle; through the `action` command, a method that can be activated from a `do` command; through the `ask` command, which refers to an interaction between agents; and finally, the `update` command, which updates an attribute at each cycle.

Another important method, is the movement in the environment. For this to occur, the map must be imagined as a matrix with several squares, each with a values. The ant must be able to check in each of these squares both the availability of sugar and the spatial availability, using the `empty` command.

Figure 2 presents a portion of the *Sugarscape* simulation implemented in the GAMA platform. In the left section, we find the *Users Models*, which allows developers to create customized ABSs, while a green button initiates the simulation. In the right section, there is the code where agent attributes can be configured, such as the random values for vision and metabolism of each agent, illustrating the use of the `<-` operator for value assignment. Additionally, the use of commands like `update` and `reflex` is evident, as mentioned earlier. Furthermore, there is the definition of agent appearance and the termination of their life.



```
species ant{
    // The minimum values for some ants attributes and the random assignment of the ants' characteristics
    int vision_ant min:1  <-  rnd (vision);
    float matabolism_ant min: 1.0<- rnd(metabolism);
    float inicial_energy min: 5.0<- rnd(max_sugar);
    // Ant energy variation
        cell my_cell <- one_of(cell);
    float sugar min: 0.00 max: max_energy <- inicial_energy update: sugar - matabolism_ant + my_cell.pSugar;
    // movement of ants
    init{
        location <- choose_cell().location;
    }
    cell choose_cell {
    list<cell> avaliable_cell <- my_cell.neighbours[vision_ant] where (empty(ant inside (each)));
     cell cell_with_max_sugar <-    avaliable_cell with_max_of (each.pSugar);
   if (my_cell = nil){
        return one_of(my_cell.neighbors);}
   if (cell_with_max_sugar.pSugar < my_cell.pSugar or cell_with_max_sugar.pSugar = my_cell.pSugar) {
            return my_cell;
        } else {
            return cell_with_max_sugar;}
}
    // ant appearance
    aspect default {
        draw circle(1.0) color: #darkred;
        draw string (sugar with_precision 1 ) size: 3 color: #black;
    }
    // The end of agent´s life
    reflex end_of_life when: (sugar < 1)  {
        do die;
    }
}
```

Figure 2. The SugarScape code on the GAMA platform

3.3. Grid Species

The *Grid* species, represented by the green color in Figure 1, is unique compared to other species, as it has unique attributes and behaviors that are essential for defining the environment where the simulation takes place. Unlike conventional species, it is automatically generated and is present in all simulations. The *Grid* is responsible for defining the characteristics of the map. In general, it is a structure that represents a grid or spatial mesh on which agents can move and interact in a simulation. This grid is composed of cells, where each cell can contain different attributes, such as the amount of available resources, the presence of obstacles, or other relevant information to the ABS.

By default, the *Grid* species has a series of attributes, such as the number of rows and columns, as well as the predefined color. However, all these characteristics can be modified if we define the species in the code. When manipulating the *Grid*, we are actually altering a matrix that, by default, has dimensions of 100x100. In other words, all changes will permeate every value of the matrix. Additionally, when defining the *Grid* species, it is necessary to specify which neighbor rule will be adopted for the cells. GAMA supports the following neighbor rules: *Von Neumann*, where each cell perceives 4 neighbor cells (horizontally and vertically); *Hexagon*, perceiving 6 neighbor cells; and *Moore*, perceiving 8 neighbor cells.

In Sugarscape, the *grid* species is defined with dimensions of 50x50. Each cell has attributes representing the current amount of sugar and the maximum amount of sugar it can hold. Depending on the version of the implemented simulation, the sugar growth rate can be added as an attribute, which varies based on both consumption and progressive recovery. This feature is implemented using the `update` command, providing greater dynamism to the simulation. The maximum sugar amount is initialized from the CSV file loaded in the global species. A value of zero indicates the absence of sugar in the cell. The adopted neighborhood rule is the *Von Neumann*, where neighboring cells are considered by the agent when searching for cells with sugar to move to.

