Final Report

INDY-1 Environmental Simulation

Selene Espinoza, Jake Dunkley, JaDante Hendrick, Anna Alquisiras

College of Computing and Software Engineering Kennesaw State University

CS 4850/03 Spring 2023

Prof. Sharon Perry

https://environmental-simulation.github.io/

Table of Contents

Problem statement

Artificial Intelligence (AI) is a new field of Computer Science that is being utilized in multiple academic fields. This project is intended as a proof of concept for the utilization of AI in the field of biology, specifically studying the ecology and seasonal behavior of animals. If the behavior of a “cell” of animal units, such as a wolf pack or a rabbit burrow, is studied and learned through an AI model using environmental inputs and behavioral outputs, then it is hypothetically possible to create software that would simulate the response from those models to fabricated environments, and consequently, simulate animal behavior with real world consequences.

Planning and designing

Originally, the plan was first to simply create models to train and allow them to operate in an environment. However, after speaking with Dr. Green, a biologist at the college, it became clear that there would need to be better control for specific variables and terminology. As such, the development was split into two phases: the Learning Phase and the Reacting Phase. The Learning Phase was intended to be where the AI models are created. In a professional application of this concept, this step would be solely under the jurisdiction of biologists and credited academia working with access to large amounts of data to process into a model. Afterward, the models would be used in the Reacting Phase, wherein these models would be brought into the GUI-based software, intended to be used by any entity desiring to see the effects of a hypothetical situation. This can theoretically be for academic purposes, such as in teaching students about AI or animal behavioral patterns, or it could also theoretically be used for prospective land development, to understand possible consequences from developing previously wild land.

Project Plan

Final Deliverables

1. Fully operational software, deployed and in production
2. Project Documentation

Milestone Events

1. By Feb 17th be able to access all project environments
    - Upload to GitHub. All developers should have shared git environments to be able to push branches to the repository.
    - Complete Design Documentation.
    - Defining and Reviewing requirements.
2. By March 3rd framework setup
    - Research data to begin generating a training data set.
    - Basic UI dummy functions
    - Basic Simulation Setup
    - CNN review
3. By March 17th have a prototype ready to present to peers
    - Review prototype design
    - Operational sub menus and basic agents complete.
    - Test prototype
    - Documentation updated
4. By March 24th have Final Integration / Full Documentation /
    - Final CNN algorithm to process the data.
    - Be able to comment on all code functions for better readability and maintainability.
    - All functions listed are written out in documentation.
    - Draft of final report

Meeting Schedule Date/Time

Remotely, the members will meet on Tuesdays or Thursdays from 2:00 PM – 3:00 PM. The team will meet every class period on Mondays and Wednesday, in-person. Besides that, the team will have discussions online, via Discord.

Collaboration and Communication Plan

Communication and collaboration tools

- GitHub, Google Drive
- Discord, Microsoft teams

Development tools

- Visual Studio Code

Learning Phase dev

The first thing to do for the building of the Learning Phase, where the models would be trained, is research. Studying what parameters to use, the range of the animals studied (wolf and rabbit, in this demo) that they explored each day, and the way that the time could be handled by the Summer and Winter seasons. Due to the difference in both behavior and environment, both seasons were created as their own independent simulations. The range itself was simple to figure out, using the range that an animal tends to explore from the center of their territory, and modifying it by the scale of each square in the environmental grid, to find how far of a range the animal group should "see" their environment. The chosen environmental inputs for the model was Human Development, Prey/Predator Population, and Vegetation. Added as a final input, was the proximity to another animal group of the same species, because in real life, animal packs tend to stay in their own territory.