3.4. Experiment

For an ABS to be effective and representative, it's desirable that it's visual and encompasses all the characteristics defined in the various species of the simulation. This requires a precise definition of the experiment, represented by the purple color in Figure 1, where the user has the ability to configure inputs, outputs, and behaviors as needed for the simulation. In GAMA, there are four main types of experiment that can be implemented to meet different needs and contexts [Taillandier et al. 2019].

- *GUI*: It offers a graphical user interface that allows interaction with the simulation. This facilitates parameter manipulation and real-time observation of results, providing a more intuitive and immersive experience.
- *Batch*: It allows running the simulation multiple times, with the possibility of prechanging parameter values for each run. This approach is useful for comparative analyses and evaluating results in different scenarios, providing insights into the model's behavior under various conditions.
- *Test*: It focuses on performing utility tests to ensure the quality and integrity of the simulation. This approach is essential for validating the accuracy and robustness of the model, identifying any potential flaws or inconsistencies that could compromise the results.
- *Memorize*: It maintains a detailed record of each step taken during the simulation. This allows the user to review and modify any stage of the process as needed, providing greater control and flexibility in conducting the simulation.

When defining the experiment, it's necessary to specify the type of simulation adopted. Additionally, the visual and interactive configurations of the simulation are also defined. This includes determining how relevant information will be presented through graphs and monitors, which play a role in interpreting the data and making informed decisions based on the results obtained.

In Sugarscape, the type of simulation adopted is the GUI. Through the graphical interface generated by GAMA, it is possible to explore and analyze the behavior of the ants in the environment. This choice allows for direct interaction with the simulation through display elements. The displays, which are graphical representations of some elements of the simulation, facilitate understanding and experimentation with different configurations to investigate various aspects of the phenomenon under study.

In the experimental species of Sugarscape, four display elements are defined. The first display, is called *display grid*, is a visual representation of the simulation's grid, showing the cells and agents. The other three displays specify graph elements that

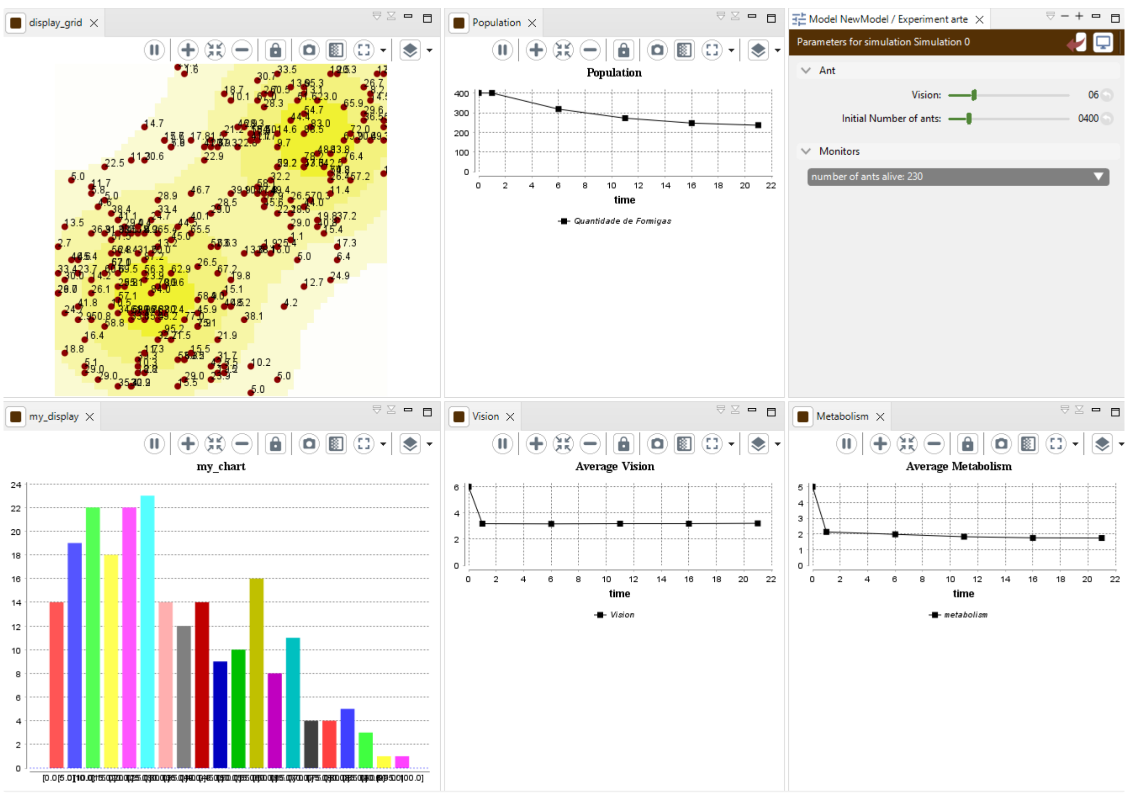


Figure 3. SugarScape simulation made on GAMA

display aggregated data from the various agents collected through the definition in the global species: *population* is a line graph that shows the number of ants in the simulation over time; *vision* is a line graph showing the average vision of the ants over time; and metabolism is also a line graph, displaying the average metabolism of the ants.

Besides display elements, the experimental species also allows for the definition of monitoring elements. These elements show values throughout the execution of the simulation, providing the user with a quick and convenient way to monitor the simulation's progress and make decisions based on the presented data. In Sugarscape, a monitor is defined to display the number of ants in the simulation.

Figure 3 show us a visual representation of the ongoing simulation is presented, comprising four distinct monitors. The first monitor, labeled *Display grid*, graphically displays the distribution of ants on the map grid, accompanied by a histogram that assesses the energy distribution among live ants. Subsequent monitors, *Population*, *Average Metabolism*, and *Average Vision*, provide the variation in the population of live agents, average metabolism, and average vision of ants over time. Additionally, the figure includes interactive parameters allowing users to adjust the agents' vision range and modify the initial quantity of ants, alongside a real-time monitor showing the quantity of live ants in the simulation, providing an instant view of the current population status.

4. Recommendations for Beginners in GAMA

Developing on the GAMA platform can present a significant challenge, even for those familiar with ABMs. This is partly because the platform is still under development and may exhibit errors in executing its commands. Additionally, the available documentation can be confusing for beginners, lacking in information and organization, which contributes to an inadequate understanding for the developer.

During the development process of Sugarscape, it was necessary to find workarounds for some issues. One such issue involved the `neighbours` command, which was supposed to return the neighboring cells to the agent's current position. However, this command did not worked as expected, ignoring the predefined configuration specified for the grid. This issue was reported to the GAMA developers[5] through the platform's mailing list, and they acknowledged the problem with the command and suggested an alternative method to achieve the desired execution. Another problem reported to the developers[6] but still unresolved is the use of a histogram graph, which cannot be configured to adapt to the constantly varying energy levels of the agents.

Given this, a recommendation for beginners on the GAMA platform is to interact and post questions on the mailing list. However, it is important to note that responses from the GAMA developers often take a significant amount of time, and the answers may be insufficient for problem resolution. For example, the query regarding the histogram took three months to receive an inconclusive response about its application.

The scarcity of tutorials and explanatory articles about the functionality and development within the platform hinders the initial access for people interested in using GAMA for their simulations. Therefore, it is recommended to organize your questions and seek out forum discussions with other users of the platform. Often, these users are helpful and share their own experiences, which facilitates understanding and reduces the learning curve.

5. Conclusion

This work presented the elements of the GAMA platform that are essential for developing an ABS. To this end, the development of Sugarscape, a simple and widely known ABS, was reported. A diagram was built to highlight the essential structure of an ABM in GAMA, so that beginners on this platform can use this paper as an introductory guide to develop simulations.