Reacting Phase dev

First, the engine in which to create the software was decided. As it was fairly simple, a mixture of OpenGL and Imgui was used. These provided a window and clickable buttons, which was serviceable enough. The software would primarily run the models per each "cell" representing the animal packs in the simulation. In order to lower the amount of computation and inputs needed by the model in each cell, an algorithm was used to reduce the amount of inputs to only four, one per cardinal direction. These inputs were using the inputs of half of the total inputs, with regard to the cardinal direction needed. For example, for a cell to process a "North" input of Vegetation, the Vegetation values of all the squares within range of the same row of the cell, then one row up, and another row up, and so on, until the edge of the range is reached, is all weighted depending on how far "North" each square is, and this value is then given as the "North Vegetation" value, one of four for the Vegetation

layer. Something similar can be done for the animal cell proximity, though without the limits of range in this example, since usually animals will consider that range to be too close anyway.

Requirements Lists

1. Learning Phase

1.1. Animal model (Model A)

1.1.1. Contains multiple inputs for the model (A_I)

1.1.1.1. Environmental layers, four inputs of each to match cardinal directions (A_{En}, A_{Ew}, A_{Ee}, A_{Es} for each layer A_E)

1.1.1.2. Distance from another cell (A_T)

1.1.2. Contains multiple outputs for the model (A_O)

1.1.2.1. Two outputs for vector movement (A_{Vx}, A_{Vy})

1.1.2.2. Four outputs for action to take next iteration (A_m, A_s, A_d, $A_n \in A_A$)

1.1.3. Has weights in fully connected hidden layers (Nodes in A_N)

1.1.3.1. Randomized weights at first

1.1.4. Loss function uses known observance values of specific packs/groups of animals and their location (Loss of A_L)

1.1.5. For each seasonal iteration, the AI is adjusted to learn to be more accurate with its predictions

1.1.5.1. Summer & Winter seasons running concurrently but separately, and operating off the same model

1.1.6. Range variable to tell cell how far to "see" (A_R)

1.1.6.1. Range is scaled based off of biological data of average travel distance per day and the scale of the map

1.1.7. Saved as an independent XML file

1.1.7.1. Contains the name and length of the name

1.1.7.2. Contains the “range” variable”

1.1.7.3. Contains the scale of maps

1.1.7.4. Contains the input names

1.1.7.5. Contains the node weights of the hidden layers

1.2. Animal cell (Cell object C)

1.2.1. Contains AI model (C_A)

1.2.1.1. AI is shared with all other cells of the species (C^1_A, C^2_A, … $C^n_A = A$)

1.2.2. Observes the surrounding environment for model inputs

1.2.2.1. Cardinal direction inputs are created that scale with direction using range's calculations (C_{En}, C_{Ee}, C_{Es}, C_{Ew} becomes A_{En}, A_{Ee}, A_{Es}, A_{Ew} respectively for each layer of C_E)

1.2.2.2. Range from next closest cell (C_T becomes A_T)

1.2.3. Chooses actions based on its surroundings (C_O)

1.2.3.1. Vector-based movement, moves in a direction if it exceeds a threshold, horizontal or vertical, positive = East, North (C_{Ox} and C_{Oy} takes from A_{Vx}, A_{Vy})

1.2.3.2. Merge, Split, Die, and doing nothing (C_{Oa} takes highest value from A_A)

1.2.4. Checks for proximity to expected movement and adds each of these distances to make a final loss value (C_L becomes A_L)

1.3. Learning environment (Environments E)

1.3.1. Two dimensional grid (Grid E_{xy})

1.3.2. Contains layers of environmental values and the observance values of packs (E^1, E^2, … $\in$ E)

1.3.3. Follows commands from cells

1.3.3.1. Merge deletes adjacent cell and itself and creates a new cell, Split creates a new cell, Die also deletes a cell (C_{Oa})

1.3.3.2. Has a move function that a cell can call to move itself to a new location if possible (Uses C_{Ox} and C_{Oy} if not traveling beyond the edge of the map)

1.3.4. A timeline that has the environment change slightly so as to understand how cells will react and compare to real observance values and pack actions (E_T)

1.3.4.1. 95% of maps in the timeline will be for testing

1.3.4.2. 5% will be for evaluation and understanding model accuracy

2. Reacting Phase

2.1. Animal models (A^1, A^2, etc.)

2.1.1. Loadable from XML

2.1.2. Can automatically scale weights if some input data is not given in order to attempt to reproduce results with fewer inputs

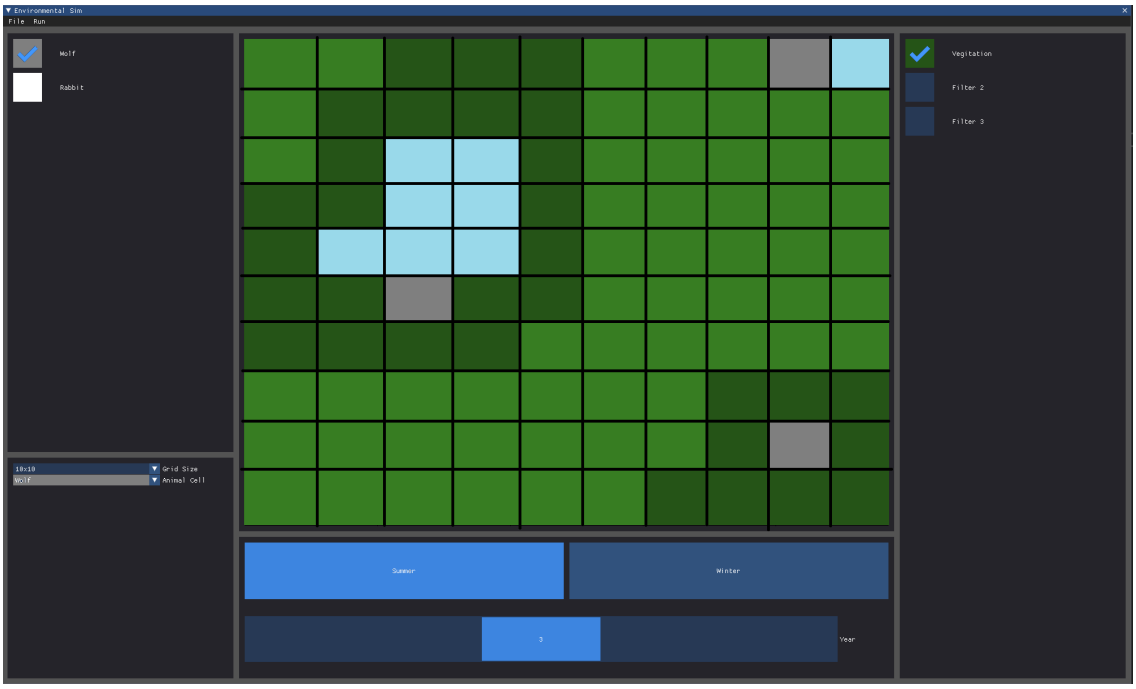
2.1.3. Does not change or "learn" in this phase

2.1.4. Interacts with the simulation map using cells and model much like in learning phase, but without the loss function and change to weights

2.2. Simulated environment (Map M)
    2.2.1. Map is loadable from file
    2.2.2. Contains layers and input names (Set of M_E)
2.3. Software presentation (Program S)
    2.3.1. Can display a simulated map from file (Map M)
        2.3.1.1. Can load environmental layers from map and allow the user to "activate" and "deactivate" them (Boolean M_A array for whether each layer is in use)
        2.3.1.2. Layer displays can be turned on and off to be seen on the environment simulation or not (Boolean M_V array for whether each layer is visible)
        2.3.1.3. Layers are visible through monochrome color gradient on map (M_V)
        2.3.1.4. Mouse over squares for info on each visible layer grid space
        2.3.1.5. Layers can be connected manually if needed (M_{E1}, M_{E2}, … can connect to A_E)
    2.3.2. Can load animal models from file (Model A^1, A^2, etc.)
        2.3.2.1. Displays backpropagation ratios of model AI in subwindow
            2.3.2.1.1. Selectable inputs and color gradients that represent effects on output (A_O)
        2.3.2.2. Fully displays all nodes (A_I, A_N, A_O)
            2.3.2.2.1. Hides hidden nodes by default (A_N)
        2.3.2.3. Allows placement of Animal cells onto map or randomly scatters a set amount
    2.3.3. Can create animal cells on the simulation grid
        2.3.3.1. Placeable cells by clicking
        2.3.3.2. Placeable "scatter" cells of an input amount randomly around the grid
    2.3.4. Operates from a timeline (S_T)
        2.3.4.1. Animal models react over time from starting positions of their cells and can be placed specifically in different positions
        2.3.4.2. Map does not change (Phase 2 addition of timeline change)
        2.3.4.3. Timeline is split between Summer and Winter (S_{Ts} and S_{Tw} respectively)