Indeed, while the GAMA platform facilitates the development of ABMs, especially when compared to object-oriented languages, it does require dedication and study from the developer to understand its elements and perform basic operations. The absence of materials that assist the beginner developer in GAMA contributes to the difficulty of accessibility of the platform for those developing their first ABM. Therefore, works like this are relevant to contribute to the popularization, accessibility, and development of GAMA.

References

Bousquet, F. c., Bakam, I., Proton, H., e Le Page, C. (1998). Cormas: Common-pool resources and multi-agent systems. In Pasqual del Pobil, A., Mira, J., e Ali, M., editors, *Tasks and Methods in Applied Artificial Intelligence*, pages 826–837, Berlin. Springer.

[5]`https://groups.google.com/g/gama-platform/c/jOFr9ju7sS4/m/LekjUGQnAAAJ`

[6]`https://groups.google.com/g/gama-platform/c/fSDZzeQjs_E/m/wSdQZUVHAgAJ`

Costa, A. C. R. (2023). The tupinambá: An exercise in societal modeling. In *Anais do XVII Workshop-Escola de Sistemas de Agentes, seus Ambientes e Aplicações (WESAAC 2023)*, pages 44–54.

Daudé, E., Chapuis, K., Taillandier, P., Tranouez, P., Caron, C., Drogoul, A., Gaudou, B., Rey-Coyrehourcq, S., Saval, A., e Zucker, J.-D. (2019). ESCAPE: exploring by simulation cities awareness on population evacuation. In *Proceedings of the 16th International Conference on Information Systems for Crisis Response and Management (ISCRAM 2019)*, pages 76–93.

Epstein, J. M. e Axtell, R. L. (1996). *Growing Artificial Societies: Social Science from the Bottom Up*. MIT Press.

Gaudou, B., Huynh, N. Q., Philippon, D., Brugière, A., Chapuis, K., Taillandier, P., Larmande, P., e Drogoul, A. (2020). COMOKIT: A modeling kit to understand, analyze, and compare the impacts of mitigation policies against the COVID-19 epidemic at the scale of a city. *Frontiers in Public Health*, 8.

Kleiboer, M. (1997). Simulation methodology for crisis management support. *Journal of Contingencies and Crisis Management*, 5(4):198–206.

Klügl, F. e Bazzan, A. L. C. (2012). Agent-based modeling and simulation. *AI Magazine*, 33(3):29–40.

Li, J. e Wilensky, U. (2009a). Netlogo sugarscape 1 immediate growback model. Center for Connected Learning and Computer-Based Modeling, Northwestern University, Evanston, IL.

Li, J. e Wilensky, U. (2009b). Netlogo sugarscape 2 constant growback model. Center for Connected Learning and Computer-Based Modeling, Northwestern University, Evanston, IL.

Macal, C. M. (2016). Everything you need to know about agent-based modelling and simulation. *Journal of Simulation*, 10(2):144–156.

Pereira, A. H., Nardin, L. G., e Sichman, J. S. a. (2011). Coordination of agents in the robocup rescue: A partial global approach. In *2011 Workshop and School of Agent Systems, their Environment and Applications*, pages 45–50.

Taillandier, P., Gaudou, B., Grignard, A., Huynh, Q.-N., Marilleau, N., Caillou, P., Philippon, D., e Drogoul, A. (2019). Building, composing and experimenting complex spatial models with the GAMA platform. *GeoInformatica*, 23:299–322.

Teixeira, L. e Santos, F. (2020). Uma simulação com agentes para estudar a propagação da COVID-19 em ibirama(SC). In *Anais do XVII Encontro Nacional de Inteligência Artificial e Computacional*, pages 176–187, Porto Alegre, RS, Brasil. SBC.

Wilensky, U. (1999). NetLogo. Center for Connected Learning and Computer-Based Modeling, Northwestern University, Evanston, IL.

Wilensky, U. e Rand, W. (2015). *An introduction to agent-based modeling: modeling natural, social, and engineered complex systems with NetLogo*. Mit Press.