Screen Mockup

Prototype Grid



Gantt Chart

Project Name: INDY-1 Environmental Simulation Engine with AI/ML

Deliverable	Tasks	Complete%	Current Status Memo	Assigned To	*Project Environment*			02/10	*Framework*		02/24	03/03	Prototype		03/17	*Final Integration*		03/24	C-Day 04/07	
Requirements	Meet with team	100%		ALL	20	12														
	Define requirements	100%		ALL																
	Review requirements	100%		ALL		10	15													
	Sign off on requirements	100%		ALL			5	10												
	Fully Complete Design Docs	70%		Anna, Selene			5	10												
Project design	Begin work on CNN	0%		Jake				10	4											
	Basic Window Functions / UI	0%		Selene, JaDante				5	10	10										
	Save/laod	0%		JaDante			6	6	10	5	5									
	Basic Sim. Setup	0%		Jake, Selene						10	10									
	CNN protoype	0%		Jake, Selene							5	10	5			5				
	Observable timeline	0%									5	10	5			5				
	Clickable Sub Menus	0%									5	10	5			5				
	Prototpye basic agents	0%									5	10	5			5				
	End of data collection	0%		Anna, Selene							5	10	5			5				
Development	Fully operational sub menus	0%										8	5	10		5	10			
	Basic Agents	0%										8	10	20	20	10	20	20		
	Finish Website	0%													10	10		10	10	
	CNN final	0%													10	10		10	10	
	Test product	0%												8	5	20	8	5	20	
Final report	Presentation Documents	0%														15			15	10	10
	Presentation preparation	0%																			10
	Poster preparation	0%																			10
	Final report submission to D2L and project owner	0%																			5
			Total work hours	**525**	**20**	**22**	**31**	**41**	**24**	**25**	**40**	**66**	**40**	**38**	**45**	**55**	**40**	**38**	**45**	**55**	**10**	**35**

* formally define how you will develop this project including source code management

Version Control

GitHub Desktop: Local file changes reflected in environmental-simulation.github.io repository
Documents: Google Docs

Final product

The final product was quite simple. A GUI allows the loading of the animal models and environmental layers. The squares are clickable, to allow one to place cells wherever the user may want them. Afterwards, the processing of the models over a time span can be performed, with the cells using their respective species model shared amongst them to understand what the next action should be. These models were trained off real world data, meaning that it is possible to create a loss function and find a proper comparison to find how well the AI fits to real life. Hopefully, this proof of concept of AI learning and how well it can fit to real world data inspires similar programs and uses for machine learning.

Conclusion

Some of the end product was unfortunately scaled down due to scope. However, ultimately this was a particularly successful proof of concept of what AI can do for academic fields, specifically in animal behavior, as it already has in many other fields. While there should be no claims that this program or any form of machine learning can “fix” or “improve” the methodology and research that has gone into the related subject matter, it is still possible that software such as this can provide some additional data processing or predictive techniques for biologists. Hopefully, with further application of this technology, simulations of how deeply humanity has affected nature around itself can help with conservation strategies.